International Journal on Advances in Systems and Measurements

2010 vol. 3 nr. 1&2

The *International Journal on Advances in Systems and Measurements* is published by IARIA.
ISSN: 1942-261x
journals site: http://www.iariajournals.org
contact: petre@iaria.org

Sponsored by IARIA
www.iaria.org

International Journal on Advances in Systems and Measurements
Volume 3, Number 1 & 2, 2010

- Jean-Henry Morin, University of Geneva - CUI, Switzerland
- Igor Podebrad, Commerzbank, Germany
- Leon Reznik, Rochester Institute of Technology, USA
- Chi Zhang, Juniper Networks, USA

Sensor Networks

- Steven Corroy, University of Aachen, Germany
- Mario Freire, University of Beira Interior, Portugal / IEEE Computer Society - Portugal Chapter
- Jianlin Guo, Mitsubishi Electric Research Laboratories America, USA
- Zhen Liu, Nokia Research – Palo Alto, USA
- Winston KG Seah, Institute for Infocomm Research (Member of A*STAR), Singapore
- Radosveta Sokkulu, Ege University - Izmir, Turkey
- Athanasios Vasilakos, University of Western Macedonia, Greece

Electronics

- Kenneth Blair Kent, University of New Brunswick, Canada
- Josu Etxaniz Maranon, Euskal Herriko Unibertsitatea/Universidad del Pais Vasco, Spain
- Mark Brian Josephs, London South Bank University, UK
- Michael Hubner, Universitaet Karlsruhe (TH), Germany
- Nor K. Noordin, Universiti Putra Malaysia, Malaysia
- Arnaldo Oliveira, Universidade de Aveiro, Portugal
- Candid Reig, University of Valencia, Spain
- Sofiene Tahar, Concordia University, Canada
- Felix Toran, European Space Agency/Centre Spatial de Toulouse, France
- Yousaf Zafar, Gwangju Institute of Science and Technology (GIST), Republic of Korea
- David Zammit-Mangion, University of Malta-Msida, Malta

Testing and Validation

- Cecilia Metra, DEIS-ARCES-University of Bologna, Italy
- Krzysztof Rogoz, Motorola, Poland
- Rajarajan Senguttuvan, Texas Instruments, USA
- Sergio Soares, Federal University of Pernambuco, Brazil
- Alin Stefanescu, University of Pitesti, Romania
- Massimo Tivoli, Universita degli Studi dell'Aquila, Italy

Simulations

- Tejas R. Gandhi, Virtua Health-Marlton, USA
- Ken Hawick, Massey University - Albany, New Zealand
- Robert de Souza, The Logistics Institute - Asia Pacific, Singapore
- Michael J. North, Argonne National Laboratory, USA

International Journal on Advances in Systems and Measurements
Volume 3, Numbers 1 & 2, 2010

CONTENTS

SoC yield Improvement

Using TMR Architectures for Manufacturing Defect Tolerance in Logic Cores

Julien Vial Arnaud Virazel Alberto Bosio Luigi Dilillo Patrick Girard Serge Pravossoudovtich
Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier – LIRMM
Université de Montpellier II / CNRS
161, rue Ada
34392 Montpellier Cedex 5, France
Email: {vial, virazel, bosio, dilillo, girard, pravo}@lirmm.fr URL: http://www.lirmm.fr/~w3mic

***Abstract*—Manufacturing processes in the nanoscale era are less reliable leading to lower yields. As memories are the most important contributor to SoC (System-on-Chip) yield, fault tolerance techniques based on redundancy are generally used to improved memory yield. Conversely, logic cores embedded in SoC usually do not have these important features and manufacturing defects affecting these cores decrease the yield of the entire SoC. Therefore, meaningful techniques for SoC yield improvement must also consider logic cores. In this paper, we propose and investigate the use of TMR (Triple Modular Redundancy) architectures for logic cores to increase the overall SoC yield. We also propose to improve the defect tolerance capabilities of TMR architectures by partitioning logic cores and adding voters on logic core's cuts. Results show that this improvement of the TMR architecture is very fruitful to improve its tolerance capability with a low overhead in term of silicon area. Results obtained on SoC examples (ISCAS'85 and ITC'99 benchmark circuits as logic cores merged with memory cores with different rates) demonstrate the interest of using TMR architectures for logic cores for SoC yield enhancement.**

Keywords-system-on-chip; logic cores; manufacturing defects; yield ramp-up; fault-tolerance; TMR; test of tolerant architecture.

I. INTRODUCTION

System on Chip (SoC) has becoming a widely accepted architecture for complex and heterogeneous systems that include digital, analog, mixed-signal, radio frequency, micromechanical, and other types of components on a single piece of silicon. The context of this study is related to SoCs that are composed of two types of cores: memory and logic cores, where memory cores occupy the major silicon area. This is confirmed by the SIA (Semiconductor Industry Association) roadmap, which forecasts a memory density of up to 90% *w.r.t.* the overall SoC area in the next few years [1].

Despite the efficiency of their design and manufacturing processes, SoCs may be affected by defects leading to yield loss. To increase SoC yield, memory cores usually embed fault tolerance techniques, based on hardware redundancy (spare rows and columns) [2]. With the technology reaching nano dimension even the logic cores, embedded in a SoC, become a challenge for the overall SoC yield level. Thus, to achieve a better yield, also logic cores have to be designed with some fault tolerant techniques in order to tolerate manufacturing defects.

Existing fault tolerant techniques are commonly used to tolerate on-line faults [3]. They use redundancies, *i.e.*, the property of having spare resources that perform a given function and tolerate defects. Fault tolerance techniques are generally classified depending on the type of redundancy. Basically, four types of redundancy are considered: software, information, temporal and hardware [4].

In software redundancy, error detection and recovery are based on replicating application processes on a single or multiple computers [5].

In information redundancy, additional data are used. For example, the use of error-correcting codes requires extra bits that need to be added to the original data bits [4]. Nevertheless, the use of error-correcting codes is widely used in memory context; its application to logic cores requires an important design effort associated to a high area overhead used to predict and compute the code computation.

Temporal redundancy consists in forcing the system to repeat a given operation and then compare the results with those of the previous operation [6]. Such a redundancy is able to tolerate transient or intermittent errors but not permanent errors. Since manufacturing defects lead to permanent errors, this solution is not suitable for the target of our study.

Hardware redundancy consists in modifying the design by adding additional hardware. For example, instead of having a single processor, three processors are embedded to perform the same operation. The failure of one processor is tolerated thanks to a voter that chooses the majority outputs [4].

In [7, 8, 9, 10], we have considered the well-known TMR fault tolerant architecture in order to tolerate manufacturing defects in logic cores. In this paper, we extend this study to the SoC context. We first determine the set of conditions to be satisfied in order to successfully resort to TMR to ramp-up the overall SoC yield. These conditions are related to the testability of the TMR version of logic cores. Therefore, these conditions are evaluated by using an ad-hoc ATPG procedure. The evaluation has been done on several SoC

examples with different memory ratio. Based on these results, we first analyze the impact of the robustness of the voter. Then, we propose to improve the fault tolerance of TMR architectures by partitioning logic cores. Results obtained on SoC examples (composed of ISCAS'85 and ITC'99 benchmark circuits as logic cores and different memory ratio) demonstrate the interest of using TMR architectures for logic cores in the SoC context.

The remaining of the paper is organized as follows. In Section II, we detail the TMR approach. Section III shows the impact of using TMR architectures for logic cores on the overall SoC yield. Section IV details the test strategy targeting TMR architectures. Section V presents a set of experimental results. Section VI presents an analysis of voter robustness and an improvement of the TMR structure to make it able to tolerate more defects. Finally, concluding remarks are given in Section VII.

II. THE TMR APPROACH

In this section, we first present the TMR architecture and then we show how many defects it can tolerate.

A. Basic principle

Several hardware fault tolerant architectures have been proposed in the literature [11]. Generally, the degree of fault tolerance is defined as the maximum number of faults that can be tolerated in the system [12]. The classical hardware redundancy architecture is the NMR (N Modular Redundancy). A NMR structure is a fault tolerant architecture based on N modules performing the same function. The outputs of these modules are compared by using a majority voter. In general, this architecture can tolerate at least $(N - 1) / 2$ defects.

The case of $N = 3$ is called TMR and has been widely studied and used in practical applications [13, 4]. The inputs to three identical modules are tied together receiving thus the same data, and their outputs feed a majority Voter (V) circuit as shown in Figure 1.

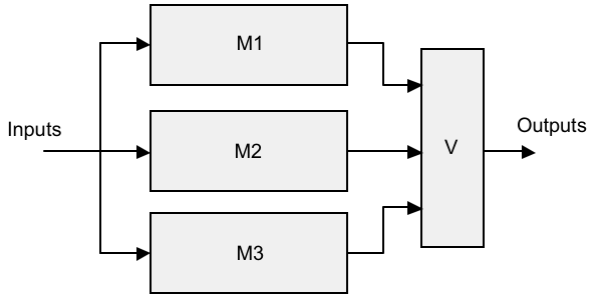


Figure 1. TMR principle

As a result, the TMR architecture significantly reduces the error probability at the primary outputs of the system. The erroneous value propagated by a defective module can be masked thanks to the presence of the two other fault-free modules. In the simplest structure, the voter is the weak point. If a fault appears in the voter, the TMR structure can be possibly faulty. To avoid this type of problem, the voter can be realized in software or with more robust design techniques [14].

B. How many defects can be tolerated?

Basically, the TMR architecture can tolerate one defect but, in practice, it can tolerate more than one defect. In fact, if there are two defects, the TMR may function properly depending on the nature and the location of the defects. Two defects are simply tolerated by the TMR if their induced errors cannot simultaneously drive the majority voter. On the other hand, two defects are not tolerated if there are located in two different modules and then propagate an error towards identical outputs on each module.

- Let P be the input pattern.
- Let Ω_i be the set of erroneous outputs in the module i due to the first defect when P is the input (i = 1, 2, 3).
- Let Ω_j be the set of erroneous outputs in the module j due to the second defect when P is the input (j = 1, 2, 3).

Under these constraints, the capability to tolerate two defects is formalized as follows:

- If $i = j$, defects are in the same module and, consequently, are tolerated.
- If $i \neq j$ and $\Omega_i \cap \Omega_j = \varnothing$, defects are tolerated.
- If $i \neq j$ and $\Omega_i \cap \Omega_j \neq \varnothing$, defects are not tolerated.

In Figure 2, two examples are shown with the same pattern P feeding the three modules. The voter has been omitted. Each defect is modeled as a stuck-at fault (f_1 and f_2 respectively). In the case of Figure 2.a, f_1 is propagated towards $O1$ in the first module and f_2 is propagated towards $O2$ in the second module. The majority voter receives two correct values and one wrong value. The outputs of the TMR are therefore correct and, consequently, faults f_1 and f_2 are tolerated.

In the case of Figure 2.b, f_1 is propagated towards $O1$ and $O2$ while f_2 is propagated towards $O2$. The voter receives one wrong value for $O1$ and two wrong values for $O2$. So, the value on the second output of the TMR is faulty. Consequently, faults f_1 and f_2 are not tolerated.

As a result, two faults are tolerated when there is no pattern able to propagate errors, coming from the two faults in different modules, toward identical outputs in each module.

In the case of more than two defects, multiple defects can be handled by considering all possible fault couples between them. For example, three defects are supposed to behave as three couples of defects. If we assume now that all these fault couples have non-masking behaviors (which is very unlikely in practice), we can handle multiple defects by simply considering all fault couples independently. Moreover, three defects are not observable if the three associated fault couples are not observable individually. Such assumption is reasonable as it relies on the same principles (seldom masking phenomena) than the single stuck-at fault assumption with respect to multiple stuck-at faults.

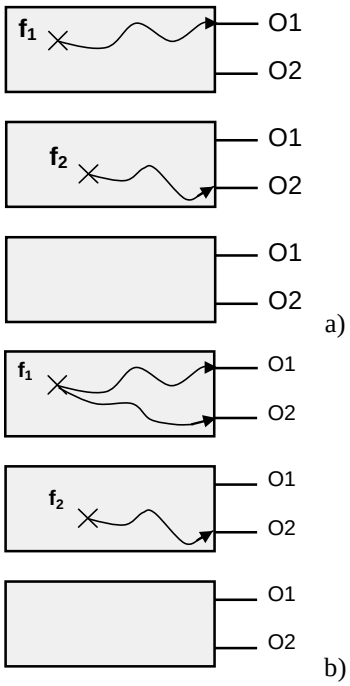


Figure 2. Two stuck-at faults are a) tolerated and b) not tolerated

III. USING TMR ARCHITECTURES IN SOC CONTEXT

In this section, we investigate the interest of using TMR architectures in order to tolerate manufacturing defects in logic cores and consequently improve the overall SoC yield. We therefore analyze the conditions to be fulfilled to achieve benefits in implementing such a TMR architecture instead of using a simple non-tolerant architecture.

A SoC composed of memory and logic cores has a yield expressed by:

$$Y_{SoC} = Y_L \times Y_M \tag{1}$$

where Y_M is the yield of memory cores and Y_L is the yield of logic cores. These two yield factors are computed by considering the contribution of the whole memory cores and the whole logic cores:

$$Y_M = \prod_{i=1}^{mc} Y_{Mi} \quad and \quad Y_L = \prod_{i=1}^{lc} Y_{Li} \tag{2}$$

where *mc* is the number of memory cores and *lc* is the number of logic cores. Assuming that memories embed fault tolerance technique, Y_M can hence be considered close to 100%. Consequently, to further increase the SoC yield, we may resort to TMR architectures for tolerating manufacturing defects in logic cores.

Let A_L be the area of logic cores in the original SoC, A_M be the area of memory cores and A_V the area of voters needed for the TMR implementation in the fault tolerant SoC (SoC_{TMR}). The resulting area overhead of the TMR architecture will be:

$$A_O = \frac{A_M + 3A_L + A_V}{A_M + A_L} = M + \frac{3A_L + A_V}{A_M + A_L} \tag{3}$$

with $M = A_M / (A_M + A_L)$ representing the memory occupancy ratio in the SoC. Thus, if we triplicate all logic cores to implement TMR architectures on a given silicon area (*e.g.*, a wafer) containing several SoCs, we can have n SoC_{TMR} having a yield equal to Y_{SoCTMR} ($Y_{SoCTMR} \times n$ SoC_{TMR} having a correct behavior). On the other hand, without TMR we can have $A_O \times n$ SoC having a yield equal to Y_{SoC} ($Y_{SoC} \times A_O \times n$ defect-free SoC). TMR architectures are therefore worthwhile only if:

$$\frac{Y_{SoCTMR}}{A_o} > Y_{SOC} \tag{4}$$

As $Y_{SoCTMR} \leq 1$, using TMR architectures for yield improvement is interesting only if:

$$Y_{SOC} \leq \frac{1}{A_O} \tag{5}$$

Let us now compute Y_{SoCTMR} and Y_{SoC} by using the Poisson distribution to model the defect distribution on the SoC. It would not be completely accurate to use the Poisson distribution for large circuits due to clustering effects on defects. More accurate calculation could be obtained by using, *e.g.*, the negative binomial distribution model [15, 16]. But, for a first and rough evaluation this is reasonable. In our case, the Poisson distribution is a discrete probability distribution that defines the probability that a number of manufacturing defects occurs in a fixed area if these defects occur with a known probability. Let X be the number of manufacturing defects. Let λ be the average number of expected defects for a given silicon area. Then, $\lambda = n \times p$ with n being the number of logic gates (or transistors) and p being the average defect rate of a gate (or a transistor). Let $P\{X = k\}$ be the probability that the circuit has k manufacturing defects. If n is high and p is low, the binomial distribution becomes the Poisson distribution:

$$P\{X = k\} = e^{\lambda} \times \frac{\lambda^k}{k!} \tag{6}$$

In the original SoC, the presence of a fault in logic cores makes the entire system faulty. So, Y_L is the probability that there is no defect inside logic cores:

$$Y_L = P\{X = 0\} = e^{\lambda_L} \tag{7}$$

Consequently, Y_{SoC} becomes:

$$Y_{SoC} = e^{\lambda_L} \times Y_M \tag{8}$$

The computation of Y_{SoCTMR} is more complex as it depends on *i)* the ability of the TMR to tolerate manufacturing defects and *ii)* the yield of voters. Consequently, Y_{SoCTMR} can be expressed as follows:

$$Y_{SoCTMR} = Y_T \times Y_V \times Y_M \tag{9}$$

where Y_T is the yield of the triplication without considering the voter and Y_V is the yield of voters. Let us first compute Y_T as the probability that no defects occur plus the probability that the TMR tolerates all the defects, Y_T is formally defined as follows:

Probability that there are 2 defects

$$Y_T = \underbrace{P\{X=0\}}_{\text{Probability that there is no defect}} + \underbrace{P\{X=1\}}_{\text{Probability that there is 1 defect}} + R \times \overbrace{P\{X=2\}} + R^3 \times \underbrace{P\{X=3\}}_{\text{Probability that there are 3 defects}} + \ldots$$

n defects are equivalent to C_n^2 couples of defects

$$Y_T = e^{\lambda_T} \times \left(1 + \lambda_T + R\frac{(\lambda_T)^2}{2!} + R^3\frac{(\lambda_T)^3}{3!} + \ldots\right) \tag{10}$$

with R being the probability that two defects are tolerated, *i.e.*, R reflects the tolerance capability of the TMR architecture.

There are three times more gates (or transistors) into a TMR architecture than into a non-redundant logic core. So, by substituting λ_T by $3\lambda_L$ we obtain:

$$Y_T = e^{-3\lambda_L} \times \left(1 + 3\lambda_L + \sum_{i=2}^{\infty} R^{C_i^2} \times \frac{(3\lambda_L)^i}{i!}\right) \tag{11}$$

Y_V is the probability that there is no defect inside voters:

$$Y_V = P\{X=0\} = e^{\lambda_V} \times \frac{(\lambda_V)^0}{0!} = e^{\lambda_V} \tag{12}$$

and, by substituting λ_V by $\lambda_L \times A_V / A_L$ we obtain:

$$Y_V = e^{\lambda_L \frac{A_V}{A_L}} \tag{13}$$

Consequently, by considering Eq. (3), (9), (11) and (13) and by substituting $\lambda_L = (1-M)\lambda_{SoC}$, Y_{SoCTMR} becomes:

$$Y_{SoCTMR} = e^{(A_O - M)\lambda_{SoC}} \times \left(1 + 3(1-M)\lambda_{SoC} + \sum_{i=2}^{\infty} R^{C_i^2} \times \frac{[3(1-M)\lambda_{SoC}]^i}{i!}\right) \times Y_M \tag{14}$$

and with $Y_{SoC} = e^{\lambda_{SoC}} \Rightarrow \lambda_{SoC} = -\ln Y_{SoC}$:

$$Y_{SoCTMR} = e^{(A_O - M)\ln Y_{SoC}} \times \left(1 - 3(1-M)\ln Y_{SoC} + \sum_{i=2}^{\infty} R^{C_i^2} \times \frac{[-3(1-M)\ln Y_{SoC}]^i}{i!}\right) \times Y_M \tag{15}$$

To summarize, implementing TMR architectures for logic cores can improve the overall SoC yield if two conditions are satisfied. First, $Y_{SoC} \leq 1/A_O$; this condition is related to the area overhead needed to implement TMR architectures for logic cores. Secondly, $Y_{SoCTMR} \geq A_O \times Y_{SoC}$ with Y_{SoCTMR} depending on Y_{SoC}, A_O, M and R as shown in Eq. 15. To satisfy these conditions, we have to analyze the impact of A_O, M and R on Y_{SoCTMR}. A_O and M are easily computed but R requires determining the percentage of tolerated couple of defect. In fact, the problem is how can we determine the percentage of tolerated couple of defects for a given TMR architecture of logic cores. As shown in the example of Figure 2, tolerated defects lead to fault-free values at the outputs of the TMR architecture thus corresponding to the untestable defects. Consequently the problem of determining the percentage of couple of tolerated defects is equivalent to determine the untestable ones. In the next section we investigate the test issues related to TMR architectures in order to determine this percentage.

IV. TEST OF TMR ARCHITECTURES

In this section we detail specificities of the test of TMR architectures. Especially, we present how the fault list and the ATPG procedure have been created.

A. Test specificities

Testing tolerant architectures should be addressed in a different way compared to the test of standard circuits. Hereafter, we present the peculiarities of TMR testing. For the sake of simplicity, we consider that all modules are structurally identical. Nevertheless, further considerations are still valid also in case of structurally different redundant modules. The first peculiarity of a TMR architecture induced by its redundant nature makes the test approach very different from the test of a classical structure without redundancy. In this type of structure, a fault affecting only one module is masked thank to the two other fault-free modules.

Testing a TMR architecture depends on its final use. In a classical way of use, the goal is to tolerate potential on-line

defects (permanent and/or transient). Each module should be fault-free after manufacturing. Due to the intrinsic impossibility to test single stuck-at fault, the architecture has to be modified during test [3] as shown in Figure 3.a. In this modified architecture, redundancy is removed and each single stuck-at fault becomes testable. Each module is individually tested.

In the proposed way of use, the goal is to increase the yield by tolerating permanent defects due to an imperfect manufacturing process. In this case, the test consists in testing globally the TMR structure in order to determine what are the couples of defects which are tolerated or not (determination of the R value). In this case, the architecture is not modified for test purpose like the one presented in Figure 3.b. Finally, we remind that the TMR basically tolerates one single defect and optionally two or more defects as previously discussed. For example, if several defects are located in the same module, the TMR still works properly.

To summarize, in the first approach, the question to be answered during the test of the TMR is: "*Are there one or more than one manufacturing defects in each module of the TMR architecture?*".

In the second approach, the question becomes: "*Does the logic core pass the test despite the presence of one or more than one manufacturing defects?*".

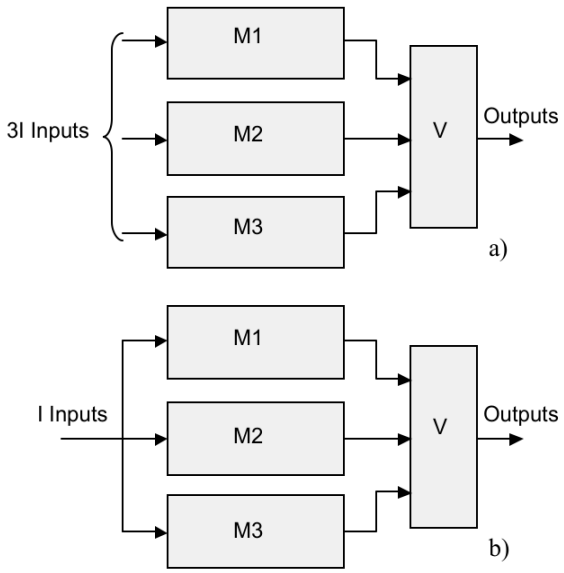


Figure 3. Test of a) an on-line fault tolerant structure and b) a manufacturing fault tolerant structure

B. Fault Model

The fault model widely used in structural testing to detect a manufacturing defect is the single stuck-at fault model. An ATPG based on single stuck-at fault model generates patterns able to detect all single stuck-at faults. Such patterns are not efficient in the context of multiple stuck-at faults occurring in a redundant TMR architecture. Since the single stuck-at fault is by definition tolerated (untestable) in the TMR architecture, a different fault model has to be considered. This fault model must be testable on primary outputs of the TMR, and also be representative of actual manufacturing defects. In Figure 2 we have seen that a couple of stuck-at faults can be observed (not tolerated) on primary outputs of the TMR and also can be testable.

In the following, we refer to a couple of stuck-at fault as fault pair [17]. If there are two stuck-at faults in the structure, these two faults are a fault pair. If there are more than two stuck-at faults, these multiple stuck-at faults can be gathered in fault pairs (under the assumption that no masking effect occurs between faults). For example, three stuck-at faults f_1, f_2, and f_3 are equivalent to three fault pairs $\{f_1, f_2\}$, $\{f_1, f_3\}$ and $\{f_2, f_3\}$. In general, n stuck-at faults are equivalent to C_n^2 fault pairs.

To be testable and thus not tolerated, the two faults composing the fault pair must affect two different modules and propagate toward common module's outputs as shown in Figure 2.

C. Size of the fault list

Let us consider that each module has n single stuck-at faults. The whole number of stuck-at faults in the three modules is $3n$. As $\{f_1, f_2\} = \{f_2, f_1\}$, the total number ψ of fault pairs is:

$$\Psi = C_{3n}^2 = \frac{3n!}{(3n-2)!\times 2!} = \frac{9n^2 - 3n}{2} \qquad (16)$$

We have now to compute the percentage of untestable (tolerated) fault pairs to determine if the TMR architecture is worthwhile for yield improvement. So, we simply generate test patterns able to detect all testable fault pairs. The remaining fault pairs are untestable. Since ψ is quadratic to n, the use of a classical ATPG to detect the entire set of fault pairs will be unfeasible due to a huge CPU time. In the next sub-section, we show how to handle this particular point.

D. ATPG procedure

In order to determine the convenience of using the TMR architecture to handle manufacturing defects, we have first to determine the untestable fault pairs. An ATPG targeting the fault pair model has to be run. Since the goal of this work was not to develop an ATPG for fault pairs, we have adapted an ATPG tool targeting single stuck-at faults to target fault pairs.

Considering that a fault pair is composed of two stuck-at faults (f_1, f_2), the proposed approach is to inject a permanent fault (f_1) in one module of the TMR architecture by modifying the *netlist*. Then, an ATPG is run to test all stuck-at faults in presence of the permanent fault f_1. This process is repeated until we have injected all stuck-at faults in one module.

The simplicity of this approach is obtained at the cost of high simulation time since n (number of single stuck-at faults

in one module) full ATPG runs are needed. To reduce this drawback, we have shortened the fault pair list to be handled as follows:

- When there is only one faulty module, the outputs of this module are masked by the voter. Thus, all fault pairs, which impact only one module, are structurally untestable. These fault pairs can be removed from the ATPG fault list.
- Symmetries of the TMR architectures are used to reduce the fault pair list of the ATPG. As the module inputs are tied together during test, fault pairs are equivalent when their two stuck-at faults have the same location in two different modules. Therefore, equivalent fault pairs are removed from the ATPG fault list.

With the help of these simplifications, the size of the fault pair list becomes:

$$Nb\_Faute = \frac{n^2 + n}{2} \tag{17}$$

The list of fault pairs can be further reduced by determining and removing fault pairs that are structurally untestable. Let us consider the fault pair $\{f_1, f_2\}$ and their output cones (list of outputs where the fault effect may be propagated) $\{\phi_1, \phi_2\}$. Due to the presence of the voter, if $\phi_1 \cap \phi_2 = \emptyset$, the fault pair $\{f_1, f_2\}$ is structurally untestable and therefore, can be removed from the fault pair list. In practical cases, the number of structurally untestable fault pairs is quite large leading to a huge improvement of the overall ATPG performance.

To summarize, Figure 4 presents the ATPG algorithm with the fault pair model.

V. TMR INTEREST FOR SOC YIELD IMPROVEMENT

In the previous section we state the constraints to be fulfilled in order to use TMR to increase SoC yield. In this section we evaluate those constraints on a set of benchmark SoCs.

```
create the fault pair list;

for (each fault pair {f1, f2}) {
        Compute output-cones {ϕ1, ϕ2} of f1 and f2;
        if (ϕ1 ∩ ϕ2 ≠ Ø) add fault pairs to the fault list ζ;
}

for (each single stuck-at fault f of module 1) {
        inject the permanent fault f in the VERILOG description;
        add all fault pair Є ζ with (f = f1 or f = f2) to the ATPG fault list;
        run ATPG;
}

create report;
```

Figure 4. ATPG algorithm

For the analysis, both ISCAS'85 and ITC'99 (combinational part only) benchmark circuits are used as logic cores in a SoC. These benchmark circuits are simply cloned three times and majority voters have been added to implement a TMR architecture. Results of the ATPG runs are reported in Table 1 for different values of *M* (Memory occupancy ratio).

The first column lists the benchmark circuit name used as logic core. Second and third columns show the number of stuck-at faults in one logic core (# SAF) and the number of fault pairs in the ATPG fault list (# ATPG FP) using reduction techniques proposed in [8]. The next columns give for three *M* scenarios (50%, 70% and 90%):

- $1/A_O$: Y_{SoC} limit under which it is interesting to implement a TMR as presented in Eq. 5.

TABLE I. INTEREST OF USING TMR ARCHITECTURES FOR SOC YIELD IMPROVEMENT

Logic core	ATPG data		M = 50%			M = 70%			M = 90%		
	# SAF	# ATPG FP	$1/A_O$ (%)	Rmin (%)	R (%)	$1/A_O$ (%)	Rmin (%)	R (%)	$1/A_O$ (%)	Rmin (%)	R (%)
c1908	1812	1.30M	49.02	90.68	56.42	61.58	86.36	56.42	82.78	65.75	56.42
c2670	2852	1.87M	46.19	95.88	75.95	58.86	94.00	75.95	81.10	84.92	75.95
c3540	3438	4.91M	49.50	89.54	54.09	62.03	84.67	54.09	83.06	61.59	54.09
c5315	4970	3.44M	48.08	92.69	93.20	60.68	89.31	93.20	82.24	73.10	93.20
c6288	6250	18.2M	49.63	89.24	38.02	62.15	84.23	38.02	83.13	60.49	38.02
c7552	7438	7.40M	48.90	90.95	84.92	61.46	86.76	84.92	82.71	66.74	84.92
b04	1477	535k	46.30	95.73	84.32	58.96	93.78	84.32	81.17	84.35	84.32
b05	2553	1.17M	48.08	92.69	88.65	60.68	89.31	88.65	82.24	73.10	88.65
b07	1120	399k	45.66	96.59	81.91	58.34	95.05	81.91	80.78	87.56	81.91
b11	1308	703k	47.73	93.35	74.50	60.35	90.28	74.50	82.03	75.54	74.50
b12	2777	857k	46.51	95.41	95.45	59.17	93.31	95.45	81.30	83.17	95.45
b13	835	59.6k	44.54	97.86	96.94	57.24	96.90	96.94	80.06	92.23	96.94

- *Rmin*: minimum value of *R* (percentage of untestable fault pairs) above which it is beneficial to use a TMR architecture.
- *R*: tolerance of the TMR architecture. It corresponds to the percentage of untestable fault pairs provided by the ATPG procedure.

On the base of these APTG results, we are now able to determine if implementing TMR architectures for logic cores will improve the overall SoC yield. This is simply done by comparing columns *Rmin* and *R* in Table 1 for each *M* scenario. It appears (concerned cases are highlighted by shaded boxes) that using TMR for logic cores in SoC is suitable for 2 logic core examples when $M = 50\%$, 3 logic core examples when $M = 70\%$ and 5 logic core examples when $M = 90\%$ among the 12 logic core examples. Therefore, TMR architectures are promising solutions for SoC yield improvement. In the next, section we propose to improve the fault tolerance of the TMR to achieve higher yield benefit.

VI. TMR IMPROVEMENT AND DISCUSSION

As shown in the previous section, TMR architecture seems to be a promising fault tolerant structure for logic core to improve the overall SoC yield. Hereafter, we discuss possible improvements of the basic TMR architecture by first analyzing the impact of the yield of the voters and secondly, by partitioning logic cores to increase the tolerance of the TMR architecture. Then, we analyze the SoC yield improvement. Finally, we extend the study to the general SoC context where several logic cores are embedded in a SoC.

A. Voter yield impact

For all equations and results presented before, the voters added to implement TMR architectures for logic cores have been considered to be possibly defective with the same defect density than the overall SoC. In this sub-section, we provide results by also considering that voters can be fault-free as they are manufactured with robust design techniques.

Results are reported in Table 2. The first column lists the benchmark circuit names used as logic cores in a SoC with three *M* scenarios (50, 70 and 90%). Then, for each *M* scenario, Table 2 first recalls $1/A_O$, values *Rmin* and *R* of Table 1 and then gives values *Rmin* and *R* obtained when voters are defect-free ($Y_v = 1$). By comparing again columns *Rmin* and *R* in Table 2, using TMR architectures for logic cores is suitable for SoC yield improvement if voters are designed in a robust way (Y_v close to 100%).

B. Fault tolerance improvement

To improve the tolerance of TMR architectures, we propose to increase the number of untestable fault-pairs by reducing the combinational depth of logic cores. This is done by partitioning each logic core into two or three equivalent blocks so as to increase the tolerance of the TMR architecture. As shown in Figure 5, majority voters are added on circuit's edge cut.

An important feature of partitioning the circuit is that the tolerance of fault pairs increases when the number of partition increases as well. For example, in the case of double TMR (Figure 5.b), cores (modules) are divided into two equivalent blocks. Each block operates independently and a manufacturing defect in the first block has no impact on the second block.

In a basic TMR architecture, the percentage of untestable fault pairs is always greater than 33.33% as two stuck-at faults in the same module are untestable. In a double TMR architecture, if we consider that the first fault f_1 impacts *M1'*, then the fault pair $\{f_1, f_2\}$ can be detected (not tolerated) if and only if f_2 is in *M2'* or *M3'*. Conversely, if f_2 is in *M1'*, *M1''*, *M2''* or *M3''*, the fault pair is necessarily tolerated due to the presence of the voters. Therefore, the percentage of tolerated fault pairs in a double TMR architecture is always greater than 66.66%.

TABLE II. IMPACT OF THE VOTER YIELD ON THE SOC YIELD IMPROVEMENT

Logic core	M = 50%					M = 70%					M = 90%				
	$1/A_O$ (%)	Rmin (%)	R (%)	$Y_v = 1$		$1/A_O$ (%)	Rmin (%)	R (%)	$Y_v = 1$		$1/A_O$ (%)	Rmin (%)	R (%)	$Y_v = 1$	
				Rmin (%)	R (%)				Rmin (%)	R (%)				Rmin (%)	R (%)
c1908	49.02	90.68	56.42	88.61	56.42	61.58	86.36	56.42	83.35	56.42	82.78	65.75	56.42	58.52	56.42
c2670	46.19	95.88	75.95	89.48	75.95	58.86	94.00	75.95	84.75	75.95	81.10	84.92	75.95	62.40	75.95
c3540	49.50	89.54	54.09	88.46	54.09	62.03	84.67	54.09	83.10	54.09	83.06	61.59	54.09	57.82	54.09
c5315	48.08	92.69	93.20	88.91	93.20	60.68	89.31	93.20	83.83	93.20	82.24	73.10	93.20	59.85	93.20
c6288	49.63	89.24	38.02	88.42	38.02	62.15	84.23	38.02	83.04	38.02	83.13	60.49	38.02	57.64	38.02
c7552	48.90	90.95	84.92	88.65	84.92	61.46	86.76	84.92	83.42	84.92	82.71	66.74	84.92	58.69	84.92
b04	46.30	95.73	84.32	89.45	84.32	58.96	93.78	84.32	84.70	84.32	81.17	84.35	84.32	62.26	84.32
b05	48.08	92.69	88.65	88.91	88.65	60.68	89.31	88.65	83.83	88.65	82.24	73.10	88.65	59.84	88.65
b07	45.66	96.59	81.91	89.63	81.91	58.34	95.05	81.91	84.99	81.91	80.78	87.56	81.91	63.08	81.91
b11	47.73	93.35	74.50	89.02	74.50	60.35	90.28	74.50	84.01	74.50	82.03	75.54	74.50	60.33	74.50
b12	46.51	95.41	95.45	89.39	95.45	59.17	93.31	95.45	84.60	95.45	81.30	83.17	95.45	61.97	95.45
b13	44.54	97.86	96.94	89.95	96.94	57.24	96.90	96.94	85.50	96.94	80.06	92.23	96.94	64.46	96.94

In the case of a triple TMR architecture (see Figure 5.c), circuits (modules) are divided into three equivalent blocks. If f_1 impacts *M1'*, the fault pair $\{f_1, f_2\}$ is necessarily tolerated if f_2 impacts *M1'*, *M1''*, *M2''*, *M3''*, *M1'''*, *M2'''* and *M3'''*. Consequently, the percentage of tolerated fault pairs is always higher than 77.77%.

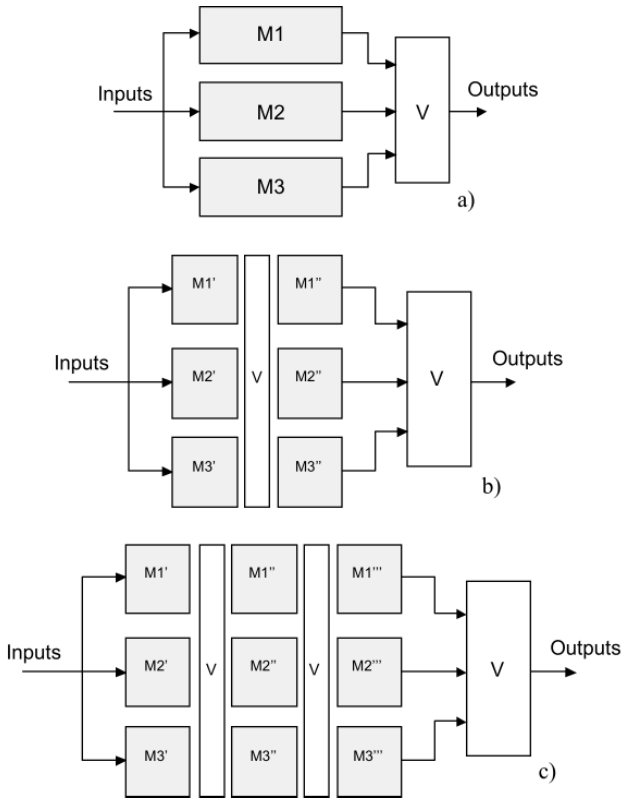


Figure 5. Improvement of the TMR architecture
a) basic, b) double and c) triple TMR architectures

More formally, if circuits are partitioned into *k* blocks, then the percentage of tolerated fault pairs is always higher than:

$$Tol\_FP = 100 \times \frac{3k-2}{3k} \quad (18)$$

With this technique, the tolerance of the TMR architecture increases with the number of partitions. The main drawback of this technique is the requirement of additional voters. In the next sub-section, we show that the overhead of silicon area due to additional voters still remains very low (less that 3% for large benchmark circuits).

The circuit partitioning consists in three steps. First, we transform the circuit into a hypergraph. A hypergraph is a generalization of a graph, where the set of edges is replaced by a set of hyperedges. A hyperedge extends the notion of an edge by allowing more than two vertices to be connected together. Formally, a hypergraph $H = (V, Eh)$ is defined as a set of vertices V and a set of hyperedges Eh, where each hyperedge is a subset of the vertices set V, and the size of an hyperedge is the cardinality of this subset. Hypergraphs can be used to naturally represent a gate-level VLSI circuit. The vertices of the hypergraph can be used to represent the gates of the circuit, and the hyperedges can be used to represent the lines connecting these gates.

In a second step, we make the hypergraph partitioning. The proposed technique of circuit partitioning for TMR architecture improvement utilizes sh-METIS, a multilevel hypergraph partitioning algorithm based upon the multilevel paradigm [18]. The quality of the partitioning produced by sh-METIS in terms of cut size (the size of the hyperedge cut is representative of the area overhead of the proposed solution), the computational time needed to partition large combinational circuits, and the availability of sh-METIS in a free access on the Web site of the University of Minnesota have motivated our choice for this academic tool. In a performance comparison of their multilevel partitioning algorithm with other state-of-the-art partitioning schemes, the authors of sh-METIS reported that their algorithm produces partitioning that are on the average 6% to 23% better (in terms of cut size) than existing algorithms, and often requires 4 to 10 times less time than that required by other schemes.

The final step consists in adding voters in the place of hyper-cuts. The number of added voters is equal to number of hyper-cuts that is optimal using sh-METIS software.

The ATPG procedure presented in Section IV.D has been run on *Double TMR* architecture, *i.e.*, logic cores have been partitioned into two modules (having the same size). Experimental results are reported in Table 3. The first column of the table lists the benchmark circuit names used as logic cores in a SoC with three *M* scenarios (50, 70 and 90%). Then, for each *M* scenario, Table 3 (see the last page of the paper) provides results for both *Basic TMR* and *Double TMR*. As previously, each sub-column gives values $1/A_O$, *Rmin* and *R*.

A first comment regarding these results is that the partitioning of logic cores increases the probability *R*. For example, when *M* = 90%, 11 logic core examples make the use of TMR architectures suitable for SoC yield improvement. Of course, if the number of partitions increases, the tolerance of the TMR increases too. This is shown in Table 4 that provides results using *Basic*, *Double* and *Triple* TMR for a unique *M* scenario (70%).

A second comment is that we can analyze the impact of the logic core partitioning on the area overhead (A_O). From data in Table 3 (column $1/A_O$) we have computed the area overhead needed to implement a *Double TMR* compared to a *Basic TMR* and done it for the different memory ratio. From these results, we notice that the area overhead is low for large benchmark circuits: less than 4.53% when *M* = 50%, less than 3.45% when *M* = 70% and less than 1.55% when *M* = 90%.

TABLE III. INTEREST OF USING LOGIC CORE PARTITIONING FOR SOC YIELD IMPROVEMENT

Logic core	M = 50%		M = 70%		M = 90%	
	Basic TMR	*Double TMR*	*Basic TMR*	*Double TMR*	*Basic TMR*	*Double TMR*

	$1/A_O$ (%)	Rmin (%)	R (%)	$1/A_O$ (%)	Rmin (%)	R (%)	$1/A_O$ (%)	Rmin (%)	R (%)	$1/A_O$ (%)	Rmin (%)	R (%)	$1/A_O$ (%)	Rmin (%)	R (%)	$1/A_O$ (%)	Rmin (%)	R (%)
c1908	49.02	90.68	56.42	47.85	93.13	89.49	61.58	86.36	56.42	60.46	89.96	89.49	82.78	65.75	56.42	82.10	74.74	89.49
c2670	46.19	95.88	75.95	45.77	96.46	92.46	58.86	94.00	75.95	58.45	94.85	92.46	81.10	84.92	75.95	80.84	87.06	92.46
c3540	49.50	89.54	54.09	48.78	91.22	88.36	62.03	84.67	54.09	61.35	87.15	88.36	83.06	61.59	54.09	82.64	67.70	88.36
c5315	48.08	92.69	93.20	47.73	93.35	94.30	60.68	89.31	93.20	60.35	90.28	94.30	82.24	73.10	93.20	82.03	75.54	94.30
c6288	49.63	89.24	38.02	49.38	89.84	76.35	62.15	84.23	38.02	61.92	85.11	76.35	83.13	60.49	38.02	82.99	62.66	76.35
c7552	48.90	90.95	84.92	48.66	91.48	94.40	61.46	86.76	84.92	61.24	87.53	94.40	82.71	66.74	84.92	82.58	68.65	94.40
b04	46.30	95.73	84.32	45.25	97.10	93.19	58.96	93.78	84.32	57.94	95.79	93.19	81.17	84.35	84.32	80.52	89.43	93.19
b05	48.08	92.69	88.65	47.73	93.35	89.09	60.68	89.31	88.65	60.35	90.28	89.09	82.24	73.10	88.65	82.03	75.54	89.09
b07	45.66	96.59	81.91	44.05	98.32	91.23	58.34	95.05	81.91	56.75	97.56	91.23	80.78	87.56	81.91	79.74	93.90	91.23
b11	47.73	93.35	74.50	45.66	96.59	87.75	60.35	90.28	74.50	58.34	95.05	87.75	82.03	75.54	74.50	80.78	87.56	87.75
b12	46.51	95.41	95.45	46.19	95.88	96.27	59.17	93.31	95.45	58.86	94.00	96.27	81.30	83.17	95.45	81.10	84.92	96.27
b13	44.54	97.86	96.94	44.35	98.05	97.36	57.24	96.90	96.94	57.05	97.18	97.36	80.06	92.23	96.94	79.94	92.93	97.36

TABLE IV. BASIC, DOUBLE AND TRIPLE TMR COMPARISONS

Logic core	M = 70%					
	Basic TMR		*Double TMR*		*Triple TMR*	
	$1/A_O$ (%)	R (%)	$1/A_O$ (%)	R (%)	$1/A_O$ (%)	R (%)
c1908	61.58	56.42	60.46	89.49	59.92	96.50
c2670	58.86	75.95	58.45	92.46	57.94	93.90
c3540	62.03	54.09	61.35	88.36	60.28	93.78
c5315	60.68	93.20	60.35	94.30	60.24	96.38
c6288	62.15	38.02	61.92	76.35	61.69	84.14
c7552	61.46	84.92	61.24	94.40	61.01	96.22
b04	58.96	84.32	57.94	93.19	57.44	95.53
b05	60.68	88.65	60.35	89.09	59.81	93.30
b07	58.34	81.91	56.75	91.23	56.47	94.11
b11	60.35	74.50	58.34	87.75	57.44	93.91
b12	59.17	95.45	58.86	96.27	58.24	97.79
b13	57.24	96.94	57.05	97.36	56.75	97.83

C. SoC yield improvement

To illustrate and prove the interest of using TMR architectures, we have considered, as case study, two benchmark circuits (c5315 with M = 70% and b05 with M = 90%) as the logic cores in a SoC. We have computed the yield improvement (Y_I) reachable if a fault-tolerant SoC (using TMR for logic cores) is manufactured instead of a classical SoC (only memory cores are fault-tolerant):

$$Y_I = 100 \times \frac{Y_{SoCTMR} - Y_{SoC} \times A_O}{Y_{SoC} \times A_O} \quad (17)$$

The graphs in Figures 6 and 7 show the yield improvement for the two SoC examples. In each graph, *Basic* and *Triple TMR* implementations are considered. Moreover, voters are considered to be either possibly faulty or fault-free.

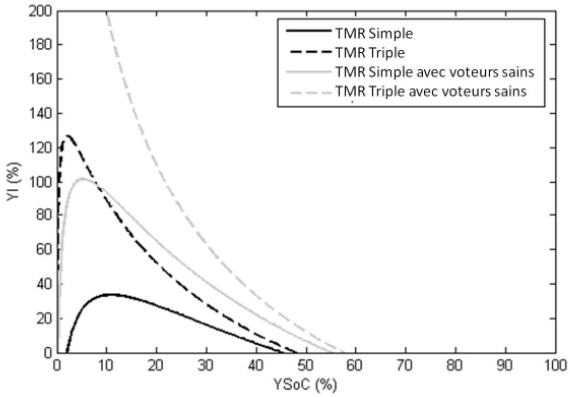


Figure 6. Yield improvement for the c5315 with M = 70%

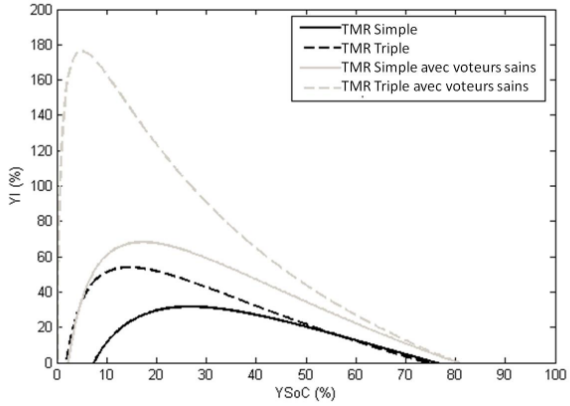


Figure 7. Yield improvement for the b05 with M = 90%

For example, let us consider the b05 benchmark circuit used as unique logic core. From Figure 7, if the initial yield Y_{SoC} = 30%, the yield improvement (Y_I) is about 31.28% for a Basic TMR implementation (59.17% when voters are fault-free) and about 42.69% for a Triple TMR implementation (90.49% when voters are fault-free). These results clearly show the interest of using TMR architectures for SoC yield improvement.

D. General SoC context

Until now, we have considered simple SoC examples composed by only one logic core. In the general SoC context, several logic cores are embedded. Consequently, the probability R (R_i) has to be computed for each logic core i translated into a TMR architecture. Then, we have to compute the value R (R_{eq}) representing the fault tolerance of the whole logic part by applying the following equation:

$$R_{eq} = 1 - \frac{\sum_{i=1}^{lc} \left[(1 - R_i) \times C_{3 \times n_i}^2 \right]}{C_{\sum_{j=1}^{lc} n_j}^2} \quad (19)$$

where lc is the number of logic cores and n_i is the number of single stuck-at faults in the logic core i. We also have to compute the $Rmin$ value ($Rmin_{eq}$), which represents the minimum value of R_{eq} above which it is beneficial to use TMR architectures, by using Eq. 15.

For example, let us assume a SoC with M = 70% and two logic cores C_1 (c7552) and C_2 (b05). From data reported in Table 1, we extract that R_1 = 84.92% and R_2 = 88.65%. If considered as independent, these two cores have no interest to successfully resort to TMR, *i.e.*, $R < Rmin$ in Table 1. Now, if these cores are embedded in the same SoC, R_{eq} = 90.90% and $Rmin_{eq}$ = 87.45%. So, as $R_{eq} > Rmin_{eq}$ the TMR versions of C_1 and C_2 are suitable for SoC yield improvement.

It is also important to notice that C_1 and C_2 can be viewed as two partitions of a logic core C. Each partition is not enough fault tolerant ($R_i < Rmin_i$) but C tolerates enough fault pairs as $R_{eq} > Rmin_{eq}$. Consequently, this example demonstrates the interest of using logic core partitioning to improve the fault tolerance of the TMR.

VII. CONCLUSION

In this paper analyzes the use of TMR architectures for logic cores to improve the overall SoC yield. We have computed the necessary conditions that make TMR architectures more attractive compared to non-tolerant logic cores. These conditions are related to *i)* the initial SoC yield (Y_{SoC}), *ii)* the memory ratio (M), *iii)* the area overhead needed to implement the TMR (A_O) and *iv)* the tolerance of the TMR implementation (R). A dedicated ATPG targeting TMR architectures has been used to evaluate the R parameter for different values of A_O and M. We have also analyzed the impact of the voter yield and the interest of partitioning logic cores to increase the fault tolerance of the TMR architecture. Results have shown that using TMR architectures for logic cores can be very fruitful to improve the overall SoC yield.

REFERENCES

[1] Semiconductor Industry Association (SIA), "International Technology Roadmap for Semiconductors (ITRS)", http://itrs.net, 2007.

[2] S. Shoukourian et Al., "SoC Yield Optimization via an Embedded-Memory Test and Repair Infrastructure", Proc of IEEE Design & Test of Computer, pp. 200-207, 2004.

[3] C. E. Stroud and A. E. Barbour, "Design for Testability and Test Generation for Static Redundancy System Level Fault Tolerant Circuits", Proc. of IEEE International Test Conference, pp. 812-818, 1989.

[4] I. Koren and C. Krishna, "Fault Tolerant Systems", Morgan Kauffman Publisher, 2007.

[5] T. Tsai, "Fault tolerance via N-modular software redundancy", Proc. of IEEE International Symposium on Fault Tolerant Computing, pp. 201 1998.

[6] S. Laha and J. H. Patel, "Error correction in arithmetic operations using time redundancy", Proc. of IEEE Fault Tolerant Computing Symposium, pp. 298-305, 1983.

[7] J. Vial, A. Bosio, P. Girard, C. Landrault, S. Pravossoudovitch and A. Virazel, "Yield Improvement, Fault-Tolerance to the Rescue?", Proc. of IEEE International On-Line Testing Symposium, pp. 165-166, 2008.

[8] J. Vial, A. Bosio, P. Girard, C. Landrault, S. Pravossoudovitch and A. Virazel, "Using TMR Architectures for Yield Improvement", Proc. of IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp. 7-15, 2008.

[9] J. Vial, A. Virazel, A. Bosio, P. Girard, C. Landrault and S. Pravossoudovitch, "Using TMR Architectures for SoC yield Improvement", International Conference on Advances in System Testing and Validation Lifecycle, pp 155-160, 2009.

[10] J. Vial, A. Virazel, A. Bosio, P. Girard, C. Landrault and S. Pravossoudovitch, "Is TMR Suitable for Yield Improvement?", IET Computers and Digital Techniques, Vol. 3, No 6, November 2009, pp. 581-592.

[11] P. K. Lala, "Self-Checking and Fault-Tolerant Digital Design", Morgan Kauffman Publisher, 2000.

[12] P. K. Lala, "Fault Tolerant and Fault Testable Hardware Design", Prentice-Hall International, 1985.

[13] R. E. Lyons and W. Vanderkulk, "The Use of Triple-Modular Redundancy to Improve Computer Reliability", IBM Journal of Research and Development, Vol. 6, No 2, April 1962, pp.200-209.

[14] J. M. Cazeaux, D. Rossi and C. Metra, "New High Speed CMOS Self-Checking Voter", Proc. of IEEE International On-Line Testing Symposium, pp. 58-63, 2004.

[15] I. Koren, Z. Koren and C. H. Stapper, "A Unified Negative-Binomial Distribution for Yield Analysis of Defect-Tolerant Circuits", IEEE Transaction of Computers, Vol. 42, No 6, 1993, pp. 724-734.

[16] P. de Gyvez, "Integrated Circuit Manufacturability", John Wiley & Sons Inc, 1999.

[17] L. Fang and M. S. Hsiao, "Bilateral Testing of Nano-scale Fault Tolerant Circuits", Proc. of IEEE Defect and Fault Tolerance in VLSI Systems, pp. 309-317, 2006.

[18] G. Karypis, R. Aggarwal, V. Kumar and S. Shekhar, "Multilevel Hypergraph Partitioning: Applications in VLSI Domain", Technical Report, Department of Computer Science, University of Minnesota, 1998.

Numerical studies of the metamodel fitting and validation processes

Bertrand Iooss
Electricité de France - EDF R&D
Chatou, France
bertrand.iooss@edf.fr

Loïc Boussouf
Altran
Toulouse, France
loic.boussouf@gmail.com

Vincent Feuillard
EADS IW
Suresnes, France
vincent.feuillard@eads.net

Amandine Marrel
Institut Français du Pétrole
Rueil-malmaison, France
amandine.marrel@ifp.fr

Abstract

Complex computer codes, for instance simulating physical phenomena, are often too time expensive to be directly used to perform uncertainty, sensitivity, optimization and robustness analyses. A widely accepted method to circumvent this problem consists in replacing cpu time expensive computer models by cpu inexpensive mathematical functions, called metamodels. In this paper, we focus on the Gaussian process metamodel and two essential steps of its definition phase. First, the initial design of the computer code input variables (which allows to fit the metamodel) has to provide adequate space filling properties. We adopt a numerical approach to compare the performance of different types of space filling designs, in the class of the optimal Latin hypercube samples, in terms of the predictivity of the subsequent fitted metamodel. We conclude that such samples with minimal wrap-around discrepancy are particularly well-suited for the Gaussian process metamodel fitting. Second, the metamodel validation process consists in evaluating the metamodel predictivity with respect to the initial computer code. We propose and test an algorithm, which optimizes the distance between the validation points and the metamodel learning points in order to estimate the true metamodel predictivity with a minimum number of validation points. Comparisons with classical validation algorithms and application to a nuclear safety computer code show the relevance of this new sequential validation design.

Keywords - Metamodel, Gaussian process, discrepancy, optimal design, Latin hypercube sampling, computer experiment.

1. Introduction

With the advent of computing technology and numerical methods, investigation of computer code experiments remains an important challenge. Complex computer models calculate several output values (scalars or functions), which can depend on a high number of input parameters and physical variables. These computer models are used to make simulations as well as predictions, uncertainty analyses or sensitivity studies [3].

However, complex computer codes are often too time expensive to be directly used to conduct uncertainty propagation studies or global sensitivity analysis based on Monte Carlo methods. To avoid the problem of huge calculation time, it can be useful to replace the complex computer code by a mathematical approximation, called a metamodel [29], [15]. Several metamodels are classically used: polynomials, splines, generalized linear models, or learning statistical models like neural networks, regression trees, support vector machines [5]. One particular class of metamodels, the Gaussian process (Gp) model, extends the kriging principles of geostatistics to computer experiments by considering the correlation between two responses of a computer code depending on the distance between input variables [29]. Numerous studies have shown that this interpolating model provides a powerful statistical framework to compute an efficient predictor of code response [30], [19].

From a practical standpoint, fitting a Gp model implies estimation of several hyperparameters involved in the covariance function. This optimization problem is particularly difficult in the case of a large number of inputs [5], [19]. Several authors (for example [31] and [5]) have shown that the space filling designs are well suited to metamodel fitting. However, this class of design, which aims at obtaining the better coverage of the points in the space of the input variables, is particularly large, ranging from the well known Latin Hypercube Samples to low discrepancy sequences [5]. At the moment, no theoretical result gives the type of initial design, which leads to the best fitted Gp metamodel in terms of metamodel predictivity. In this work, we propose to give some numerical results in order to answer to this fundamental question.

Another important issue we propose to address concerns the optimal choice of the test sample, i.e., the set of simulation design, which allows the most accurate metamodel validation using the minimal number of additional test observations. The validation of a metamodel is an essential step in practice [15]. By estimating the metamodel predictivity, we obtain a confidence degree associated with the use of the metamodel instead of the initial numerical model. Two validation methods are ordinarily used: the test sample approach [11] and the cross validation method [23], [27]. In this paper, we propose to perform numerical studies of the metamodel predictivity with respect to these validation methods.

In the following section, we present the Gp model. In the third section, we present several criteria to optimize the choice of the initial input design. On two analytical examples, we evaluate the numerical performance of this optimal design in terms of Gp metamodel predictivity. In the fourth section, we look at the metamodel validation problem. Our solution consists in minimizing the number of test observations by using the recent algorithm of [6], called the sequential validation design. We illustrate the relevance of this new design by performing intensive simulation on two analytical functions and an industrial example. Finally, a conclusion summarizes our results and gives some perspectives for this work.

2. Gaussian process metamodeling

Let us consider n realizations of a computer code. Each realization $y(\boldsymbol{x}) \in I\!R$ of the computer code output corresponds to a d-dimensional input vector $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathcal{X}$, where $\mathcal{X}$ is a bounded domain of $I\!R^d$. The n points corresponding to the code runs are called the experimental design and are denoted as $\boldsymbol{X}_s = (\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)})$. The outputs will be denoted as $Y_s = (y^{(1)}, \ldots, y^{(n)})$ with $y^{(i)} = y(\boldsymbol{x}^{(i)}) \; \forall \; i = 1..n$. Gaussian process (Gp) modeling treats the deterministic response $y(\boldsymbol{x})$ as a realization of a random function $Y(\boldsymbol{x})$, including a regression part and a centered stochastic process. This model can be written as:

$$Y(\boldsymbol{x}) = f(\boldsymbol{x}) + Z(\boldsymbol{x}). \tag{1}$$

The deterministic function $f(\boldsymbol{x})$ provides the mean approximation of the computer code. In our study, we use a one-degree polynomial model where $f(\boldsymbol{x})$ can be written as follows:

$$f(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^{d} \beta_j x_j \ ,$$

where $\boldsymbol{\beta} = [\beta_0, \ldots, \beta_k]^t$ is the regression parameter vector. It has been shown, for example in [21] and [19], that such a function is sufficient, and sometimes necessary, to capture the global trend of the computer code.

The stochastic part $Z(\boldsymbol{x})$ is a Gaussian centered process fully characterized by its covariance function: $\mathrm{Cov}(Z(\boldsymbol{x}), Z(\boldsymbol{u})) = \sigma^2 R(\boldsymbol{x}, \boldsymbol{u})$, where σ^2 denotes the variance of Z and R is the correlation function that provides interpolation and spatial correlation properties. To simplify, a stationary process $Z(\boldsymbol{x})$ is considered, which means that correlation between $Z(\boldsymbol{x})$ and $Z(\boldsymbol{u})$ is a function of the distance between $\boldsymbol{x}$ and $\boldsymbol{u}$. Our study is focused on a particular family of correlation functions that can be written as a product of one-dimensional correlation functions R_l:

$$\mathrm{Cov}(Z(\boldsymbol{x}), Z(\boldsymbol{u})) = \sigma^2 R(\boldsymbol{x} - \boldsymbol{u}) = \sigma^2 \prod_{l=1}^{d} R_l(x_l - u_l).$$

This form of correlation functions is particularly well adapted to get some simplifications of integrals in analytical uncertainty and sensitivity analyses [20]. More precisely, we choose to use the generalized exponential correlation function:

$$R_{\boldsymbol{\theta},\boldsymbol{p}}(\boldsymbol{x} - \boldsymbol{u}) = \prod_{l=1}^{d} \exp(-\theta_l |x_l - u_l|^{p_l}),$$

where $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_d]^t$ and $\boldsymbol{p} = [p_1, \ldots, p_d]^t$ are the correlation parameters (also called hyperparameters) with $\theta_l \geq 0$ and $0 < p_l \leq 2 \;\; \forall \; l = 1..d$. This choice is motivated by the wide spectrum of shapes that such a function offers.

If a new point $\boldsymbol{x}^* = (x_1^*, \ldots, x_d^*) \in \mathcal{X}$ is considered, we obtain the predictor and variance formulas:

$$I\!E[Y_{\mathrm{Gp}}(\boldsymbol{x}^*)] = f(\boldsymbol{x}^*) + \boldsymbol{k}(\boldsymbol{x}^*)^t \boldsymbol{\Sigma}_s^{-1} (Y_s - f(\boldsymbol{X}_s)) \ , \tag{2}$$

$$\mathrm{Var}[Y_{\mathrm{Gp}}(\boldsymbol{x}^*)] = \sigma^2 - \boldsymbol{k}(\boldsymbol{x}^*)^t \boldsymbol{\Sigma}_s^{-1} \boldsymbol{k}(\boldsymbol{x}^*) \ , \tag{3}$$

with Y_{Gp} denoting $(Y | Y_s, \boldsymbol{X}_s, \boldsymbol{\beta}, \sigma, \boldsymbol{\theta}, \boldsymbol{p})$,

$$\begin{aligned} \boldsymbol{k}(\boldsymbol{x}^*) &= [\mathrm{Cov}(y^{(1)}, Y(\boldsymbol{x}^*)), \ldots, \mathrm{Cov}(y^{(n)}, Y(\boldsymbol{x}^*))]^t \\ &= \sigma^2 [R_{\boldsymbol{\theta},\boldsymbol{p}}(\boldsymbol{x}^{(1)}, \boldsymbol{x}^*), \ldots, R_{\boldsymbol{\theta},\boldsymbol{p}}(\boldsymbol{x}^{(n)}, \boldsymbol{x}^*))]^t \end{aligned}$$

and the covariance matrix

$$\boldsymbol{\Sigma}_s = \sigma^2 \left(R_{\boldsymbol{\theta},\boldsymbol{p}} \left(\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)} \right)_{i=1..n, j=1..n} \right) \ .$$

The conditional mean (Eq. (2)) is used as a predictor. The variance formula (Eq. (3) corresponds to the mean squared error (MSE) of this predictor and is also known as the kriging variance. This analytical formula for MSE gives a local indicator of the prediction accuracy. More generally, Gp model provides an analytical formula for the distribution of the output variable at any arbitrary new point. This distribution formula can be used for sensitivity and uncertainty analysis [20].

Regression and correlation parameters $\boldsymbol{\beta}$, σ, $\boldsymbol{\theta}$ and $\boldsymbol{p}$ are ordinarily estimated by maximizing likelihood functions [5]. This optimization problem can be badly conditioned and difficult to solve in high dimensional cases ($d > 5$) [19]. Moreover, the estimation algorithms are particularly sensitive to the input design. The following section proposes to deal with this input design problem.

3. Initial design for the metamodel fitting

For computer experiments, selecting an experimental design is a key issue in building an efficient and informative metamodel. In this section, we describe the different properties than a computer experimental design has to reach. Some numerical tests support our discussion.

3.1. Latin hypercube sampling

Contrary to the Simple Random Sample (SRS, also called crude Monte Carlo sample), which consists of n independently and identically distributed samples, the well known Latin Hypercube Sample (LHS) consists in dividing the domain of each input variable in n equiprobable strata, and in sampling once from each stratum [22]. The LHS of a random vector $\boldsymbol{X} = (X_1, \ldots, X_d)$, denoted $(\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)})$, gives a sample mean $m = \frac{1}{n} \sum_{i=1}^{n} Y^{(i)}$ for the output $Y = y(\boldsymbol{X})$ with a smaller variance than the sample mean of a SRS [32]. Figure 1 shows 10 samples of two random variables, X_1 and X_2, obtained with SRS and LHS schemes. We can see that the result of LHS is more spread out and does not display the clustering effects found in SRS.

However, LHS does not reach the smallest possible variance for the sample mean. Since it is only a form of stratified random sampling and it is not directly related to any criterion, it may also perform poorly in metamodel estimation and prediction of the model output. Therefore, some authors have proposed to enhance LHS not only to fill space in one dimensional projection, but also in higher dimensions [25]. One powerful idea is to adopt some optimality criterion applied to LHS, such as entropy, integrated mean square

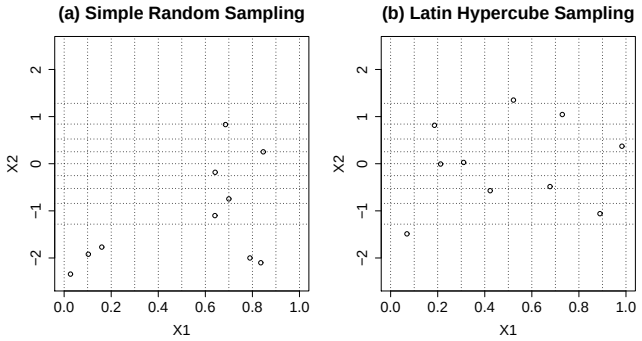


Fig. 1. Examples of two ways to generate a sample of size $n = 10$ from two variables $\mathbf{X} = [X_1, X_2]$ where X_1 has a uniform distribution $\mathcal{U}[0,1]$ and X_2 has a normal distribution $\mathcal{N}(0,1)$. Equprobable stratas are shown in each dimension.

error, minimax and maximin distances, etc. For instance, the maximin criterion consists in maximizing the minimal distance between the points [13]. This leads to avoid situations with too close points. The paper [24] examines some optimal maximin distance designs constructed within the class of Latin hypercube arrangements. The conceptual simplicity of these designs has led to their large popularity in practical applications [14].

3.2. Low-discrepancy Latin hypercube samples

Alternative metamodel-independent criteria, based on discrepancy measures, consist in judging the uniformity quality of the design. Discrepancy can be seen as a measure between an initial configuration and an uniform one. It is a comparison between the volume of intervals and the number of points within these intervals [8]. There exists different kinds of definition using different forms of intervals or different norms in the functional space. Discrepancy measures based on L_2 norms are the most popular in practice because they can be analytically expressed and are easy to compute. Among them, two measures have shown remarkable properties [12], [4], [5]:

- the centered L^2 discrepancy

$$D^2(\boldsymbol{X}_s(n)) = \left(\frac{13}{12}\right)^d - \frac{2}{n}\sum_{i=1}^{n}\prod_{k=1}^{d}\left(1+\frac{1}{2}|u_k^{(i)}-\frac{1}{2}|-\frac{1}{2}|u_k^{(i)}-\frac{1}{2}|^2\right) + \frac{1}{n^2}\sum_{i,j=1}^{n}\prod_{k=1}^{d}\left(1+\frac{1}{2}|u_k^{(i)}-\frac{1}{2}|+\frac{1}{2}|u_k^{(j)}-\frac{1}{2}|-\frac{1}{2}|u_k^{(i)}-u_k^{(j)}|\right) \quad (4)$$

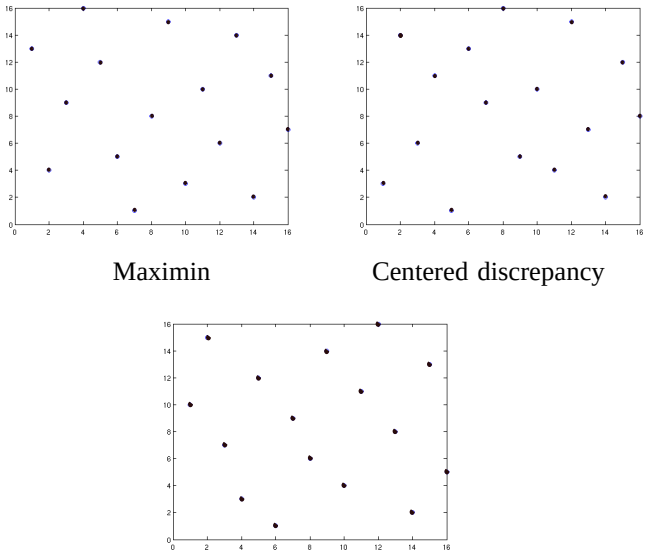
where $\boldsymbol{X}_s(n)$ denotes the input learning sample with n input vectors and $\left(u_k^{(i)}\right)_{i=1..n,k=1..d}$ are the normalized values in $[0,1]$ of the design $\boldsymbol{X}_s(n) = \left(x_k^{(i)}\right)_{i=1..n,k=1..d}$;

- the wrap-around L^2 discrepancy

$$W^2(\boldsymbol{X}_s(n)) = \left(\frac{4}{3}\right)^d + \frac{1}{n^2}\sum_{i,j=1}^{n}\prod_{k=1}^{d}\left[\frac{3}{2}-|u_k^{(i)}-u_k^{(j)}|(1-|u_k^{(i)}-u_k^{(j)}|)\right], \quad (5)$$

which allows to suppress bound effects (by wrapping the unit cube for each coordinate).

The optimization of LHS can be done following different methods: choice of the best (in terms of the chosen criteria) LHS amongst a large number of different LHS, columnwise-pairwise exchange algorithms, genetic algorithms, simulated annealing, etc [12], [17]. In our tests, we have found that the simulated annealing algorithm (with a geometrical temperature descent and with a slight noise on the initial condition) gives the best results for all the criteria [18]. Figure 2 gives some examples of two-dimensional LHS of size $n = 16$, optimized following three different criteria with the simulated annealing algorithm. We see that uniform repartitions of the points are nicely respected.



Maximin | Centered discrepancy

Wrap-around discrepancy

Fig. 2. Visual comparisons of LHS ($d = 2$, $n = 16$) optimized following three different criteria (below each figure).

3.3. Projection properties of space filling designs

In addition to the space filling property on the sample space, one important property of the initial designs is their robustness to the dimension decrease. A LHS structure for the space filling design is not sufficient because it only guarantees good repartitions for one-dimensional projections, and not for the other dimensions of projection. Indeed, LHS ensures that each of the input variables has all proportion of its range which is represented (equiprobable stratas are created for each input

variable). In contrary, no equiprobable stratas are created in the various multi-dimensional spaces of the input variables.

We then argue that the sample points of a space filling design have to be well spread out when projected onto a subspace spanned by a subset of coordinate axes. This property is particularly important when the initial design is made in dimension d and the metamodel fitting is made in a smaller dimension (see an example in [1]). In practice, this is often the case because the initial design may reveal with screening methods the useless (i.e., non influent) input variables that we can neglect during the metamodel fitting step [26]. Moreover, when a selection of input variables is made during the metamodel fitting step (as for example in [19]), the new sample, solely including the retained input variables, has to keep good space filling properties.

Figure 3 compares the two-dimensional projections of the maximin LHS and low wrap-around discrepancy LHS (called WLHS) with $n = 100$ points and different initial dimensions (from $d = 3$ to 15). The reference criterion values are given for $d = 2$. For dimension larger than 2, we compute the new criterion values by considering all the two-dimensional projections of the initial design. A robust criterion to the dimension decrease would lead to a small increase of the criterion value. The criteria behave very diferently between the two types of design:

- 2D projection criteria of WLHS regularly and slightly deteriorate. Then, 2D projections of WLHS made in dimensions close to 2 keep rather good space-filling properties.
- 2D projection criteria of maximin LHS sharply and strongly deteriorate from the first dimension increase at $d = 3$. Then 2D projection criteria of maximin LHS remain stable at poor values for larger dimensions.

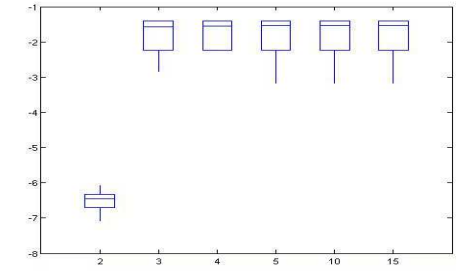
Similar tests with different sample sizes n and for the three-dimensional and four-dimensional projections have led to the same conclusions. All these results show that a WLHS is the preferable initial design for fitting a computer code metamodel in high dimensional cases.

3.4. Numerical studies on toy functions

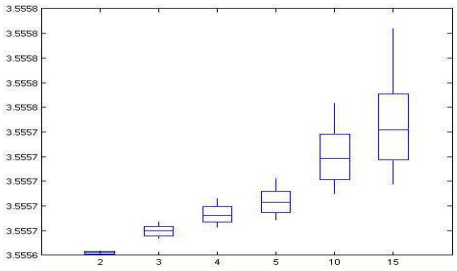
At present, we perform two numerical studies to evaluate the impact of an inadequate design on the metamodel fitting process. For the metamodel, we use the Gp model Y_{Gp} described in §2. The quality of the metamodel predictor is measured by the so-called predictivity coefficient Q_2 (i.e., the determination coefficient R^2 computed on a test sample), which gives the percentage of the output variance explained by the metamodel:

$$Q_2 = 1 - \frac{\sum_{i=1}^{n_t}[y(\tilde{\boldsymbol{x}}^{(i)}) - \hat{Y}_{\text{Gp}}(\tilde{\boldsymbol{x}}^{(i)})]^2}{\sum_{i=1}^{n_t}[\bar{y} - y(\tilde{\boldsymbol{x}}^{(i)})]^2} \qquad (6)$$

with $(\tilde{\boldsymbol{x}}^{(1)}, \ldots, \tilde{\boldsymbol{x}}^{(n_t)})$ the test sample of size n_t, $\hat{Y}_{\text{Gp}} = I\!E(Y_{\text{Gp}})$ the Gp predictor (Eq. (2)) and $\bar{y}$ the mean of the output test sample $(y(\tilde{\boldsymbol{x}}^{(1)}), \ldots, y(\tilde{\boldsymbol{x}}^{(1)}))$.



Maximin LHS



Low wrap-around discrepancy LHS (WLHS)

Fig. 3. Criterion values (up: maximin, bottom: wrap-around discrepancy) obtained with 2D projections of designs coming from two types of LHS (containing $n = 100$ points), with different dimensions: $d = 2, 3, 4, 5, 10, 15$. Boxplots are obtained by repeating 100 optimizations using different initial LHS.

3.4.1. A two-dimensional test case. Our first test involves a two-dimensional analytical function (called the irregular function):

$$f(\boldsymbol{x}) = \frac{e^{x_1}}{5} - \frac{x_2}{5} + \frac{x_2^6}{3} + 4x_2^4 - 4x_2^2 + \frac{7x_1^2}{10} + x_1^4 + \frac{3}{4x_1^2 + 4x_2^2 + 1}$$

with $\boldsymbol{x} \in [-1, 1]^2$. Figure 4 represents the irregular function.

We have made several comparisons between random LHS and different space filling designs before fitting a metamodel [18]. In the following, we show our results concerning the random LHS and the WLHS, which has provided the best results. For a size n of the learning sample and each type of design, we repeat 100 times the following procedure: we generate an initial input design of n observations, we obtain n outputs with the toy function, we fit a Gp metamodel (1), and we evaluate its predictivity coefficient Q_2 using a test sample of large size ($n_t = 10000$). Therefore, for each type of LHS, we obtain 100 values of Q_2 whose mean and variance give us the efficiency and robustness of the design in terms of Gp quality.

The initial LHS design optimized with the wrap-around discrepancy (Eq. (5)) has given us the best results. In Figure 5, we compare the predictivity coefficients obtained with non optimized LHS (random LHS) and those obtained with optimized LHS (WLHS). The size of the design increases from $n = 10$ to $n = 46$ (by step of 4), which leads to a

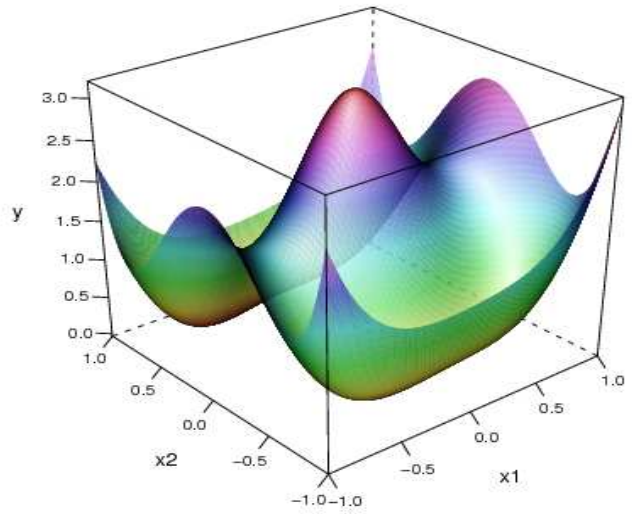


Fig. 4. Graphical representation of the irregular function on $[-1;1]^2$.

regular increase of Q_2. For each size n, the boxplot represents the summary of the 100 values of Q_2. In the all range of n, Q_2 of the WLHS are better than the random LHS ones. Furthermore, much smaller variances (boxplots are smaller) are shown for WLHS and lead to the conclusion that these designs are more robust than others. This property is rather natural because there are much less variability between the 100 different WLHS than between the 100 different random LHS (because of the optimization process). Differences are particularly important for sizes $n = 30$ and $n = 34$: the WHS lead to very competitive Gp metamodels ($Q_2 \sim 0.95$ and boxplot width ~ 0.05) while random LHS give uncompleted metamodels ($Q_2 \sim 0.9$ and boxplot width ~ 0.2).

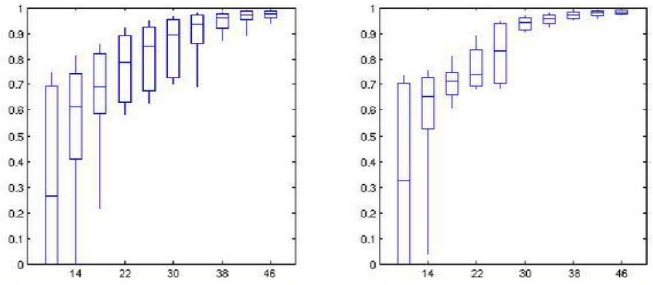


Fig. 5. For the irregular function, Gp Q_2 evolution in function of the learning sample size n and for two types of LHS (left: random LHS; right: WLHS).

3.4.2. A five-dimensional test case. Our second test involves a five-dimensional analytical function (called the g-Sobol 5d function):

$$f(\boldsymbol{x}) = \sum_{i=1}^{5} \frac{|4x_i - 2| + a_i}{1 + a_i}$$

with $a_1 = 1$, $a_2 = 2$, $a_3 = 3$, $a_4 = 4$, $a_5 = 5$, $\boldsymbol{x} \in [0,1]^5$.

We have made several comparisons between random LHS and different space filling designs before fitting a metamodel [18]. In the following, we show our results concerning the random LHS and the WLHS, which has provided the best results. For a size n of the learning sample and each type of design, we repeat 100 times the following procedure: we generate an initial input design of n observations, we obtain n outputs with the toy function, we fit a Gp metamodel (1), and we evaluate its predictivity coefficient Q_2 using a test sample of large size ($n_t = 10000$). Therefore, for each type of LHS, we obtain 100 values of Q_2 whose mean and variance give us the efficiency and robustness of the design in terms of Gp quality.

As in the previous section, the initial LHS design optimized with the wrap-around discrepancy (Eq. (5)) has given us the best results. In Figure 6, we compare the predictivity coefficients obtained with non optimized LHS (random LHS) and those obtained with optimized LHS (WLHS). The size of the design increases from $n = 22$ to $n = 40$ (by step of 2), which leads to a regular increase of Q_2. For each size n, the boxplot represents the summary of the 100 values of Q_2. In the all range of n, Q_2 of the WLHS are better than the random LHS ones. Furthermore, much smaller variances are shown for WLHS and lead to the conclusion that these designs are more robust than others. For small sample sizes, the Q_2 differences reach 0.2 between the two types of design: $Q_2(\text{LHS}) \sim 0.6$ and $Q_2(\text{WLHS}) \sim 0.8$. In industrial applications, such a difference makes the distinction between "bad" (unacceptable) metamodels and good ones. The latter can be used for example for quantitative sensitivity studies.

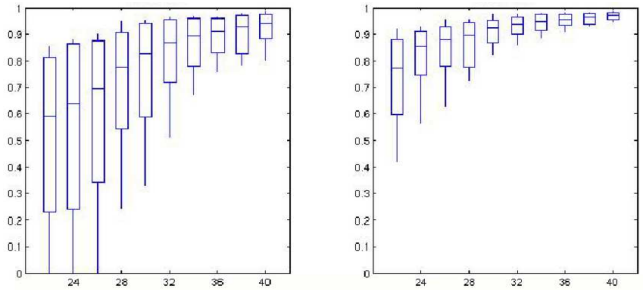


Fig. 6. For the g-Sobol 5d function, Gp Q_2 evolution in function of the learning sample size n and for two types of LHS (left: random LHS; right: WLHS).

3.5. Conclusion of numerical tests

In conclusion of our numerical study, the LHS optimized with the wrap-around discrepancy has provided efficient results for the Gp metamodel fitting, even in high dimension. Furthermore, we have found that this design guarantees correct repartitions of the points for all the two-dimensional projections, while other types of LHS (like maximin) have bad

repartitions for these projections. Other types of LHS can also provide good results but less systematically [18]. For instance, [7] has studied quasi-Monte Carlo samples (Sobol suites and Halton sequences) and has shown that these sequences are less performant than other space filling designs in terms of the Gp metamodel fitting.

Of course, such designs have to be seen as initial ones. If possible, in a second step, adaptive designs can improve metamodel predictivity in a very efficient way [18], for instance by choosing new simulation points in poorly predicted areas.

4. Test sample selection for metamodel validation

In practical cases, only a small number of simulations can be performed with the computer code in order to fit a metamodel. Once the metamodel has been built, estimating its predictivity is an important issue. Indeed, a safe use of this metamodel to answer to uncertainty or sensitivity problems requires a precise estimation of its capabilities. In this section, we make a discussion on algorithms of predictivity estimation.

4.1. Classical validation methods

Let us consider the d-dimensional input vector $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathcal{X}$, where $\mathcal{X}$ is a bounded domain of $\mathbb{R}^d$ and $y(\boldsymbol{x}) \in \mathbb{R}$ is the computer code output. We suppose that a metamodel $\hat{Y}(\boldsymbol{x})$ has been fitted using $\left((\boldsymbol{x}^{(1)}, y(\boldsymbol{x}^{(1)})), \ldots, (\boldsymbol{x}^{(N)}, y(\boldsymbol{x}^{(N)}))\right)$, a N-size learning sample of computer code experiments.

The test sample approach consists in comparing the metamodel predictions on simulation points not used in the metamodel fitting process. This gives some prediction residuals (which can be finely analyzed) and global quality measures as the metamodel predictivity coefficient Q_2 (Eq. (6)). Such test points set is called a test sample (or also validation sample or prediction sample). This method requires new calculations with the computer code and the first question we have to face up is the sufficient number of prediction points to obtain the required accuracy of our global validation measures. For cpu time expensive code, it can be difficult to provide a sufficient number of test points. Some convergence visualisation tools of the global validation measures can be used to answer to this first question.

Another important question for the test sample approach is the localization of these test points. The usual practice is to choose an independent Monte Carlo sample for the test sample. However, if the sample size is small, the proposed points can be badly localized, for example near learning points or leaving large space domain unsampled. A fine strategy could be to use, as the test sample, a space filling design (which consists in filling the input variable space $\mathcal{X}$ as uniformly as possible). Unfortunately, this solution does not avoid the possibility of too strong proximity between learning points and test points. Such proximity would lead to too optimistic quality measures, and consequently to a biased prectivity estimation.

The second solution to validate a metamodel, the cross validation method, is extremely popular in practice because it avoids new calculations on the computer code. The cross validation method proposes to divide the initial sample on a learning sample and a test sample. A metamodel is estimated with the points in the new learning sample and prediction residuals are obtained via the new test sample. This process is repeated several times by using other divisions of the learning sample. Finally all the prediction residuals can be used to compute the global predictivity measures. The leave-one-out procedure is a particular case of the cross validation method where just one observation is left out at each step.

The first drawback of the cross validation method is its cost, which can become large due to many metamodel fitting processes. Moreover, if the initial design has a specific geometric structure (which aims at optimizing the metamodel fitting), the deletion of points from the learning sample causes the breakdown of the specific design structure while creating the new learning sample. Indeed, the new learning sample does not have the adequate statistical and geometric properties of the initial design and the metamodel fitting process might fail. This could lead to too pessimistic quality measures.

To sum up, the test sample method requires too many new prediction points (to avoid too optimistic validation criteria), while the cross-validation method can provide too pessimistic validation criteria. Therefore, to solve this dilemma, an heuristic new solution has been introduced in [10], [9] and is presented in the next section.

4.2. A new optimized validation design

Retaining the test sample method, we limit its main drawback by minimizing the number of necessary points in the test sample. In this goal, an algorithm allows the specification of new design points decreasing the discrepancy of an initial design [6]. This sequential algorithm gives us at each step the prediction point furthest away from the other points of the design. The algorithm performs its optimization process in the space $\mathcal{X}$ of the input variables $\boldsymbol{x}$. By choosing the future prediction points in the unfilled zone of the learning sample design, we aim at capturing the right metamodel predictivity using only a small number of additional points. Note that such ideas have also been proposed in [28] for different purposes.

We have not theoretically studied the computational efficiency of this algorithm over the computational efficiency of the traditional methods (introduced in the previous section). However, our intuition is that mean square error computed by this algorithm avoids the biases, which could be caused by too strong proximities between the test sample points and between test sample points vs. learning sample points.

Let us consider $\boldsymbol{X}_f(n_f) = (\boldsymbol{x}_f^{(i)})_{i=1..n_f}$ a low discrepancy sequence of n_f points in $[0,1]^d$. A low discrepancy sequence is a deterministic design constructed to uniformly fill the space with regular patterns. Among all the low discrepancy sequence, Halton, Hammersley, Faure and Sobol sequences are the most famous [16]. In the following, we will use the

Hammersley sequence which, on a few tests, have shown better properties than the others [6]. The chosen discrepancy measure is the centered L^2 discrepancy $D^2(\cdot)$ (Eq. (4)).

To obtain an additional point of the initial N-size sample, noticed $\boldsymbol{X}_s(N)$, we use the following algorithm:

1) For $i = 1, \ldots, n_f$,
   - $\boldsymbol{X}_s(N+1) = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\} \cup \boldsymbol{x}_f{}^{(i)}$;
   - compute $\mathrm{Dif}_i = D^2(\boldsymbol{X}_s(N+1)) - D^2(\boldsymbol{X}_s(N))$;
2) select i^* such that $i^* = \underset{i=1,\ldots,n_f}{\arg\min}\ \mathrm{Dif}_i$;
3) obtain the new point $\boldsymbol{x}_f{}^{(i^*)}$.

This algorithm is repeated sequentially to obtain N_{test} test points, by updating the initial design and the low discrepancy sequence. For example, for the second point, we reinitalize the design by the following: $\boldsymbol{X}_s(N+1) = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\} \cup \boldsymbol{x}_f{}^{(i^*)}$ and $\boldsymbol{X}_f(n_f - 1) = \{\boldsymbol{x}_f{}^{(1)}, \ldots, \boldsymbol{x}_f{}^{(n_f)}\} \backslash \boldsymbol{x}_f{}^{(i^*)}$.

This algorithm just consists in adding to the initial design some points of a low discrepancy sequence by minimizing the discrepancy differences between the initial and the new design. The size of the low discrepancy sequence is required to be as large as possible, especially if d is large. Figure 7 gives an example of the specification with our algorithm of $N_{\text{test}} = 4$ new points (the crosses) inside an initial Monte Carlo design ($N = 46$, $d = 2$). One of the advantage of this algorithm is its size-independence (related to the number of added points): the sequence of added points is deterministic and will be always the same for the same $\boldsymbol{X}_f(n_f)$. In the following, the design obtained using this algorithm is called the sequential validation design.

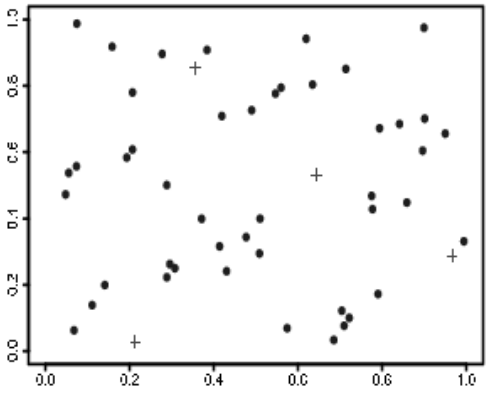


Fig. 7. Example of the sequential algorithm: $N = 46$, $d = 2$, $N_{\text{test}} = 4$. The bullets are the points of the initial design while the crosses are the new specified points.

4.3. Numerical studies on toy functions

4.3.1. A two-dimensional test case. To compare the sequential validation design with other test designs for the metamodel validation purpose, we first perform an analytical test using a two-dimensional toy function, called the cosin2 function:

$$f(\boldsymbol{x}) = \cos(10x_1) + \sin(10x_2) + x_1 x_2\ ,\ (x_1, x_2) \in [0,1]^2.$$

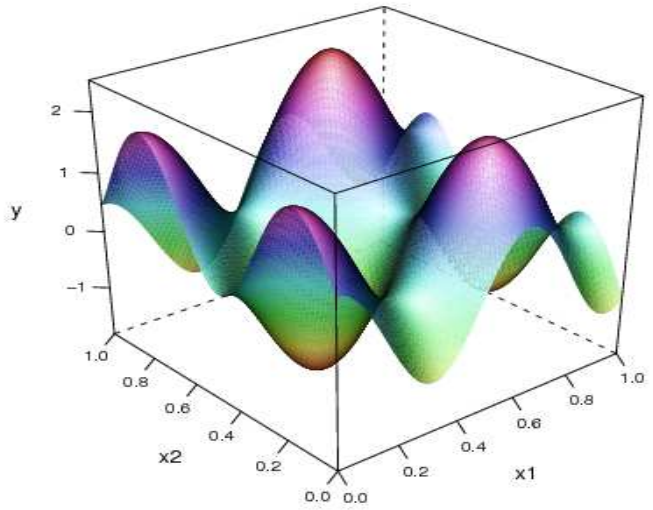


Fig. 8. Graphical representation of the cosin2 function on $[0;1]^2$.

Figure 8 represents the cosin2 function.

Gp metamodels (1) are fitted using learning samples of differents sizes N_{BA}: N_{BA} ranges from 10 to 40 allowing a wide variety of metamodel predictivity coefficients Q_2, from 0 (null predictivity) to 1 (perfect predictivity). The initial 10-size design is a maximin LHS. The other learning designs (of increased size) are obtained by sequentially adding points to the design, while maintaining the LHS properties of the design and keeping some optimality properties (maximizing the mean distance from each design point to all the other points in the design [17]). Choosing an initial maximin LHS design, while we have shown in Section 3 that WLHS is better than maximin LHS for the Gp fitting process, is not in contradiction with our objectives in this section: our goal is now to study the Gp metamodel validation. Anyway, we are not able to keep the properties of maximin LHS or WLHS when gradually increasing the size of the learning sample.

The black line in Figure 9 shows the evolution of Q_2 in function of the learning sample size. This reference value for the predictivity coefficient has been computed for each metamodel by taking its mean over 100 test samples of size $N_{\text{test}} = 1000$. The Q_2 estimation by a leave one out procedure (pink line) strongly underestimates the exact Q_2 for $N_{\text{BA}} < 30$. This is certainly due to the small number of points: leave one out is pessimistic in this case because each point deletion has a strong impact on the metamodel fitting process. The red curve gives the Q_2 estimation using the sequential validation design described in the previous paragraph (with a Hammersley sequence of size $n_f = 10000$).

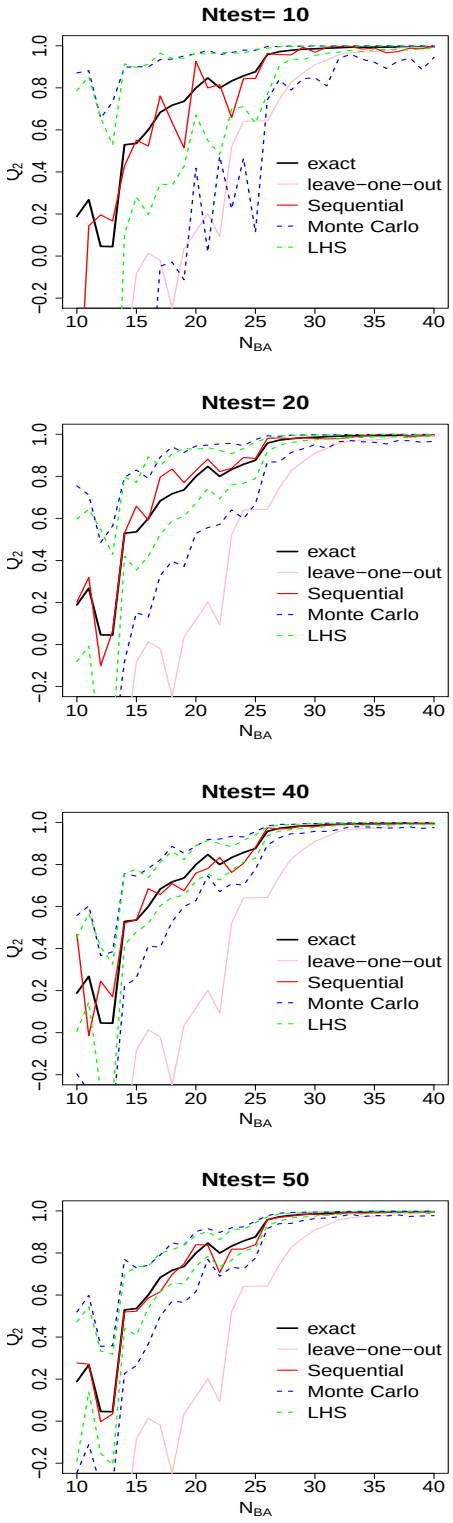


Fig. 9. For the cosin2 function, Gp predictivity coefficient (Q_2) in function of the learning sample size N_{BA}, estimated from different test sample sizes N_{test}. The dashed curves (blue and green) give the minimal and maximal values obtained with 100 repetitions of the random test design (Monte Carlo and LHS).

Results are greatly satisfactory for $N_{\text{test}} \geq 20$: the sequential validation design gives precise Q_2 estimates in all cases and outperforms a crude Monte Carlo or LHS design. The green curves correspond to the minimal and maximal values obtained with 100 repetitions using an optimized LHS as the test design. As expected, these intervals are more reduced than the intervals obtained using a crude Monte Carlo sample as the test design (blue curves). As N_{test} increases, these intervals contract, but always show the superiority of the sequential validation design, especially for low metamodel predictivity ($Q_2 < 0.9$ and $N_{\text{BA}} < 25$).

4.3.2. An eight-dimensional test case. We perform now a second numerical test using the g-Sobol function in eight-dimension (called the g-Sobol 8d function):

$$f(\boldsymbol{x}) = \sum_{i=1}^{8} \frac{|4x_i - 2| + a_i}{1 + a_i}$$

with $a_1 = a_2 = 3$, $a_i = 0$ for $(i = 3, \ldots, 8)$, $\boldsymbol{x} \in [0,1]^8$.

A Gp metamodel (1) is fitted on a learning sample (maximin LHS) of size $N_{\text{BA}} = 40$. We compute the reference value of the predictivity coefficient by taking its mean over 100 test samples of size $N_{\text{test}} = 1000$ and obtain $Q_2^{\text{ref}} = 0.83$. We then apply the sequential validation design described previously (with a Hammersley sequence of size $n_f = 10000$) by adding $N_{\text{test}} = 50$ new points to the design, and we obtain $Q_2^{\text{seq50}} = 0.85$, which is close to the true value. We compare this result with 100 crude Monte Carlo samples of the same size ($N_{\text{test}} = 50$), which give the 90% confidence interval $[0.79, 0.91]$ for Q_2^{MC}. This last result is rather large and shows the insufficient number of points if we choose a crude Monte Carlo design.

Figure 10 shows the evolution of the estimated Q_2 for test bases with different sizes, ranging from $N_{\text{test}} = 10$ to $N_{\text{test}} = 50$. The solid red line shows the results obtained with the sequential validation design while the dotted blue lines show the 100 sequentially increased crude Monte Carlo samples. This figure illustrates the poor estimates we obtain when using small size ($N_{\text{test}} < 50$) of Monte Carlo samples for validation. On the contrary, the sequential validation design allows to obtain a good approximation of the true predictivity coefficient even for small test sample sizes. Results are precise for $N_{\text{test}} \geq 25$.

4.4. Application to a nuclear safety computer code

In this section we apply our algorithms on a complex computer model used for nuclear reactor safety. It simulates a hypothetical thermal-hydraulic scenario on Pressurized Water Reactors: a large-break loss of primary coolant accident (see Fig. 11) for which the output of interest is the peak cladding temperature. This scenario is part of the Benchmark for Uncertainty Analysis in Best-Estimate Modelling for Design, Operation and Safety Analysis of Light Water Reactors [2] proposed by the Nuclear Energy Agency of the Organisation for Economic Co-operation and Development (OCDE/NEA). It has

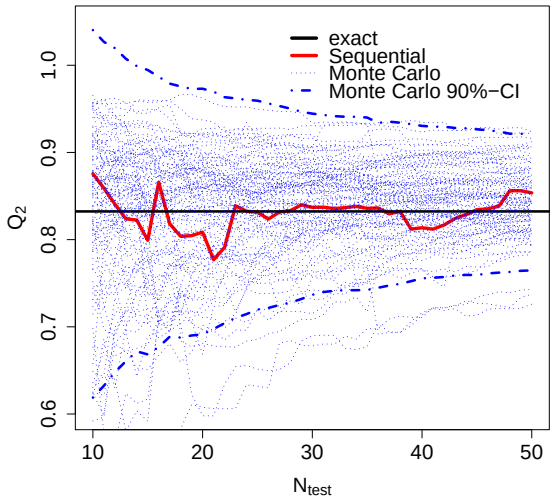


Fig. 10. For the g-Sobol 8d function, Gp predictivity coefficient (Q_2) in function of the test sample size N_{test}, for two types of validation design: sequential (red) and crude Monte Carlo (blue). Dotted blue lines correspond to 100 different crude Monte Carlo samples).

been implemented on the french computer code CATHARE2 developed at the Commissariat à l'Energie Atomique (CEA). Figure 12 illustrates 100 CATHARE2 simulations (by varying input variables of the accidental scenario) giving the cladding temperature in function of time.

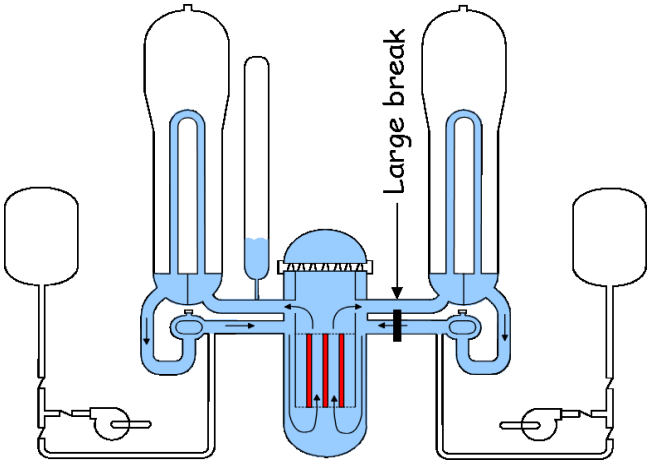


Fig. 11. Illustration of a large-break loss of primary coolant accident on a nuclear Pressurized Water Reactor.

In our exercise, a Gp metamodel (1) of the first peak cladding temperature (which is a scalar variable) has to be estimated with $N = 100$ simulations of the computer model (the input design is a maximin LHS). The cpu time is twenty minutes for each simulation with a standard computer (Pentium IV PC). The complexity of the computer model lies in the high-dimensional input space. $d = 53$ random

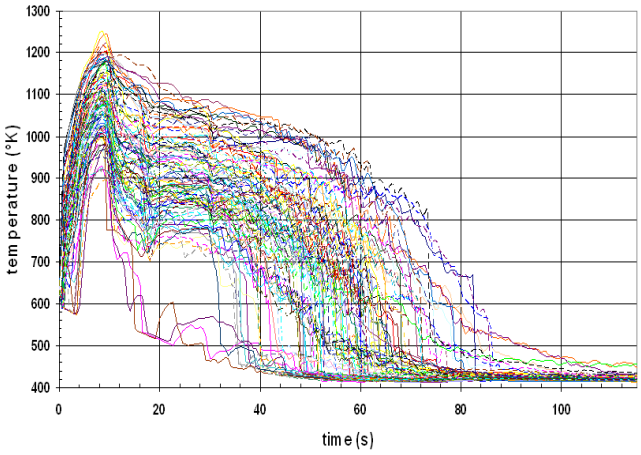


Fig. 12. 100 output curves (cladding temperatures in function of time) of the CATHARE2 code. The output variable of interest for the reactor safety is the first peak of the cladding temperature.

input variables are considered: physical laws essentially, but also initial conditions, material properties and geometrical modeling. Their probability distributions are either normal or log-normal, and both are truncated. Such a number of input variables is rather large for the metamodel fitting problem. This difficult fit (due to the high dimensionality and small learning sample size) can be made thanks to the algorithm of [19], specifically devoted to this situation. The obtained Gp metamodel (1) contains a linear regression part (including 7 input variables) and a centered Gp model with a generalized exponential covariance function (including 6 input variables). The reference quality of this Gp model is measured via an additional 1000-size test sample, which gives $Q_2^{\text{ref}} = 0.66$.

Figure 13 shows the evolution of the estimated Q_2 for test bases with different sizes, ranging from $N_{\text{test}} = 10$ to $N_{\text{test}} = 95$. The sequential validation design gives coarse estimations for all the test design sizes and begins to give precise results for $N_{\text{test}} \geq 40$. Some inadequacies, which remain when $N_{\text{test}} \in [75, 90]$, have to be finely analyzed in a further work. In any cases, sequential validation design estimations are clearly less hazardous than using a crude Monte Carlo test sample to validate the metamodel: the 90%-confidence intervals obtained using Monte carlo samples show extremely large variation ranges (because of the high dimensionality of the input space: $d = 53$). Q_2 estimation using a Monte Carlo test sample can lead to a strongly erroneous result. Same results have been obtained using optimized LHS for the test design instead of a crude Monte Carlo sample.

5. Conclusion and future works

In this paper, we have proposed to look at two practical problems when fitting a metamodel to small-size data samples: the initial design and the validation method choices. These

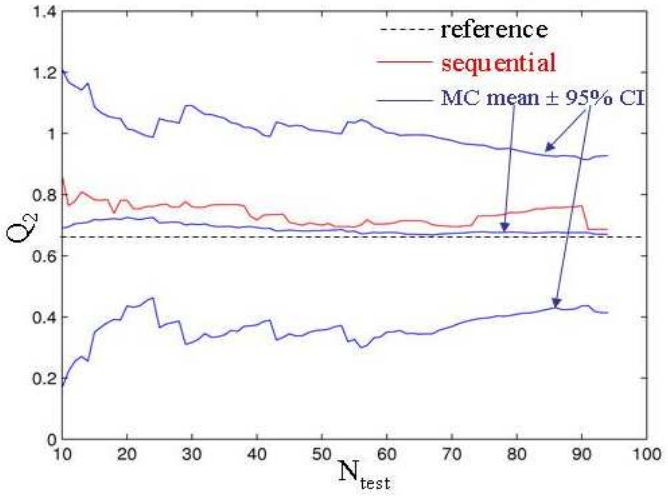


Fig. 13. For the nuclear safety computer code application, estimation of the metamodel (Gp) predictivity coefficient (Q_2) in function of the test sample size N_{test}, for two types of validation design: sequential (red) and crude Monte Carlo (blue).

problems are relevant when a cpu time expensive code has to be replaced by a simplified model with negligible cost (I;e., a metamodel). Such replacement is useful to resolve optimisation, uncertainty propagation or sensitivity analysis issues.

We have first paid attention to the initial input design. Our numerical tests concentrate on the popular Gp metamodel and on the LHS. This type of design, developed thirty years ago, is the most widely used in industrial applications. We have shown that an excellent way to optimize its properties, in the objective of the best metamodel fit, is to minimize some discrepancy measures, especially the wrap-around L^2 discrepancy. An alternative strategy, if possible, would be to use some adaptive designs [18]. For the Gp metamodel, this kind of design is well-adapted due to the availability of the variance expression (the MSE of the metamodel predictor, see Eq. (3)).

Secondly, we have looked at the metamodel validation process and have shown that the test sample approach can provide erroneous results for small sizes of the test sample. Moreover, the leave one out approach can strongly underestimate the metamodel predictivity for small sizes of the whole database. We have proposed to use a recent algorithm, called the sequential validation design, which puts prediction points in the unfilled zones of the learning sample design. Therefore, a minimal number of points is required to obtain a good estimation of the metamodel predictivity. Our numerical tests on analytical functions and real application cases have shown that the sequential validation design outperforms the classical metamodel validation methods, especially in high dimensional context. For our analytical functions, the sequential validation design gives precise estimate of the metamodel predictivity with a test sample size $N_{\text{test}} \geq 25$, while for our industrial application, the minimal bound is $N_{\text{test}} \geq 40$.

Further works are necessary to more deeply study the validation designs (other test functions with different effective dimensionality and complexity). Moreover, it would be useful to find a criterion to determine when the sequential validation design can be ended. Finally, the ultimate goal of such studies will be to define a global strategy of allocating simulation points between the metamodel fitting step and the metamodel validation step.

Acknowledgments

This work was supported by the Simulation program managed by CEA/DEN/DISN. We thank P. Bazin and A. de Crecy from the CEA Grenoble/DER/SSTH who have provided the CATHARE2 application.

References

[1] C. Cannamela, J. Garnier and B. Iooss. Controlled stratification for quantile estimation. *Annals of Applied Statistics 2*, 1554–1580, 2008.

[2] A. De Crécy, P. Bazin, H. Glaeser, T. Skorek, J. Joucla, P. Probst, K. Fujioka, B.D. Chung, D.Y. Oh, M. Kyncl, R. Pernica, J. Macek, R. Meca, R. Macian, F. DAuria, A. Petruzzi, L. Batet, M. Perez, and F. Reventos. Uncertainty and sensitivity analysis of the LOFT L2-5 test: Results of the BEMUSE programme. *Nuclear Engineering and Design 12*, 3561–3578, 2008.

[3] E. De Rocquigny, N. Devictor, and S. Tarantola, editors. *Uncertainty in industrial practice*. Wiley, 2008.

[4] K-T. Fang. Wrap-around L_2-discrepancy of random sampling, Latin hypercube and uniform designs. *Journal of Complexity 17*, 608–624, 2001.

[5] K-T. Fang, R. Li, and A. Sudjianto. *Design and modeling for computer experiments*. Chapman & Hall/CRC, 2006.

[6] V. Feuillard. *Analyse d'une base de données pour la calibration d'un code de calcul*. Thèse de l'Université Pierre et Marie Curie - Paris VI, 2007.

[7] J. Franco. *Planification dexpériences numériques en phase exploratoire pour la simulation des phénomènes complexes*. Thèse de l'Ecole Nationale Supérieure des Mines de Saint-Etienne, 2007.

[8] F. Hickernell. A generalized discrepancy and quadrature error bound. *Mathematics of Computation 67*, 299–322, 1998.

[9] B. Iooss. Numerical study of the metamodel validation process. In W.E. Biles, A. Saltelli, and C. Dini, editors, *Proceedings of the First International Conference on Advances in System Simulation (SIMUL 2009)*, 100-105, Porto, Portugal, september 2009.

[10] B. Iooss, L. Boussouf, A. Marrel, and V. Feuillard. Numerical study of algorithms for metamodel construction and validation. In S. Martorell, C. Guedes Soares, and J. Barnett, editors, *Safety, reliability and risk analysis - Proceedings of the ESREL 2008 Conference*, pages 2135–2141, Valencia, Spain, september 2008. CRC Press.

[11] B. Iooss, F. Van Dorpe, and N. Devictor. Response surfaces and sensitivity analyses for an environmental model of dose calculations. *Reliability Engineering and System Safety 91*, 1241–1251, 2006.

[12] R. Jin, W. Chen, and A. Sudjianto. An efficient algorithm for constructing optimal design of computer experiments. *Journal of Statistical Planning and Inference 134*, 268–287, 2005.

[13] M. E. Johnson, L. M. Moore, and D. Ylvisaker. Minimax and maximin distance design. *Journal of Statistical Planning and Inference 26*, 131–148, 1990.

[14] B. Jones and R. T. Johnson. Design and analysis for the Gaussian process model. *Quality and Reliability Engineering International 25*, 515–524, 2009.

[15] J.P.C. Kleijnen and R.G. Sargent. A methodology for fitting and validating metamodels in simulation. *European Journal of Operational Research 120*, 14–29, 2000.

[16] C. Lemieux. *Monte Carlo and quasi-Monte Carlo sampling*. Springer, 2009.

[17] M. Liefvendahl and R. Stocki. A study on algorithms for optimization of Latin hypercubes. *Journal of Statistical Planning and Inference 136*, 3231–3247, 2006.

[18] A. Marrel. *Mise en œuvre et utilisation du métamodèle processus gaussien pour l'analyse de sensibilité de modèles numériques*. Thèse de l'INSA Toulouse, 2008.

[19] A. Marrel, B. Iooss, F. Van Dorpe, and E. Volkova. An efficient methodology for modeling complex computer codes with Gaussian processes. *Computational Statistics and Data Analysis 52*, 4731–4744, 2008.

[20] A. Marrel, B. Iooss, B. Laurent, and O. Roustant. Calculations of Sobol indices for the Gaussian process metamodel. *Reliability Engineering and System Safety 94*, 742–751, 2009.

[21] J. D. Martin and T. W. Simpson. Use of kriging models to approximate deterministic computer models. *AIAA Journal 43*, 853–863, 2005.

[22] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics 21*, 239–245, 1979.

[23] M. Meckesheimer, A.J. Booker, R.R. Barton, and T.W. Simpson. Computationally inexpensive metamodel assessment strategies. *AIAA Journal 40*, 2053–2060, 2002.

[24] M. D. Morris and T. J. Mitchell. Exploratory designs for computational experiments. *Journal of Statistical Planning and Inference 43*, 381–402, 1995.

[25] J.-S. Park. Optimal Latin-hypercube designs for computer experiments. *Journal of Statistical Planning and Inference 39*, 95–111, 1993.

[26] G. Pujol. Simplex-based screening designs for estimating metamodels. *Reliability Engineering and System Safety 94*, 1156–1160, 2009.

[27] M.I. Reis dos Santos and A.M.O. Porta Nova. Statistical fitting and validation of non-linear simulation metamodels: A case study. *European Journal of Operational Research 171*, 53–63, 2006.

[28] G. Rennen. Subset selection from large datasets for kriging modeling. *Structural and Multidisciplinary Optimization 38*, 545–569, 2009.

[29] J. Sacks, W.J. Welch, T.J. Mitchell, and H.P. Wynn. Design and analysis of computer experiments. *Statistical Science 4*, 409–435, 1989.

[30] T. Santner, B. Williams, and W. Notz. *The design and analysis of computer experiments*. Springer, 2003.

[31] T. Simpson, J. Peplinski, P. Kock, and J. Allen. Metamodel for computer-based engineering designs: survey and recommandations. *Engineering with Computers 17*, 129–150, 2001.

[32] M. Stein. Large sample properties of simulations using Latin hypercube sampling. *Technometrics 29*, 143–151, 1987.

Single and Multiple Antenna Relay-Assisted Techniques for Uplink and Downlink OFDM Systems

Andreia Moço, Sara Teodoro, Adão Silva, Hugo Lima and Atílio Gameiro
Instituto de Telecomunicações
Aveiro University, UA
Campo Universitário, 3810-391 Aveiro, Portugal
e-mail: avm@av.it.pt, steodoro@av.it.pt, asilva@av.it.pt, hlima@av.it.pt and amg@ua.pt

***Abstract*— In this paper we propose and assess the performance of relay-assisted schemes designed for both the uplink and downlink OFDM based systems, using efficient distributed space-frequency block coding protocols. We consider the use of an antenna array at the base station and a single antenna at the user terminal. At the relay node we consider either single antenna or an antenna array. We assume that some of the user terminals deployed in a certain area could act as relaying-able terminals for the communication of other users. Two types of relay-assisted protocols are considered: equalize-and-forward and decode-and-forward. The optimal maximum ratio combining coefficients are derived for the proposed relay-assisted schemes. The performance of these cooperative schemes is evaluated under realistic scenarios, considering typical pedestrian scenarios based on WiMAX specifications and using channel convolutional turbo code. The proposed schemes are also compared against the non-cooperative OFDM based systems. Numerical results show that the availability of antenna arrays at the relays significantly improves the cooperative systems performance, which outperform the non-cooperative ones in most studied scenarios.**

Keywords - OFDM, relay, multiple antennas, cooperation, downlink and uplink.

I. INTRODUCTION

Wireless systems are one of the key components for enabling the information society. To meet the service requirements of future multimedia applications, Orthogonal Frequency Division Multiplexing (OFDM) is being adopted in various kinds of broadband wireless systems [2][3].

Cooperative communications is one of the fastest growing areas of research, and it is likely to be a key enabling technology for efficient spectrum use in the years to come. The key idea in user-cooperation is that of resource sharing among multiple nodes in a network [4][5]. It is commonly agreed that the provision of the broadband wireless component of the future wireless systems will probably rely on the use of multiple antennas at transmitter/receiver side. Multiple-input, multiple-output (MIMO) wireless communications are effective at mitigating the channel fading and thus improving the cellular system capacity. By configuring multiple antennas at both the base station (BS) and user terminal (UT), the channel capacity may be improved proportionally to the minimum number of the antennas at the transmitter and receiver [6]. However, considering a conventional cellular architecture with co-located antennas there is significant correlation between channels in some environments.

Cooperative diversity is a promising solution for wireless systems to increase capacity, extend coverage and provide fairness, namely for the scenarios where the direct link is in outage and the inter-user channel has good quality [4]. It can be achieved through cooperation of users, which share their antennas and thereby create a virtual antenna array (VAA) system. It is well known that the wireless channel is quite bursty, i.e., when a channel is in a severe fading state, it is likely to stay in the sate for a while. Thus, when a source cannot reach its destination due to severe fading, it will not be of much help trying by leveraging repeating-transmission protocols. However, if a third terminal that receives the data from the source could help via a channel that is independent from source to destination link, the chance for a successful transmission would increase, thus improving the overall system performance [7].

Research on cooperative diversity can be traced back to the pioneering papers of van der Meulen [8] and Cover [9] on the information theoretic properties of the relay channel. Explicit cooperation of neighbouring nodes was considered in [10][11]. Considering various degrees of knowledge of channel state information (CSI), they show that significant gains can be achieved over non-cooperative direct transmission. In [12] the authors have resorted, to solve the half duplex constraint, to a two-stage protocol and have shown in several papers that the use of cooperation provides a spatial diversity gain.

The concept of VAA or virtual MIMO has been introduced in [13], derived from the principles of relaying channel. Recently, the use of space-time block coding (STBC) implemented in a distributed fashion providing user cooperation has been proposed [14][15][16]. However, despite the extensive literature on cooperative relaying diversity, most of the work only considers single antenna relays. Studies and practical schemes with relays equipped with an antenna array are relatively scarce. Furthermore, most of the proposed schemes are not targeting OFDM based systems and/or have been assessed under unrealistic scenarios.

In [17] a theoretical diversity-multiplexing trade-off study is presented for a cooperative system with 1 and 2 antennas in a single relay scheme. In [18], a multiple antenna half-duplex relay–assisted maximum ratio combining (MRC) transmission scheme, that uses the decode-and-forward protocol has been proposed to increase

coverage. However, this scheme requires the knowledge of the instantaneous CSI prior to the transmission at the relays. Furthermore, in [19] the Rayleigh performance of a single-relay cooperative scenario with multiple-antenna nodes has been investigated, and pairwise error probability expressions were derived. Recently, a robust MIMO relay for multipoint-to-multipoint communication in wireless networks, and based on amplify and forward (AF) protocol, has been proposed [20].

In this paper we provide a detailed description of the work that we have started and presented at [1], on single/multiple antenna relay-assisted cooperative schemes designed for the uplink (UL) OFDM systems, and at [21], on single antenna relay-assisted cooperative schemes designed for the downlink (DL) OFDM systems. Moreover, we extend the latter scheme from single antenna relay to multiple antenna relay. We consider that some of the user terminals deployed in a certain area could act as relaying-able terminals for the communications of other users. We also assume that these terminals are available idle users connected to the network. The considered relay-assisted (RA) protocols are: equalize-and-forward (EF) and decode-and-forward (DF). Their performance is evaluated under realistic scenarios, considering typical pedestrian scenarios based on WiMAX specifications and channel turbo coding. These schemes are also compared against either SIMO or MISO OFDM non-cooperative uplink or downlink systems, respectively. Simulations reveal that the proposed cooperative schemes outperform the non-cooperative ones in most of the studied scenarios and that the availability of two-antenna arrays at the relay nodes dramatically increases the cooperative systems performance. Cooperating is especially advantageous when the source-to-relay link is better than the source-to-destination link, so that the relay receives data correctly and can have a considerable impact on the final performance.

The remaining paper is organized as follows: in section II we present a general description of the proposed multiple antenna relay-assisted for uplink and downlink systems. In sections III and IV, we derive and analyze the link equations for those relay-assisted schemes for uplink and downlink, respectively. The optimal relay-assisted MRC coefficients are derived. In section V, we assess the performance, in terms of BER, in several communication scenarios. The first part is devoted to discuss the uplink results while second one is for downlink. Finally, the main conclusions are pointed out in section VI.

II. System Model

Figs. 1 and 2 depict the proposed multiple antenna relay-assisted scheme for both the uplink and downlink OFDM based systems, which consist of a user terminal equipped with a single antenna, a base station equipped with an antenna array and a relay node which can have either one or two antennas.

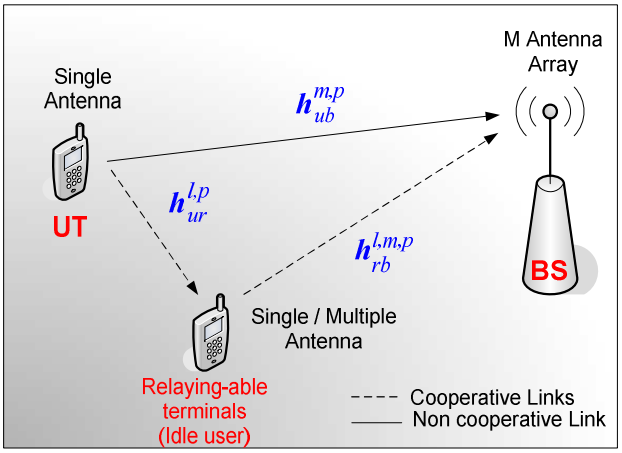


Figure 1. Single/double antenna relay-assisted system model for uplink.

Fig. 1 shows the system model for the uplink. We consider that N_c subcarriers are assigned to the UT. From this scenario, three main links can be identified: the direct UT-to-BS link, represented by $h_{ub}^{m,p}$, $m=1,2$ channels; the UT-to-RN link, formed by channels $h_{ur}^{l,p}$, $l=1,2$; and the RN-to-BS link, represented by channels $h_{rb}^{l,m,p}$, $l=1,2$, $m=1,2$, where index p denotes the p^{th} subcarrier. If the RN has a single antenna, then L=1 and the UT-to-RN and RN-to-BS links are represented by channels h_{ur}^{p} and $h_{rb}^{m,p}$ $,m=1,2$, respectively.

Fig. 2 is analogous to Fig. 1, but for the downlink. Since the BS is equipped with 2 transmit antennas (M=2), the non-cooperative link is a 2×1 MISO channel. It is represented by the $h_{bu}^{m,p}$ $,m=1,2$ channels. The cooperative links, BS-to-RN and RN-to-UT, are represented by $h_{br}^{m,l,p}$, $l=1,2$, $m=1,2$ and $h_{ru}^{l,p}$, $l=1,2$, respectively.

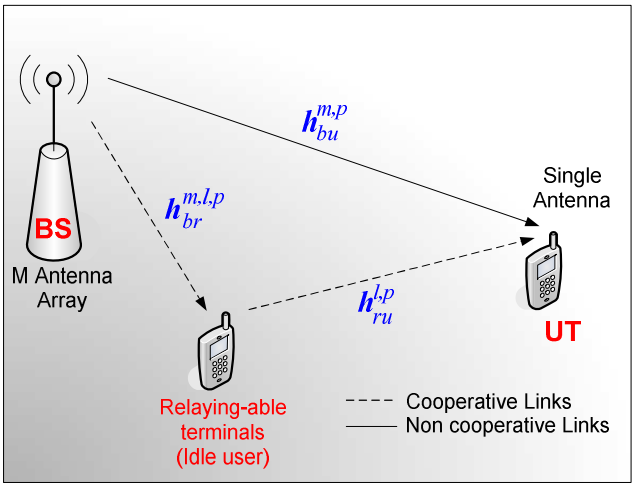


Figure 2. Single/double antenna relay-assisted system model for downlink.

If the RN has a single antenna, the BS-to-RN and RN-to-UT links are represented by channels $h_{br}^{m,p}$, $m=1,2$, and h_{ru}^{p} , respectively. Since we assume the relay is half-duplex, the communication cycle for both aforementioned cooperative schemes requires two phases:

- In the first one, the source, UT or BS depending on whether UL or DL systems are considered, broadcasts its own data at full power to its destination (BS or UT), and also to the relay node, which does not transmit data during this stage. For the downlink systems, the data are space-frequency encoded before transmission.
- During the second phase, the relay node can help the source by forwarding the information, also at full power, to the destination node (BS or UT), whereas the source is idle. If the RN is equipped with a 2-antenna array, the data to be transmitted to the BS or UT must also be space-frequency encoded.

The mapping scheme with two transmit antennas used in this work is shown in Table I [23] , where s_p is either the soft or hard decoded data estimated at the relay node on the p^{th} subcarrier for equalize-and-forward or decode-and-forward, respectively. Factor $1/\sqrt{2}$ is included in order to constrain the transmitted energy to one and $(.)^*$ denotes complex conjugation.

TABLE I: SFBC MAPPING SCHEME.

	Antenna 1	**Antenna 2**
Subcarrier *p*	$s_p/\sqrt{2}$	$-s_{p+1}^*/\sqrt{2}$
Subcarrier *p+1*	$s_{p+1}/\sqrt{2}$	$s_p^*/\sqrt{2}$

III. UPLINK RELAY-ASSISTED SCHEMES

We analyze two half-duplex multiple antenna relay-assisted schemes: equalize-and-forward and decode-and-forward. Throughout this paper, we shall refer to these schemes as 1×*L*×*M* RA EF and 1×*L*×*M* RA DF, depending on the number of antennas at the relay node (*L*) and BS (*M*).

Without loss of generality, hereinafter the mathematical formulation refers to a generic subcarrier p .

A. Relay-Assisted Equalize-and-Forward

In this work we consider that the RN node is either equipped with a single antenna or a 2-antenna array. For the first case, the amplify-and-forward protocol studied in [22] is equivalent to the RA EF protocol proposed here. However, if the signal at the relay is collected by two antennas, doing just a simple amplify-and-forward it is not the best strategy. We need to perform some kind of equalization to combine the received signals before re-transmission.

According to the communication cycle adopted in this paper, the received signal during the first communication phase at the BS antenna m , is found to be

$$y_{BS}^{m,p}(t) = d_p h_{ub}^{m,p} + n_{BS}^{m,p}(t), \quad (1)$$

where d_p is the data symbol for the p^{th} subcarrier, with unit power and $h_{ub}^{m,p}$ represents the complex flat Rayleigh fading non-cooperative channel of the p^{th} subcarrier on antenna m. Samples $n_{BS}^{m,p}(t)$ are complex additive white Gaussian noise (AWGN) on antenna *m* with zero mean and variance σ_{d1}^2.

The received signal at the RN, at instant t and antenna *l*, is

$$y_{RN}^{l,p}(t) = d_p h_{ur}^{l,p} + n_R^{l,p}(t), \quad (2)$$

where $n_R^{l,p}$ are AWGN samples added at the RN with zero mean and variance σ_r^2 . The received signals at the relay antennas at instant t are combined using MRC. The resulting soft decision variable, s_p , is expressed as

$$s_p = \underbrace{\alpha_p d_p \Gamma_{ur}^p}_{\text{Desired Signal}} + \underbrace{\alpha_p \sum_{l=1}^{L} h_{ur}^{l,p*} n_R^{l,p}(t)}_{\text{Relay Noise}}, \quad (3)$$

with $\Gamma_{ur}^p = \sum_{l=1}^{L} \left| h_{ur}^{l,p} \right|^2$ and α_p is a constant whose purpose is to constrain the transmitted power by the RN to one, given by

$$\alpha_p = \frac{1}{\sqrt{{\Gamma_{ur}^p}^2 + \sigma_r^2 \Gamma_{ur}^p}}. \quad (4)$$

The following actions that occur at the RA EF relay differ, depending on its number of receiving antennas (*L*).

1) Single Antenna Relay

In the single antenna scenario (*L*=1), the 1×1×*M* RA EF can conclude its action by re-transmitting the signal in (3) to the BS. Therefore, the received signals at BS antenna m and instant $t+T_s$, are expressed as

$$y_{BS}^{m,p}(t+T_s) = h_{rb}^{m,p} s_p + n_{BS}^{m,p}(t+T_s), \quad (5)$$

where $h_{rb}^{m,p}$ represents the complex flat Rayleigh fading channel between the RN and antenna m of the BS;

$n_{BS}^{m,p}(t+T_s)$ is AWGN added at $t+T_s$ on antenna m for the p^{th} subcarrier, with zero mean and variance σ_{d2}^2; and, T_s is a symbol transmission duration.

The total noise power of (5), conditioned to a specific channel realization, is related to σ_{d1}^2 by

$$\sigma_{y,p}^2 = \sigma_{d1}^2 \underbrace{(\beta_1 + \alpha_p^2 \Gamma_{ur}^p \left|h_{rb}^{m,p}\right|^2 \beta_2)}_{\beta_{m,p}}. \quad (6)$$

Coefficient β_1 is used to relate the variance of the noise that is added at the BS at instants t and $t+T_s$, $\sigma_{d2}^2 = \beta_1 \sigma_{d1}^2$. In the same manner, β_2 relates the noise that is added at the relay with the one added at BS at instant $t+T_s$, i.e., $\sigma_r^2 = \beta_2 \sigma_{d1}^2$.

At the BS side, MRC is used to combine the received signals from the direct and 2-hop links. Note that MRC is the optimal technique to maximize the overall signal-to-noise ratio (SNR). Since the noise variance of the received signals at the BS at instants t and $t+T_s$ is different, the coefficients that maximize the overall SNR are given by

$$g^{m,p}(t+kT_s) = \begin{cases} \dfrac{h_{ub}^{m,p*}}{\sigma_{d1}^2}, & k=0 \\ \dfrac{\alpha_p \Gamma_{ur}^p}{\sigma_{d1}^2 \beta_{m,p}} h_{rb}^{m,p*}, & k=1 \end{cases}. \quad (7)$$

The resulting soft decision variable, at the output of the MRC detector, may be expressed as

$$\hat{d}_p = \underbrace{\frac{d_p}{\sigma_{d1}^2}\left(\Gamma_{ub}^p + \alpha_p^2 {\Gamma_{ur}^p}^2 \sum_{m=1}^{M} \frac{\Gamma_{rb}^{m,p}}{\beta_{m,p}}\right)}_{Desired\ Signal} + \underbrace{\frac{1}{\sigma_{d1}^2}\alpha_p^2 \sum_{m=1}^{M} \frac{{\Gamma_{rb}^{m,p}}^2}{\beta_{m,p}} h_{ur}^{p*} n_R^p(t)}_{Relay\ Noise} + \underbrace{\frac{1}{\sigma_{d1}^2}\sum_{m=1}^{M}\left(h_{ub}^{m,p*} n_{BS}^{m,p}(t) + \frac{\alpha_p \Gamma_{ur}^p}{\beta_{m,p}} n_{BS}^{m,p}(t+T_s)\right)}_{BS\ Noise}, \quad (8)$$

where $\Gamma_{ub}^p = \sum_{m=1}^{M} \left|h_{ub}^{m,p}\right|^2$.

2) Two Antenna Relay

In this case before transmission, the RN encodes the soft data estimates represented by (3) with the SFBC encoding algorithm according to the scheme presented in Table I. The received signals at BS antenna m, at instant $t+T_s$, on subcarriers p and $p+1$, are given by

$$\begin{cases} y_{BS}^{m,p}(t+T_s) = \dfrac{1}{\sqrt{2}}\left(h_{rb}^{1,m,p} s_p - h_{rb}^{2,m,p} s_{p+1}^*\right) + n_{BS}^{m,p}(t+T_s) \\ y_{BS}^{m,p+1}(t+T_s) = \dfrac{1}{\sqrt{2}}\left(h_{rb}^{2,m,p+1} s_p^* + h_{rb}^{1,m,p+1} s_{p+1}\right) + n_{BS}^{m,p+1}(t+T_s) \end{cases} \quad (9)$$

where $h_{rb}^{l,m,p}$ represents the complex flat Rayleigh fading channel for the p^{th} subcarrier, between relay antenna l and BS antenna m. The OFDM systems are usually designed so that the subcarrier separation is significantly lower than the coherence bandwidth of the channel. Therefore, the fading in two adjacent subcarriers can be considered flat, i.e., we can consider $h_{rb}^{l,m,p}$ equal to $h_{rb}^{l,m,p+1}$. With this assumption, and defining $\Gamma_{rb}^{m,p}$ as $\sum_{l=1}^{L}\left|h_{rb}^{l,m,p}\right|^2$, the total noise power of (9), conditioned to a specific channel realization, is related to the variance of the noise added at the BS at t, σ_{d1}^2, by

$$\sigma_{y,p}^2 = \sigma_{d1}^2 \underbrace{(\beta_1 + \alpha_p^2 \Gamma_{ur}^p \Gamma_{rb}^{m,p} \beta_2 / 2)}_{\beta_{m,p}}. \quad (10)$$

At the BS side, MRC is also used to optimally combine the received signals from the direct and 2-hop links. The direct link processing is the same as presented for L=1 case.

For the 2-hop link, the signals for antenna m and an arbitrary pair of adjacent subcarriers p and $p+1$, are subject to space-frequency block decoding. The resulting signal is

$$\begin{cases} \hat{s}_{m,p}(t+T_s) = g^{1,m,p*}(t+T_s) y_{BS}^p(t+T_s) \\ \qquad\qquad + g_p^{2,m,p}(t+T_s) y_{BS}^{p+1*}(t+T_s) \\ \hat{s}_{m,p+1}(t+T_s) = -g^{2,m,p}(t+T_s) y_{BS}^{p*}(t+T_s) \\ \qquad\qquad + g^{1,m,p*}(t+T_s) y_{BS}^{p+1}(t+T_s) \end{cases}, \quad (11)$$

with equalization coefficients used at both instants given by

$$g^{l,m,p}(t+kT_s) = \begin{cases} \dfrac{h_{ub}^{m,p*}}{\sigma_{d1}^2}, & k=0 \\ \dfrac{\alpha_p \Gamma_{ur}^p}{\sqrt{2}\,\sigma_{d1}^2 \beta_{m,p}} h_{rb}^{l,m,p}, & k=1 \end{cases}. \quad (12)$$

The resulting soft decision variable, at the output of the MRC detector, may be expressed as

$$\hat{d}_p = \underbrace{\frac{d_p}{\sigma_{d1}^2}\left(\Gamma_{ub}^p + \frac{1}{2}\alpha_p^2 {\Gamma_{ur}^p}^2 \sum_{m=1}^{M}\frac{\Gamma_{rb}^{m,p}}{\beta_{m,p}}\right)}_{\text{Desired Signal}} + \underbrace{\frac{1}{2\sigma_{d1}^2}\alpha_p^2 \sum_{m=1}^{M}\frac{{\Gamma_{rb}^{m,p}}^2}{\beta_{m,p}}\sum_{l=1}^{L=2} h_{ur}^{l,p*} n_R^{l,p}(t)}_{\text{Relay Noise}} + \underbrace{\frac{1}{\sigma_{d1}^2}\sum_{m=1}^{M}\left(h_{ub}^{m,p*} n_{BS}^{m,p}(t) + \frac{\alpha_p \Gamma_{ur}^p}{\sqrt{2}\beta_{m,p}}\tilde{n}_{BS}^{m,p}(t+T_s)\right)}_{\text{BS Noise}}, \quad (13)$$

where $\tilde{n}_{BS}^{m,p}(t+T_s)$ is the residual noise which results from SFBC decoding.

B. Relay-Assisted Decode and Forward

In contrast to RA EF, in the RA DF scheme, the relay hard decodes the incoming signals before forwarding them.

The communication cycle for RA DF is as follows. Firstly, the UT transmits at full power for T_s of the time. The expressions for the signals at the receiving antennas of the BS and RN for the first phase are the same as presented in (1) and (2), respectively.

During the second phase, the RN demodulates, decodes and forwards the received signals. Thus, the received signals at the RN at instant t are combined using MRC, and the soft decision in the RN, s_p, is also given by (3). Now, if s_p is successfully detected, then the data to be retransmitted by the RN, here represented by $\bar{s}_p$, is the correct data d_p that the UT transmitted. If the number of transmit antennas of the RN is one, then the RN concludes its action by forwarding $\bar{s}_p$ to the BS. When the RN is equipped with two antennas, the data must be encoded with the presented SFBC scheme before the transmission.

1) Single Antenna Relay

In the single antenna scenario, the received signals at BS antenna m and instant $t+T_s$ are expressed as

$$y_{BS}^{m,p}(t+T_s) = h_{rb}^{m,p}\bar{s}_p + n_{BS}^{m,p}(t+T_s), \quad (14)$$

where $h_{rb}^{m,p}$ and $n_{BS}^{m,p}(t+T_s)$ hold the same meaning as in (5). The total noise power of $y_{BS}^{m,p}(t+T_s)$ is $\sigma_{d1}^2\beta_1$.

For RA DF, the soft decision variable at the UT is given by

$$\hat{d}_p = \underbrace{\frac{1}{\sigma_{d1}^2}\left(d_p\,\Gamma_{ub}^p + \bar{s}_p\frac{1}{\beta_1}\sum_{m=1}^{M}\Gamma_{rb}^{m,p}\right)}_{\text{Desired Signal}} + \underbrace{\frac{1}{\sigma_{d1}^2}\sum_{m=1}^{M}\left(h_{ub}^{m,p*} n_{BS}^{m,p}(t) + \frac{1}{\beta_1} n_{BS}^{m,p}(t+T_s)\right)}_{\text{BS Noise}}. \quad (15)$$

2) Two Antenna Relay

Since the two-antenna array at the RN requires additional SFBC for transmission in the RN-to-BS link, the 2-hop link contributions at BS antenna m, at instant $t+T_s$, can be obtained from (9) replacing s_p by $\bar{s}_p$. The 2-hop decoded signals are also given by (11). As in RA EF, the BS combines the direct and 2-hop link signals using MRC whose coefficients are given by

$$g^{l,m,p}(t+kT_s) = \begin{cases} \dfrac{h_{ub}^{m,p*}}{\sigma_{d1}^2} & ,k=0 \\ \dfrac{h_{rb}^{l,m,p}}{\sigma_{d1}^2\beta_1} & ,k=1 \end{cases}. \quad (16)$$

The final soft decision variable for subcarrier p is

$$\hat{d}_p = \underbrace{\frac{1}{\sigma_{d1}^2}\left(d_p\,\Gamma_{ub}^p + \bar{s}_p\frac{1}{2\beta_1}\sum_{m=1}^{M}\Gamma_{rb}^{m,p}\right)}_{\text{Desired Signal}} + \underbrace{\frac{1}{\sigma_{d1}^2}\sum_{m=1}^{M}\left(h_{ub}^{m,p*} n_{BS}^{m,p}(t) + \frac{1}{\sqrt{2}\beta_1}\tilde{n}_{BS}^{m,p}(t+T_s)\right)}_{\text{BS Noise}}. \quad (17)$$

As it can be seen from (15) and (17), in the outage case when the relay fails to decode the data correctly, it cannot help the UT for the current cooperation round. In such case, the BS will get interference from the cooperative path and cooperation will not be beneficial. For the particular case where $\bar{s}_p = d_p$, i.e., the data sent by UT is successful decoded at the RN, (17) reduces to (18). In practical systems this can be achieved when the link UT-to-RN has high quality.

$$\hat{d}_p = \underbrace{\frac{1}{\sigma_{d1}^2}\left(\Gamma_{ub}^p + \frac{1}{2\beta_1}\sum_{m=1}^{M}\Gamma_{rb}^{m,p}\right)d_p}_{\text{Desired Signal}} + \underbrace{\frac{1}{\sigma_{d1}^2}\sum_{m=1}^{M}\left(h_{ub}^{m,p*} n_{BS}^{m,p}(t) + \frac{1}{\sqrt{2}\beta_1}\tilde{n}_{BS}^{m,p}(t+T_s)\right)}_{\text{BS Noise}}. \quad (18)$$

Note that for this particular scenario, this RA scheme can achieve a diversity order of 6 (4 by RN-to-BS link and 2 by the direct one), and an antenna gain of approximately 6 dB.

IV. DOWNLINK RELAY-ASSISTED SCHEMES

Similarly to the previous section, we analyze the two relay-assisted schemes for downlink system, presented in Fig. 2, using also EF and DF relay protocols. These schemes are referred as $M\times L\times 1$ RA EF and $M\times L\times 1$ RA DF, depending on the number of antennas at the relay node and BS, L and M respectively. As for the uplink schemes, we consider the cases with 1 and 2 antennas at the relay. The formulation for single antenna RN was first derived in [21]. The main contribution of this section is to extend the formulation for 2 antennas at the RN.

A. Relay-Assisted Equalize-and-Forward

In the single-antenna relay-assisted scheme, the amplify-and-forward protocol is equivalent to the RA EF protocol proposed in [21]. For the relay case with 2 antennas, we need to equalize and estimate the received data at the relay and only after that retransmit the coded data. However, no hard decision is made at the RN.

During the first phase the BS transmits at full power for T_s of the time. Thus, the received signal at the UT, at instant t, is given by

$$\begin{cases} y_{UT}^{p}(t)=\frac{1}{\sqrt{2}}\left(h_{bu}^{1,p}\,d_p-h_{bu}^{2,p}\,d_{p+1}^{*}\right)+n_{UT}^{p}(t) \\ y_{UT}^{p+1}(t)=\frac{1}{\sqrt{2}}\left(h_{bu}^{2,p+1}\,d_p^{*}+h_{bu}^{1,p+1}\,d_{p+1}\right)+n_{UT}^{p+1}(t) \end{cases}, \quad (19)$$

where d_p is the data symbol for the p^{th} subcarrier, with unit power; $h_{bu}^{m,p}$, with $m=1,2$, represents the complex Rayleigh flat fading channel between m^{th} BS antenna and UT; and, samples $n_{UT}^{p}(t)$ are zero mean complex additive AWGN samples on the UT with variance σ_{d1}^2.

The received signals at the l^{th} relay antenna (which can be l=1 or l=1,2, depending whether the RN is equipped with 1 or 2 antennas), at instant t, are given by

$$\begin{cases} y_{Rl}^{p}(t)=\frac{1}{\sqrt{2}}\left(h_{br}^{1,l,p}\,d_p-h_{br}^{2,l,p}\,d_{p+1}^{*}\right)+n_{Rl}^{p}(t) \\ y_{Rl}^{p+1}(t)=\frac{1}{\sqrt{2}}\left(h_{br}^{2,l,p}\,d_p^{*}+h_{br}^{1,l,p}\,d_{p+1}\right)+n_{Rl}^{p+1}(t) \end{cases}, \quad (20)$$

where $h_{br}^{m,l,p}$, with $m=1,2$ and $l=1,2$ represents the complex Rayleigh flat fading channel between m^{th} BS antenna and l^{th} RN antenna, and, $n_{Rl}^{p}(t)$ represents zero mean complex additive AWGN samples on the l^{th} relay antenna, with variance σ_r^2.

During the second phase, the RN comprises several tasks over the received signals, which include: equalization plus space-frequency decoding, power normalization and forwarding. Since the mathematical formulation for RA EF depends on the availability of a 2-antenna array at the relay node, we treat these cases separately, deriving the final decision variables for both schemes, with L=1 and L=2.

1) Single Antenna Relay

When the relay is equipped with a single antenna, i.e., L=1, the soft decision variable in this terminal is expressed as

$$\begin{cases} s_p=g_r^{1,p^*}y_R^{p}(t)+g_r^{2,p}y_R^{p+1^*}(t) \\ s_{p+1}=-g_r^{2,p}y_R^{p^*}(t)+g_r^{1,p^*}y_R^{p+1}(t) \end{cases}, \quad (21)$$

where the equalization coefficients are defined as

$$g_r^{m,p}=\frac{h_{br}^{m,p}}{\sqrt{2}\Gamma_{br}^{p}}, \quad (22)$$

with m=1,2 and $\Gamma_{br}^{p}=\frac{1}{2}\sum_{m=1}^{2}\left|h_{br}^{m,p}\right|^2$ represents the BS-to-RN equivalent channel gain. The expression for s_p can be simplified to the expression that follows

$$s_p=d_p+\tilde{n}_{R1}^{p}, \quad (23)$$

where $\tilde{n}_{R1}^{p}$ is the noise that results from SFBC decoding with variance $\sigma_{\tilde{r}1}^2=\frac{\sigma_r^2}{\Gamma_{br}^{p}}$.

The following task that the RN does is to normalize the overall transmit power of $\hat{s}_p$ to one. The pertinent normalization constant is

$$\alpha_p=\frac{1}{\sqrt{1+\frac{\sigma_r^2}{\Gamma_{br}^{p}}}}. \quad (24)$$

At the UT, for the cooperative link the received signal, on subcarrier p, is given by

$$y_{UT}^{p}(t+T_s)=h_{ru}^{p}\alpha_p s_p+n_{UT}^{p}(t+T_s), \quad (25)$$

where h_{ru}^{p} represents the complex flat Rayleigh fading channel between RN and UT; $n_{UT}^{p}(t+T_s)$ is AWGN noise

added in the UT, during the second phase, with zero mean and variance σ_{d2}^2.

The total noise variance of (25), conditioned to a specific channel realization is referred to as $\sigma_{y,u}^2$ and is found to be

$$\sigma_{y,u}^2 = \alpha_p^2 \left|h_{ru}^{1,p}\right|^2 \sigma_r^2 / \Gamma_{br}^p + \sigma_{d2}^2 . \quad (26)$$

Now we define factor β_1 as the relation between the noise variance added at the UT at instants t and $t+T_s$. In the same way, β_2 is defined as the relation between σ_{d1}^2 and the variance of the noise added at the RN. In (27) we express $\sigma_{y,u}^2$ as a function of σ_{d1}^2.

$$\sigma_{y,u}^2 = \beta_{y,u}\sigma_{d1}^2 = \left(\frac{\alpha_p^2 \left|h_{ru}^p\right|^2 \beta_2}{\Gamma_{br}^p} + \beta_1 \right) \sigma_{d1}^2 . \quad (27)$$

At the UT, MRC is used to combine the received signal from the direct path and the 2-hop cooperative links. Thus, the estimated signal for an arbitrary pair of adjacent subcarriers, at instants t and $t+T_s$, after the space-frequency combining scheme being applied, is expressed as

$$\begin{cases} \hat{d}_p = g^{1,p^*}\left(t+T_s\right) y_{UT}^p\left(t+T_s\right) + \\ \qquad g^{1,p^*}\left(t\right) y_{UT}^p\left(t\right) + g^{2,p}\left(t\right) y_{UT}^{p+1^*}\left(t\right) \\ \hat{d}_{p+1} = g^{1,p^*}\left(t+T_s\right) y_{UT}^p\left(t+T_s\right) - \\ \qquad g^{2,p}\left(t\right) y_{UT}^{p^*}\left(t\right) + g^{1,p^*}\left(t\right) y_{UT}^{p+1}\left(t\right) \end{cases} \quad (28)$$

where the equalization coefficients $g^{m,p}\left(\mathrm{n}+kT_s\right)$ are defined as

$$g^{m,p}(t+kT_s) = \begin{cases} \dfrac{\dfrac{h_{bu}^{m,p}}{\sqrt{2}\sigma_{d1}^2}}{\dfrac{1}{2\sigma_{d1}^2}\Gamma_{bu}^p + \dfrac{1}{\sigma_{y,u}^2}\alpha_p^2\left|h_{ru}^p\right|^2}, & k=0 \\ \dfrac{\alpha_p \dfrac{h_{ru}^p}{\sigma_{y,u}^2}}{\dfrac{1}{2\sigma_{d1}^2}\Gamma_{bu}^p + \dfrac{1}{\sigma_{y,u}^2}\alpha_p^2\left|h_{ru}^p\right|^2}, & k=1 \end{cases} . \quad (29)$$

Here we used normalized coefficients to allow higher order modulations. The resulting soft decision variable, for subcarrier *p*, may be expressed as

$$\hat{d}_p = d_p + \psi \underbrace{\frac{\left|h_{ru}^p\right|^2 \alpha_p^2}{\beta_{y,u}} \tilde{n}_R^p(t)}_{\text{Relay Noise}} + \underbrace{\psi \frac{h_{ru}^{p^*}\alpha_p}{\beta_{y,u}} n_U^p(t+T_s) + \psi \tilde{n}_U^p(t)}_{\text{UT Noise}} , \quad (30)$$

with $\psi = \dfrac{1}{\Gamma_{bu}^p + \dfrac{1}{\beta_{y,u}}\left|h_{ru}^{1,p}\right|^2 \alpha_p^2}$ and $\Gamma_{bu}^p = \dfrac{1}{2}\sum_{m=1}^{2}\left|h_{bu}^{m,p}\right|^2$.

2) Two Antenna Relay

When the RN is equipped with a 2-antenna array (i.e., L=2), the soft decision variable at the l^{th} relay antenna (l=1,2), is expressed as follows

$$\begin{cases} s_p = \sum_{l=1}^{2}\left(g_r^{1,l,p^*} y_{Rl}^p\left(t\right) + g_r^{2,l,p} y_{Rl}^{p+1^*}\left(t\right)\right) \\ s_{p+1} = \sum_{l=1}^{2}\left(-g_r^{2,l,p} y_{Rl}^{p^*}\left(t\right) + g_r^{1,l,p^*} y_{Rl}^{p+1}\left(t\right)\right) \end{cases} , \quad (31)$$

where the equalization coefficients, $g_r^{m,l,p}$, for m=1,2, are defined as

$$g_r^{m,l,p} = \frac{\left(\dfrac{h_{br}^{m,l,p}}{\sqrt{2}}\right)}{\dfrac{1}{2}\sum_{l=1}^{2}\sum_{m=1}^{2}\left|h_{br}^{m,l,p}\right|^2} . \quad (32)$$

After this processing, the RN retransmits $\hat{s}_p$ according to the SFBC mapping scheme and sends it to the UT. Thus, the received signals in the UT, at instant $t+T_s$, on subcarriers p and p+1, are given by

$$\begin{cases} y_{UT}^p\left(t+T_s\right) = \dfrac{1}{\sqrt{2}}\left(h_{ru}^{1,p}\, s_p - h_{ru}^{2,p}\, s_{p+1}^*\right) + n_{UT}^p\left(t+T_s\right) \\ y_{UT}^{p+1}\left(t+T_s\right) = \dfrac{1}{\sqrt{2}}\left(h_{ru}^{2,p}\, s_p^* + h_{ru}^{1,p}\, s_{p+1}\right) + n_{UT}^{p+1}\left(t+T_s\right) \end{cases} . \quad (33)$$

The total noise variance of the received signal in UT, during the second phase, is also referred as $\sigma_{y,u}^2$ and now is given by

$$\sigma_{y,u}^2 = \beta_{y,u}\sigma_{d1}^2 = \left(\frac{\alpha_{EF,p}^2 \Gamma_{ru}^p \beta_2}{\sum_{n=1}^{2} \Gamma_{br}^{l,p}} + \beta_1 \right) \sigma_{d1}^2, \quad (34)$$

with $\Gamma_{br}^{l,p} = \frac{1}{2}\sum_{m=1}^{2} \left| h_{br}^{m,l,p} \right|^2$.

At the UT side, MRC is used to combine the received signals from both phases, obtaining

$$\begin{cases} \hat{d}_p = \sum_{k=0}^{1} \left(\begin{array}{l} g^{1,p^*}\left(t+kT_s\right) y_{UT}^p\left(t+kT_s\right) \\ \quad + g_p^{2,p}\left(t+kT_s\right) y_U^{p+1^*}\left(t+kT_s\right) \end{array} \right) \\ \hat{d}_{p+1} = \sum_{k=0}^{1} \left(\begin{array}{l} -g^{2,p}\left(t+kT_s\right) y_{UT}^{p^*}\left(t+kT_s\right) \\ \quad + g^{1,p^*}(t+kT_s) y_U^{p+1}\left(t+kT_s\right) \end{array} \right) \end{cases}, \quad (35)$$

where now the equalization coefficients are given by

$$g^{q,p}\left(\mathrm{t}+kT_s\right) = \begin{cases} \dfrac{\dfrac{h_{bu}^{q,p}}{\sqrt{2}\sigma_{d1}^2}}{\dfrac{1}{\sigma_{d1}^2}\Gamma_{bu}^p + \dfrac{1}{\sigma_{y,u}^2}\Gamma_{ru}^p}, & k=0 \\ \dfrac{\dfrac{h_{ru}^{q,p}}{\sqrt{2}\sigma_{y,u}^2}}{\dfrac{1}{\sigma_{d1}^2}\Gamma_{bu}^p + \dfrac{1}{\sigma_{y,u}^2}\Gamma_{ru}^p}, & k=1 \end{cases}, \quad (36)$$

with $\Gamma_{ru}^p = \frac{1}{2}\sum_{l=1}^{2} \left| h_{ru}^{l,p} \right|^2$ and q represents the antenna script for BS (for k=0) and for RN (k=1).

The final expression for signal estimation would be

$$\hat{d}_p = d_p + \underbrace{\psi \sum_{l=1}^{2} \left(\frac{\alpha_p^2}{\beta_{y,u}} \Gamma_{ru}^p \tilde{n}_{Rl}^p(t) \right)}_{\text{Relay Noise}} + \underbrace{\psi \frac{\alpha_p}{\beta_{y,u}} \tilde{n}_U^p(t+T_S) + \tilde{n}_U^p(t)}_{\text{UT Noise}}, \quad (37)$$

with $\psi = \dfrac{1}{\Gamma_{bu}^p + \dfrac{\alpha_p^2}{\beta_{y,u}}\Gamma_{ru}^p}$.

B. Relay-Assisted Decode-and-Forward

In this scheme, during the second phase, the relay first demodulates and hard decodes the incoming signals before forwarding it. Mathematical expressions for the estimated symbols for schemes with L=1 and L=2, are made separately.

1) Single Antenna Relay

When the RN is equipped with a single antenna, the soft decision variable at the relay is expressed in (21). The hard decision of s_p, also represented by $\overline{s}_p$, is forwarded to the UT. The received signal at UT is

$$y_{UT}^p\left(t+T_s\right) = h_{ru}^p\, \overline{s}_p + n_{UT}^p\left(t+T_s\right). \quad (38)$$

Signals received in both phases are then combined using MRC, resulting in (28), but now with the equalization coefficients defined as

$$g^{q,p}(t+kT_s) = \begin{cases} \dfrac{\dfrac{h_{bu}^{q,p}}{\sqrt{2}\sigma_{d1}^2}}{\dfrac{1}{\sigma_{d1}^2}\Gamma_{bu}^p + \dfrac{1}{\sigma_{d2}^2}\left|h_{ru}^{1,p}\right|^2}, & k=0 \\ \dfrac{\dfrac{h_{ru}^{p^*}}{\sigma_{d2}^2}}{\dfrac{1}{\sigma_{d1}^2}\Gamma_{bu}^p + \dfrac{1}{\sigma_{d2}^2}\left|h_{ru}^{1,p}\right|^2}, & k=1 \end{cases}. \quad (39)$$

The final expression for the estimated signal is given by

$$\hat{d}_p = \underbrace{\psi\left(\Gamma_{bu}^p s_p + \frac{1}{\beta_1}\left|h_{ru}^p\right|^2 \overline{s}_p \right)}_{\text{Desired Signal}} + \underbrace{\psi \tilde{n}_U^p(t) + \frac{\psi}{\beta_1} h_{ru}^{p^*} n_U^p(t+T_s)}_{\text{UT Noise}}, \quad (40)$$

with $\psi = \dfrac{1}{\Gamma_{bu}^p + \left|h_{ru}^p\right|^2 \Big/ \beta_1}$.

The optimal situation occurs when the data is successfully detected at the RN, i.e., $\overline{s}_p = s_p$. On other hand, when the relay fails in decoding the data correctly, the UT will get interference from the cooperative path and cooperation will not be beneficial.

2) Two Antenna Relay

Considering the RA scheme with a 2-antennas relay, the decoding expressions for the relay received signals, in the first phase, is expressed as (31) with the equalization coefficients defined as (32). After that, the RN hard decodes and then re-encodes the data and forwards it to the UT. If the decoded data at the RNs is correct, the data to be retransmitted by the RN antenna *l*, i.e., $\bar{s}_p$, is the data d_p that the BS broadcasted.

Signals received in both phases are then combined using MRC as in (35), with the coefficients defined as

$$g^{q,p}(t+kT_s)=\begin{cases}\dfrac{\dfrac{h_{bu}^{q,p}}{\sqrt{2}\sigma_{d1}^2}}{\dfrac{1}{\sigma_{d1}^2}\Gamma_{bu}^p+\dfrac{1}{\sigma_{d2}^2}\Gamma_{ru}^p}, & k=0\\ \dfrac{\left(\dfrac{h_{ru}^{q,p}}{\sqrt{2}\sigma_{d2}^2}\right)}{\dfrac{1}{\sigma_{d1}^2}\Gamma_{bu}^p+\dfrac{1}{\sigma_{d2}^2}\Gamma_{ru}^p}, & k=1\end{cases}. \quad (42)$$

The final expression for the estimated signal for subcarrier *p* may be expanded as

$$\hat{d}_p=\underbrace{\psi\left(\Gamma_{bu}^p d_p+\frac{1}{\beta_2}\Gamma_{ru}^p\bar{s}_p\right)}_{\text{Desired Signal}}+\underbrace{\psi\tilde{n}_U^p(t)+\frac{\psi}{\beta_2}\tilde{n}_U^p(t+T_s)}_{\text{UT Noise}}, \quad (43)$$

with $\psi=\dfrac{1}{\Gamma_{bu}^p+\Gamma_{ru}^p/\beta_2}$.

As in the previous scheme, RA DF with *L*=1, the optimal situation occurs when the data is successfully detected at the RN, i.e., $\bar{s}_p=d_p$ and for this specific situation (43) reduces to

$$\hat{d}_p=\underbrace{d_p}_{\text{Desired Signal}}+\underbrace{\psi\tilde{n}_U^p(t)+\frac{\psi}{\beta_2}\tilde{n}_U^p(t+T_s)}_{\text{UT Noise}}. \quad (44)$$

V. NUMERICAL RESULTS

In order to evaluate the performance of the presented relay-assisted schemes we considered a typical pedestrian scenario, based on WiMAX specifications. The main simulation settings are summarized in Table II.

With the aim of having the same spectral efficiency between both cooperative and non-cooperative schemes, we used 2 different modulation and coding modes: for the relay-assisted schemes we used QPSK and a CTC, with a block size of (7200, 3600); for the non-cooperative schemes we used BPSK and a CTC with block a size of (3600, 1800). For both cases, the code rate was set to 1/2 and a Max Log MAP algorithm with 8 iterations was used.

TABLE II: MAIN SIMULATIONS PARAMETERS.

WiMAX General Signal Definitions [24]	• FFT size: 1024; • number of available carriers: 400; • sampling frequency: 11.20 MHz; • useful symbol duration: 91.43μs; • cyclic prefix length: 11.43μs; • overall OFDM symbol duration: 102.86μs; • sub-carrier separation: 10.94 kHz; • number of OFDM symbols per block: 9;		
Nodes's Antenna Array Size	• UT: 1 Tx/Rx antenna • RN: 1 ou 2 Tx/Rx antennas (*L*=1 or 2) • BS: 2 Tx/Rx antennas (*M*= 2)		
Channel Model	• ITU pedestrian model B for 3km/h [25] • modified tap delays according to the sampling frequency defined for WiMAX.		
UT and RN velocities	3 km/h		
Channel Code	Convolutional Turbo Code (CTC)		
Channel Decoder	Max Log MAP algorithm with 8 iterations		
Modulation and coding schemes for (CTC)	**Constellation**	**Code rate**	**CTC code size (N, K)**
	BPSK	1/2	(3600,1800)
	QPSK	1/2	(7200, 3600)

Concerning the MIMO model and for the uplink system, we extended the ITU time model to space-time, assuming that the distance between antenna elements is far apart enough to assume *LM* independent channels, i.e, we assume independent fading processes. We consider that the UT has just one antenna, the BS has 2 receiving antennas (*M* =2) and the relay can have either 1 or 2 antennas (*L*=1 or 2). We further assume perfect channel state information at both RN and UT and the overall transmitted power is normalized to 1. For the downlink system, we use spatial transmitter correlation matrix with an average angle of departure (AoD) of 50°, the standard deviation of AoD set to 8°, and a spatial receiver matrix (at RN and UT when equipped with an antenna array) with an average angle of arrival (AoA) of 67.5º, standard deviation of AoA set to 68°, and antenna spacing at both transmitter and receiver set to 0.5 wavelength.

The results of the relay-assisted and non-cooperative schemes are presented in terms of the average BER as function of E_b/N_0, where E_b is the received energy per bit

and $N_0/2$ the bilateral power spectral density of the noise added at the BS in the direct link. Since the aim is to compare the performance using single or multiple antenna relay in scenarios with different link qualities, we present results considering different values of E_b/N_0 for each links: direct link Source-to-Destination (E_b/N_0); Source-to-RN (E_b/N_R), and RN-to-Destination (E_b/N_2), as shown in Table IV. We define and focus our simulation efforts on three scenarios, which will be referred to as scenarios I (all links with similar quality), II (RN has a good connection to the source) and III (RN have a good connection to the source and destination). These scenarios definitions hold for UL and DL communications as presented in Table III. Relay assignment algorithms were not considered in this work, i.e., algorithms to select the best terminals to cooperate. However, some relay assignment algorithms can be found in [26].

We compare the proposed relay-assisted schemes against the non-cooperative or co-located uplink and downlink systems. In the former, the non-cooperative system is represented by the SISO and 1×*M* MRC point-to-point communication systems, i.e., cellular systems with a single antenna at the UT and a BS with *M* antennas, where the signals of each antenna are MRC combined.

In the latter, the non-cooperative systems used as reference are the 2×1 and 2×2 MRC SFBC Alamouti point-to-point communication systems. Moreover, we compare the single antenna relay-assisted schemes against the multiple antenna ones. For the sake of simplicity, the numerical results used for assessing the performance of the RA schemes in the UL and DL communication will be presented in two separate sections.

TABLE III: UPLINK AND DOWNLINK SIMULATION SCENARIOS.

Scenario for UL/DL	E_b/N Relations with E_b/N_0	Observations
I	$E_b/N_0=E_b/N_R = E_b/N_2$	All links exhibit similar quality
II	$E_b/N_R = E_b/N_0+10dB$	RN has a good connection to the Source Node
III	$E_b/N_R = E_b/N_2 = E_b/N_0 +10dB$	RN has a good connection to the Source and Destination Nodes

TABLE IV: EB/N NOTATION FOR EACH LINK.

DL	UL	E_b/N
BS-to-UT	UT-to-BS	E_b/N_0
BS-to-RN	UT-to-RN	E_b/N_R
RN-to-UT	RN-to-BS	E_b/N_2

1) *Uplink Relay-Assisted Schemes*

In order to assess RA schemes in UL communication, we consider the different scenarios presented in Table III. The first one assumes that all links have the same quality (scenario I), i.e., $E_b/N_0= E_b/N_R= E_b/N_2$. In the second one, the quality of the UT-to-RN link is made 10 dB higher than the direct link, i.e., $E_b/N_R= E_b/N_0+10dB$. This second scenario is more realistic, since the RN is assumed to be near the UT. In practical cellular systems the cooperative mode is only activated if an active UT can see an idle one with a high link quality. In the third scenario, the quality of the UT-to-RN and RN-to-BS links is made 10 dB higher than the direct link, i.e., $E_b/N_R= E_b/N_2=E_b/N_0+10$ dB.

Fig. 3 shows results for the case at which all links have similar quality. It suggests that the systems whose relay is equipped with an antenna array outperform the ones that use single antenna relays. In fact, the existence of an antenna array at the RN dramatically improves the UT-to-RN link, because it turns this channel into 1×2 SIMO, which provide both 2-order diversity and antenna gain of approximately 3dB. Also, Fig. 3 shows that 1×1×2 RA DF is not working properly, since the information that is erroneously hard decoded at the relay will be seen as interference at the BS and will not improve the system behavior. In this scenario, EF schemes are preferable, as the relay only takes a soft decision over the incoming data and the hard-decision is left for the BS (after combining both signals received from direct and cooperative links).For a BER target of 10^{-4}, the 1×2×2 RA EF scheme yields about 3.6 dB of increase in E_b/N_0, over the non cooperative 1×2 MRC system.

Fig. 4 shows the performance of cooperative schemes for scenarios where the UT-to-RN link is significantly better than the other links. We consider the case where $E_R/N_0=E_b/N_0+10dB$ and $E_b/N_0=E_{b2}/N_0$. Since the UT-to-RN link is improved with respect to the previous scenario, all RA systems reveal relative performance improvements.

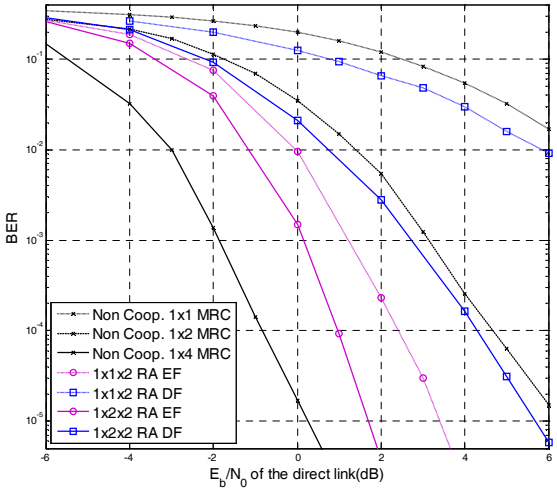


Figure 3. Performance comparison of RA schemes for the uplink: when $Eb/N_0= E_b/N_R= E_b/N_2$ (Scenario I).

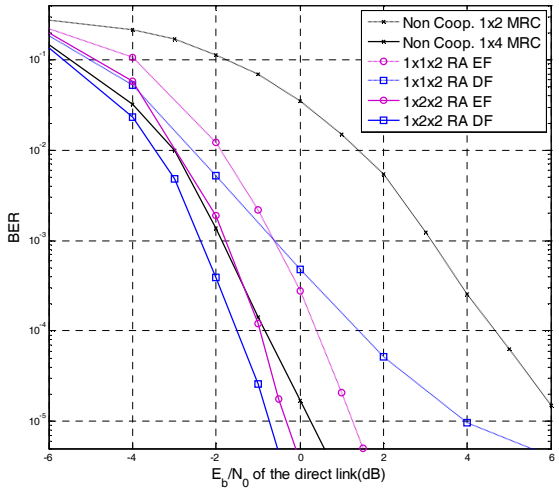


Figure 4. Performance comparison of RA schemes for the uplink: when $E_b/N_R = E_b/N_0+10$dB (Scenario II).

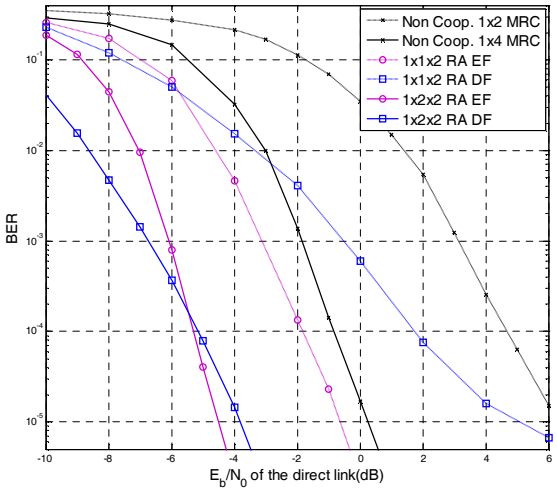


Figure 5. Performance comparison of single and multiple antenna RA schemes for the uplink: when $E_b/N_R = E_b/N_0+10$dB and $E_b/N_2 = E_b/N_0+10$dB (scenario III).

In the 1×1×2 RA schemes, we observe that, for increasingly higher E_b/N_0 values (above 0 dB), 1×1×2 RA EF outperforms 1×1×2 RA DF. The explanation for the 1×1×2 RA DF malfunction is the fact that the UT-to-RN SISO channel is still not good enough to allow for proper hard decoding at the RN. In the cases where the relay has a 2-antenna array, the UT-to-RN link is not the performance bottleneck anymore. The RA schemes are once again improved by the use of 2 antennas at the relay and are now better than non cooperative 1×2 MRC systems.

The performance of the 1×2×2 EF non cooperative is similar to the one obtained with the 1×4 MRC systems. Moreover, the 1×2×2 DF is even slightly better than 1×4 MRC in the studied range. This non cooperative system can achieve a diversity order of 4 and an antenna gain of 6dB, while the 1×2×2 DF can achieve a diversity order of 6 and an antenna gain of 6 dB for the case where the data is successfully decoded at RN. For a BER target of 10^{-4}, 1×2×2 RA DF yields a gain of 6 dB and 0.7 dB with respect to non cooperative 1×2 MRC and 1×4 MRC, respectively.

Fig. 5 summarizes results for the scenario when E_b/N_2 and E_b/N_R are equal to E_b/N_0+10 dB (scenario III). It evidences the fact that the proposed 1×2×2 RA schemes are clearly better than the 1×1×2 RA ones, clearly showing the benefit of using a 2-antenna array at the RN. It also confirms that RA DF outperforms RA EF for low E_b/N_0 values and the opposite situation occurs as E_b/N_0 increases. For a BER target of 10^{-4}, 1×2×2 RA schemes offer a gain of about 4 dB with respect to non cooperative 1×4 MRC.

2) Downlink Relay-Assisted Schemes

To assess the RA schemes in DL communication, we also consider the different scenarios presented in Table III.

Fig. 6 shows performance results for RA and non-cooperative schemes for scenario I. From this figure we can see that both RA schemes have a performance that is between non cooperative 2×1 and 2×2 MRC SFBC systems. For *L*=1, RA EF is preferable, whereas for *L*=2, RA DF is better and also outperforms 2×1 MRC SFBC. The explanation for the DF improvement relates to its high sensitivity to the cooperative link. As its reliability improves (as happens when the BS-to-RN link becomes MIMO 2×2 instead of MISO 2×1), the hard-decoding process that takes place at the relay is more effective and less likely to produce decoding errors, which will be regarded as interference noise at the UT side. RA DF with *L*=2 yields a performance improvement greater than 6 dB with respect to SISO, for a BER target of 10^{-4}.

In Fig. 7 we show the schemes performance under the definitions of scenario II. The choice of this scenario for downlink derives from the fact that, in most real situations, the cooperative link has higher transmission quality conditions than the direct link. From this figure we can observe that the performance of both RA schemes is improved in comparison with the previous scenario and that all schemes also outperform non cooperative 2×1 MRC SFBC. This is due to the fact that in the case that the links between BS and RNs are highly reliable, most information is successfully detected at the RN.

We also observe that cooperative systems perform almost the same. Note that in the RN-to-UT link, where most of the errors occur, the schemes behave as a SISO for 1 relay scheme and as a MISO for 2-antenna relay scheme, which have similar performances for low SNRs.

It is interesting to observe that the cooperative schemes outperform the co-located MIMO system, already expected for this scenario.

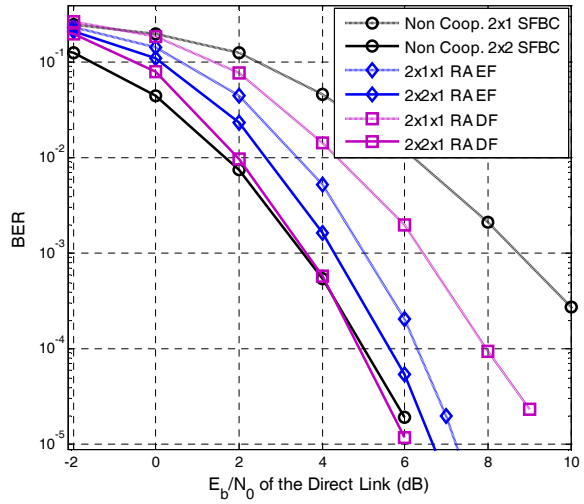


Figure 6. Performance comparison of single and multiple antenna RA schemes for the downlink: when $E_b/N_0 = E_b/N_R = E_2/N_0$ (scenario I).

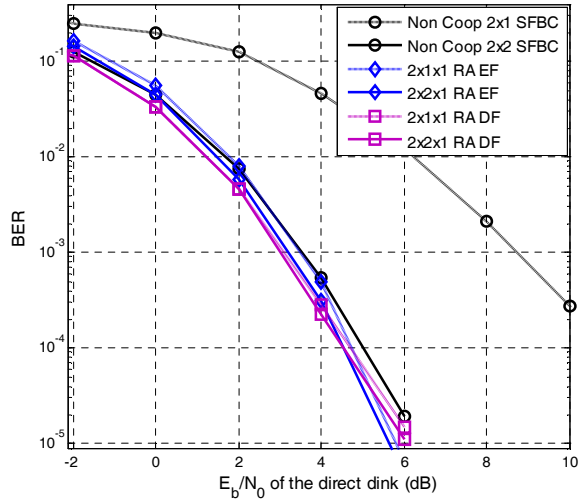


Figure 7. Performance comparison of single and multiple antenna RA schemes for the downlink: when $E_b/N_R = E_b/N_0 + 10dB$ (scenario II).

Despite almost all information being successfully detected at the relay node, we particularly note that in the reference MIMO case the channels of the two receiver antennas are strongly correlated, while for the relay based systems the channels between RN-to-UT and BS-to-UT links are uncorrelated, increasing the diversity order and thus the overall system performance. Note that for the downlink scenario uncorrelated channels between both receiver and transmitter antennas are not assumed.

Further performance improvements are verified when the whole cooperative link is now 10 dB better than the direct link, as is illustrated in Fig. 8, where the results for scenario III are summed up.

In contrast with Fig. 7, Fig. 8 exhibits a slight performance improvement of RA DF with respect to RA EF. This is due to the fact that DF is decoding properly and

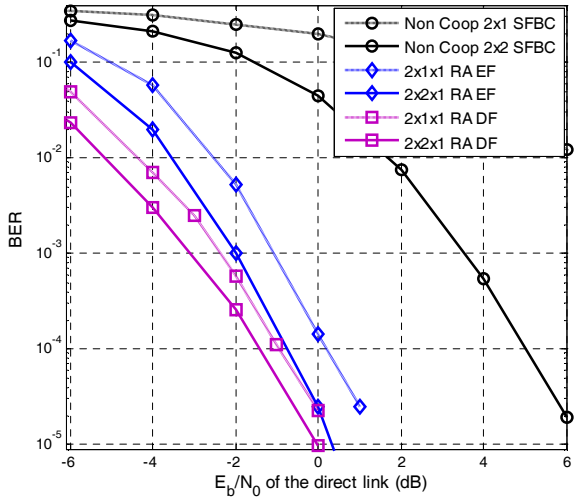


Figure 8. Performance comparison of single and multiple antenna RA schemes for the downlink: when $E_b/N_R = E_b/N_0 + 10dB$ and $E_b/N_2 = E_b/N_0 + 10dB$ (scenario III).

effectively eliminating the RN thermal noise, whereas the EF scheme is being limited by the impact of that noise. We also observe that, due to the antennas correlation and the high quality of the cooperative links (BS-to-RN and RN-to-UT are MIMO 2×L and L×1 channels, respectively), the gains of RA systems with L=2 with respect to L=1 are smaller in scenario III than in scenario I.

The aforementioned results suggest that the RA schemes are severely limited by the relative quality of the cooperative links, specially the BS-to-RN link. Thus, the success of the cooperative schemes lies upon the choice of the relay in the network. Ideally, it should be chosen so that the BS-to-RN link is highly reliable.

V. Conclusion

We proposed and evaluated single and multiple antenna relay-assisted schemes designed for both the UL and DL OFDM based systems. For each configuration, two types of relay-assisted protocols were analyzed: equalize and forward and decode and forward. These schemes were evaluated under realistic scenarios based on WIMAX specifications and compared against the non-cooperative/co-located SISO, MISO, SIMO and MIMO systems.

Concerning UL systems, results have shown that all the proposed relay-assisted schemes have better performances than the non-cooperative ones in the studied scenarios. Furthermore, it was shown that RA DF schemes outperform RA EF when E_b/N_0 is low and the quality of the UT-to-RN link is of good quality. Otherwise, RA EF based relays are preferable.

For DL systems, results have also shown that all the proposed relay-assisted schemes perform better than the non-cooperative ones. In scenarios where all the links have approximately the same quality the RA EF outperforms the RA DF. However, when the BS-to-RN link has good quality is preferable to implement RA DF instead. In short, the UL

and DL results have shown that it is possible to achieve dramatic improvements in systems performance if the proposed cooperative schemes are implemented, especially if the relays are equipped with two-antenna arrays.

It is clear from the presented results that the proposed cooperative schemes can be used to increase the coverage and provide fairness, especially in scenarios where the quality of the direct link is poor, as in urban environments cluttered with buildings. Also, it is crucial to select the best terminal to cooperate, i.e., the one with the higher source-relay link quality in order to achieve better performances. Thus, efficient algorithms to select the best terminal are absolutely fundamental in practical cellular systems.

ACKNOWLEDGMENT

The authors wish to acknowledge the support of the European project Enhanced Wireless Communication Systems Employing Cooperative Diversity – CODIV, FP7/ICT/2007/215477, Portuguese Cooperative and Antenna Diversity for Broadband Wireless Networks – CADWIN, PTDC/EEA-TEL/099241/2008 and Portuguese Foundation for Science and Technology (FCT) with a grant for the second author.

REFERENCES

[1] H. Lima, A. Moço, A. Silva, and A. Gameiro, "Performance assessment of single and multiple antenna relays for the uplink OFDM systems", in *Proc. of IARIA International Conf. on Networking and Services (ICNS)*, Porto, Portugal, Sept. 2009.

[2] R. Laroia, S. Uppala, and J. Li, "Designing a mobile broadband wireless access network", *IEEE Signal Processing Magazine*, Vol. 21, No. 5, Sept. 2004, pp. 20-28.

[3] H. Liu and G. Li, *OFDM-based broadband wireless networks*, John Wiley & Sons, Inc., 2005.

[4] F. H. P. Fitzek and M.D. Katz, *Cooperation in wireless networks: principles and applications*, Springer, 2006.

[5] CODIV- Enhanced Wireless Communication Systems Employing Cooperative Diversity project website. [Online]. Available: https://www.ict-codiv.eu.

[6] G. J. Foschini and M. J. Gans, "On limits of wireless communications in a fading environment when using multiple antennas", *Wireless Personal Communications Magazine*, Vol. 6, No. 3, Mar. 1998, pp. 311-335.

[7] K. J. Ray Liu, Ahmed K. Sadek, Weifeng Su, and Andres Kwasinski, *Cooperative communications and networking*, Cambridge University Press, New York, 2009.

[8] E. C. van der Meulen, "Three-terminal communication channels", *Advances in Applied Probability*, Vol. 3, No. 1, 1971, pp. 120-154.

[9] T. M. Cover and A. A. E. Gamal, "Capacity theorems for the relay channel", *IEEE Transactions on Information Theory*, Vol. 25, No. 5, Sept. 1979, pp. 572-584.

[10] A. Sendonaris, E. Erkip, and B. Aazhang, "User cooperation diversity—Part I: System description," *IEEE Trans. Commun*., Vol. 51, No. 11, Nov. 2003, pp. 1927–1938.

[11] A. Sendonaris, E. Erkip, and B. Aazhang, "User cooperation diversity—Part II: Implementations aspects and performance analyses," *IEEE Trans. Commun*., Vol. 51,No. 11, Nov. 2003, pp. 1939–1948.

[12] J. N. Laneman, D. N. C. Tse, and G. W. Wornell, "Cooperative diversity in wireless networks: Efficient protocols and outage behaviour," *IEEE Transactions on Information Theory*, Vol. 50, No. 12, Dec.2004, pp. 3062-3080.

[13] M. Dohler, *Virtual antenna arrays*, Ph.D. Thesis, King's College London, London, UK, Nov. 2003.

[14] G. S. Rajan and B. S. Rajan, "Distributed Space-Time Codes for cooperative networks with partial CSI", in *Proc. of Wireless Communications and Networking Conference*, Hong Kong, Mar. 2007.

[15] O. S. Shin, A. M. Chan, H. T. Kung, and V. Tarokh, "Design of an OFDM cooperative space-time diversity system", *IEEE Transactions on Vehicular Technology*, Vol. 56, No. 4, July 2007, pp. 2203-2215.

[16] M. Hayes, S. K. Kassim, J. A. Chambers, and M. D. Macleod, "Exploitation of Quasi-Orthogonal Space Time Block Codes in virtual antenna arrays: Part I-Theoretical capacity and throughput gains", in *Proc. of IEEE Vehicular Technology Conference Spring*, Marinara Bay, Singapore, May 2008.

[17] M. Yuksel and E. Erkip, "Diversity-multiplexing tradeoff in multiple-antenna relay systems", in *Proc. of International Symposium on Information Theory, Seattle*, USA, July 2006, pp. 1154-1158.

[18] J. Zhao, M. Kuhn, A. Wittneben, and G. Bauch, "Cooperative transmission schemes for decode-and-forward relaying", in *Proc. of IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, Athens, Greece, Sept. 2007.

[19] H. Muhaidat and M. Uysal, "Cooperative diversity with multiple-antenna nodes in fading relay channels", *IEEE Transactions on Wireless Communications*, Vol. 7, No. 8, Aug. 2008, pp. 3036-3046.

[20] B. K. Chalise and L. Vandendorpe, "MIMO relay design for multipoint-to-multipoint communications with imperfect channel state information," *IEEE Trans. Signal Process*., Vol. 57, July 2009, pp. 2785–2796.

[21] S. Teodoro, A. Silva, J. M. Gil, and A. Gameiro, "Virtual MIMO schemes for downlink space-frequency coding OFDM systems", in *Proceedings of Personnal, Indoor and Mobile Radio Communications*, Tokyo, Japan, Sept. 2009.

[22] A. Moço, S. Teodoro, A. Silva, and A. Gameiro, "Performance evaluation of virtual MIMO schemes for the UL OFDMA based systems", in *Proc. of IARIA International Conference on Wireless and Mobile Communications*, Athens, Greece, Aug. 2008.

[23] S. Kaiser, "Space frequency block coding and code division multiplexing in OFDM systems", in *Proc. IEEE Global Telecommunications Conferenc*, San Francisco, USA, Dec. 2003.

[24] IEEE 802.16, Part 16: Air Interface for Broadban Wireless Access Systems, p802.16Rev2/D2, Dec. 2007.

[25] 3GPP TS 36.201 V8.1.0, 3rd Generation Partnership Project; Technical Specification Group Radio Access Netwotwork; Evolved Universal Terrestrial Radio Access (E-UTRA);LTE Physical Layer - General Description, Nov. 2007.

[26] C. K. Lo, S. Vishwanath, and R. Heath, Jr. "Relay subset selection in wireless networks using partial decode-and-forward transmission", *IEEE Transaction on Vehicular Tecnology*, Vol. 58, No. 2, Feb. 2009, pp. 697-704.

Flow Time Prediction for a Single-Server Order Picking Workstation using Aggregate Process Times*

R. Andriansyah, L. F. P. Etman, and J. E. Rooda
Systems Engineering Group
Eindhoven University of Technology
Den Dolech 2, 5600 MB Eindhoven, The Netherlands
Email: r.andriansyah@tue.nl; l.f.p.etman@tue.nl; j.e.rooda@tue.nl

***Abstract*—In this paper we propose a simulation modeling approach based on aggregate process times for the performance analysis of order picking workstations in automated warehouses with first-in-first-out processing of orders. The aggregate process time distribution is calculated from tote arrival and departure times. We refer to the aggregate process time as the effective process time. An aggregate model uses the effective process time distributions as input to predict tote and order flow times. Results from experimental settings show that the aggregate model accurately predicts the mean and variability of tote and order flow times. As a case study, we develop an aggregate model to predict flow times for a real, operating warehouse. The resulting flow time predictions give satisfactory accuracy for both tote and order flow times. Meaningful insights are obtained for improving the performance of the warehouse.**

***Keywords*—Order picking; Polling system; Simulation; Aggregation; Performance analysis**

I. Introduction

Order picking has been identified as the most expensive process in a warehouse. It is estimated that 55% of the total warehouse operating expenses is caused by order picking only [2]. Even in automated warehouses, order picking remains a very capital intensive operation [3]. This fact alone highlights the importance of performance analysis and improvement of order picking systems.

In this paper we consider a product-to-picker, end-of-aisle, unit-load order picking system [4], with *totes* as unit-loads. An AS/RS (Automated Storage/Retrieval System) is used to retrieve product totes from a storage area. The totes are then transported using conveyors to an order picking workstation. At the workstation, a picker takes the required amount of products from the totes. Afterwards, totes with remaining items are stored back by the AS/RS.

For automated warehouses, the existing literature mostly focuses on the AS/RS [5]. Koh, Kwon and Kim [6] developed an analytical model for a miniload AS/RS with a horse-shoe style buffer. Park, Foley, and Frazelle [7] analyzed the performance of a miniload AS/RS with two-class storage. Bozer and Cho [8] derived closed-form analytical results to evaluate the performance of AS/RS under stochastic demand. Hur et al. [9] developed an $M/G/1$ queueing model to estimate the performance of a unit-load AS/RS. Other references on performance analysis of similar AS/RS are available in the recent review by Roodbergen and Vis [4].

An order picking workstation can be regarded as a special type of *polling system*, where a number of queues is attended by a single server in a certain order. Several analytical queueing models of such systems exist, see e.g., references [10] [11] [12] [13]. Typically these methods consider gated or exhaustive service policies, or a combination of the two. Another variation is the $k-$limited polling system where the server continues to work at a queue until either a predefined number of customers k is served or until the queue becomes empty (see e.g., [14]). These polling variants, however, do not fully correspond to the order picking workstation we consider. In our case, a picker always completes an order before starting to pick items for the next order. Hence, a picker may be idle at one queue (i.e., waiting for the remaining totes to arrive) while other queues are filled with totes.

We present a simulation model for quantifying the mean and variability of tote and order flow times for this type of order picking workstation. A key aspect of our model is that we do not model in detail the various outages that contribute to the flow time performance. That is, the human pickers, picking faults, setup times, picking equipment failures, etc. are not modeled in every detail. In practice, these are typically difficult to quantify [15]. Instead, we model them by means of an aggregate process time distribution. The idea is that we want to obtain the aggregate process time distribution from tote arrival and departure events of the order picking workstation in operation. Here we start from the concept of EPT (Effective Process Time) by Hopp and Spearman [16] and the concept of measuring EPT distribution from arrival and departure events [17], using a sample path equation [18].

Gu et al. [19] concluded in their recent literature review that studies describing validated or applied design models, and practical case studies will give important contributions to warehouse research in the future. This paper includes an extensive warehouse case study based on data obtained from a real, operating warehouse.

The remainder of this paper is organized as follows. Section II describes the order picking workstation. Section III

*This article is an extended version of [1].

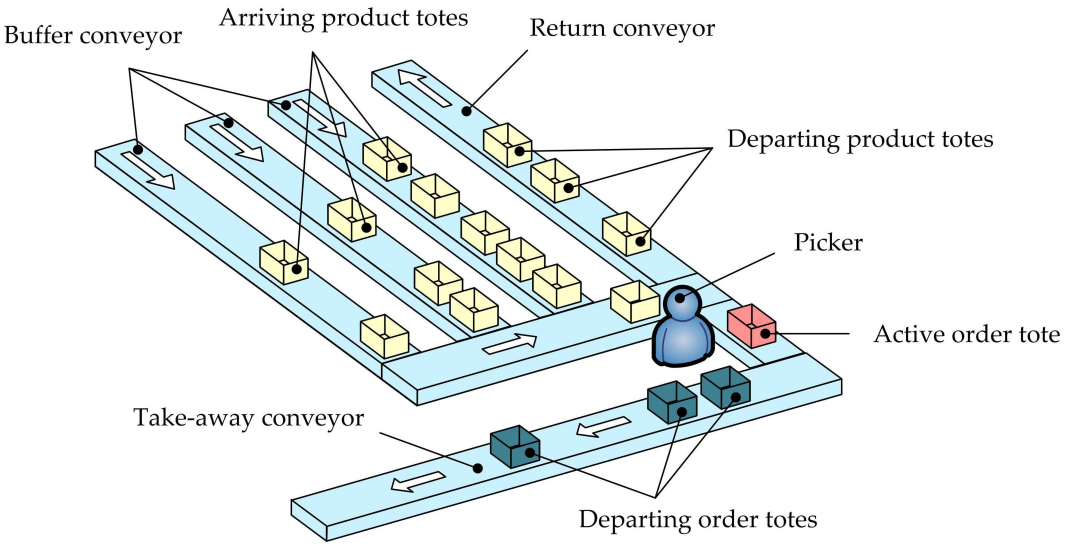


Fig. 1. Order picking workstation.

describes the simulation model. Section IV elaborates the aggregation method and the EPT measurement method. Section V discusses a number of validation experiments. Section VI provides a case study to see the performance of the proposed method in a realistic setting. Section VII concludes the paper.

II. System description

Figure 1 shows the layout of the order picking workstation under study. This system can be classified as a *product-to-picker* system [20]. Pickers work to fulfill *orders*. An order consists of a number of *order lines*. The number of order lines in an order is referred to as *order size*. Order sizes may vary significantly. Internet orders, for example, may have a small order size while orders from supermarkets may have a very large order size. An order line represents the required number of *items* from a certain SKU (Stock Keeping Unit). A *product tote* contains items of the same SKU type.

At the order picking workstation, arriving product totes form queues on buffer conveyors. Once the picker and the required product tote are available, the product tote will be removed from the buffer conveyor and transported to the pick position where the picker stands. The picker then picks a number of required items from the product tote and puts them in an *order tote*. The picker works on one order at a time until all lines of the order have been picked and the order is said to be finished. When an order is finished, the picker moves the finished order tote to a take-away conveyor that brings the order tote to a consolidation area. It is possible that more than one order tote is needed to fulfill an order due to the number of required items in that order or the size of the items being picked. In that case we assume the order has been split into suborders accordingly, which means that in the remainder of this paper we assume that every order corresponds to a single order tote. If a product tote is not yet empty after item-picking, the tote will be returned to the storage area using a *return conveyor*.

Order picking workstations have a typical characteristic that distinguishes them from ordinary manufacturing workstations. An order picking workstation receives a number of product totes for different orders. In the type of system that we consider here, the picker can only pick items from the product totes that belong to the order currently being processed, known as the *active order*. As such, only product totes required to fulfill the active order are sent in a FIFO (First-In-First-Out) sequence from the buffer conveyor to the pick position, while all other product totes wait in the queue. If there are no product totes in the queue that belong to the active order, then the picker will be idle although totes for other orders may be present. In this system, totes of three orders may arrive simultaneously at the buffer conveyor. They are sorted such that the picker always has access to the totes of the active order.

Once all order lines of the active order are finished, the next order is processed following a FIFO sequence. Subsequently, product totes for this new order are sent to the pick position. Note that only one active order is allowed in the system under consideration as shown in Figure 1. In other order picking systems it might be possible that more than one active order is processed, allowing the picker to pick items for multiple orders simultaneously. We do not consider these here.

Two performance measures are particularly of interest for this order picking workstation, namely the tote and order flow times. Tote flow time is defined as the total time spent by a tote at the order picking workstation, which starts when a tote arrives at the workstation and ends when it departs the workstation. Order flow time is defined as the time required to complete an order, which starts when the first product tote of an order arrives at the workstation and ends when the last product tote of the order has left the workstation. A complete order means that all items required for the order have been picked into the order tote.

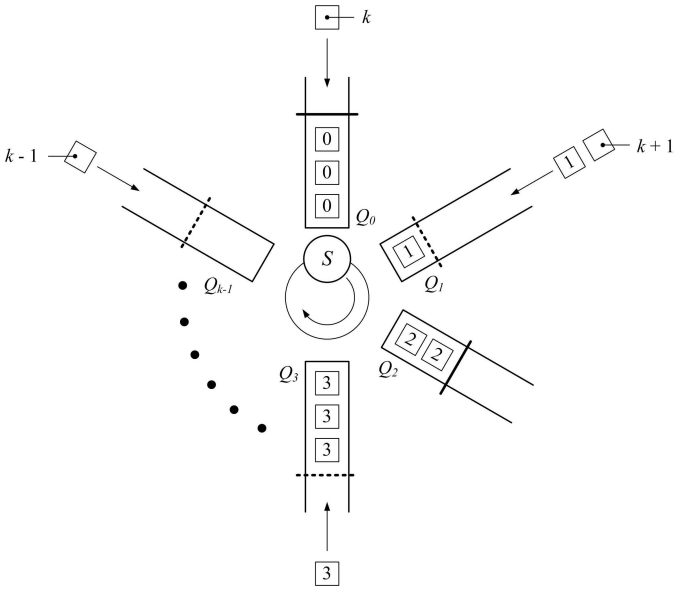


Fig. 2. Order picking workstation as a polling system.

III. Simulation Model Description

Figure 2 shows the simulation model representation of the order picking workstation. The workstation is modeled as a polling system with a single server S and k infinite queues. The queues are denoted by $Q_i,\ i = 0, 1, 2, ..., k-1$. The number of queues k indicates the maximal number of orders for which product totes *simultaneously* arrive in the workstation; that is, order IDs of arriving product totes may be shuffled. Totes arrive with a rate of λ. Each tote has an *id* that denotes the order ID to which the tote belongs. All arriving totes with the same *id* are put into the same queue.

When the first tote of a new order enters a queue, a *gate* is immediately set for that queue. The gate indicates the number of totes required for the order, which equals to the order length. The gate is kept open until all totes for the corresponding order have arrived at the queue. Once the last tote of the order has arrived, the gate is closed. In Figure 2 an open gate is represented by a dotted line and a closed gate is represented by a solid line in the queue.

A new order is created each time the gate for another order has been closed. The variable *id* is increased by one and the totes arriving for new order are put in queue Q_i where $i =$ *id* modulo k. In Figure 2, for example, the gates of orders 0 and 2 have been closed and thus two new orders can be started. If the number of queues $k = 5$ (as in Figure 2), then the new orders 5 and 6 are put into queues Q_0 and Q_1.

The server attends the queues in a cyclic direction, causing the orders to be served in FIFO sequence. The server will switch to the next queue only if the gate for the current queue has been closed and all totes in front of the gate have been served. If the server is done processing all totes in front of the gate but the gate is still open, then the server will become idle at the queue. In this case, the server waits until the remaining totes for the queue arrive.

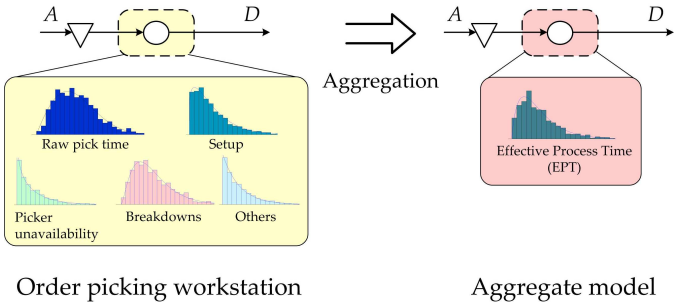


Fig. 3. Aggregation method.

IV. Aggregation Method and EPT Measurement

The process time used in this paper represents an aggregation of all components that contribute to the processing time at the order picking workstation. We refer to the aggregate process time as the effective process time or EPT for short (see [16]). Jacobs et al. [17] presented an algorithm to compute EPT realizations directly from arrival and departure events for infinitely buffered workstations with single-lot processing. Subsequent studies using this concept have been conducted for equipments in manufacturing lines with blocking [18], equipments in assembly lines [21], and batch equipments [22]. The former two studies employed sample path equations to calculate EPT realizations. We will do so here as well.

An order picking workstation is characterized by several process time components (see Figure 3). At the core of the process is the time required for picking items, which is referred to as the raw pick time. In addition to the raw pick time, pickers may require some setup time (change-over time) between processing of orders. Conveyor systems may break down, causing unavoidable delays. Picker availability is also an issue since it is likely that a picker is sometimes not present at the workstation. In our *aggregate model* (see Figure 3) these components are aggregated into a single EPT distribution. The idea is then to reconstruct the EPT distribution directly from tote arrival and departure times registered at the operating order picking workstation under consideration, with the obvious advantage that one does not need to quantify each component contributing to the process time.

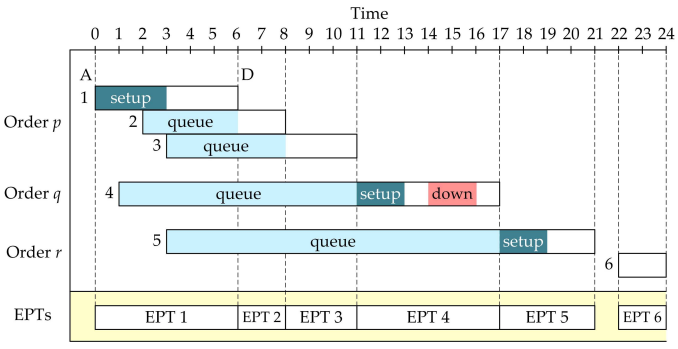


Fig. 4. Gantt chart example.

An EPT realization is calculated for each departing tote, which equals the total amount of time a tote *claims* capacity even if the tote is not yet in physical process. When EPT realizations for all departing totes have been obtained, an EPT distribution with mean t_e and squared coefficient of variation c_e^2 is created. We typically assume a gamma distribution, but other distributions may equally well be used. A gamma distribution is relatively easy to construct since the scale and shape parameters are readily obtainable from the mean and variance of the empirical EPT realizations.

Figure 4 shows an example of arrivals and departures of six totes at an order picking workstation. Totes 1, 2, and 3 belong to order p, Tote 4 belongs to order q, and Totes 5 and 6 belong to order r. An arrival A_i occurs at the moment a product tote i enters the buffer conveyor of the order picking workstation. A departure D_i occurs when item picking has been finished and the respective product tote i is moved to the return conveyor or to the take-away conveyor (see Figure 1).

EPT realizations are calculated using the following sample path equation:

$$\text{EPT}_i = D_i - \max\{A_i, D_{i-1}\} \quad (1)$$

here D_i denotes the time epoch of i^{th} departing tote. A_i denotes the arrival epoch of the corresponding i^{th} departing tote. The bottom part of Figure 4 illustrates how EPT realizations are obtained using Equation (1).

The first tote of an order typically may have a different EPT distribution compared to the other totes in an order. The reason is that each time a picker starts working on a new order, a number of extra activities are performed. These activities include moving the active order tote to the take away conveyor, scanning the barcode of a new order tote to be used for the next order, and placing the order tote at the pick position. Furthermore, pickers may leave their workstations for a break after finishing an order. These are setup activities, which usually only take place in preparation of picking items from the first product tote of a new order. Consequently, EPTs of the first tote typically include a setup time whereas the remaining totes do not. Therefore, we sort EPTs into EPTs for the first totes and EPTs for the remaining totes. So in our aggregate model we will use two distribution functions, accordingly, which we refer to as the 1^{st} *tote difference* EPT approach.

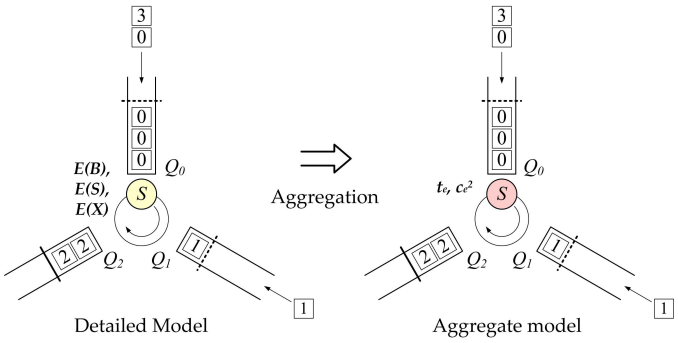


Fig. 5. EPT-based aggregation.

V. Validation experiments

This section presents simulation experiments to validate the proposed aggregate modeling method. First, we create a *detailed model* to be used as a test case representing the "real-life" operating order picking workstation. We simulate this detailed model at a certain utilization level (referred to as the *training point*) to generate tote arrival and departure events. Subsequently, these events are used as input for the sample path equation to calculate EPT realizations. Two gamma EPT distributions are constructed namely for the first totes and the remaining totes. Next, we simulate the detailed model at various utilization levels to measure the mean and variability of tote and order flow times.

In the aggregate model, we use the EPT distributions to sample the aggregate process time for totes that are being processed. The aggregate model is then simulated at the same utilization levels as the detailed model. We compare the mean and variability of tote and order flow times from the aggregate model with those of the detailed model. In this way, we assess the accuracy of flow time predictions by the aggregate modeling method.

A. Detailed model

The detailed model represents the real system under study, namely the order picking workstation with a number of process time components including raw picking time, setups and disturbances. This system is modeled as a polling system with three queues shown in Figure 5. As such, we assume

TABLE I
Data of order sizes and their frequencies.

Size	Freq.	Size	Freq.	Size	Freq.	Size	Freq.	Size	Freq.
1	331	11	80	21	20	31	11	41	11
2	243	12	67	22	20	32	6	42	10
3	257	13	41	23	12	33	5	43	11
4	181	14	24	24	13	34	12	44	12
5	195	15	34	25	7	35	11	45	7
6	208	16	42	26	10	36	12	46	2
7	147	17	19	27	6	37	7	47	3
8	91	18	14	28	12	38	9	48	5
9	134	19	17	29	20	39	8	49	3
10	90	20	27	30	5	40	6	50	1

that totes for three orders are generated simultaneously to the workstation, each with an exponential rate of $\frac{1}{3}\lambda$. Orders have different sizes as given in Table I.

The server S in Figure 5 represents a picker, which is characterized by the mean values of raw pick time $E(B)$, order tote setup $E(S)$, and other disturbances $E(X)$. The raw pick time is assumed to be gamma distributed with a mean of 17.5 seconds and an SCV (Squared Coefficient of Variation) of 0.8. The SCV is defined as the variance divided by the square of mean raw pick time. An order tote setup is performed each time a picker starts working on a new order. We assume that the order tote setup is uniformly distributed between 10.0 and 15.0 seconds. Other disturbances such as incorrect product tote administration, unreadable barcode on the product tote, distraction from other pickers, etc. occur during item picking. These disturbances are assumed to take place on average every 30 minutes, with a duration of on average 2 minutes. Both times are assumed to be exponentially distributed.

This model has been implemented using the process algebra based simulation language χ (Chi) 1.0 [23]. χ uses a pseudo-random number generator based on Mersenne Twister [24] to generate samples from distributions. But other simulation packages may of course be used as well.

The experimental setup used for the detailed model is as follows. The arrival and departure data are generated in a single simulation run of 1,000,000 totes. To measure the mean and variability of tote and order flow times we perform 30 simulation runs of 300,000 totes and a warm up period of 30,000 totes at utilization levels ranging from 0.30 to 0.95.

B. Measured EPT

To measure EPT realizations we first generate arrival and departure events from the detailed model at a training point of $0.8\delta_{\max}$, where $\delta_{\max}$ is the maximum throughput of the detailed model. Through simulation we obtain $\delta_{\max} = 0.05$ totes per second. Arrival and departure events of 1,000,000 product totes are then generated. Subsequently, EPT realizations are calculated using Equation (1).

We apply the 1st tote difference as explained in Section IV. Two EPT distributions with parameters $t_{e,1} = 31.15$ seconds, $c^2_{e,1} = 0.59$ and $t_{e,2+} = 18.69$ seconds, $c^2_{e,2+} = 1.61$ are obtained for the first and remaining totes of orders, respectively. Suppose now we do not apply the 1st tote difference. That is, we do not distinguish between EPT realizations of the first totes and the remaining totes of orders. Without the 1st tote difference we obtain one EPT distribution with parameters $t_e = 20.08$ seconds and $c^2_{\mathrm{e}} = 1.44$.

Figure 6 shows the CDF (Cumulative Distribution Function) of EPT realizations with and without 1st tote difference. With the 1st tote difference we obtain two significantly different EPT distributions for the first and remaining totes of orders. Without the 1st tote difference, the EPT distribution of all totes is very similar to the EPT distribution of the remaining totes using the 1st tote difference. This is because the number of EPT realizations of remaining totes is significantly larger than the first totes.

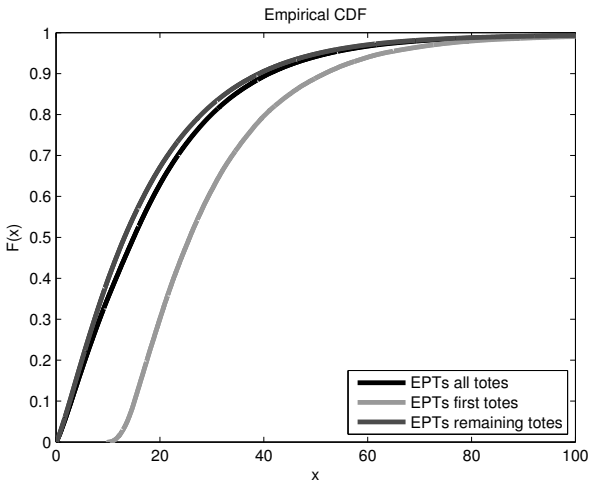


Fig. 6. CDF of EPT realizations.

C. Analytical EPT

The mean EPT for this system can be obtained analytically by applying the EPT formula for nonpreemptive and preemptive outages consecutively [16]. Order tote setup is a *nonpreemptive outage* because the setup only occurs *between* picking. Other disturbances, on the contrary, can be seen as a *preemptive outage* since they occur *during* picking. Since the formula in [16] assumes no distinction between job types, the resulting analytical mean EPT is as if the 1st tote difference is not applied.

Let t_0, σ_0^2, and c_0^2 be the mean raw pick time, its variance, and its SCV, respectively. The order tote setup is characterized by the mean setup time t_{s}, its variance σ_{s}^2 and the number of jobs between setup N_{s} (or the mean order size from Table I). The mean EPT t_{e}, effective variance σ_{e}^2, and effective SCV c_{e}^2 after including the order tote setup (nonpreemptive outage) are [16]:

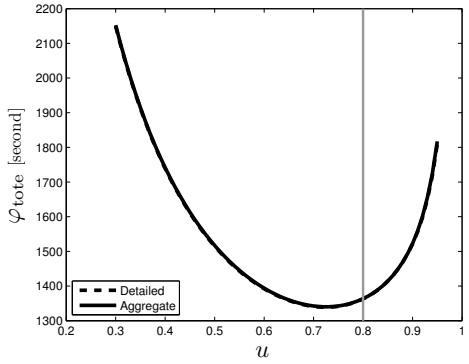
$$t_{\mathrm{e}} = t_0 + \frac{t_{\mathrm{s}}}{N_{\mathrm{s}}},\ \sigma_{\mathrm{e}}^2 = \sigma_0^2 + \frac{\sigma_{\mathrm{s}}^2}{N_{\mathrm{s}}} + \frac{N_{\mathrm{s}}-1}{N_{\mathrm{s}}^2}t_{\mathrm{s}}^2,\ c_{\mathrm{e}}^2 = \frac{\sigma_{\mathrm{e}}^2}{t_{\mathrm{e}}^2} \quad (2)$$

Next we include other disturbances (preemptive outage) in the EPT calculation. t_{e}, σ_{e}^2, and c_{e}^2 obtained previously become t_0, σ_0^2, and c_0^2. Mean EPT t_{e} and effective SCV c_{e}^2 after including the preemptive outage are [16]:

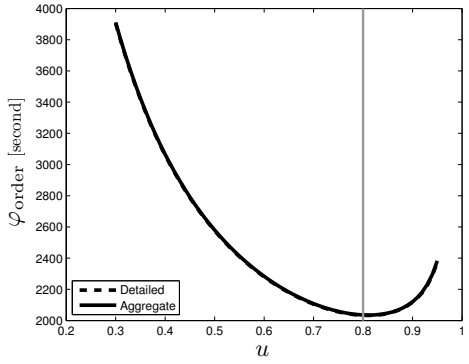
$$A = \frac{m_{\mathrm{f}}}{m_{\mathrm{f}} + m_{\mathrm{r}}},\ t_{\mathrm{e}} = \frac{t_0}{A},\ c_{\mathrm{e}}^2 = c_0^2 + (1 + c_{\mathrm{r}}^2)A(1-A)\frac{m_{\mathrm{r}}}{t_0} \quad (3)$$

where m_{f} is the mean time between two consecutive disturbances, m_{r} is the mean repair time, and c_{r}^2 is the SCV of the repair times.

Applying the above formula with assumptions used in the detailed model, we obtain $t_{\mathrm{e}} = 20.06$ seconds and $c_{\mathrm{e}}^2 = 1.43$. Comparing these values with the measured EPT without 1st tote difference (see Section V-B), we get errors of 0.11% and 0.81% for t_{e} and c_{e}^2, respectively. This result validates our method of measuring EPT realizations from tote arrival and departure events.

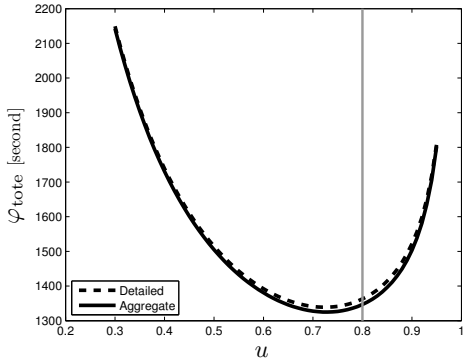


(a) Tote flow time.

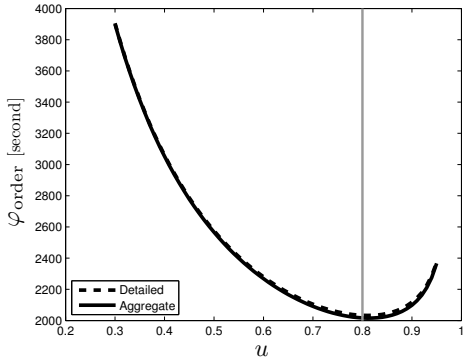


(b) Order flow time.

Fig. 7. Flow time prediction with 1st tote difference.



(a) Tote flow time.



(b) Order flow time.

Fig. 8. Flow time prediction without 1st tote difference.

D. Aggregate model

The aggregate model comprises a single-server with aggregate process times sampled from EPT distributions. With the 1st tote difference, two EPT distributions with means and SCVs $t_{e,1}$, $c^2_{e,1}$, and $t_{e,2+}$, $c^2_{e,2+}$ are used for the first and remaining totes, respectively. Without 1st tote difference, the EPT distribution has mean t_{e} and SCV c^2_{e}. The sampled aggregate process time represents the duration in which the capacity is claimed by a tote.

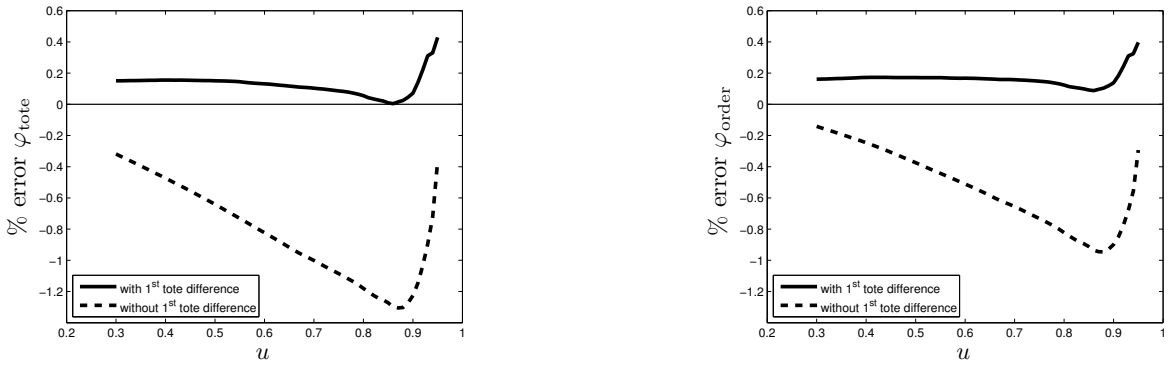
The aggregate model is simulated at the same utilization levels as the detailed model (see Section V-A). At each utilization level, 30 simulation runs of 300,000 totes and a warm up period of 30,000 totes are performed. We evaluate the flow time prediction accuracy of the aggregate model with and without 1st tote difference by comparing the flow times from the aggregate model with that of the detailed model.

E. Flow time prediction

Figure 7 shows the tote and order flow time predictions *with* 1st tote difference. The first tote of an order is assigned with an aggregate process time that is significantly larger than the remaining totes (see the values of $t_{e,1}$ and $t_{e,2+}$ in Section V-B). This imposes a longer flow time for the first totes of orders. Therefore the remaining totes have to wait longer before they are processed. Consequently, the aggregate model correctly predicts both tote and order flow times. Errors for mean and variability of flow time prediction are less than 0.5% and 3.0%, respectively for both tote and order flow times.

Figure 8 shows the tote and order flow time predictions *without* 1st tote difference. Flow time predictions by the aggregate model are consistently lower than the flow times from the detailed model. This observation can be explained as follows. The processing time for all totes in the aggregate model are sampled from an EPT distribution with parameters $t_e = 20.08$ seconds and $c^2_{\text{e}} = 1.44$ (see Section V-B). However, this t_{e} is significantly lower than the mean EPT of the first totes of orders $t_{e,1} = 31.15$ seconds when using 1st tote difference. This causes the aggregate model to underestimate the flow times of the first totes of orders because they are processed much faster in the aggregate model than in the detailed model. The flow times of the remaining totes are affected as well. These totes have shorter waiting time in the buffer and therefore their flow times become lower as well.

Figure 9 compares the percentage error in flow time predictions with and without 1st tote difference. It is clear that the 1st tote difference approach increases the flow time prediction accuracy.



(a) Totes. (b) Orders.

Fig. 9. Percentage error of flow time prediction

F. Effect of order size distribution

We investigate the effect of using different order size distributions on the accuracy of flow time prediction by the aggregate model. Geometric and uniform distributions are used for this purpose. A geometric distribution allows us to model an order pattern with many small orders (e.g., internet orders, slow-moving products) or an order pattern with many large orders (e.g., supermarket orders and fast-moving products). A uniform distribution allows us to model an order pattern with a predefined maximum order length.

The geometric distribution for order size is given by its probability mass function:

$$P\{X = n\} = (1-p)^{(n-1)}p$$

where n is the order size and two values of p are used namely 0.8 and 0.2. With $p = 0.8$ the order size distribution is short-tailed and most orders will have a size of 1 tote. On the contrary, with $p = 0.2$ the order size distribution is long-tailed and most orders will have a size larger than 1 tote.

For the uniform distribution, we set the maximum order size $n_{\max} = 20$. As such, the probability that an order size takes any value between 1 and 20 is fixed at 0.05.

Each order size distribution is used in a simulation experiment with 30 replications of 300,000 totes and a warm up period of 30,000 totes. Again, we evaluate flow time predictions from two aggregate models namely with and without 1st tote difference. In each simulation replication, the mean and variability of individual flow times are calculated as φ and c^2_φ, respectively. We take the average of all 30 replications to get the mean values of both performance measures $\bar{\varphi}$ and $\bar{c}^2_\varphi$. Subsequently, a two sample t-test at significance level $\alpha = 0.05$ is conducted to compare the mean flow time from the detailed model $\bar{\varphi}_{\text{D}}$ with that of the aggregate model $\bar{\varphi}_{\text{A}}$. The following two-sided hypothesis is tested.

$$H_0 : \bar{\varphi}_{\text{D}} = \bar{\varphi}_{\text{A}}$$
$$H_1 : \bar{\varphi}_{\text{D}} \neq \bar{\varphi}_{\text{A}}$$

The results are shown in Tables II and III. Prediction errors (in %) are indicated in the columns labeled with $\%\ e$.

The aggregate model with 1st tote difference predicts the mean flow times of both tote and order significantly better than the one without 1st tote difference. This can be seen from the resulting p-value of the t-test. Without 1st tote difference, the p-value is significant at some utilization levels ($p < 0.05$). At those values we reject H_0 and conclude that the mean flow times from the detailed and aggregate model are different. However, all p-values are not significant for the aggregate model with 1st tote difference. Hence, we cannot reject H_0 and accept that the mean flow time of the detailed model is similar to the mean flow time predicted by the aggregate model.

The errors for flow time SCV are larger for the order size distribution that has high probability of small orders (see columns $\%\ e\ \bar{c}^2_{\varphi_{\text{A}}}$ and $\%\ e\ \bar{c}^2_{\varphi_{\text{A1}}}$ in Tables II and III). For this type of order size distribution, setups between orders are performed more frequently. The error occurs because the gamma distributed EPTs do not correspond fully to the setup time, which is a convolution of a uniform and a gamma distribution. Consequently the errors of flow time variability increase as the EPT distribution is sampled more frequently. In this case, a more detailed fit for the EPT distribution of the first totes is required. We refer to [25] for alternative EPT distribution fits. However, if the probability of having small orders is low, then using a gamma distribution is sufficient.

For geometrically distributed order size with $p = 0.8$, the errors for mean flow time prediction without the 1st tote difference $\%\ e\ \bar{\varphi}_{\text{A}}$ are all positive. That is, the predicted flow times from the aggregate model consistently overestimate the real flow times. This can be explained as follows. Recall that without 1st tote difference all EPT realizations are collected into a single bucket. In the case of geometric distribution with $p = 0.8$, most orders have a size of 1 tote. Therefore, most EPT realizations are high because EPTs include setup time for orders with size of 1. The resulting EPT distribution has a high mean EPT t_{e}. Since only one EPT distribution is used for sampling the aggregate process time, totes that do not require setup time (remaining totes of an order) also have high aggregate process times. This causes extra waiting for totes in the buffer and consequently higher flow times.

TABLE II
TOTE FLOW TIME (IN SECONDS) AND ITS VARIABILITY.

u	Detailed (D)		Aggregate without 1st tote difference (A)					Aggregate with 1st tote difference (A1)				
	$\bar{\varphi}_{\mathrm{D}}$	$\bar{c}^2_{\varphi_{\mathrm{D}}}$	$\bar{\varphi}_{\mathrm{A}}$	$\bar{c}^2_{\varphi_{\mathrm{A}}}$	% e $\bar{\varphi}_{\mathrm{A}}$	% e $\bar{c}^2_{\varphi_{\mathrm{A}}}$	p-value	$\bar{\varphi}_{\mathrm{A1}}$	$\bar{c}^2_{\varphi_{\mathrm{A1}}}$	% e $\bar{\varphi}_{\mathrm{A1}}$	% e $\bar{c}^2_{\varphi_{\mathrm{A1}}}$	p-value
Geometric $p = 0.8$												
0.3	200.47	2.35	203.47	2.30	1.50	−2.32	**0.00**	200.42	2.34	−0.02	−0.73	0.92
0.4	179.69	1.81	182.82	1.76	1.74	−3.06	**0.00**	179.62	1.79	−0.04	−1.39	0.86
0.5	174.53	1.38	177.71	1.33	1.82	−4.09	**0.00**	174.37	1.35	−0.09	−2.54	0.67
0.6	180.91	1.05	184.06	0.99	1.74	−5.69	**0.00**	180.68	1.00	−0.12	−4.49	0.58
0.7	200.80	0.80	203.87	0.73	1.53	−8.07	**0.00**	200.57	0.74	−0.11	−7.21	0.63
0.8	245.98	0.63	248.88	0.57	1.18	−10.33	**0.00**	246.08	0.57	0.04	−9.24	0.89
0.9	379.11	0.60	381.60	0.54	0.66	−10.03	0.27	380.24	0.55	0.30	−8.44	0.62
0.95	625.63	0.66	630.62	0.62	0.80	−5.61	0.57	637.12	0.63	1.84	−3.70	0.25
Geometric $p = 0.2$												
0.3	1071.00	1.22	1064.40	1.23	−0.61	0.86	**0.03**	1070.40	1.22	−0.06	0.08	0.83
0.4	875.53	1.07	868.49	1.08	−0.80	1.16	**0.01**	874.89	1.07	−0.07	0.07	0.80
0.5	771.43	0.91	763.87	0.93	−0.98	1.43	**0.00**	770.88	0.91	−0.07	−0.02	0.81
0.6	719.18	0.76	710.85	0.77	−1.16	1.71	**0.00**	718.62	0.76	−0.08	−0.17	0.80
0.7	707.24	0.61	697.83	0.62	−1.33	1.87	**0.00**	706.43	0.60	−0.11	−0.43	0.72
0.8	744.15	0.46	732.88	0.46	−1.52	1.49	**0.00**	742.79	0.45	−0.18	−1.32	0.61
0.9	899.89	0.33	886.05	0.33	−1.54	−0.41	**0.01**	896.99	0.31	−0.32	−4.03	0.56
0.95	1184.60	0.31	1173.30	0.32	−0.96	2.79	0.46	1187.20	0.31	0.22	−0.67	0.87
Uniform $n_{\max} = 20$												
0.3	1161.10	0.90	1157.60	0.90	−0.30	0.22	**0.05**	1163.00	0.90	0.16	−0.21	0.28
0.4	938.48	0.80	934.43	0.80	−0.43	0.29	**0.01**	939.91	0.79	0.15	−0.26	0.31
0.5	815.37	0.69	810.86	0.69	−0.55	0.29	**0.00**	816.49	0.69	0.14	−0.38	0.36
0.6	746.40	0.58	741.42	0.58	−0.67	0.18	**0.00**	747.17	0.58	0.10	−0.59	0.51
0.7	716.53	0.48	710.72	0.48	−0.81	−0.09	**0.00**	716.55	0.47	0.00	−0.94	0.99
0.8	730.08	0.37	722.78	0.37	−1.00	−0.90	**0.00**	728.51	0.36	−0.22	−1.79	0.26
0.9	845.82	0.29	836.96	0.28	−1.05	−4.59	**0.01**	841.58	0.28	−0.50	−5.57	0.21
0.95	1089.00	0.31	1089.40	0.31	0.05	0.40	0.97	1087.90	0.30	−0.10	−3.29	0.94

TABLE III
ORDER FLOW TIME (IN SECONDS) AND ITS VARIABILITY.

u	Detailed (D)		Aggregate without 1st tote difference (A)					Aggregate with 1st tote difference (A1)				
	$\bar{\varphi}_{\mathrm{D}}$	$\bar{c}^2_{\varphi_{\mathrm{D}}}$	$\bar{\varphi}_{\mathrm{A}}$	$\bar{c}^2_{\varphi_{\mathrm{A}}}$	% e $\bar{\varphi}_{\mathrm{A}}$	% e $\bar{c}^2_{\varphi_{\mathrm{A}}}$	p-value	$\bar{\varphi}_{\mathrm{A1}}$	$\bar{c}^2_{\varphi_{\mathrm{A1}}}$	% e $\bar{\varphi}_{\mathrm{A1}}$	% e $\bar{c}^2_{\varphi_{\mathrm{A1}}}$	p-value
Geometric $p = 0.8$												
0.3	272.23	1.65	275.21	1.64	1.09	−0.75	**0.00**	272.16	1.64	−0.02	−0.56	0.91
0.4	233.51	1.33	236.61	1.31	1.33	−1.17	**0.00**	233.42	1.31	−0.04	−1.14	0.84
0.5	217.59	1.05	220.76	1.03	1.46	−1.96	**0.00**	217.41	1.02	−0.08	−2.23	0.68
0.6	216.79	0.81	219.94	0.78	1.45	−3.44	**0.00**	216.55	0.78	−0.11	−4.20	0.59
0.7	231.56	0.63	234.62	0.59	1.32	−6.02	**0.00**	231.32	0.58	−0.10	−7.12	0.64
0.8	272.85	0.51	275.78	0.47	1.07	−8.99	**0.00**	272.98	0.46	0.05	−9.49	0.86
0.9	402.98	0.52	405.54	0.47	0.64	−9.52	0.26	404.19	0.47	0.30	−8.54	0.60
0.95	648.27	0.61	653.30	0.57	0.78	−5.37	0.57	659.89	0.58	1.79	−3.54	0.25
Geometric $p = 0.2$												
0.3	1917.30	0.42	1911.20	0.43	−0.32	1.29	0.12	1917.20	0.42	0.00	0.15	0.99
0.4	1510.30	0.38	1503.50	0.39	−0.45	1.83	**0.03**	1510.00	0.39	−0.02	0.17	0.92
0.5	1279.30	0.34	1271.80	0.35	−0.58	2.34	**0.01**	1278.90	0.34	−0.03	0.07	0.90
0.6	1142.40	0.30	1134.20	0.31	−0.72	2.91	**0.00**	1142.00	0.30	−0.04	−0.11	0.87
0.7	1070.10	0.25	1060.70	0.26	−0.88	3.46	**0.00**	1069.40	0.25	−0.07	−0.44	0.77
0.8	1061.80	0.20	1050.40	0.21	−1.07	3.36	**0.00**	1060.50	0.20	−0.13	−1.62	0.64
0.9	1182.30	0.16	1168.30	0.16	−1.18	0.91	**0.01**	1179.50	0.15	−0.24	−5.15	0.58
0.95	1452.10	0.19	1440.80	0.19	−0.78	4.00	0.46	1454.80	0.18	0.19	−0.86	0.87
Uniform $n_{\max} = 20$												
0.3	2728.70	0.15	2729.70	0.15	0.04	0.39	0.75	2732.20	0.15	0.13	−0.33	0.25
0.4	2106.20	0.14	2106.40	0.14	0.01	0.56	0.94	2108.90	0.14	0.13	−0.46	0.25
0.5	1742.50	0.13	1742.10	0.13	−0.02	0.69	0.84	1744.60	0.13	0.12	−0.69	0.26
0.6	1512.30	0.11	1511.30	0.12	−0.06	0.75	0.58	1513.90	0.11	0.11	−1.03	0.34
0.7	1366.50	0.10	1364.60	0.10	−0.13	0.60	0.26	1367.10	0.10	0.05	−1.65	0.66
0.8	1292.30	0.09	1288.90	0.09	−0.27	−0.27	**0.04**	1291.10	0.08	−0.09	−3.07	0.45
0.9	1339.30	0.09	1334.00	0.08	−0.40	−5.35	0.13	1335.10	0.08	−0.32	−8.11	0.20
0.95	1553.20	0.13	1556.90	0.13	0.23	1.90	0.81	1551.70	0.12	−0.10	−3.76	0.91

VI. Case Study

A case study with data obtained from an operating automated warehouse is used to illustrate the applicability of our method in a real warehouse setting. The warehouse shown in Figure 10 distributes slow-moving products to a number of supermarkets in the Netherlands. Three processing units are present in the warehouse, namely miniloads, a conveyor loop, and order picking workstations. Miniloads provide temporary storage spaces for product totes. The conveyor loop transports product totes from the miniload to the order picking workstations, and the other way around. Three order picking workstations, with a similar structure as shown in Figure 1, are available to process customer orders. We predict the flow time of totes and orders at the order picking workstation using an aggregate simulation model. These flow times exclude the time spent while retrieving the totes from the miniload and the time spent by the totes while traveling on the conveyor loop. That is, the tote and order flow times start when a tote and the first tote of an order arrive at the order picking workstations, respectively.

A. Data processing

The data consist of event logs collected via Programmable Logic Controllers (PLC) from all processing units in the warehouse. From this PLC data we extract tote arrival and departure events at the order picking workstations. EPT realizations are then calculated using Equation 1. Other parameters are also extracted from the event data, including the interarrival times of totes, order lengths, and the order release strategy. These parameters are the input for the aggregate simulation model to predict tote and order flow times. Subsequently, we compare the predicted flow times with the flow times measured from the data. This demonstrates the prediction accuracy of the method when applied to the data from a real, operating warehouse. Figure 11 depicts the flow chart of activities performed in this case study.

Arrival and departure events from three working days are extracted from the PLC data for all three order picking workstations. An event consists of type (arrival or departure), time, order ID, order length, and tote ID. Some recorded events may be inconsistent or extreme outliers; e.g., a tote may have an arrival recorded without a departure, or the other way around. Extreme outliers are present when some exceptionally large delays occur between two events, for instance due to lunch breaks. These breaks occur also when there are some totes still waiting in the buffer. The outliers cause very large values for the EPT of the next required tote waiting to be served, the tote flow times of all totes in the buffer, and the order flow times pertaining to the totes in the buffer. Therefore, we filter the arrival and departure events to exclude inconsistent events and large delays between two events if they are longer than 60 seconds. This threshold has been chosen based on the observation that it is very unlikely that there is no arrival or departure event at all within 60 seconds from the previous event. Only 2.8% of all arrival and departure events are discarded due to data filtering. The remaining arrival and departure events (97.2%) are used to extract EPTs, interarrival times, order lengths, and order release strategy.

The calculated EPTs are sorted based on the 1st tote difference rule into EPTs of the first totes and EPTs of the remaining totes. We observed that many large EPTs occur when not all totes for the active order are present in the buffer as the picker starts picking. That is, the EPTs are smaller when the picker finds all totes for the active order present in the buffer. This observation holds for both the EPTs of the first totes and the EPTs of the remaining totes. As such, we further sort the EPTs based on the *completeness* of totes in the buffer when the picker starts picking a tote. This results

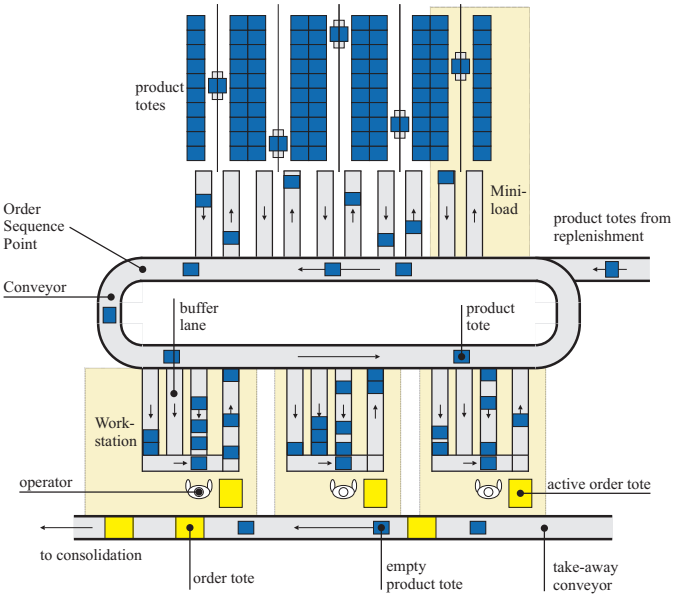


Fig. 10. Layout of an automated warehouse.

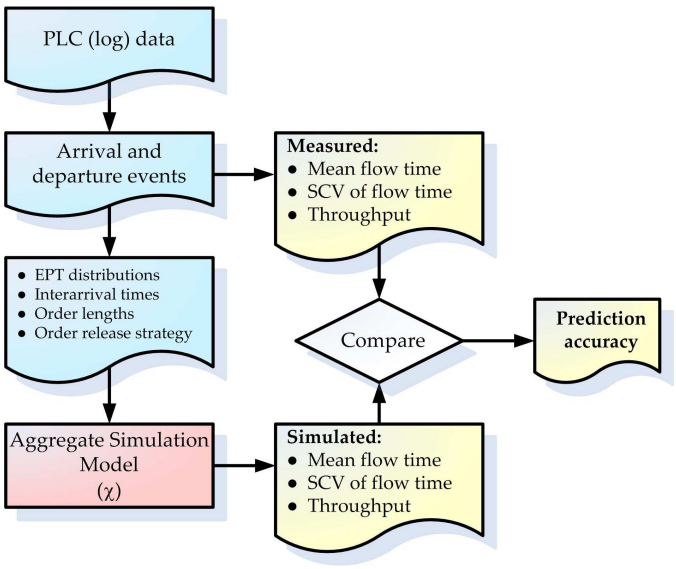


Fig. 11. Case study flow chart.

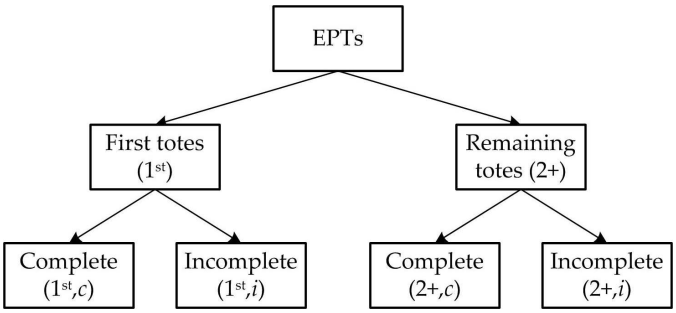


Fig. 12. Sorting of EPTs.

in four types of EPT as shown in Figure 12, namely EPTs first totes complete $(1^{\text{st}}, c)$, EPTs first totes incomplete $(1^{\text{st}}, i)$, EPTs remaining totes complete $(2+, c)$, and EPTs remaining totes incomplete $(2+, i)$. We use shifted gamma distributions to represent all four EPT distributions because the EPTs can never be smaller than a certain value. Hence, using a shifted gamma distribution with the minimum EPT value as offset should produce a better fit than using a gamma distribution as previously done in Section V. Figure 13 visualizes the EPT distributions gathered from workstation 1. The EPTs have been normalized due to data confidentiality. We can see that there are significant differences between all four EPT distributions. It is therefore important to distinguish the EPTs based on the 1^{st} tote difference approach and completeness.

The interarrival distribution can be easily extracted from the tote arrival times at the workstation. However, these interarrival times alone are not sufficient. We also need to reconstruct the order release strategy used in the operating warehouse. Such strategy determines to which order the next arriving tote belongs. Since tote and order flow times are affected by the sequence in which totes arrive, the flow time prediction accuracy also depends on how accurate the order release strategy is modeled in the aggregate simulation model as compared to the reality. The order release strategy is reconstructed also from the arrival and departure events, as follows. We create several so-called *buckets* for each order length. When the first tote a new order arrives at the workstation, the order length of that order is registered. Afterwards, we count the number of totes arriving subsequently for this order until the arrival of the first tote of the next order. The resulting number of totes is then collected into the corresponding bucket based on the order length. Next, an empirical distribution function of the number of totes is created for each bucket. These distributions will be used in the aggregate simulation model to sample the number of totes to be generated for the active order before generating the first tote of the next order.

To assess the quality of the reconstructed order release strategy, we compare the interarrival time of orders from the real data with the simulation. The order interarrival time is defined as the time between the arrival of the first totes of new orders. The result is depicted in Figure 14. The interarrival times have been normalized due to data confidentiality. The reconstructed order release strategy resembles the order release strategy used in the operating warehouse.

B. Flow time prediction

The aggregate simulation model shown in Figure 15 is used to predict the tote and order flow time of the operating warehouse. We model the system as a closed queueing network with two sequential servers namely the miniload and the workstation. The input parameters used in each server are shown in the figure.

The miniload generates new totes for the workstation if there is a space available in the finite buffer of the workstation. For each tote generated, the miniload determines the interarrival time of the tote, the tote ID, the order ID, and the order length. Since totes from multiple orders are generated simultaneously, the order release strategy reconstructed previously is used to determine the order ID indicating the order to which a tote belongs. The order release strategy works as follows. Suppose

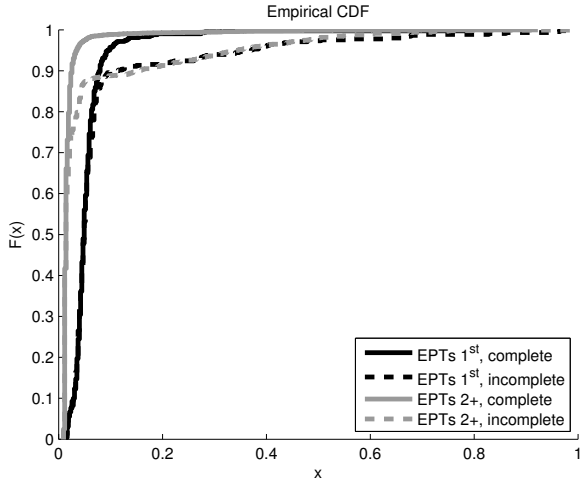


Fig. 13. CDF of four types of EPT from workstation 1.

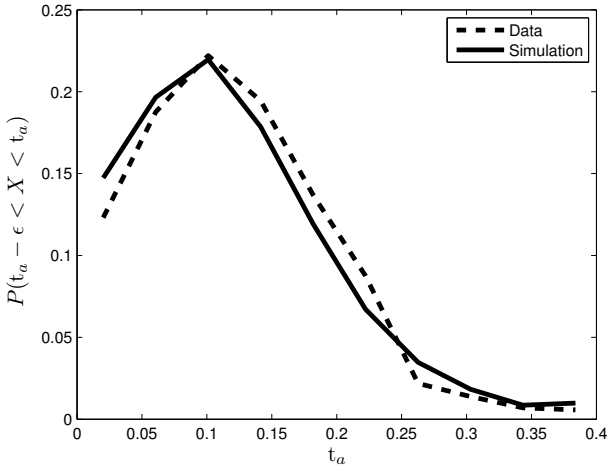


Fig. 14. Distribution of interarrival times of orders.

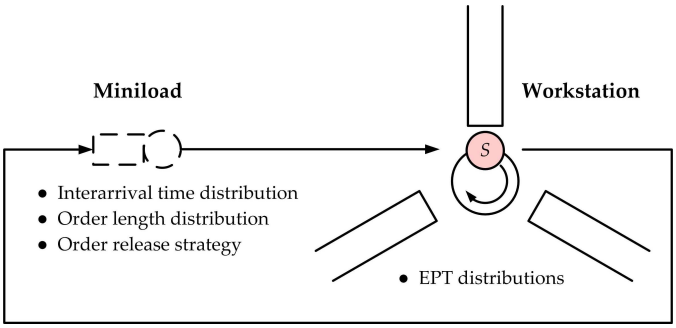


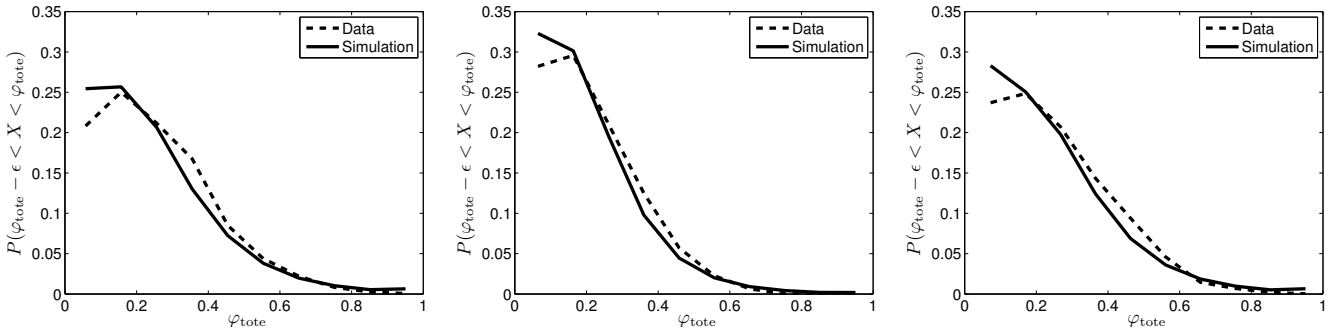
Fig. 15. Aggregate simulation model of the operating warehouse.

the system is empty and the miniload generates the first tote of order X with an order length of 5 totes. At that moment we say that a new order, i.e., order X, has entered the system. We sample from the corresponding bucket, that is the bucket for order length 5, the number of totes N to be generated for order X before generating the next order. Suppose we sample $N = 3$, this means the next three generated totes will belong to order X. The fourth generated tote belongs to a new order, e.g., order Y. Now two orders are in the system and the miniload will decide based on a certain probability whether the tote generated next belongs to order X, order Y, or another new order. If all totes of order X has been generated, then the value N will be sampled again from the correct bucket based on the order length of order Y. The interarrival time of totes are sampled from the interarrival time distribution.

The workstation is modeled as a polling system with a finite buffer. The EPT for each tote being processed is sampled from one of the four EPT distributions, depending on the type of the tote and the completeness of totes for the active order. For example, if the tote is the first tote of an order and not all totes for this order are present in the buffer at the start of picking, then the EPT for this tote will be sampled from the distribution of EPTs 1st incomplete.

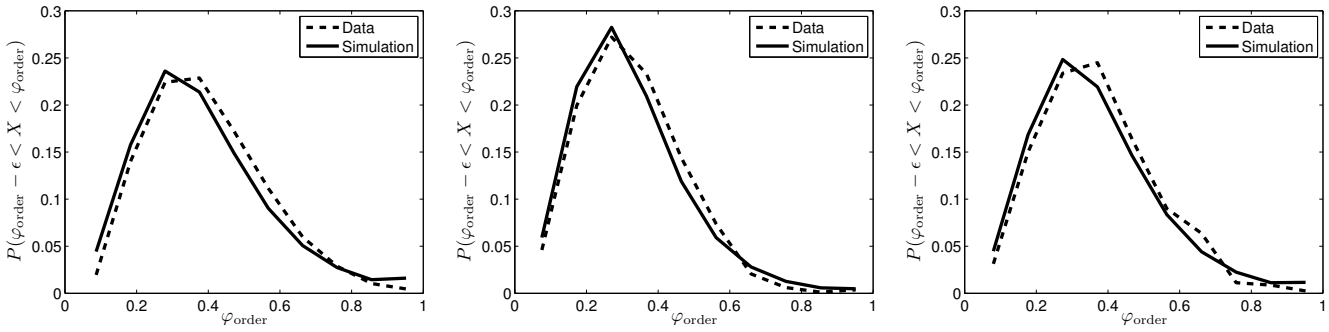
The aggregate simulation model has been run with 50 replications each with a run length of 1,000,000 totes excluding a warm-up period of 300,000 totes for all three workstations. The predicted flow times from the aggregate simulation model are then compared with the real flow times from the data.

The resulting flow time distributions shown in Figures 16 and 17 suggest that the aggregate simulation model accurately predicts the flow time of totes and orders. Indeed, the errors of mean tote and order flow time prediction are less than 5.5%. We also observe that the tote flow time variability is better predicted than orders flow time variability. That is, in Figure 16 the aggregate model consistently overestimates the occurrence of small values of tote flow times.



(a) Workstation 1. (b) Workstation 2. (c) Workstation 3.

Fig. 16. Tote flow time distributions.



(a) Workstation 1. (b) Workstation 2. (c) Workstation 3.

Fig. 17. Order flow time distributions.

C. Discussion

We found that tote and order overtaking exist in the real warehouse. Overtaking of totes occur when the picker picks a product tote that did not arrive the earliest for the active order. Any time loss due to overtaking is not accounted for in the EPT calculation using Equation 1. Overtaking of orders occur if upon completion of an order the picker processes the next order that is not the oldest in the buffer. We measured the average percentage of tote and order overtaking from the three workstations to be 18.3% and 4.6%, respectively. This may explain the underestimation of mean tote and order flow time prediction since overtaking is not considered in the aggregate simulation model.

Another insight from the case study is that pickers hesitate to wait for totes. If totes for the active order are not yet complete in the buffer by the time the picker starts picking, it is very likely that the picker will leave the workstation for a while. Suppose that a tote for the active order arrives shortly after the picker leaves, then the EPT for this tote will include the time when the picker was leaving the workstation. This will cause the EPT for this tote to become very large. To account for this phenomenon, we have sorted EPTs based on the completeness of totes in the buffer. In practice, one may want to improve the order release strategy such that the totes for the active order arrive more frequently than the totes for other orders. One may also consider allowing pickers to work on multiple orders simultaneously, hence reducing the likeliness that the picker becomes idle waiting for the required totes.

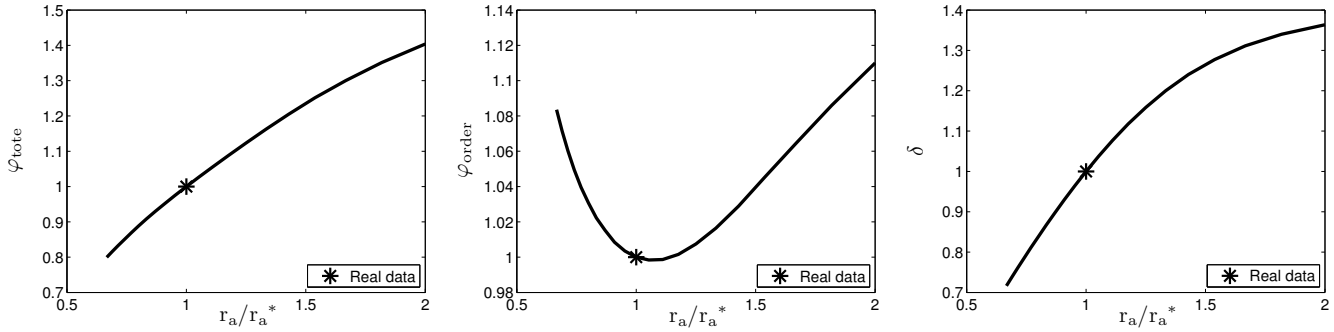
In this case study, the aggregate simulation model is able to predict the tote and order flow time with satisfactory accuracy based on real data of three working days. The aggregate model may further be used to analyze the effect of different interarrival rates, order length distributions, order release strategies, etc. on the system throughput and flow times. As an example, Figure 18 shows the predicted performance of workstation 1 under various interarrival rates. One may also be interested in analyzing the flow time distribution for a specific order length. Furthermore, EPTs can be calculated in real-time to monitor the performance of an operating order picking workstation such that any deviations from the expected performance (e.g., extremely large EPTs) can be detected and the necessary corrective actions can be performed timely.

VII. Conclusion and future work

In this paper, we propose a method to predict the mean and variability of tote and order flow times for a single-server order picking workstation by means of a simulation model that is based on an aggregate process time distribution. Arrival and departure data of totes are the only input required to calculate the aggregate process time distribution. Inspired by [16], we refer to the aggregate process time as Effective Process Time. We actually distinguish two types of EPT realizations namely for the first totes and for the remaining totes of orders, which we refer to as the 1st tote difference EPT method. We have demonstrated in the simulation validation study that this separation is important because the EPTs of the first totes are not identically distributed with the EPTs of the remaining totes. Therefore we sort the EPTs into two EPT distributions. We then fit a gamma distribution to these empirical EPT distribution data, but in principle any other suitable distributions may be used. The two gamma EPT distributions are used to sample the aggregate process time in the aggregate simulation model. We find that the proposed method accurately predicts the mean and variability of tote and order flow times.

We apply the method to data obtained from a real, operating warehouse. The flow time prediction by the aggregate simulation model has satisfactory accuracy, even with the relatively small amount of arrival and departure data. Practical insights for performance improvement are proposed based on the observation of EPTs from the real data. The resulting EPT distributions represent the actual pick rate of an order picking workstation, which for performance analysis purposes can be compared to the expected pick rate. The aggregate simulation model can further be used to evaluate the order picking workstation's performance under different settings.

The proposed method can also be used for manufacturing workstations processing a number of different product types, where switching from one product type to another requires



(a) Mean tote flow time. (b) Mean order flow time. (c) Throughput.

Fig. 18. Predicted performance of workstation 1 under various interarrival rates.

a setup time (see e.g., [26]). Here, the first job of a product type will have a different process time distribution than the remaining jobs. Hence, the 1st tote difference EPT approach used in this paper is also valuable in other systems.

Current advancements in order picking technology have allowed multiple orders to be processed simultaneously by a picker, which is often the case in warehouses with fast-moving products. Also, tote routings in large-scale automated warehouses may cause orders at the workstation to be processed not in a FIFO sequence. Performance analysis of these types of order picking workstation is subject to future work.

ACKNOWLEDGMENT

This work has been carried out as part of the FALCON project under the responsibility of the Embedded Systems Institute with Vanderlande Industries as the industrial partner. This project is partially supported by the Netherlands Ministry of Economic Affairs under the Embedded Systems Institute (BSIK03021) program. We would like to thank Ivo Adan, Jacques Resing, and Liqiang Liu of the Department of Mathematics and Computer Science at the Eindhoven University of Technology, and Roelof Hamberg from the Embedded Systems Institute for invaluable discussions during the model development. We also thank Bruno van Wijngaarden from Vanderlande Industries for providing the data used in the case study.

REFERENCES

[1] R. Andriansyah, L. F. P. Etman, and J. E. Rooda, "Simulation model of a single-server order picking workstation using aggregate process times," in *1st International Conference on Advances in System Simulation (SIMUL 2009)*, pp. 23–31, 2009.

[2] J. A. Tompkins, J. A. White, Y. A. Bozer, E. H. Frazelle, and J. M. A. Tanchoco, *Facilities Planning*, 2nd ed. New Jersey: John Wiley and Sons, 2003.

[3] M. Goetschalckx and J. Ashayeri, "Classification and design of order picking systems," *Logistics World*, pp. 99–106, 1989.

[4] K. J. Roodbergen and I. F. A. Vis, "A survey of literature on automated storage and retrieval systems," *European Journal of Operational Research*, vol. 194, no. 2, pp. 343–362, 2009.

[5] A. C. Caputo and P. M. Pelagagge, "Management criteria of automated order picking systems in high-rotation high-volume distribution centers," *Industrial Management and Data Systems*, vol. 106, no. 9, pp. 1359–1383, 2006.

[6] S. G. Koh, H. M. Kwon, and Y. J. Kim, "An analysis of the end-of-aisle order picking system: multi-aisle served by a single order picker," *International Journal of Production Economics*, vol. 98, pp. 162–171, 2005.

[7] B. C. Park, R. D. Foley, and E. H. Frazelle, "Performance of miniload systems with two-class storage," *European Journal of Operational Research*, vol. 170, pp. 144–155, 2006.

[8] Y. A. Bozer and M. Cho, "Throughput performance of automated storage/retrieval systems under stochastic demand," *IIE Transactions*, vol. 37, pp. 367–378, 2005.

[9] S. Hur, Y. H. Lee, S. Y. Lim, and M. H. Lee, "A performance estimation model for AS/RS by M/G/1 queuing system," *Computers and Industrial Engineering*, vol. 46, pp. 233–241, 2004.

[10] M. Eisenberg, "Queues with periodic service and changeover time," *Operations Research*, vol. 20, no. 2, pp. 440–451, 1972.

[11] M. J. Ferguson and Y. J. Aminetzah, "Exact results for nonsymmetric token ring systems," *IEEE Transactions on Communications*, vol. 33, no. 3, pp. 223–231, 1985.

[12] T. Hirayama, S. J. Hong, and M. M. Krunz, "A new approach to analysis of polling system," *Queueing Systems*, vol. 48, pp. 135–158, 2004.

[13] E. M. M. Winands, I. J. B. F. Adan, and G. J. van Houtum, "Mean value analysis for polling systems," *Queueing Systems*, vol. 54, pp. 35–44, 2006.

[14] M. van Vuuren and E. M. M. Winands, "Iterative approximation of *k*-limited polling systems," *Queueing Systems*, vol. 55, pp. 161–178, 2007.

[15] B. Rouwenhorst, B. Reuter, V. Stockrahm, G. J. van Houtum, R. J. Mantel, and W. H. M. Zijm, "Warehouse design and control: framework and literature review," *European Journal of Operational Research*, vol. 122, pp. 515–533, 2000.

[16] J. W. Hopp and M. L. Spearman, *Factory Physics: Foundation of Manufacturing Management*, 3rd ed. London: McGraw Hill, 2008.

[17] J. H. Jacobs, L. F. P. Etman, E. J. J. van Campen, and J. E. Rooda, "Characterization of operational time variability using effective processing times," *IEEE Transactions on Semiconductor Manufacturing*, vol. 16, no. 3, pp. 511–520, 2003.

[18] A. A. A. Kock, L. F. P. Etman, and J. E. Rooda, "Effective process times for multi-server flowlines with finite buffers," *IIE Transactions*, vol. 40, no. 3, pp. 177–186, 2008.

[19] J. Gu, M. Goetschalckx, and L. F. McGinnis, "Research on warehouse design and performance evaluation: a comprehensive review," *European Journal of Operational Research*, vol. 203, no. 3, pp. 539–549, 2010.

[20] J. P. van den Berg, "A literature survey on planning and control of warehousing systems," *IIE Transactions*, vol. 31, pp. 751–762, 1999.

[21] A. A. A. Kock, "Effective process times for aggregate modeling of manufacturing systems," Ph.D. dissertation, Eindhoven University of Technology, Eindhoven, 2008.

[22] J. H. Jacobs, L. F. P. Etman, E. J. J. van Campen, and J. E. Rooda, "Quantifying variability of batching equipment using effective process times," *IEEE Transactions on Semiconductor Manufacturing*, vol. 19, no. 2, pp. 269–275, 2006.

[23] A. T. Hofkamp and J. E. Rooda, "Chi 1.0 reference manual," Eindhoven University of Technology, Eindhoven, Systems Engineering Report 2008-04, 2008. [Online]. Available: http://se.wtb.tue.nl/sewiki/chi

[24] M. Matsumoto and T. Nishimura, "Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Transactions on Modeling and Computer Simulation*, vol. 8, no. 1, pp. 3–30, 1998.

[25] E. L. W. Blom, "EPT and flowtime distribution," Master's thesis, Eindhoven University of Technology, 2007.

[26] E. Lefeber and J. E. Rooda, "Controller design for switched linear systems with setups," *Physica A*, vol. 363, pp. 48–61, 2006.

Two-Level Validation and Data Acquisition for Microscopic Traffic Simulation Models

Stefan Detering, Lars Schnieder, Eckehard Schnieder
Institute for Traffic Safety and Automation Engineering
Technische Universität Braunschweig
38106 Braunschweig, Germany
E-mail: Detering@iva.ing.tu-bs.de, L.Schnieder@iva.ing.tu-bs.de, E.Schnieder@tu-bs.de

***Abstract*—Advanced driver assistance systems can have consequential effects on traffic flow. For the design, optimization and evaluation of these systems investigative simulations are necessary. Previous calibration and validation methods for these simulations utilized either microscopic or macroscopic measurement data. This paper's purpose is to argue that the formerly held calibration and validation perspectives with regard to traffic simulations are incomplete. Moreover, these assistance systems have their own set of particular requirements, and require the simultaneous consideration of microscopic and macroscopic system behavior. Therefore, this paper presents a new measurement concept that is needed to gain the required data necessary for proper calibration and validation. This concept advocates simultaneous measurements sourced in both a vehicle (microscopic) and overall traffic (macroscopic) perspective. Microscopic measurement results obtained by an equipped vehicle are presented.**

***Index Terms*—Traffic simulation; validation; calibration; microscopic and macroscopic perception, equipped vehicle**

I. Introduction

In this paper we extend our previous work [1][2]. The extensions and new contributions are as follows: the simulation requirements for investigation of recent developed advanced driver assistance systems (ADAS) are presented in a more structural method, therefore the difference between the previous and the new calibration and validation method becomes more clear. The measurement concept is presented in more detail and first empirical results from own test drives with an equipped vehicle are presented.

To date, exclusively the driver and the physical behavior of the vehicle have determined the overall driving behavior. More and more, it is becoming the trend to have assistance systems take over functions in areas of longitudinal and lateral dynamics of the vehicle. An already established system in the field of longitudinal guidance is the Adaptive Cruise Control (ACC) system. Using ACC, the driver chooses a target speed, and depending upon traffic conditions, the system automatically accelerates the vehicle to this target speed. If there is a preceding vehicle the vehicle decelerates and follows the preceding vehicle at a safe distance. Radar or laser sensor systems are used for distance and speed measurement. The driver assistance system uses distance or relative speed to the preceding vehicle as input variable. The combined vehicles' behavior as a system determines the macroscopic traffic characteristic, e.g. traffic flow or mean-speed. Therefore, the driver assistance system also influences these characteristics (see Fig. 1). Since the introduction of these systems to the market, there have been many simulation investigations concerning the influences of ACC on traffic flow. For an overview of the past work considering the effect of conventional ACC systems on traffic flow see [3][4].

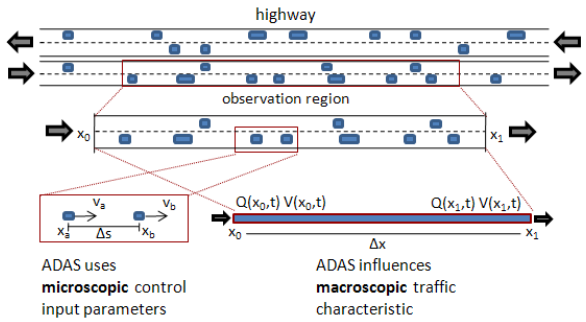


Fig. 1. Driver assistance system perception at the microscopic and macroscopic levels

In recent years, initial proposals have been presented for advanced ACC systems which recognize the current traffic state and adapt system parameters to optimize traffic flow [5]. Simulation investigations are carried out for design, optimization and evaluation, and are essential for the improvement of the system. Models employed for traffic simulations have been examined for over fifty years. However, the investigations associated with the advanced driver assistance systems, make new requirements essential which have not been considered over the last several decades since those simulation investigations were designed with different objectives in mind. This paper investigates these new requirements and proposes a measurement concept to fulfill these requirements.

This paper is structured as follows. In Section 2 an overview of current simulation models and tools is presented, which are most popular for examination of recently developed ADAS. In order to obtain quantitative results, a calibration and validation of the simulation model is mandatory, and this significance is presented in Section 3. Subsequently, Section 4 describes three typical simulation investigation cases, and based on this knowledge identifies the new requirements for simulation investigations for this kind of assistance system which have not

yet been considered. Section 5 presents a new measurement concept to fulfill the determined requirements. First results from an own equipped vehicle are presented.

II. Traffic Flow Modeling

In this section the most common traffic stream features are explained and different model structures are classified. Concluding the most widely-used simulation tools are presented.

A. Measures for Traffic Stream Features

One makes a distinction in general between microscopic and macroscopic measured variables. When considering the microscopic driving condition it is necessary to have: the position on the street s, the speed v as well as the acceleration (positive values) or deceleration (negative values) a of the vehicle as a function of time. In the case of the consideration of two successive vehicles, the distance Δs as well the relative speed Δv are also considered.

When considering the macroscopic traffic state, the traffic flow, sometimes also referred to as "traffic volume", density, as well as space-mean speed are essential. The dimensions are defined in detail as follows:

The number of vehicles N_q observed at a fixed location during a time interval Δt is the traffic flow Q (veh/h) for this interval:

$$Q = \frac{N_q}{\Delta t} \tag{1}$$

The traffic density k (veh/km) is the instantaneous number of vehicles N_k for a given road segment with the length Δx :

$$k = \frac{N_k}{\Delta x} \tag{2}$$

k and Q yield the space-mean speed v (km/h):

$$v = \frac{Q}{k} \tag{3}$$

For a detailed discussion about traffic variables see e.g. [6][7][8].

B. Traffic Simulation Models and Tools

Modeling of traffic reaches back to the middle of the 20th century when mainly physicists began to describe the phenomena of traffic by differential equations. There exist fundamentally different model structures that may be classified in agreement with e.g. [6][8][9]:

1) Microscopic traffic models
2) Macroscopic traffic models
3) Hybrid traffic models
4) Mesoscopic traffic models

Microscopic traffic models simulate the behavior of single vehicle-driver-units and describe their interactions by rule-bases that specify acceleration or velocity. Generally the dynamics are based on local variables like distance or relative speed to the front or rear vehicle.

Macroscopic traffic models neglect individual vehicles. They are devoted to aggregate state variables like density and flow for representing the collective behavior of vehicles. The equations are derived from the laws of nature and are structurally often similar to fluid mechanics.

Hybrid traffic models are a combination of macroscopic and microscopic approaches. Regions of the traffic network are modeled either in a macroscopic or microscopic manner, depending on which manner is more reasonable. For example at on and off ramps it may be interesting to regard the microscopic interactions of cars entering or leaving the highway, whereas traffic dynamics on the highway between ramps is sufficiently modeled by a macroscopic approach.

Mesoscopic traffic models combine the properties of microscopic and macroscopic traffic models. Single vehicles are simulated, but instead of considering vehicle-vehicle interactions macroscopic relationships are used to determine vehicle behavior.

Traffic simulation tools offer a network model for storing the data for the relevant traffic route properties as well as a traffic demand model for determining the amount of vehicles depending on time and day. The movement of the vehicles is determined by the behavior model. The behavior model itself consists of sub-models. In microscopic traffic models these are models for car-following, lane changing and route choice behavior.

The most well-known car-following models include Gazis Herman, Gipps, Wiedemann, IDM as well as the Cellular Automata ([10][11][12][13][14]). Popular lane changing algorithms have been presented by Sparmann, Gipps and for Cellular Automata ([15][16][17]). An integrated approach including car-following and lane changing behavior is investigated by [18].

At this time, the commercially most successful traffic simulation tools like VISSIM, PARAMICS and AIMSUN use the microscopic approach for the behavior model. Some of them also offer a macroscopic approach. Additionally, there are simulation tools, such as PELOPS, which have not been widely marketed, which use a more detailed description of human and vehicle behavior than in other microscopic models, therefore this type is called nanoscopic model.

III. Calibration and Validation of Traffic Models

The behavior of the driver-vehicle units is determined generally by the model behavior and especially by the parameters for the individual models. In order to obtain reliable simulation results, it is necessary to prove the validity of the selected parameters for each individual application [19].

Calibration involves adjusting the model parameters so that the simulation is sufficiently accurate when compared to actual behavior. It is necessary that a comparable set of actual measurement data for the selected route or network section is available for calibration. The goal is that the deviation is minimized between the model result and the recorded, actual measurement data. For this, an application adapted measured variable needs to be selected, such as measurement of travel times, velocities, time gaps or waiting periods. In general the measurement data, which originates from the varying behavior of the individual vehicles, illustrate the traffic behavior in an

aggregated form. A calibration according to non-aggregated, real comparative data (such as car-following or lane change behavior) seldom takes place, if at all, since this data often is not available. As shown below, a representation of the actual behavior on both microscopic and macroscopic levels is difficult to achieve.

The validation of model parameters immediately follows the calibration. Therefore, an additional set of data measurements is necessary. Calibration and validation data sets should also be collected under comparable conditions and "the core problem should be directly or indirectly described" [9]. For validation purposes, the simulation parameters must not be changed beyond this point. The simulation results are to be compared to the second measurement data set. The model gives valid results when the previously determined error measurement bound is not exceeded. Quantitative statements can only be made from a sufficiently validated model.

The account described above is based on information from [9]. The American equivalent is [20]. The makeup of both documents is very similar. A large difference originates from the validation evaluation of the traffic model. While [20] "presumes that the software developer has already completed this validation of the software and its underlying algorithms in a number of research and practical applications", [9] says that the "Validation [...] therefore serves for determination of the reliability of attainable statements in the particular application case" and "The calibration and validation step is of crucial importance for the reliability of reachable statements in every application case."

IV. Simulation Requirements for Investigation of Recent Developed ADAS

Although the models employed for traffic simulations have been examined for over fifty years, previous research completed on simulation models consider either the microscopic or the macroscopic level during calibration and validation. In general four different approaches can be identified:

- The approach of calibrating a *macroscopic simulation* with empirical *microscopic data* is not possible as macroscopic simulations do not model the details of the microscopic level. Therefore this approach will not be discussed in the following.
- Approach I deals with the calibration and validation of a *microscopic simulation* with empirical *microscopical data* (c.f. Subsection IV-A)
- Approach II deals with the calibration and validation of a *macroscopic simulation* with empirical *macroscopic data* (c.f. Subsection IV-B)
- Approach III deals with the calibration and validation of a *microscopic simulation* with empirical *macroscopic data* (c.f. Subsection IV-C)

The existing approaches are discussed in the following section along with their specific pros and cons. With this knowledge the new two-level approach is explained in Subsection IV-D.

A. Calibration and validation following approach I

There are a lot of investigations considering car-following behavior. Calibration and validation is performed for one or several car-following models and the differences between measuring data and simulation results are evaluated. The investigation of lane changing behavior seldom takes place.

Figure 2 shows the approach for calibration and validation on microscopic level. This investigation is only possible with microscopic simulation models, since macroscopic models neglect the details of the microscopic level. Calibration can be seen as a closed loop which involves the iterative execution of the following steps: First a microscopic simulation run is performed. In the second step the microscopic parameters empirically observed in the field are compared against the set of data resulting from the simulation. As stated before the investigation on the microscopic level is often restricted to the investigation of car-following behavior. Depending on the use case it may also be necessary to consider lane changing behavior. The deviation is compared against a predefined threshold.

In case the deviation between empirical results and simulation results is above the threshold in a third step, a thorough analysis of the underlying causes is needed. This is the crucial step in the calibration process. In order to properly identify the reasons of the deviation a deep knowledge of the qualitative correlations of the model parameters is required. Especially for the causal analysis, a good understanding of the traffic simulation model is necessary. The fourth step is the model adjustment. As soon as the cause has been identified the simulation can be adjusted. This can either be an adjustment of the parameters (parameter variation), or a change in the underlying mathematical functions relating the parameters to each other (structural variation). After having adjusted the model we have to start with step one again and perform a new simulation run.

In the case the deviation during calibration is below the threshold, the validation of microscopic model parameters immediately follows the calibration. Calibration and validation are closely related to each other. The calibrated model is transferred into a new, but comparable situation. For this situation an additional set of measured (microscopic) data is necessary. For this new situation a simulation run is performed. The microscopic simulation results are to be compared to the second measurement data set. The simulation run should deliver sufficiently accurate results for the new situation as well. In case the previously defined threshold is exceeded a re-calibration is needed, thus the causal analysis is the subsequent step. The model gives valid results on the considered microscopic level when the previously determined error measurement bound is not exceeded.

Macroscopic measurement data is not considered in this approach. Therefore a validation on macroscopic level is lacking. The validation is achieved only for one single driver-vehicle unit. In the case of considering only car-following behavior the validation is only achieved for one single function

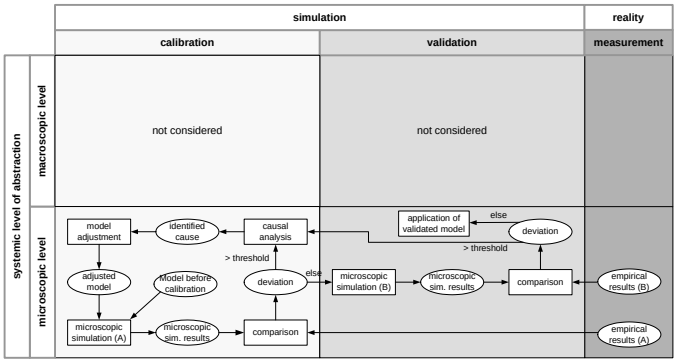


Fig. 2. Calibration and validation of a microscopic simulation model on microscopic level

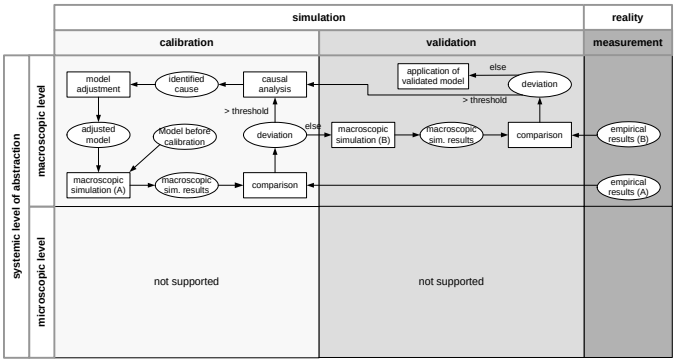


Fig. 3. Calibration and validation of a macroscopic simulation model on macroscopic level

of the driver-vehicle unit. The validation of one function as a subsystem of the complex simulation model does not result in a validation of a higher level of the model.

This conclusion is confirmed by the work of [21]. The authors of [21] present a multi-stage calibration and validation procedure. The microscopic calibration of headway behavior is one of the first steps. After completing the parameter fine-tuning to reconstruct traffic variations, the parameter values for the Mean Target Headway are raised by almost 25%, in comparison to values after headway behavior calibration. The reaction time is even raised by 59% in comparison to the previously determined value. With [21] it was therefore necessary to make an adjustment of the microscopic headway behavior in order to model the measured values from the macroscopic behavior. The question naturally arises whether or not it is advantageous to conduct a microscopic calibration and validation, and afterwards evaluate the macroscopic simulation results. As a restriction, it must be pointed out that in the investigation mentioned above the authors used measurement data from different traffic sites and different days.

In conclusion it can be stated that the validation of the simulation model does not walk hand in hand with the validation of separate sub-models.

Another crucial point is, that for the acquisition of the measurement data used for calibration and validation of car-following behavior many simplifications are used: the tracks in use are often single lane roads, sometimes the data is obtained on special test tracks or only consider inner city driving. Nevertheless these measurement data or the parameters determined by these measurement data is used for investigating traffic behavior on motorways. For example [22] found as a result of their car-following investigations that calibration with measurement data from urban rides cause strong deviations between the calculated parameters in comparision to suburban rides.

Many investigations also consider only a very small amount of car-following rides. For example [5] uses only three following-rides to determine the parameters of the model in use. When attempting to use only a very small amount of car-following rides, it does not seem possible to distinguish the behavior of different drivers (inter-driver variability) and the variation of the behavior of a single driver in the course of time (intra-driver variability).

B. Calibration and validation following approach II

Often the primary focus is not on the detailed analysis of the behavior of individual vehicles. Instead the major interest lays on the aggregated macroscopic traffic behavior. Examples for this can be found in traffic planing where the effects changes in the control of traffic light signals or modified traffic routings can be evaluated. For those analyses both microscopic and macroscopic traffic simulation models are applied. In the following Section we elaborate on how the calibration and validation of a macroscopic traffic simulation model based on macroscopic data can be performed. In the next Section we describe how a microscopic traffic simulation can be calibrated and validated taking into consideration macroscopic simulation results.

Figure 3 shows the procedure of calibrating and validating a macroscopic traffic simulation on the macroscopic level. This procedure is comparable to the the procedure of calibration and validation on microscopic level. In contrast to this only the macroscopic level is considered in this case. As the first step a macroscopic simulation is performed with standard parameters and during calibration and validation the results of the macroscopic simulation are compared to the macroscopic empirical data. Again the adjustment of the parameters of the macroscopic simulation requires a thorough and deep understanding of the model as the model's parameters often do not have an explicit relation to empirically observable parameters. Validation follows the succesful calibration. In the validation phase macroscopic simulation results are compared with a second data set of macroscopic empirical data. It should be stressed, that the macroscopic simulation does not simulate individual driver-vehicle-units and thus does not allow an analysis on the microscopic level.

However, for design and evaluation of advanced driver assistance systems optimizing traffic flow it is required that the model is valid and explains the differences of the microscopic behavior of the driver and the macroscopic behavior at the system level. This will become obvious in different driving

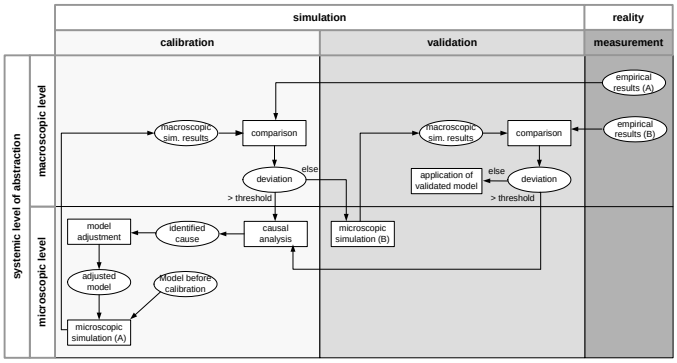


Fig. 4. Calibration and validation of a microscopic simulation model on macroscopic level

situations, where for example a following vehicle approaches the preceding vehicle, follows a preceding vehicle, brakes due to the decelaration of the preceding vehicle or accelerates to the level of the previously entered desired value as soon as no other vehicle is in front of the own vehicle. In order to show the differences of the vehicles' longitudinal dynamics performed by a human driver compared to an assistance system it is necessary to simulate the behavior of indvidual vehicles. The design of advanced driver assistance systems brings about requirements macroscopic simulations can never meet due to these principal considerations.

C. Calibration and validation following approach III

For the same investigations as mentioned in IV-B also microscopic simulation models are used. But in these cases the detailed microscopic behavior is not considered. Instead, the microscopic simulation is used only with regards to macroscopic results.

Figure 4 shows the approach for calibration and validation on macroscopic level. The calibration is comparable to the calibration process in subsection IV-A. During calibration almost the same steps have to be performed. Only the second step differs. As unlike before not the microscopic empirical results and microscopic simulation results are considered, but the macrosopic empirical results are compared to the macroscopic simulation results. For validation also a second macroscopic measurement data set is necessary. Finally in this case the model offers valid results on the considered macroscopic level when the previously determined error measurement bound is not exceeded.

Microscopic measurement data is not considered in this approach and therefore a validation on microscopic level is lacking. The validation on the macroscopic level of the simulation model does not result in a validation of the sub-models. It is possible that different combinations of the microscopic sub-models result in the same macroscopic behavior.

D. Two-level approach for calibration and validation

The review of currently available research shows that the microscopic and macroscopic levels of simulation currently co-exist and are not interwoven. Current research did not reflect that for the optimization of advanced driver assistance systems the interrelations of both levels of simulation need to be reflected. For this reason a two-level approach for calibration and validation is stipulated in this paper. It is to calibrate and validate a traffic simulation both on microscopic and macroscopic level. The microscopic simulation model simulates each vehicle individually. Each vehicle behavior in the simulation has to be valid especially concerning the crucial input factors for the advanced driver assistance system. Improving the macroscopic traffic flow with the advanced driver assistance system, the simulation has to be validated on a macroscopic level too.

Taking the ACC system as an example, the system changes the following behavior of the vehicle to its preceding vehicle. Therefore it is absolutely necessary to use a microscopic traffic model for testing such a system which includes the headway behavior of the human driver as well as the ACC system. The ACC equipped vehicles exhibit different vehicle headway behavior than those vehicles driven by humans. It must be particularly assured that human following behavior is modeled precisely as long as no 100% system equipment is considered, since the interaction between system headway behavior and human headway behavior can be of particular importance.

It seems reasonable to conduct both a microscopic calibration and validation of the vehicle headway behavior for human and ACC system behavior in the first step. This step is the same as described in subsection IV-A. Subsequently, it is possible to estimate the accuracy and reliability of the simulated vehicle headway behavior of humans and the system.

As stated in Section I the previously mentioned ACC system will influence traffic flow behavior. In recently developed ADASs the individual vehicle will even optimize traffic flow [5]. The microscopic behavior is thus changed in order to optimize the macroscopic behavior. Subsection IV-A stated that a valid model on microsopic level does not have to be valid on macroscopic level. However, for evaluating system efficiency macroscopic variables, e.g. traffic flow or travel times are used as a measurement variable. Therefore after calibrating and validating the simulation model only on microscopic level the simulation seems not yet suitable for investigation of these ADAS. Therefore a microscopic calibration and validation of the individual models and a concluding macroscopic validation appear to be meaningful. This is the reason for proposing a two-level approach of traffic simulation models.

The basic idea of a microscopic calibration and a macroscopic validation has been already described in [23] in the explanation of calibration and validation.

> "This should ideally be undertaken at both a macroscopic scale (validation) to ensure the overall behavior of the model matches that readily observable, but also at a microscopic scale, with regard to individual vehicle-vehicle interactions (the calibration, and "tuning" of the many behavioral parameters comprising the decision making processes)."

One of the first research using this approach can be seen

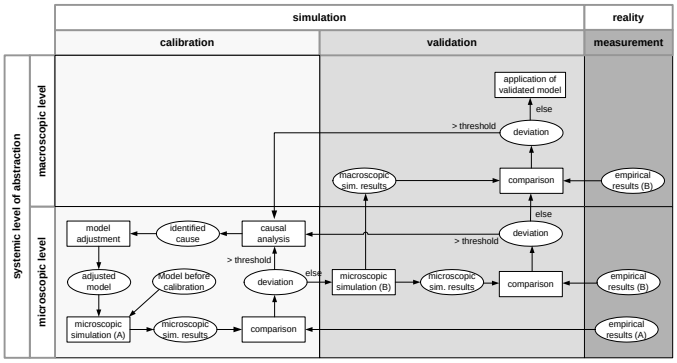


Fig. 5. Calibration and validation of a microscopic simulation model on microscopic and macroscopic level

in [24]. Witte, the author of [24], carried out a microscopic calibration and afterwards a macroscopic validation. Unfortunately, this validation is limited to a purely qualitative consideration. Moreover the validation is carried out with the help of an single-lane roundabout, therefore overtaking and merging are not considered. Witte comments that he observes relatively high traffic flows and traffic densities from the simulation. He attributes these results to the absence of trucks and the measuring data used for calibration.

The new proposed two-level approach is shown in Fig. 5. The first steps for calibrating and validating the model on microscopic level is the same procedure as in subsection IV-A. As soon as microscopic parameters have been successfully validated a validation of macroscopic variables (e.g. traffic flow, traffic density, mean speed) is possible. Again, this can be done by doing the following steps. First a microscopic simulation run is performed. This time the behavior of the sum of all individual driver-vehicle units in the simulation is measured. This simulation run yields the macroscopic variables previously identified. In a second step the simulation results are compared against empirical data observed in the field. In case the results stay within the permissible range the calibration and validation on microscopic and macroscopic level has been successful. Otherwise a re-calibration on microscopic level becomes necessary. Now it becomes obvious that calibration is only possible on the microscopic level, as well as the fact that the microscopic variables are independent variables whereas the macroscopic variables can be considered as dependent variables.

The next step is to gather both the macroscopic and microscopic real-traffic variables in such a manner that they share the same time and geographical reference. A plausible measurement concept is proposed in the next Section.

V. Measurement Concept

In general three different approaches for the acquisition of measurement data can be identified:

- The approach of acquisition of *macroscopic data* (c.f. Subsection V-A)
- The approach of acquisition of *microscopic data* (c.f. Subsection V-B)

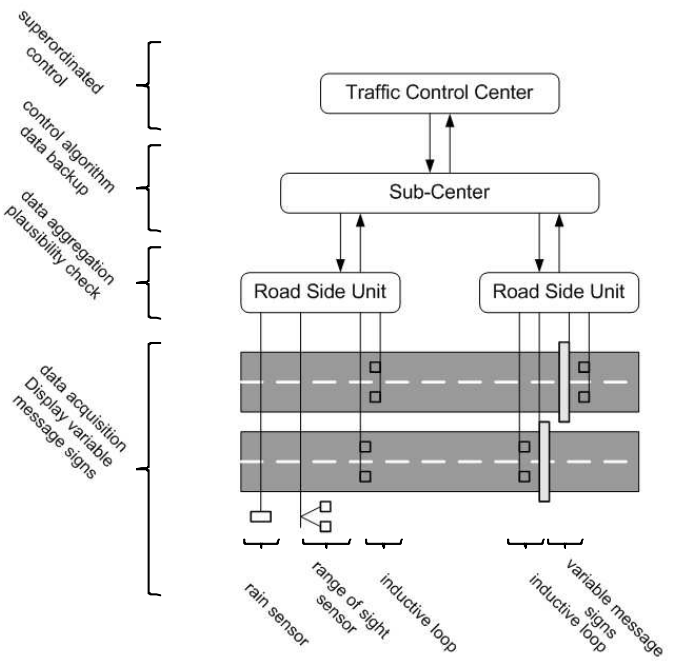


Fig. 6. System layout of a traffic control center

- The approach of acquisition of *both macroscopic and microscopic data* (c.f. Subsection V-C)

Based on the knowledge of the previous measurement concepts a new approach for acquisition of both macroscopic and microscopic data is presented. First empirical results regarding the measurement of microscopic variables are shown.

A. Acquisition of macroscopic data

Primarily, the calibration and validation of simulation models is done with macroscopic measurement data. This measurement data is often obtained by traffic centers which typically store aggregated measurement data. The general system layout of a traffic control center is shown in Fig. 6. Traffic Control Center often consists of sub-Centers and the road-side-units near the highway. The main sensor in use is the inductive loop sensor to measure traffic flow and vehicle speed, and to differ between different vehicle classes. Variable message signs are used to influence driver and traffic flow behavior. The data from the sensors is aggregated in the road side units - often is used a time span of 1 minute or more. The typical distance between the sensor systems is one or several kilometers.

B. Acquisition of microscopic data

To obtain microscopic measurement data different methods are used. This can either be a terrestrial or an aerial perspective.

Following the terrestrial perspective, test vehicles are in most cases equipped with an appropriate sensor system. The test vehicle speed, distance and the relative velocity to the vehicle driving ahead are measured by the system. Usually it is well-known by test drivers that they are being observed while driving, and in some cases they are given special driving tasks.

It is accepted that the drivers in these special test conditions do not react as they might under normal conditions. Such an investigation was conducted for example in [25]. In this investigation the suitability of 10 traffic flow models of vehicle headway behavior was tested for emulating measured data.

Another terrestrial perspective is the equipment of several test vehicles with a GPS system. The test vehicles have to drive behind each other. Therefore these kind of investigation is often performed on special test tracks. Using the GPS position data of the individual vehicles the speed, acceleration and distance information is calculated.

Following the aerial perspective, traffic observations are carried out from a helicopter. Initial investigations already have been conducted as found in [26]. The latest investigations with a video camera installed under a helicopter come from [27]. This approach has the advantage that the observed drivers remain uninfluenced as much as possible. The disadvantage exists that a great effort is needed to collect data, and inaccuracies occur when determining the distance and velocity of the observed vehicles from the video material. This inaccuracy directly influences the calibration of the traffic model.

C. Acquisition of both macroscopic and microscopic data

Presently, there are empirical investigations acquiring either macroscopic traffic data or microscopic traffic data. As previously discussed, it would make sense to simultaneously measure data from microscopic vehicle headway data and lane change data from several vehicles as well as macroscopic measurement data for the same section and the same time span.

One approach already exists for the collection of such data. In the NGSIM project [28] vehicle positions are extracted from video material obtained by several cameras and translated into vehicle tracking data. Additionally, for the road segment observed by the video macroscopic data can be calculated. In NGSIM project loop detector and weather data also have been recorded.

The data sets originate exclusively from the USA and indicate the road network layout that is typical there, as well as drivers' behavior. It is otherwise undetermined whether or not this data can universally be applied to driving scenarios in Europe or Germany. A comparable European project does not exist. Another problem with the NGSIM project exists as a result of the camera technology used. For this data only relatively short road segments a maximum of 640 m were considered. In current databases there exist two data sets from freeways, the sides measuring 500 m and 640 m, respectively. The mean-speed of the vehicles is less than 50 km/h and therefore each car is tracked for about a maximum of 50 seconds. Additionally, estimating vehicle trajectories from video data leads to measurement errors that cannot be neglected. These errors have to be taken into consideration using the data for calibration and validation [29].

D. A new approach towards the acquisition of macroscopic and microscopic data

A new approach towards the acquisition of microscopic data is the continuous and entire movement data acquisition of single vehicles on longer road segments and for greater time periods in combination with the acquisition of the macroscopic data from traffic centers. Without influencing the traffic behavior extensive data sets can be collected from the equipment of the measuring vehicles participating in the real flow of traffic. By means of suitable sensor facilities (radar, lidar), the measuring vehicles are able to capture the behavior of several surrounding vehicles precisely and pursue them continuously for a longer period.

At the same time a traffic center provides aggregate data for this road segment. This data, among other things, represents the traffic flow, truck percentage and mean vehicle speed. To simulate the real traffic flow, it is at least necessary to know the amount of vehicles entering the section under observation, and the amount of vehicles entering or leaving this section by on- or off-ramp. Figure 8 shows a road segment with the necessary inductive loop sensors as well as several equipped vehicles and observed vehicles.

Up to now test vehicles were equipped with sensor systems measuring the vehicles in front of the test vehicle. With this approach the behavior of the test driver and the test vehicle itself can be monitored, especially regarding the movement of the preceding vehicle. The test driver can be monitored for a longer time period, therefore intra-driver variability can be investigated. The disadvantage of this application is that the test driver knows that he is observed personally.

In addition to the sensors monitoring the activity ahead of the vehicle, sensors should also be mounted on the vehicle's rear in order to observe the following traffic, including the following driver's behavior. As soon as the following vehicle changes with another vehicle, the next following vehicle can be observed. This procedure has the advantage that within a short time frame the behavior of multiple drivers can be examined without the examined drivers' awareness. Therefore, the following driver's influence on the measurement can be neglected. Hence with this data the calibration of the vehicle subsequent model of the traffic simulation can be carried out. Due to the large number of observed drivers, where behavior can vary from driver to driver, the so-called inter-driver variability is determined and can be applied to the traffic simulation.

The Institute for Traffic Safety and Automation Engineering equipped a Volkswagen Passat with a radar sensor in the front and a lidar senor in the back to monitor the preceding and following vehicles. In addition a lot of measurement data from the own vehicle (speed, steering angle, actuation of throttle and brake, ...) is recorded. For verification of the measurement data a video system is recording the view to the front and back. Figure 7 shows the general system layout.

Up to now only one vehicle was equipped with the sensor system. The goal is to equip several vehicles with this system.

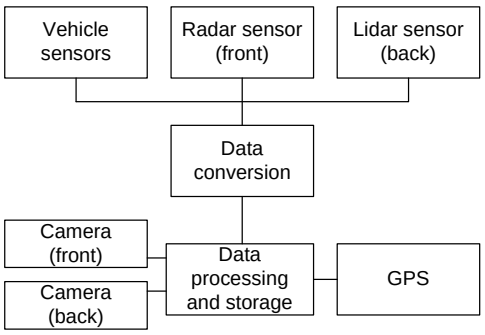


Fig. 7. General layout of the in-vehicle measurement system

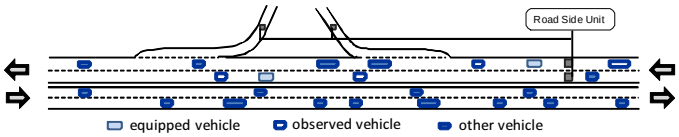


Fig. 8. Measurement concept with several cars

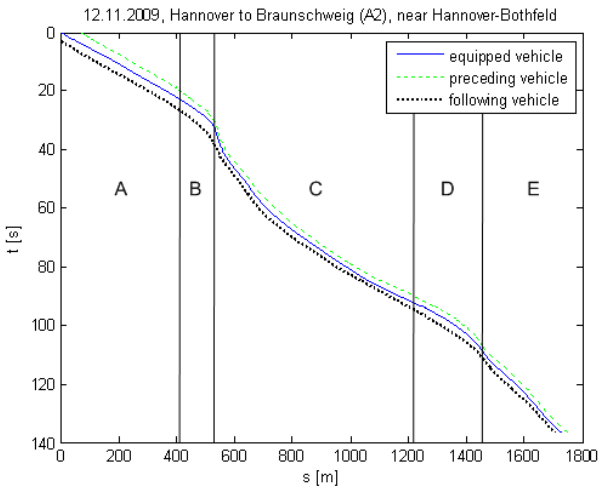


Fig. 9. Different driving situations observed by an equipped vehicle

In this case the GPS device is an important component for the measurement concept with several cars. Of course on the one hand the GPS device allows the measurement of the position of the vehicles. On the other hand it is much more important that the GPS device delivers a unique time base. This unique time base is absolutely necessary to merge the measurement data of several vehicles. The GPS time base is available almost everywhere. The advantage of this approach is that the vehicles don't have to exchange any data between each other to synchronize the measurement time.

E. Empirical results

Our equipped vehicle was tested and gathered data from trips with more than one thousand kilometers. The measurement data shows, that our measurement system is excellent for observing the own vehicle, the preceding vehicle and the following vehicle. Due to the developments in sensor technology over the past several years, testing on microscopic level is feasible with significantly lower technical effort in comparison to a few years ago.

Figure 9 shows a small section of our measurement data. The figure shows the trajectory of the movement of our equiped vehicle and the trajectory of the following and preceding vehicle in a typical stop-and-go situation on the Autobahn A2 near Hannover, Germany. In this measurement data we can identify the driving situations mentioned in Subsection IV-B. In phase A the vehicles follow with almost constant speed of about 70 km/h. In phase B the equipped vehicle and the following vehicle brake due to the deceleration of the preceding vehicle. The vehicles comes almost to standstill. In phase C the vehicles accelerate very slowly up to a speed of almost 80 km/h before starting to decelerate in phase D again. Finally the vehicles drive at almost constant speed in phase E of about 40 km/h.

With this measurement data the car-follwing behavior of our test driver and our equipped vehicle as well as the car-following behavior of the following driver and vehicle can be investigated.

VI. Conclusion and Future Work

For future investigations of ADAS it is necessary to validate the simulation at the microscopic and the macroscopic level. As a result, it has been deemed necessary to simultaneously obtain both microscopic and macroscopic data.

After verifying the qualification of our measurement system to observe car-following, the next step is to extend our analysis to lane changing behavior. With the knowledge of the preceding and following vehicle and some additional information it should also be possible to observe lane changing behavior. Afterwards several vehicles can be equipped with this measurement system. It is noteworthy that fortunately a large number of institutes and research facilities already own one or several such measuring vehicles. It is possible to use GPS signals as a uniform time and local reference. Therefore, only the coordination of a uniform data format is necessary in order to enable these vehicles to examine together traffic behavior together. With such common measurements, a new data base can be achieved for the first time.

The final goal is to coordinate the measuring vehicles on a predetermined road segment where for the same time span a traffic center provides macroscopic data for this road segment. Therefore the details for this measurement have to be developed together with experts from traffic centers and institutes, and research facilities interested in participating in such an investigation with their own test vehicles.

If video data of the segment is available it can be used as a reference. For example in Germany on the Autobahn A7, a long stretch is equipped with such a video camera system. Because the videos are used only as a supplement, a clearly larger road segment, as opposed to the NGSIM project, can be considered.

This approach makes it possible to connect the microscopic and macroscopic view. Using the evaluation of the data of all

measuring vehicles and the data from traffic centers the current traffic behavior of a large road segment can be developed. A calibration and validation on microscopic and macroscopic scale can then also be achieved.

REFERENCES

[1] S. Detering and E. Schnieder, "Two level approach for validation of microscopic simulation models," in *Advances in System Simulation, 2009. SIMUL '09. First International Conference on*, Sept. 2009, pp. 18–22.

[2] L. Schnieder and S. Detering, "Systemische kalibrierung und validierung von simulationen zur auslegung von verkehrsassistenzsystemen," in *AAET 2010 - Automatisierungssysteme, Assistenzsysteme und eingebettete Systeme für Transportmittel*. Gesamtzentrum für Verkehr e.V., 2010.

[3] P. Zwaneveld and B. van Arem, "Traffic effects of automated vehicle guidance systems - a literature survey," TNO Inro, Delft, The Netherlands, Tech. Rep. INRO-VVG 1997-17, 1997.

[4] J. VanderWerf, S. Shladover, N. Kourjanskaia, M. Miller, and H. Krishnan, "Modeling effects of driver control assistance systems on traffic," *Transportation Research Record*, vol. 1748, pp. 167–174, 2001.

[5] A. Kesting, "Microscopic modeling of human and automated driving: Towards traffic-adaptive cruise control," Ph.D. dissertation, Technische Universität Dresden, Fakultät für Verkehrswissenschaften "'Friedrich List"', 2008. [Online]. Available: http://nbn-resolving.de/urn:nbn:de:bsz:14-ds-1204804167720-57734

[6] S. Detering, *Verkehrsleittechnik - Automatisierung des Straßen- und Schienenverkehrs*, E. Schnieder, Ed. Springer, 2007, ch. Flusssteuerung im Straßenverkehr, pp. 155 – 194.

[7] C. F. Daganzo, *Fundamentals of Transportation and Traffic Operations*. Elsevier Science Ltd, 1997.

[8] D. Helbing, *Verkehrsdynamik - Neue physikalische Modellierungskonzepte*. Springer, 1997.

[9] R. Trapp, *Hinweise zur mikroskopischen Verkehrsflusssimulation*, Arbeitsgruppe Verkehrsführung und Verkehrssicherheit, Ed. FGSV Verlag, 2006.

[10] D. Gazis, R. Herman, and R. Rothery, "Nonlinear follow the leader models of traffic flow," *Operations Research*, vol. 9, pp. 545–567, 1961.

[11] P. G. Gipps, "A behavioural car-following model for computer simulation," *Transportation Research Part B: Methodological*, vol. 15, no. 2, pp. 105–111, April 1981.

[12] R. Wiedemann, *Simulation des Strassenverkehrsflusses*. Institut für Verkehrswesen der Universität Karlsruhe, 1974, vol. Heft 8 der Schriftenreihe des IfV.

[13] M. Treiber and D. Helbing, "Realistische Mikrosimulation von Straßenverkehr mit einem einfachen Modell," in *16. Symposium "'Simulationstechnik ASIM 2002"'*, D. Tavangarian and R. Grützner, Eds., Rostock, 2005, pp. 514–520. [Online]. Available: http://www.cs.wm.edu/~coppit/csci435-spring2005/project/index.php

[14] K. Nagel and M. Schreckenberg, "A cellular automaton model for freeway traffic," *Journal de Physique I*, vol. 2, no. 12, pp. 2221–2229, 1992. [Online]. Available: http://www.edpsciences.org/10.1051/jp1:1992277

[15] U. Sparmann, *Spurwechselvorgänge auf zweispurigen BAB-Richtungsfahrbahnen*, ser. Forschung Straßenbau und Straßenverkehrstechnik. Bundesminister für Verkehr, Abteilung Strassenbau, 1978, vol. 263.

[16] P. Gipps, "A model for the structure of lane-changing decisions," *Transportation Research Part B*, vol. 20, pp. 403–414, 1986.

[17] F. Mazur, D. Weber, R. Chrobok, S. F. Hafstein, A. Pottmeier, and M. Schreckenberg, "Basics of the online traffic information system autobahn.nrw," Physics of Transport and Traffic, Universität Duisburg-Essen, Tech. Rep., 2005, hannover Messe 2005.

[18] T. Toledo, H. N. Koutsopoulos, and M. Ben-Akiva, "Integrated driving behavior modeling," *Transportation Research Part C: Emerging Technologies*, vol. 15, no. 2, pp. 96 – 112, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/B6VGJ-4NGBB0J-1/2/b0ba6b4bc502c5d1de2bb3d392dee6c1

[19] S. Detering and E. Schnieder, "Requirements for precise simulation models for traffic flow optimizing adas," in *12th IFAC Symposium on Control in Transportation Systems (CTS'09)*, Redondo Beach, USA, 2009, pp. 467–471.

[20] R. Dowling, A. Skarbardonis, and V. Alexiadis, "Traffic analysis toolbox volume III: Guidelines for applying traffic microsimulation software," The Federal Highway Administration, Tech. Rep., 2004.

[21] L. Chu, H. Liu, J.-S. Oh, and W. Recker, "A calibration procedure for microscopic traffic simulation," *Proceedings of IEEE Intelligent Transportation Systems*, vol. 2, pp. 1574–1579, 2003.

[22] V. Punzo and F. Simonelli, "Analysis and comparison of microscopic traffic flow models with real traffic microscopic data," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1934, pp. 53–63, 2005.

[23] M. Brackstone and M. McDonald, "Validity of microscopic modelling of motorway traffic," *Proceedings of the Intelligent Vehicles '94 Symposium*, pp. 576–581, Oct. 1994.

[24] S. Witte, "Simulationsuntersuchnungen zum Einfluß von Fahrverhalten und technischen Abstandsregelsystemen auf den Kolonnenverkehr," Ph.D. dissertation, Universität Fridericiana zu Karlsruhe (TH), 1996.

[25] E. Brockfeld, R. Kelpin, and P. Wagner, "Performance of car following behaviour in microscopic traffic flow models," in *2nd International Symposium "'Networks for Mobility"'*, W. Möhlenbrink, F. Englmann, M. Friedrich, U. Martin, and U. Hangleiter, Eds. Universität Stuttgart, 2004, pp. 43 – 43. [Online]. Available: http://elib.dlr.de/21349

[26] J. Treiterer, "Some aspects of the stability of traffic flow," in *Beiträge zur Theorie des Verkehrsflusses : Referate anläßlich des IV. Internationalen Symposiums über die Theorie des Verkehrsflusses in Karlsruhe im Juni 1968*, W. Leutzbach and P. Baron, Eds., no. 86. Bundesmin. f. Verkehr, Abt. Straßenbau, 1969, pp. 8–13.

[27] S. Hoogendoorn, S. Ossen, and M. Schreuder, "Empirics of multianticipative car-following behavior," *Transportation Research Record*, vol. 1965, pp. 112–120, 2006.

[28] "NGSIM - Home of the Next Generation SIMimulation community," 2009. [Online]. Available: http://ngsim.fhwa.dot.gov/

[29] V. Punzo, M. T. Borzacchiello, and B. Ciuffo, "Estimation of vehicle trajectories from observed discrete positions and next-generation simulation program (NGSIM) data," in *TRB 2009 Annual Meeting*, 2009.

A Shape Grammar with Feedback Generative Model for the Design of Compact Microstrip Antennas

Adrian Muscat
University of Malta [adrian.muscat@um.edu.mt]

***Abstract*—A shape grammar based model that generates functional initial microstrip antenna designs is developed. The functionality of the designs is ensured with the inclusion of a feedback mechanism that influences the rule selection process. The grammar itself decomposes a complex shape into a chain of rectangles, whose parameters are used in a mathematical formula model that makes up the feedback mechanism. This approach to antenna design is demonstrated with the generation of compact microstrip antenna structures.**

***Keywords*-Shape Grammar; Compact Microstrip Antenna; Feedback; Intelligent Computer Aided Engineering;**

I. INTRODUCTION

The design of compact microstrip antennas is by far a manual task and in particular requires the antenna engineer to combine, articulate and evolve in a constrained volumetric space, a number of geometric shapes that yield the desired electrical characteristics. Fig.1 shows five examples of increasing design complexity. The first design is a standard full size single-band rectangular microstrip antenna that is not restricted in size and its design is a straightforward process that can be readily formalised, since the shape is always that of a parametric rectangle. The antenna in Fig.1(b) is a compact version of a larger single-band rectangular antenna. A narrower and longer rectangular patch is meandered into an S-shape. The aspect ratios of the various sections can be varied, but the shape remains the same, however distorted it may look. Therefore for example adding an extra twist in the shape cannot be simply done with a change in the value of one parameter. The antenna in Fig.1(c) is a dual-band single feed compact design which is then modified to accommodate within the same volume a separately fed third band, Fig.1(d). The design process for the latter prototypes is not obvious and difficult to formalize. Additionally the designer is rarely free to decide on the positions of the feed and shorting posts (shown as dots in fig.1) as these are generally dictated by other system design considerations. This means that the shape has to fit around the position of these components and not the other way round. The antenna in fig.1(e) is an example of a reconfigurable antenna that can be switched between wide-band and narrow-band operation. In such a case it is desired to use the minimum number of switches, while at the same time, satisfy the characteristics at a discrete number of frequency bands. The synthesis task for this case is even more difficult to formalise. Furthermore the design task is a complex one because the electrical characteristics are strongly coupled to the form and shape of the antenna structure and a small change in the topology or form can result not only in a significant change in performance, but can also render the design invalid. Because of these two reasons such designs are carried out by expert designers. Hence modifying a design to suit some additional specifications requires a significant time investment.

During the design process, the antenna engineer has at his disposal two types of Computer Aided Design (CAD) tools, empirically derived mathematical formula based models, and full-wave numerical models. The former type is limited to specific simple geometries, is very fast to compute, and to some extent relates dimensional attributes to the electrical properties. On the other hand the latter is applicable to any arbitrary geometrical shape and is very accurate, but is computationally intensive and does not give an explanation of how the device works and how it can be modified to fit the specifications. During the design of compact antennas the engineer makes use of the simple formula models to select the components and advance the process of shape evolution and manipulation. Once the initial form is developed numerical CAD and optimisation techniques are used to refine the design, or shape. It would therefore be useful to have a design tool that formalizes and mimics some of the informal processes that an antenna designer goes through. This will help speed up the synthesis task.

In [1] the use of a shape grammar with feedback based CAD system to assist the designer in the generation of valid and functional shapes is proposed and this system is demonstrated on narrow constant-width-meander-line microstrip designs. This paper extends this work to include microstrip antennas of variable line widths and therefore shapes similar to the ones in fig.1. The grammar makes use of feedback to yield an estimate of the electrical properties of arbitrary shaped meander-line microstrip antennas.

Shape grammars have been originally developed by Stiny[3] and have been used to generate architectural designs and art works that pertain to a particular style [3], [4]. They were later applied in engineering design for example in the representation of solid objects [5], the design of optimal truss structures [6], the design of coffee makers [7] and design of microresonators [8].

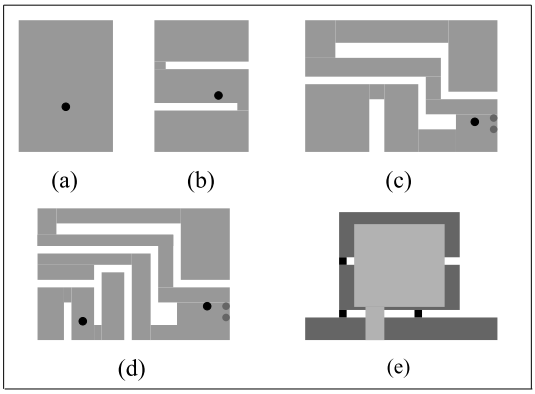


Figure 1. Microstrip and patch antenna designs (a) relatively straightforward to design and describe, (b) this design makes full use of the available space and is harder than (a) to design, (c & d) two harder designs where two modes are matched at the same point, and (e) an even harder problem where the antenna can be switched between wide-band and narrow-band operation [2].

Architectural designs have a well-defined function-form decomposition such that the generating engine first satisfies the functional requirements and then proceeds to generate the physical forms in a particular style. However most engineering artifacts do not have such a straightforward form-function decomposition and are characterised by a considerable function-form coupling. Agarwal and Cagan[9] propose shape grammars as a new framework for geometry-based engineering expert systems as an analysis and synthesis tool. It is further stated that it is the class of products characterised by a strong function-form coupling that stand to gain most from formal design tools because the lack of a pre-defined generation sequence limits human designs to a small set of valid configurations. In [9] a coupled form-function shape grammar is proposed for the design of micro-electromechanical resonators. This is used to generate designs that function as resonators. The functional requirement is enforced by making sure that the grammar includes all the necessary elements that are required for the device to function and validity is ensured in this respect. The form of the device is then altered by choosing the rules at random or through a search process. This however means that the design may not necessarily meet or be close to meeting all the required specifications for the intended applications. However, the goal in this paper is to generate designs that are close to satisfying all specifications without the use of numerical CAD. An approximate model is therefore developed and used within the generative or synthesis cycle. The system is therefore described as a shape grammar with feedback generative system.

The automation of the synthesis task in microstrip antenna design has been mainly approached through the application of the Genetic Algorithm (GA) in search of structures that yield the required specifications. Genetic algorithms are useful to explore poorly-understood search spaces. They are coded with the minimum of domain knowledge and assumptions, and use domain independent genetic operators to explore the search space. They can be very robust and persistent throughout the search but are very slow to converge to a result and are inefficient for local optimization. Systems that rely solely on the GA, therefore, require hundreds of iterations to reach their goal, if a solution exists in the defined search space. Several methods like pruning and structuring of the search space can be applied to improve the GA efficiency for CAD purposes. Some of these methods are domain specific and some are not. Johnson and Yahya Rahmat-Samii pioneered the use of the genetic algorithm (GA) for machine evolved planar microstrip antenna shapes that exhibit wide-band characteristics [10]. In this work a rectangular design space is defined in terms of a $M \times N$ matrix of pixels and the GA is allowed to search new and unknown shapes that exhibits the required specifications. Jones and Jones developed a wire antenna array language that defines the structure of Yagi antenna arrays and combines this with a genetic programming (GP) algorithm [11]. The setup is used to obtain a machine evolved classical Yagi structure as well non standard Yagi designs, where the GP is allowed to try new configurations. A similar experiment is reported by Koza [12] for loaded co-linear wire antennas and uses the concept of a turtle that drops elements along a linear track. In [13], a Knowledge-Based Genetic Algorithm (KBGA) is developed to reduce the time required to evolve novel microstrip antennas. The KBGA makes use of the abstract shape representation described in [10] and selects antenna design heuristics (rules of thumb) that influence the genetic operators. The KBGA is similar to a language of design techniques, but does not deal directly with shape. In [14] the same encoding and crossover techniques are used for the purpose of miniaturization. In [15] it is shown experimentally that this encoding method is not suitable to evolve compact structures of the types shown in Fig.1(c& d). However this does not mean that other encoding techniques do not yield results. The approaches described above are limited in three ways (a) These methods call a numerical model at each iteration and this slows down the process, (b) the quest of these papers is machine evolved designs rather than acting as a partner within the design team and therefore there is little need to link any shape modification to its properties, and (c) these methods do not deal with the geometric shape in a direct way.

In this paper the shape grammar is primarily used as an automated generative tool, where the system chooses rules to be applied based on immediate feedback on the electrical characteristics of the partial design and on the specifications to be met by the final design. The feedback is obtained through the use of the shape grammar as an analysis tool. The emergent shapes in the design are recognized and linked to an analytical equation that yields the electrical

characteristics. Recognition of the emergent shapes is carried out implicitly during the shape evolution process. This is a similar process to the procedure adopted by the human designer where first order models are used to evolve the conceptual or initial shape or form. The usefulness of this methodology lies in the fact that a large number of designs valid for the intended application can be machine generated allowing a system to explore a wide variety of configurations.

The rest of the paper is organized as follows: The microstrip antenna shape grammar is first described followed by a discussion on how shape attributes and grammar labels are used to obtain an approximate electrical analysis of the structure; next, the whole process is demonstrated with an example; in the final section conclusions and future work are discussed.

II. The Compact Microstrip Antenna Shape Grammar

A shape grammar consists of four components; the initial shape, a finite set of shapes, a finite set of symbols and a finite set of shape rules. A labeled shape is defined as a shape augmented by symbols. By successively applying the transformation rules to an evolving shape, a design is derived in the language specified by the shape grammar. Parametric shape grammars extend shape grammars to deal with shapes whose dimensions can be arbitrarily specified. Additionally *weights* can be specified for the various shapes and used to redefine shape boolean operations. *Weights* ensure that certain components are not written off by newly introduced components. The work described in this paper makes extensive use of symbols, while weights and parametric shapes are implicitly included in the rules.

The compact microstrip antenna shape grammar is to generate designs that fit in the 2D design space available and satisfy the electrical specifications to the point that the structures generated can be either efficiently optimized with the aid of a full-wave numerical model or combined further to evolve reconfigurable structures.

In order to fulfill these tasks this system should (1) ensure the inclusion of all the components necessary for the antenna to function, (2) explore the space available so as to be able to generate the shapes, and (3) trace the electrical current paths from the driving point to the radiating edges so as to be able to obtain approximate quantities for the electrical characteristics.

In this paper the grammar is limited to describe patch shapes that can be generated in 2D from an array of small square shapes or *pixels*. Fig.2 shows some example shapes that can be generated from an 12×10 array of square pixels. The dimensions of the square pixel and the size and boundaries of the array are variable and this arrangement can scale-up and accommodate many real-life cases. This system of shape representation is chosen to simplify the problem of reasoning and interpretation of shape as well as shape algebra. This representation has been often applied in studies that make use of a genetic algorithm to evolve a shape that yields the required electrical and physical characteristics, such as in device miniaturization [14] or for wide-band designs [10].

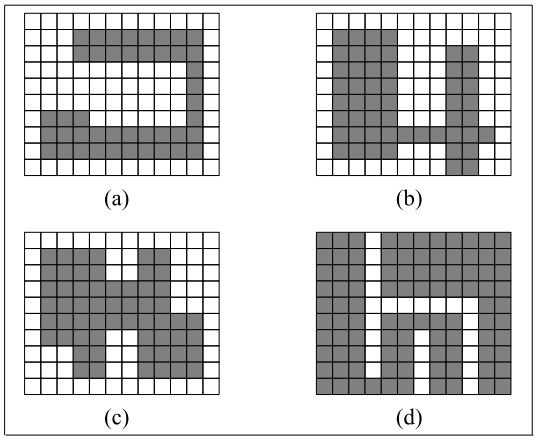


Figure 2. Shapes generated from a matrix of square pixels. The grammar described in this paper limits the number of shapes that can be generated. Shapes (a) and (d) conform with the grammar in this paper.

In this section the basic grammar that generates one pixel-wide meander line shapes and explores the design space is described first. The grammar is then extended to evolve these initial shapes to shapes characterised by wider sections, like the ones shown in fig.1(c & d) and fig.3(c). This is followed by an explanation on how feedback is coupled with the shape grammar to synthesize or generate valid shapes and results are used to demonstrate the system.

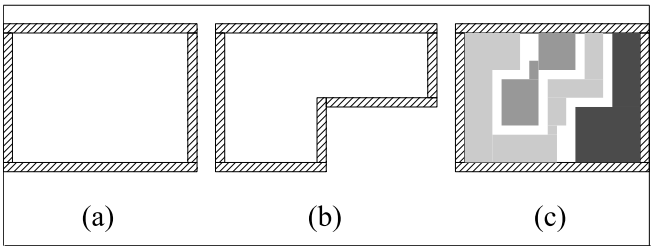


Figure 3. Examples of 2D design spaces (a) rectangular shape, (b) irregular space, and (c) Space leftover by other antenna elements - a 2D-space optimisation problem.

A. The Basic Shapes and Symbols

The proposed grammar is a parametric 2D grammar augmented by symbols. Fig.4 shows the basic shapes and symbols defined in this grammar. The basic shape is the square pixel and rectangles emerge from the union of adjacent pixels and are labeled as such. In this grammar rectangles are defined in terms of pixels rather than as a

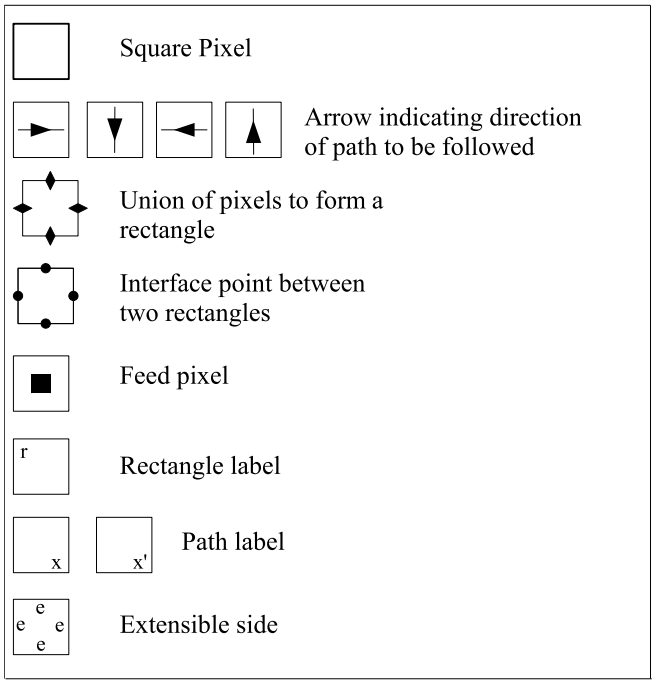


Figure 4. Grammar elements symbols and labels.

union of lines, as in [8]. The parametric term in this grammar relates to the width and length of the rectangle in pixels. Fig.5(a) shows an example of a rectangle. The symbol 'r' is a unique integer number given to an emergent rectangle as defined by the shape rules. The diamond symbol together with the arrows define how the pixels are connected to form a rectangle. The arrows define the general direction to be

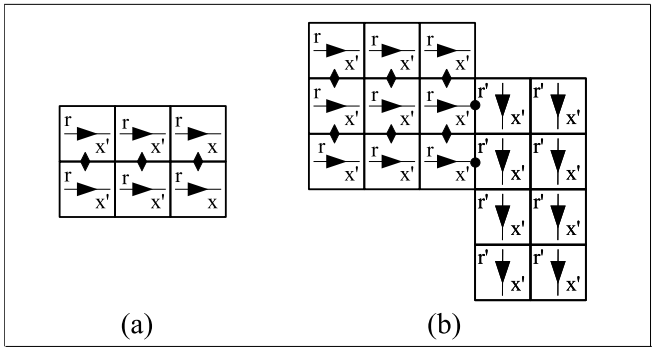


Figure 5. (a) An example of how a rectangle is defined, (b) an example of how adjacent rectangles are defined and connected.

followed and the diamonds define transverse paths. Fig.5(b) depicts two rectangles adjacent to each other and also defines the interface points, with the *dot* symbol. The rest of the labels and symbols in fig.4 are explained in the next section.

B. The Initial and Line Grammar Rules

Physically the compact microstrip antenna is made up of a patch of metal supported on a substrate and attached to a probe feed. The substrate is further backed by a ground-plane. For the structure to perform as an antenna, these four elements must be present; the feed or driving point, the patch, the substrate and ground-plane. Optional components like shorting posts and capacitive patches can also be added. In this grammar the substrate and the ground-plane are assumed to be infinitely large and assumed to be implicitly present. The inclusion of the feed point and the shape are enforced by the grammar. The grammar evolves a shape as two branches coming out of the probe feed or a shorting post. The branches are labeled *a* and *b* and the interface point between them is located at the probe feed or shorting post. This setup is suitable to model structures resonating at the fundamental mode and can also be used to model shorted patch structures. These two types of structures are the most common building blocks used in the design of compact microstrip antennas.

Fig.6 shows the initial rules that define the initial patch shape consisting of a probe feed and two branches. *Rule 1*

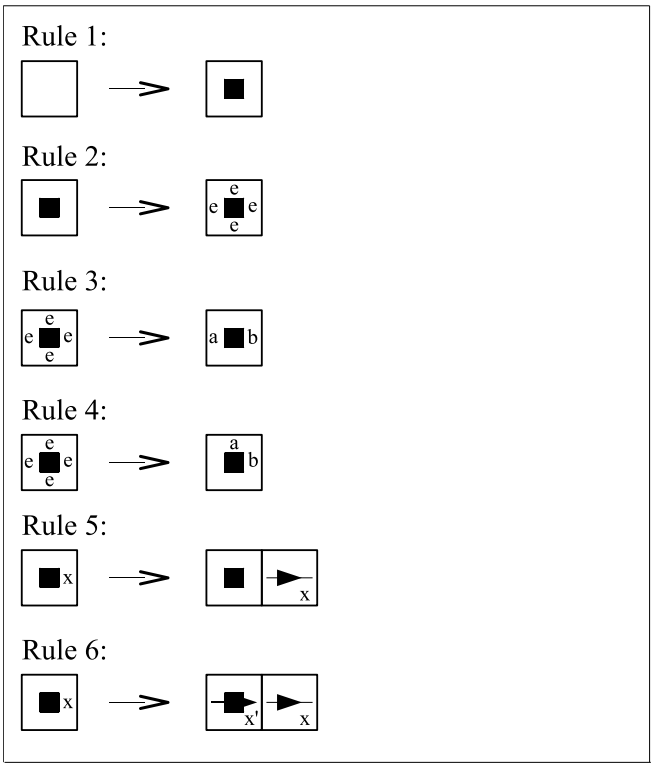


Figure 6. The initial rules.

defines a pixel as a feed pixel. *Rule 2* labels the extensible edges with the label *e*. Extensible edges in this context means that adjacent pixels can be appended to the current pixel given that these do not touch other elements belonging

to other branches or systems. *Rule 3* or *4* define the starting points for the two branches 'a' and 'b'. *Rule 5* attaches a pixel to one of the chosen sides and the first rectangle is defined. *Rule 6* attaches a pixel to the other chosen side and the second rectangle is so defined. This rectangle is composed of two pixels. The initial shape is thus defined and is composed of two branches labeled *a* and *b* and the location of the probe feed. The radiating edges or pixels are defined by labels *a* and *b*. Some possible starting shapes are shown in fig.7.

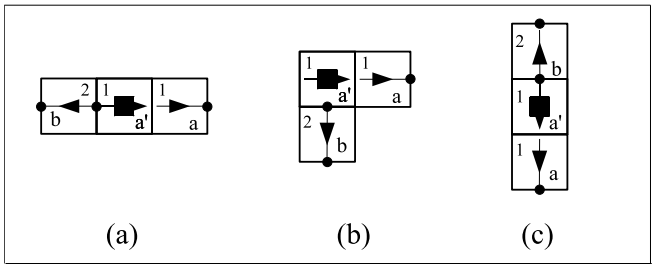


Figure 7. Some initial shapes using the initial shape grammar of fig.6.

The initial shape is further evolved or extended using the line generation rules given in fig.8. The rules are constructed using the shape and label algebra together. The rules as depicted here also assume that the side to be extended is indeed extensible. One way to implement this is to include the states of the neighbouring nodes in the rule, as depicted in fig.9. For the rest of the rules given in this paper the provision of a mechanism that is aware of the state of the neighbouring pixels is assumed and no further comments on this issue are given. Furthermore fig.10 defines three operators that transform rules to suit different orientations and an example of the *rotate* operator is given.

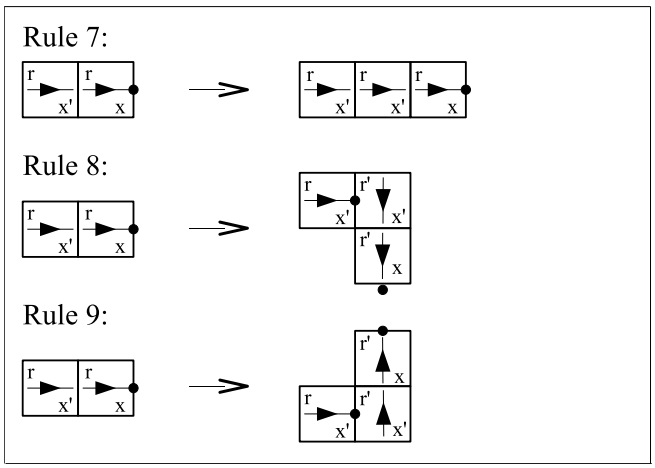


Figure 8. Grammar rules that evolve one pixel-wide meander lines.

Fig.11 illustrates the successive application of the *line generation rules* and how rectangles emerge and replace or

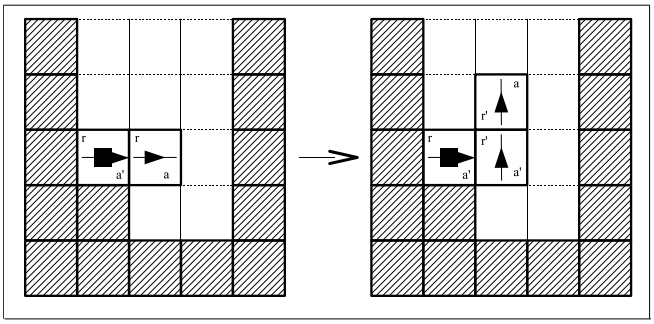


Figure 9. *Rule 9* defined with neighbouring pixels included.

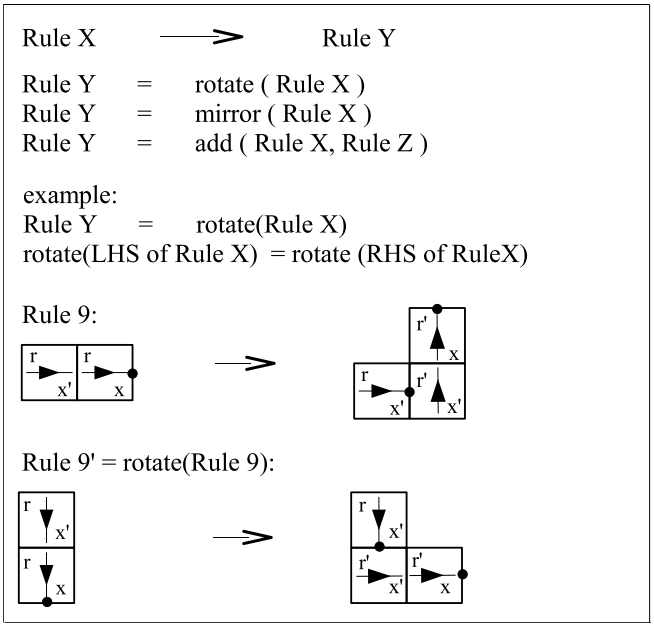


Figure 10. Rule tranformation operators and example for the *rotation* operator. This operator does not extend the shape but modifies a rule, such that it is applied in a different orientation.

modify previous ones. The arrow label indicates the direction of where next pixel is to be found. In this case rectangles are limited to a width of one pixel in one dimension. The basic shape grammar of fig.8 therefore generates the shapes discussed in [1]. The grammar ensures that two and only two branches a, b are always present; the shape evolves by considering the extension of a branch by one pixel; and a branch is extended if the new pixel to be added does not touch any other pixel that is *present* except for the one at the end of the branch. The shape grammar therefore ensures that (a) the overall shape of the patch is a meandered line of width one pixel, (b) there are no loops along the line, and (c) one feed is always present.

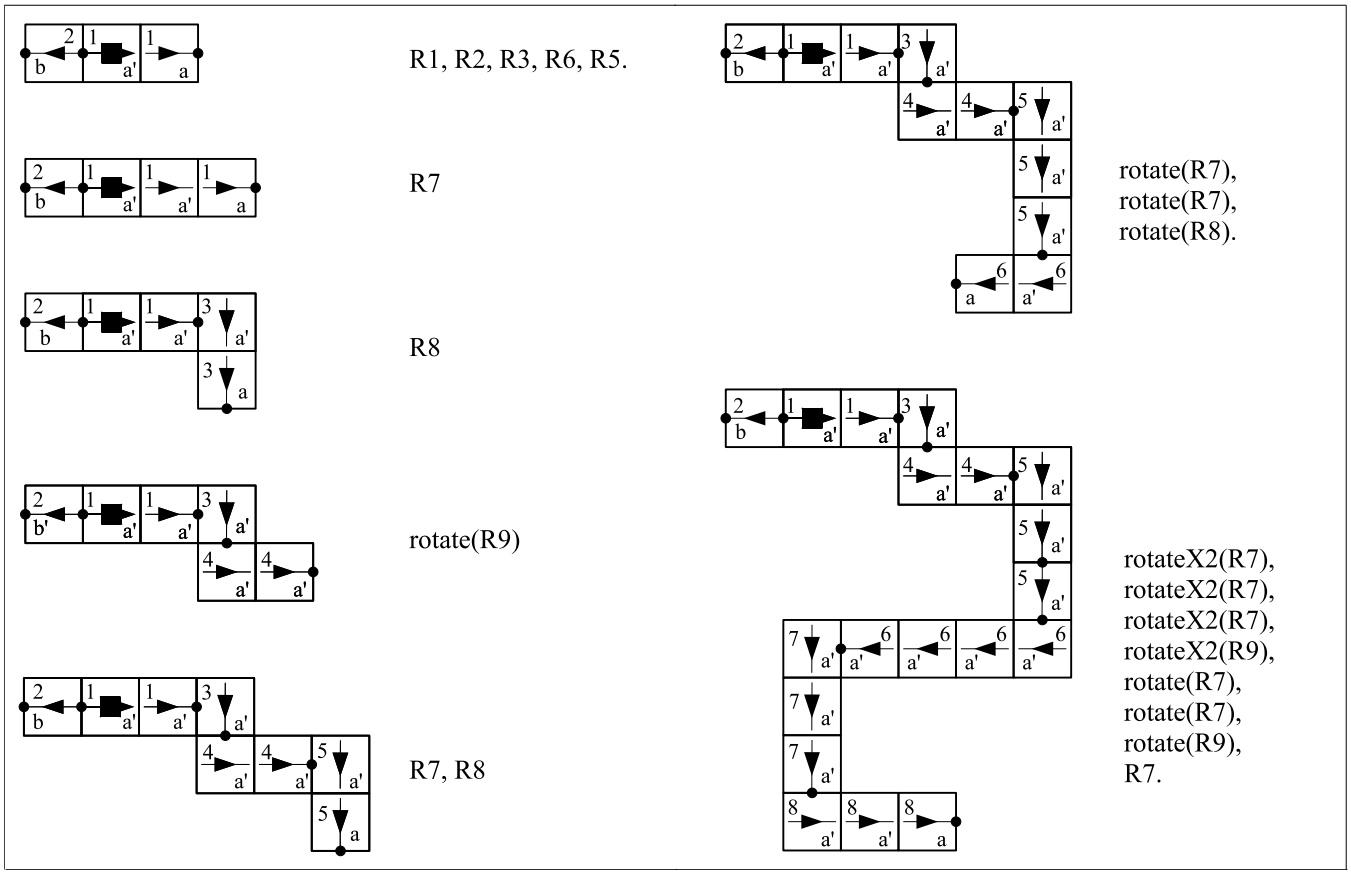


Figure 11. Application of the *line grammar* to evolve a shape.

C. The Extended grammar

The grammar described in the previous section is limited to shapes made of lines that are one pixel wide. It is desired that the grammar generates meander lines that are wider than one pixel and of non-uniform width, commonly used to optimise bandwidth and the driving point impedance. This section extends the grammar with a set of rules that further evolve a shape originally developed by the line grammar and is termed the *extended grammar*. Additionally the line grammar is implicitly used to explore the design space and therefore takes into consideration other concurrent shapes common in multi-band designs. The initial shape for the extended grammar is therefore the final shape evolved with the line grammar rules. The final shape for the extended grammar is a line defined by a chain of rectangles rather than pixels. Subsets of the extended grammar can also be defined to generate specific shapes, like for example rectangular shapes, L-shapes and C-shapes.

The extended grammar starts with a modification to the line grammar. *Rule 4* is removed from the initial rule set. Fig.12 shows *Rule 10* and *Rule 11* that define how the line in the vicinity and including the probe feed is extended.

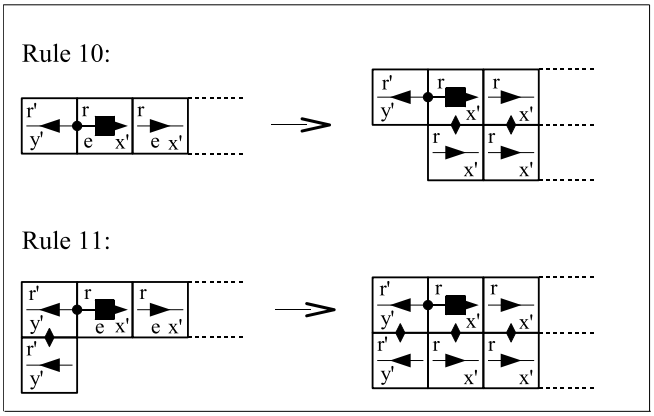


Figure 12. Shape rules in the vicinity of and including the probe feed.

Fig.13 and fig.14 give the rules for widening the line at corner sections, while fig.15 defines how branch ends are widened or extended. The rules given in figs.12, 13, 14 & 15 must be applied in pairs. Fig.16 combines rules to widen a straight run, a U shape and an S-shape. Likewise in fig.17 an S-shape is evolved into a straight run and the branch end

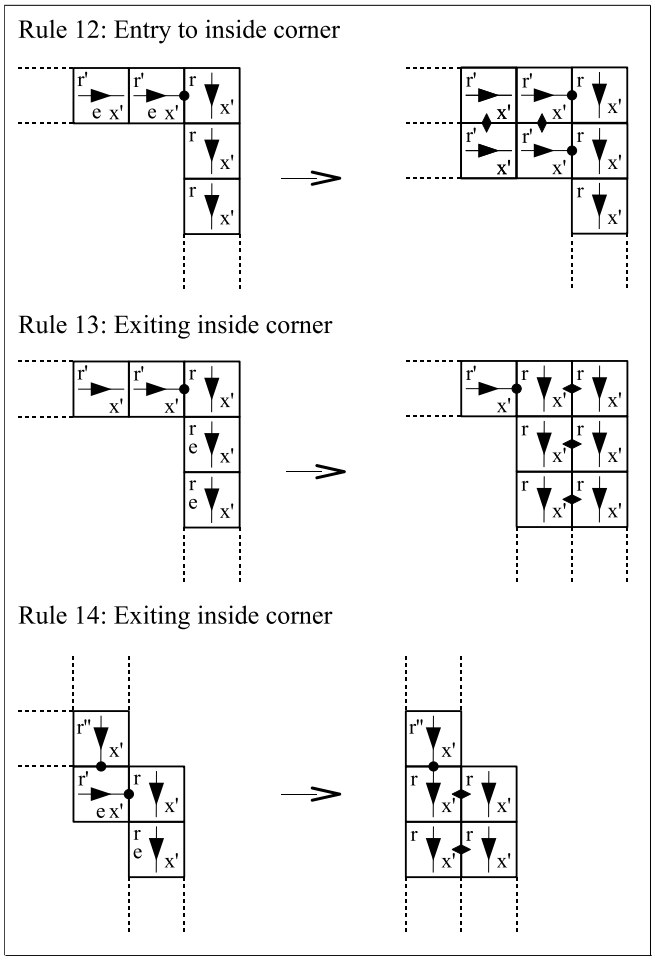


Figure 13. Shape evolution at the inside of corners.

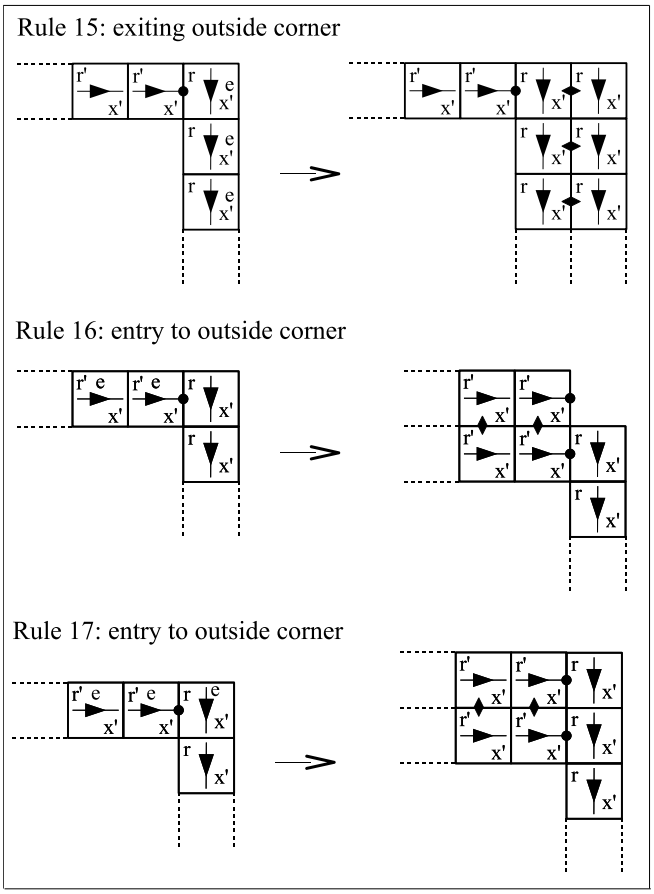


Figure 14. Shape evolution at the outside of corners.

is widened. The rules update the symbols accordingly and the chain of rectangles comes out in an implicit way as a product of the application of the rules. Fig.18 shows how the staircase line evolves into a straight *tapered* run consisting of rectangles of different widths. At this point no further rules can be applied on the right hand side other than *Rule 19* at the end of the branch. The end result may not be desired in practice but it demonstrates how rectangles emerge and replace other rectangles, while at the same time maintain the flow of arrows along the path.

In general an edge is extended as follows: an edge for extension is first chosen, based on extensibility and priority or a random list; this is followed by a search for a rule that matches the section and if the search is successful the change is carried out. Probabilities in the rule selection can be used to influence the path taken by the line grammar. Fig.19 gives some examples of shapes generated by the shape grammar. Subsets of the grammar can also be defined to describe particular classes of shapes, for example L-shape, C-Shape, G-shape. These shapes can be recognized from the number of corners and the sense of arrow directions. These features can be defined as a subset of rules, some of which having a limit on the number of rule instances that can be applied. Other shapes having other features like for example, wider branch ends than the rest of the line can be defined as a sequence of rules, that when applied generate these specific shapes.

D. The Grammar in Analysis

The antenna electrical properties of major interest are the frequency of operation, input impedance, efficiency, radiation pattern and bandwidth. In compact antenna design, radiation patterns are usually not a design variable since the designer has very limited ability to control the patterns [16] and most often exhibit omni-directional properties characterised by considerable cross-polar levels. Efficiency is a function of many variables including materials used and bandwidth is mostly a function of physical configuration, including shape characteristic and is catered for by the sequence of the grammar rules applied as described in the previous section. Furthermore, in practice, the bandwidth is usually enhanced with a matching circuit[17]. The frequency

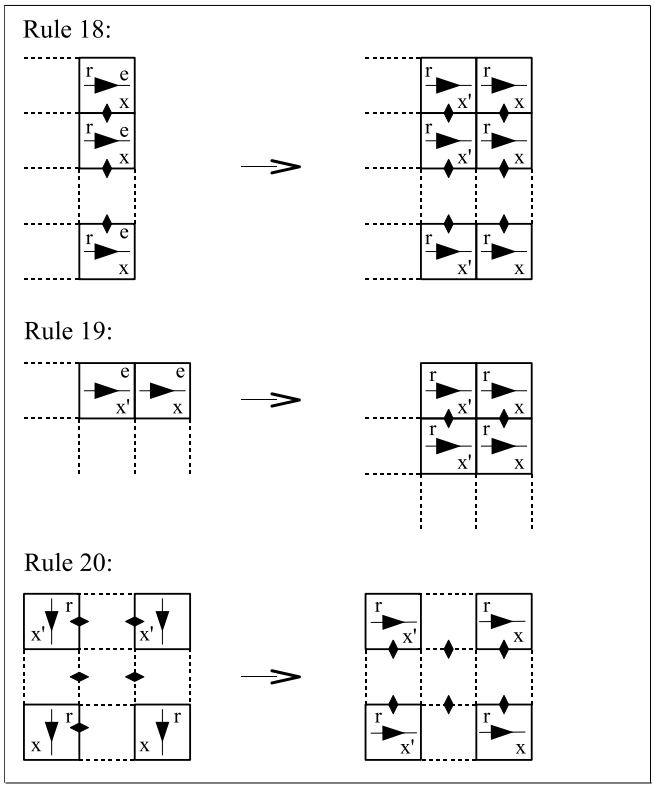


Figure 15. Shape evolution at branch end.

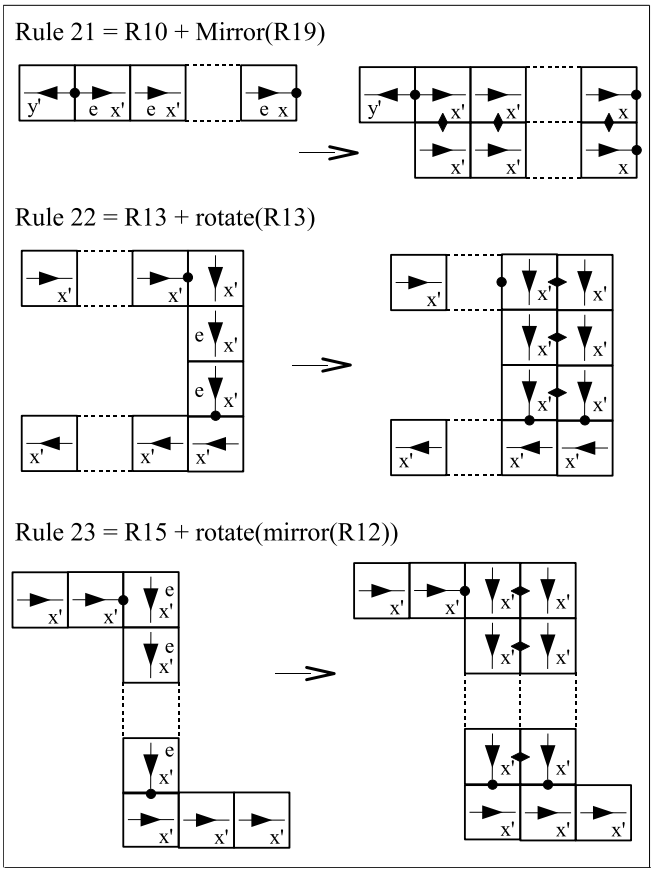


Figure 16. Set one of Modification rules in pairs.

of operation and the input impedance are the two properties that need to be considered explicitly in the analysis task and this section describes how these are estimated.

In theory the microstrip antenna resonates at an infinitely number of modes. However it is mostly used in its fundamental mode which is excited when the size of the patch is roughly half a wavelength long in at least one dimension, or quarter of a wavelength long for shorted patches. This is the concept on which the simple and natural transmission line model for the microstrip antenna is built. The transmission line model treats the antenna as a transmission line with open or shorted ends. Additionally the height of the substrate and size of the ground plane have some minor effects on the value of the frequency of resonance. The input impedance is modeled on a standing wave analysis that gives the impedance as a function of position. The effect of the patch width has a considerable effect on the input impedance and the height of the substrate usually effects the inductive part because of the higher inductance of the probe feed. These models are used as *guides* by the compact antenna designer during the initial design stage where exactness and high accuracy are not the priority. The designer follows *rules of thumb* to relate the shape of the compact antenna to these models and obtains an approximation of the electrical characteristics prior to optimisation using a numerical model. The feedback mechanism in the generative grammar reported in this paper models this analysis process and therefore can be thought of as a formalization of the transmission line model applied to the analysis of the shapes generated by the shape grammar, i.e. shapes that range from straight lines to irregular meander lines.

The main task for the algorithm is to analyze the shape and link its *geometrical attributes and dimensions* to the transmission line model. This is carried out by decomposing the shape into rectangles and tracing the current paths that are required to estimate the resonant frequency and the input impedance. These two tasks are carried out implicity by the grammar rules. The next step is to pick up the shape attributes that are used as inputs in the formula models to yield the electrical properties. The previous discussion highlighted that the main attribute used in the calculation of the frequency of resonance as well as the input impedance, is the *effective length* of the current path. Other physical characteristics of the structure that have less effect on the electrical characteristics are the height of the substrate and the shape itself - how *compact* the shape is as well as the aspect ratios of the sub-shapes.

The mechanism of this process is first discussed and

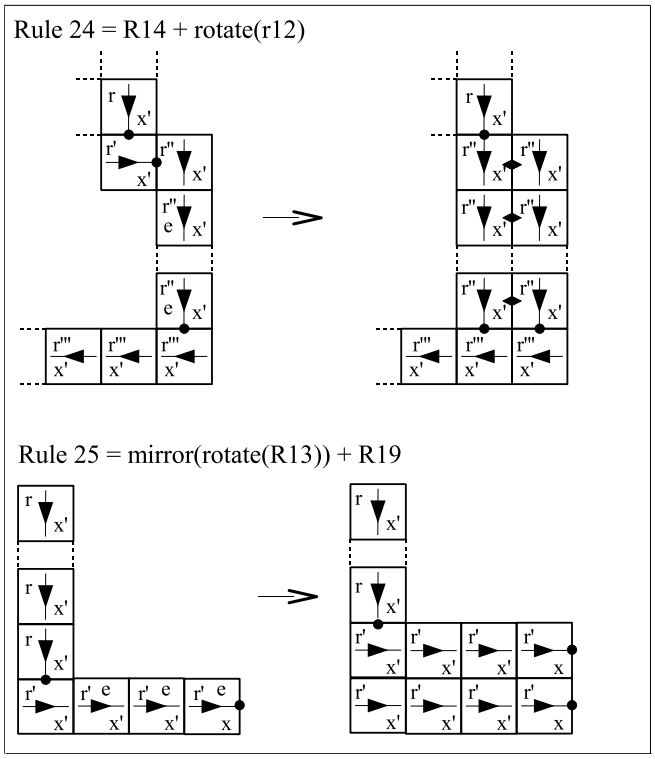


Figure 17. Set two of Modification rules in pairs.

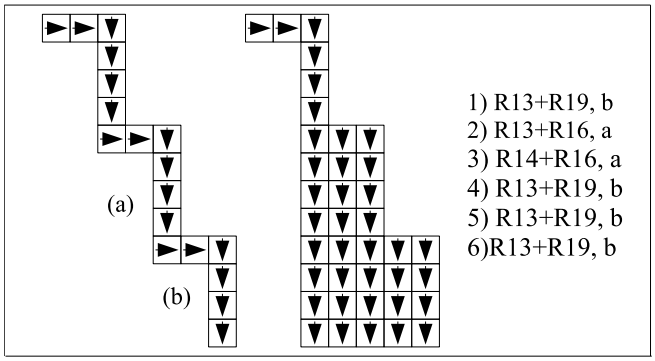


Figure 18. Stair-case line evolved to a quasi straight run by applying the extended grammar

demonstrated for shapes derived from the grammar of fig.8, and later for shapes derived by the whole grammar, i.e including the extended grammar. A set of metrics is adopted and listed below, where reference is made to fig.20.

1) Number of corners along the meandered line. This is split into two. NC_a is equal to the number of corners along branch a, and NC_b is equal to the number of corners along branch b.
2) Length of meandered line in terms of pixels. This is split in two. L_a is equal to the number of pixels between A and B, and L_b is equal to the number of pixels between A and C.
3) Separation in space of the two end points, B and C. This is equal to distance BC.
4) Angle subtended by lines AB and AC. This angle is equal to θ.

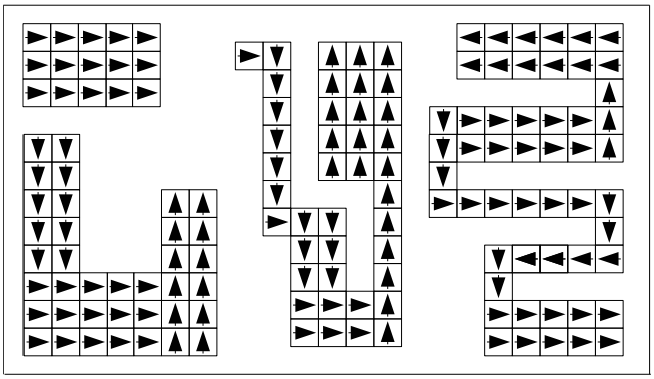


Figure 19. Examples of shapes generated by the shape grammar.

Metrics 1 and 2 are used in the calculation of the *effective lengths* and metrics 3 and 4 give an indication of the compactness of the structure.

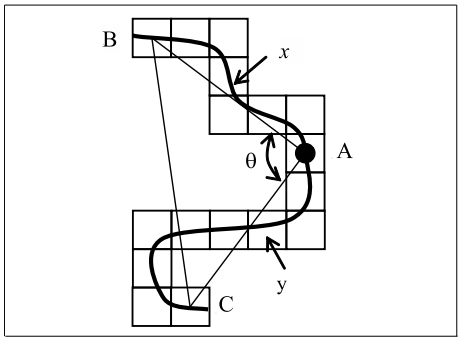


Figure 20. Metrics that are used to obtain properties of the shape.

The next step is to relate the metrics to the electrical properties, so as to be able to carry out the approximate quantitative analysis desired for the feedback mechanism. In general the relationship can be described by a weighted function of the shape properties as,

$$EP_n = \sum \textit{Weighted Shape Metrics,} \qquad (1)$$

where EP_n is the electrical property under consideration. For this purpose a set of 20 random elements are generated from the grammar without feedback of fig.8. Fig.21 shows the algorithm used to generate these elements. Table I lists the data collected for the 20 elements. The shape is described by the metrics, and the electrical properties are obtained with a Finite-difference-Time-Domain (FDTD) numerical tool developed by the author [18]. The next step is to select the shape properties that have a strong influence on

01 Generate initial shape and define branches
02 Choose a branch to extend. The choice is based on a random number and the probability of a branch being selected more often than the other is variable
03 Check if branch is extensible: If YES repeat step 02. If no extensible branch can be found terminate element
04 Extend branch in a random extensible direction
05 Decide with a probability of $P_{continue}$ whether to continue extending the branches. If YES goto step 02 else terminate element

Figure 21. Algorithm Generate Random Elements

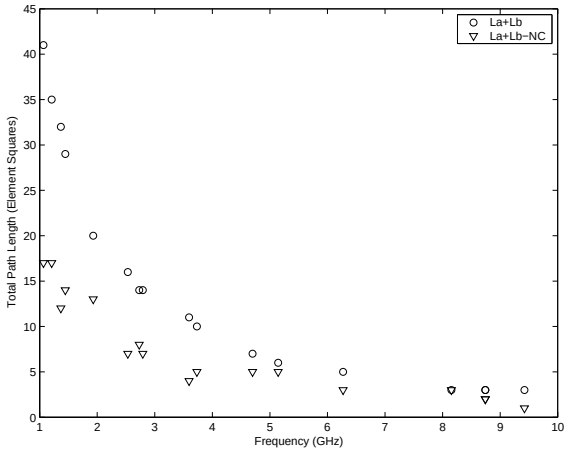


Figure 22. Scatter plot for the resonant frequency.

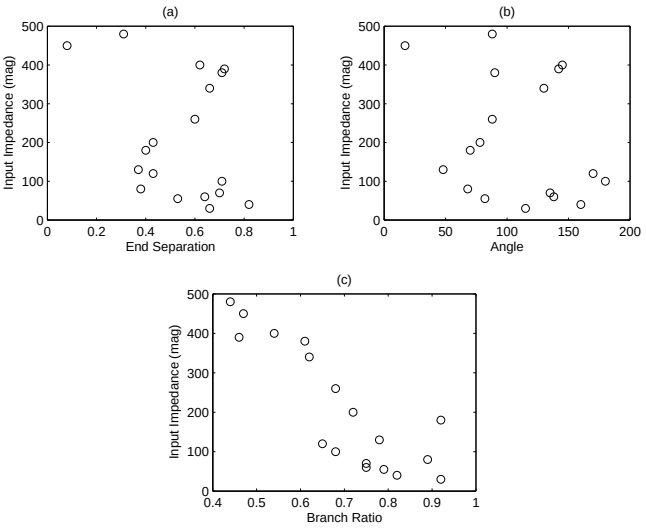


Figure 23. Scatter plots for the input impedance.

each electrical property. In general this could be posed as a regression problem. In this paper, microstrip antenna design experience discussed previously supported by scatter plots are used to guide the selection of the most significant metrics to approximate the resonant frequency and driving point impedance.

The resonant frequency, f_0 is dependent mostly on the effective length of the meandered line given by $(path_x + path_y)$ in fig.20, which in turn is a function of L_a, L_b and NC. Scatter plots for these variables against f_0 confirm a higher dependence on the branch lengths than on the number of corners present, fig.22. The relationship for f_0 derived by considering the simple transmission line model is,

$$f_0 = 3 \times 10^{-4}/((L_a + L_b + 2a_0 - (NC_a + NC_b) * a_1) * L_p * 2) \quad (2)$$

The coefficients a_0 and a_1 are obtained by minimizing the error and L_p is the width of the square pixel in millimeters. The angle θ and distance BC are ignored in this equation because the frequency's sensitivity to these is minimal.

The driving point impedance depends mostly on the relative position of the feed point along the current path. Another set of scatter plots, fig.23 confirm that, Z_{in}, depends on the relative position of the probe feed to the ends of the line. The scatter plots for the angle θ and the normalized BC metrics show a weak and inconclusive relationship with Z_{in} and therefore are not included in the model. The model does not yield a numerical value for Z_{in} but gives an indication of how far away it is from 50Ω, the system impedance. The input impedance estimate is obtained by considering the $B0 : B1$ ratio from which it is derived that when the ratio is close to 0.85 the input impedance is in close proximity to 50Ω. These two relationships are used to obtain an estimate for the frequency of resonance and the input impedance level for the structures of I. The set of designs in table I group is termed the *Tuning Set* as it is used to obtain the coefficients a_0 and a_1 in eq.2 using a trial and error approach. The values for the optimised coefficients are $a_0 = 0.6$ and $a_1 = 0.31$. The last two columns in table I give the estimated frequency, and the error. The average error is 2.96% with a standard deviation of 2.03.

A second set elements was generated using the same algorithm as that for the *tuning set*. This second set is used as a *Validation Set*. The average error in the estimated frequency is 3.15% with a standard deviation of 1.85.

The randomly generated elements for the tuning set span the frequency range from approximately 1 to $7GHz$ and a histogram of these shows that their distribution is approximately uniform especially in the 1 to $4GHz$ range. The model is therefore fitted to this wide frequency range. A plot of the averaged error against frequency shows a gentle downward slope favoring the high frequencies. The same observations are noted in the validation set. If a smaller error

Table I
TUNING SET

#	L_a	L_b	NC	NC_a	NC_b	BC	Θ	$B0:B1$	$B0/(B0+B1)$	F_0	Zin	f_0	$Error$
0	10	5	9	6	3	33.4	88	0.56	0.63	2.67	hi	2.72	1.70
1	4	2	1	0	0	17.6	90	0.61	0.66	5.14	me	5.14	0.08
2	4	1	3	2	0	14.5	132	0.47	0.71	6.4	me	6.61	3.34
3	10	3	7	6	1	33.4	129	0.41	0.71	2.85	hi	3.02	5.86
4	12	11	13	7	5	25.2	52	0.97	0.32	1.82	lo	1.82	0.17
5	11	6	6	4	2	32.1	72	0.60	0.50	2.3	hi	2.24	2.60
6	5	4	5	1	3	17.0	81	0.72	0.50	4.2	me	4.14	1.34
7	2	9	6	1	4	32.1	159	0.31	0.79	3.4	me/hi	3.49	2.69
8	4	2	3	2	0	17.6	111	0.69	0.73	5.5	me	5.62	2.22
9	9	3	5	2	2	12.6	64	0.37	0.28	3.05	hi	3.11	2.03
10	12	15	14	6	8	12.0	16	0.83	0.13	1.55	lo	1.55	0.27
11	8	2	7	6	0	28.3	122	0.43	0.80	3.83	hi	3.98	3.83
12	11	4	7	4	2	37.8	137	0.41	0.68	2.64	hi	2.60	1.56
13	15	2	7	6	1	44.1	180	0.20	0.70	2.2	hi	2.28	3.75
14	12	31	18	2	15	20.2	29	0.45	0.13	1.02	hi	0.96	5.78
15	12	9	7	3	3	29.0	57	0.75	0.37	1.92	me	1.84	4.40
16	27	8	23	16	6	36.5	78	0.31	0.32	1.29	hi	1.27	1.36
17	13	10	9	7	1	32.1	55	0.90	0.38	1.8	me	1.72	4.48
18	7	4	2	2	0	22.0	15	0.68	0.48	3.27	me/hi	3.13	4.27
19	18	16	18	11	7	28.3	71	0.95	0.24	1.35	lo	1.25	7.47

is desired the model can be fitted over a smaller frequency range. In this case the grammar will have associated with it a set of coefficient vectors from which the appropriate one has to be selected. In the modelling mode this has to be carried out at least over a two step iteration. However for the initial design phase a smaller error is generally not required.

The model is tested in the synthesis task. The simple synthesis algorithm in fig.24 was developed for this purpose. The task of the algorithm was to develop a structure that

01 Set frequency of resonance and desired input impedance
02 Start Synthesis
03 Generate initial shape at a random position
04 Obtain an estimate for the input impedance
05 If the input impedance estimate is greater than the target randomly pick one branch that can be extended, else pick the shortest branch that can be extended
06 If neither branch is extensible then terminate synthesis
07 Extend branch in a random extensible direction
08 Obtain an estimate of the resonant frequency: If estimate is less than the desired value terminate synthesis else repeat as from step 04

Figure 24. Algorithm Synthesize Elements

resonates at $2.0GHz$ and provides a close match to 50Ω. The algorithm terminates either when the specifications are satisfied or when the shape cannot be changed any more. Further more, since the feed is fixed at a given position, it may be the case that the desired input impedance is unattainable. In this case the algorithm does not attempt to shift the feed. Table II lists ten structures that have been developed by this algorithm. All ten cases were simulated

Table II
SYNTHESIS SET

#	Freq Model	Freq FDTD	Prediction Error	Target Error	Branch Ratio	Zin FDTD	Target Error
0	1.954	2.03	3.744	2.300	0.881	53.0	6
1	1.947	2.01	3.132	2.648	0.912	13.0	74
2	1.954	2.02	3.260	2.293	0.963	15.0	70
3	1.986	2.01	1.183	0.689	0.993	35.0	30
4	1.994	2.07	3.690	0.300	0.939	44.0	12
5	1.923	1.91	0.685	3.846	0.851	45.0	10
6	1.994	2.10	5.048	0.300	0.869	57.0	14
7	1.986	2.00	0.689	0.689	0.902	80.0	60
8	1.994	2.08	4.153	0.319	0.869	60.0	20
9	1.923	1.98	2.875	3.846	0.879	40.0	20

with the FDTD numerical tool and the average error in the resonant frequency is 2.85% with standard deviation of 1.5, whereas the average deviation from the $2.0GHz$ desired is 1.72% with a standard deviation of 1.44. The average error for the input impedances is 31.8% with a standard deviation of 26.2. This figure seems high. However, the input impedance is a very sensitive quantity and for this type of model it should be considered sufficiently accurate. Furthermore the accuracy required depends on where and how the model is utilized in practice. Certainly it is an area for further study.

The method described above cannot be applied directly to shapes evolved with the extended grammar and a general

method that can be applied to any shape is desired. The way the *effective length* is calculated is modified. Fig.25(a) demonstrates how this is done. The *dot* symbols that define the interface points for the rectangles are joined together by straight lines and the summation of the length of these lines gives the initial effective length, which is equivalent to $(L_a + L_b)$ in eqn.2. The final effective length is corrected in the same way as in the previous section, i.e. by considering the number of corners as well as the coefficients a_0 and a_1. To test this method the tuning set is used and the updated coefficients are $a_0 = 0.55$ and $a_1 = 0.02$. The average error is 2.82% with standard deviation of 1.95. The average error for the validation set is 3.32% with a standard deviation of 2.02%.

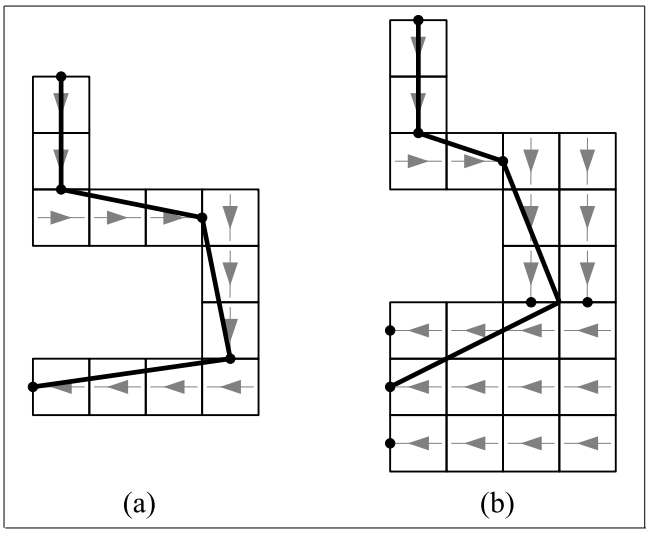


Figure 25. Evaluation of the *effective length*.

The accuracy of the formula model (eqn.2) when applied for shapes evolved with the *extended grammar* is discussed next. Three slightly different approaches are considered in calculating the *effective length* for the branches; (1) $(L_a + L_b)$ is equivalent to the minimum length end to end path passing through the sub-shapes interface points and the feed point, (2) $(L_a + L_b)$ is equivalent to the length of the path passing through the midpoint of each sub-shape interface zone (fig.25(b)), and (3) $(L_a + L_b)$ is equal to the sum of the lengths of all the rectangles that make up the shape. A set of fifty prototypes that operate over a frequency range of 1 to 4 GHz and various shapes ranging from rectangular to L-shapes and meander lines are simulated with the FDTD model and used as a second tuning set. This set is used to optimise the a_0 and a_1 coefficients. For approach (1) $a_0 = 0.60$ and $a_1 = 0.30$ and the average error is 6.73% with a standard deviation of 6.90. For approach (2) $a_0 = 0.55$ and $a_1 = 0.20$ and the average error is 4.72% with a standard deviation of 3.19, and for approach (3) $a_0 = 0.60$ and $a_1 = -0.28$ and the average error is 5.75% with a standard deviation of 8.13. For (1) and (2) the path length is underestimated and therefore the a_1 coefficients are negative. The results show that approach (2) performed best. Intuitively this approach is the closet to approximating the current path along the antenna structure. However approach (3) has the advantage of being less computationally intensive. It is also possible to obtain higher accuracies if the set is restricted to a particular type. The errors are greatest for the most irregular shapes and when the interface width between rectangles is greater than 1 pixel. The simple formula model does not take into account the various widths of the rectangles and how they scale relative to each other. The accuracy is however adequate for its intended application, i.e for the conceptual design stage. This does not mean that a better model is not possible and this issue is further discussed in the last section.

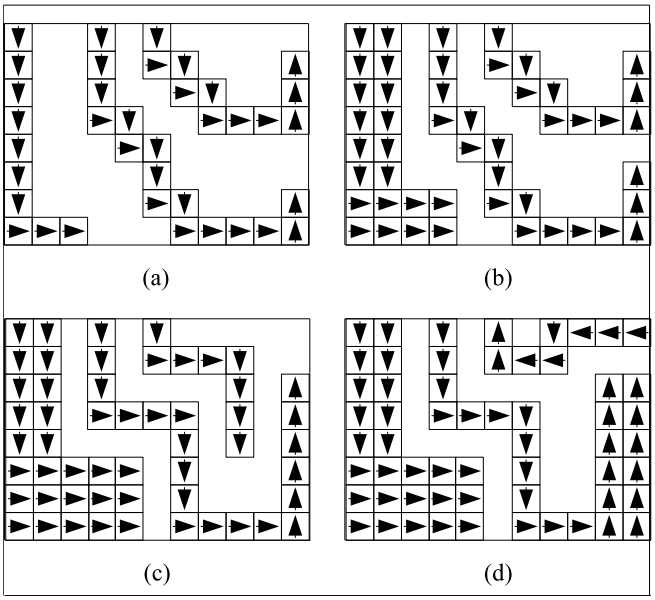


Figure 26. Stages during the design process of a tri-band separately fed structure.

This section ends with a multi-band structure. The design of multi-band structures can be either evolved separately or simultaneously. Fig.26 depicts an extracted sequence of interim designs during the evolution of a three element antenna with separate feed structures. Fig.26(a) is the result of applying the line grammar rules to the three elements simultaneously. In fig.26(b) the structure on the left-hand-side is widened but is short from satisfying the specifications and can't be further evolved. Therefore in fig.26(c) the other designs are removed and the first structure is allowed to evolve, at which point the other structures are evolved again. Fig.26(d) shows the final design, that can be further evolved if necessary. The intention of this design is to operate as a single feed dual-band antenna at $0.925GHz$ and $1.8GHz$

and a separately fed antenna for $2.45GHz$. These frequencies correspond to cellular licensed mobile communications bands and the unlicensed Industry, Scientific and Medical (ISM) band. The frequencies of operation as predicted by the shape grammar in analysis are $1.86GHz$, $1.05GHz$, and $2.47GHz$, while the respective deviations from the target values are 3.5%, 13.2% and 2.1%. These deviations are not due to errors in the analysis model, but are function of the stopping criteria in the application process of grammar rules. The two major shapes derived by the grammar are joined together to create a single feed dual-band structure and shorting planes are added. The third shape is fed separately. The height of the substrate is $4mm$ and $\varepsilon_r = 1.0$. The resulting structure, fig.27, is analyzed numerically (FDTD). The frequencies obtained from the FDTD model for the three fundamental modes are $0.85GHz$, $1.69GHz$, and $2.52GHz$ and the deviations from the grammar model for the grammar derived shapes are 18%, 9% and 2%. The high deviation for the dual-band structure is due to the increase in length when the two shapes are joined together and due to the shorting plane which is shorter than the width of the line. The input impedance is also close to the system impedance of 50Ω for both feeds. The next step for the antenna engineer is to proceed to the second phase of the design and optimise the structure so as to deliver the final design prior to measurements. Potential variables for the optimisation stage are indicated in fig.27 with lines ended in arrows. The optimisation process does not change the character of the shape itself, but varies the dimensions of the sub-shapes or rectangles.

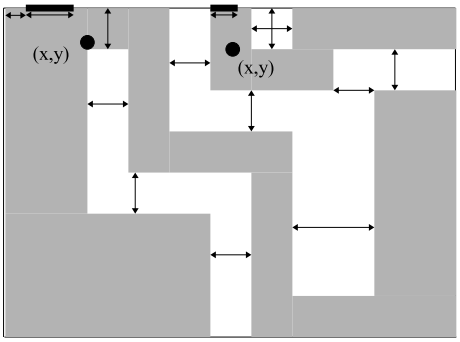


Figure 27. The numerical model ready for optimisation derived from fig.26(d). The arrows and positions for the probe feeds are suggested variables for the optimisation process. Possible variables for optimisation are indicated by the arrow-ended-lines.

III. Conclusions

This paper described an original microstrip antenna shape grammar that generates compact designs similar to meander line structures, as well as some standard shapes. The shape derived is represented as a chain of rectangles identified by labels and symbols. The decomposition into sub-shapes or rectangles is carried out by the grammar itself. The radiating edges are labeled and the nature of the shape can be derived from the orientation and sequence of the arrow symbols. A set of sub-shape and full-shape parameters are used as variables in a mathematical formula model that yields approximate values for the electrical characteristics. The result from the formula is used as feedback to guide the selection of rules and therefore shape evolution. The shape grammar is useful to synthesize and analyze compact microstrip antennas.

The shape grammar is a tool for the conceptual or initial design stage prior to the optimisation of the details. This tool formalizes and mimics some of the informal processes that an antenna designer goes through. The rule selection can be done by the machine to generate a wide variety of potentially useful designs and can form the basis of an Intelligent Computer Aided Engineering (ICAE) software. Forbus describes an ICAE system as a junior partner in the design team and the goal for the system is to capture a significant fraction of an engineer's knowledge that is later used to generate potential designs as well as modifications[19]. The shape grammar described in this paper provides such a tool and can be used to record design knowledge that can be used later. Alternatively the shape grammar can be integrated as part of a CAD software and the rules applied manually by the designer. The feedback mechanism automatically provides an estimate of the electrical properties of the antenna.

In this paper feedback is demonstrated with a simple weighted function. This function performs very well for the narrow line designs but its accuracy degrades when more variables are introduced. Not surprisingly the results show that the accuracy of the quantitative results improve if the formula is fitted over a narrow range of shapes and frequencies. Nevertheless, the accuracy of the model is still good enough for the initial design phase. On the other hand it is always desired to have a single model applicable to a wide range of devices. The inclusion of more variables and non-linear terms may improve the results. Of special interest is the use of Neural Network architectures (NN) as black boxes that replace the CAD formula. This model is suitably applied to well known and understood configurations. Geometrical dimensions are selected as inputs to a NN which in turn yields the electrical properties of the device. This approach has been shown to work for planar and microstrip antennas [20],[21],[22] and [23]. It would also be very useful to the designer if the model gives an estimate of how accurate the result is [24]. The work presented in this paper may be augmented with NN to automate the analytical derivation of the approximate model used in the generative system.

The work reported in this paper demonstrates the feasibility of a shape grammar based model that can act as an assistant to the designer. It would therefore be useful to extend this grammar to include other configurations and components such that it has a wider range of applicability.

Equally interesting should be the application of boolean shape operations to shapes generated by the grammar to extract sub-shapes that can be re-interconnected by switches. Such a tool would be very useful during the design of re-configurable antennas.

REFERENCES

[1] A. Muscat, "A shape-function grammar approach for the synthesis and modelling of pixel-microstrip-antennas," in *Advanced Engineering Computing and Applications in Sciences, 2009. ADVCOMP '09. Third International Conference on*, pp. 23 –28, 11-16 2009.

[2] J. R. Kelly, E. Ebrahimi, P. S. Hall, P. Gardner, and F. Ghanem, "Combined wideband and narrowband antennas for cognitive radio applications," in *Cognitive Radio and Software Defined Radio: Technologies and Techniques*, IET, 18th September 2008.

[3] G. Stiny and W. J. Mitchell, "The palladian grammar," *Environment and Planning B*, vol. 5, pp. 5–18, 1978.

[4] U. Flemming, "More than the sum of parts: the grammar of queen anne houses," *Environment and Planning B: Planning and Design*, vol. 14, pp. 323–350, 1987.

[5] P. A. Fitzhorn, "Formal graph languages of shape," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 4, no. 3, pp. 151–164, 1990.

[6] K. Shea and J. Cagan, "The design of novel roof trusses with shape annealing: assessing the ability of a computational method in aiding structural designers with varying design intent," *Design Studies*, vol. 20, pp. 3–23, 1997.

[7] M. Agarwal and J. Cagan, "A blend of different tastes: the language of coffemakers," *Environment and Planning B: Planning and Design*, vol. 25, pp. 205–226, 1998.

[8] M. Agarwal and J. Cagan, "A micro language: generating mems resonators by using a coupled form-function shape grammar," *Environment and Planning B: Planning and Design 2000*, vol. 27, pp. 615–626, 2000.

[9] M. Agarwal and J. Cagan, "On the use of shape grammars as expert systems for geometry-based engineering design," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 14, pp. 431–439, 2000.

[10] J. M. Johnson and Y. Rahmat-Samii, "Genetic algorithms in engineering electromagnetics," vol. 39, pp. 7–25, August 1997.

[11] E. A. Jones and W. T. Jones, "Genetic design of linear antenna arrays," *IEEE Antennas and Propagation Magazine*, vol. 42, pp. 92–100, June 2000.

[12] J. R. Koza, S. H. Al-Sakran, L. W. Jones, and G. Manassero, "Automated synthesis of a fixed-length loaded symmetric dipole antenna whose gain exceeds that of a commercial antenna and matches the theoretical maximum," in *GECCO*, (London, UK), pp. 2074–2081, ACM, July 2007.

[13] A. Muscat, *The Design of Low Gain, Wideband and Multi-band Antennas Employing Optimisation Techniques*, ch. Optimisation Based Design, pp. 120–174. Queen Mary University of London, January 2002.

[14] N. Herscovici, M. F. Osorio, and C. Peixeiro, "Miniaturization of rectangular microstrip patches using genetic algorithms," *IEE Antennas and Wireless Propagation letters*, vol. 1, pp. 94–97, 2002.

[15] A. Muscat and J. Zammit, "An efficient algorithm for the control of reconfigurable pixel microstrip antennas," in *Advances in Circuits, Electronics and Micro-electronics, 2009. CENICS '09. Second International Conference on*, pp. 44 –47, 11-16 2009.

[16] B. S. Collins, *Antennas for Portable Devices*, ch. Handset Antennas, pp. 9–57. John Wiley & Sons, 2007.

[17] S. Kingsley, D. Ireland, S. O'Keefe, R. Langley, and L. Liu, "In search of the perfect handset antenna," in *Antennas and Propagation Conference, 2008. LAPC 2008. Loughborough*, pp. 62 –65, 17-18 2008.

[18] A. Muscat, *The Design of Low Gain, Wideband and Multi-band Antennas Employing Optimisation Techniques*, ch. FDTD Model for The Patch Antenna, pp. 68–118. Queen Mary University of London, January 2002.

[19] K. D. Forbus, "Intelligent computer-aided engineering," *AI Magazine*, vol. 9, no. 3, pp. 23–36, 1988.

[20] K. C. Gupta, "Emerging trends in millimeter-wave cad," *IEEE Transactions On Microwave Theory And Techniques*, vol. 46, pp. 747–755, June 1998.

[21] H. J. Delgado and M. H. Thursby, "A novel neural network combined with fdtd for the synthesis of a printed dipole antenna," *IEEE Transactions On Microwave Theory And Techniques*, vol. 53, pp. 747–755, July 1998.

[22] S. Sagiroglu, K. Guney, and M. Erler, "Calculation of bandwidth for electrically thin and thick rectangular microstrip antennas with the use of multilayered perceptrons," *International Journal of RF and Microwave Computer Aided Engineering*, vol. 9, pp. 277–286, 1999.

[23] K. Guney, S. Sagiroglu, and M. Erler, "Generalized neural method to determine resonant frequencies of various microstrip antennas," *International Journal of RF and Microwave Computer Aided Engineering*, vol. 12, p. 131139, 2002.

[24] R. C. Booton, "Microwave cad in the year 2010 - a panel discussion," *International Journal of RF and Microwave Computer-Aided-Engineering*, vol. 9, pp. 439–448, 1999.

The Two-dimensional Superscalar GAP Processor Architecture

Sascha Uhrig, Basher Shehan, Ralf Jahr, Theo Ungerer
Department of Computer Science
University of Augsburg
86159 Augsburg, Germany
{uhrig, shehan, jahr, ungerer}@informatik.uni-augsburg.de

***Abstract*—In this paper we evaluate the new Grid Alu Processor architecture that is optimized for the execution of sequential instruction streams. The Grid Alu Processor architecture comprises an in-order superscalar pipeline front-end enhanced by a configuration unit able to dynamically issue dependent and independent standard machine instructions simultaneously to the functional units, which are organized in a two-dimensional array. In contrast to well-known coarse-grained reconfigurable architectures no special synthesis tools are required and no configuration overhead occurs. Simulations of the Grid Alu Processor showed a maximum speedup of 2.56 measured in instructions per cycle compared to the results of a comparable configured SimpleScalar processor simulator. Further improvements introduced in this paper lead to a top speedup of about 4.65 for one benchmark. Our evaluations show that with restricted hardware resources, the performance is only slightly reduced. The gain of the proposed processor architecture is obtained by the asynchronous execution of most instructions, the possibility to issue multiple depending instructions at the same cycle and the acceleration of loops.**

***Keywords*-High Performance Processors, Reconfigurable Architecture, Instruction Level Parallelism**

I. Introduction

Current processor research focuses especially on multi-core architectures. Unfortunately, these architectures perform only well with parallel applications but they cannot increase the performance of sequential programs. The **G**rid **A**lu **P**rocessor (GAP) presented in [23] is designed to improve the execution of sequential instruction streams.

In general, control flow based applications contain small data-driven code sequences that can be accelerated by special hardware. Because of the differing demands of the applications, a general accelerator for all requirements cannot be found.

Application specific processors together with hardware/software codesign methodologies [17] contain at least one functional unit or functional block, which can be adapted to a concrete application. Mostly, these functional blocks are generated at the hardware-design phase i.e., the special demands of the application's software need to be known statically. Generally, the special functions consist of very simple operations like a single MAC operation or something similar.

Dynamically reconfigurable systems [6] comprise an array of functional units that can be reconfigured during runtime. However, the configuration of the array has special demands on the development tool chain and reconfiguration is time consuming. Hence, replacement of the operation should be well scheduled, otherwise the replacement overhead exceeds the gain in execution time.

Another issue is the global clock synchronous execution, which hampers the execution of simple instructions, because the working frequency depends on the most complex operation performed in any pipeline stage therefore decelerating simple and fast instructions. Our approach applies asynchronous timing between functional units, which allows the execution of simple operations in less than a cycle.

The GAP Architecture mixes the advantages of a superscalar processor and those of a coarse-grained dynamically reconfigurable system. A special configuration unit issues synchronously instructions to the processor back-end, consisting of an array of **F**unctional **U**nits (FUs), a branch unit, and several load/store units. Thereby even very small dataflow oriented instruction sequences benefit from the reconfigurable system-like architecture of the GAP. Instructions are executed within a combination of an asynchronous array of FUs and synchronous memory access and branch units. Due to the design of the array it is possible to issue independent as well as dependent instructions in the same clock cycle. Additionally to the improved issue and execution behavior, instructions issued to the array automatically form dataflow structures similar to the accelerator blocks within an application specific processor.

The contributions of this paper are a detailed description of the GAP architecture. Additionally, several optimizations are presented and evaluated that further improve the performance and, in parallel, decrease the hardware requirements of the GAP.

The next section provides a summary of coarse-grained reconfigurable systems, which are closest to the GAP architecture. An overview of the GAP is given in Section III together with an example of its functionality. Section IV describes the GAP architecture in detail and presents several optimizations of the basic architecture. A performance evaluation is shown in Section V. Section VI concludes the paper and identifies future work.

II. Related Work

Most reconfigurable architectures are attached to a master processor as an additional unit like a coprocessor. The master processor has to fulfill three tasks:

1) It has to provide configuration data to the reconfigurable system.
2) It has to take care about the activities of the reconfigurable system.
3) It executes the program parts that cannot be directed to the reconfigurable system.

Additionally, the presence and the architecture of the reconfigurable part must be taken into account at the software development phase. Example architectures with reconfigurable parts as coprocessor are the MOLEN [24], [25], [22], the GARP [10], and the MorphoSys [11] architectures. In contrast, the XTENSA chip from Tensilica, the CHIMAERA [9] project, and the PRISC processor [14] use reconfigurable functional units within the processor pipeline, see [6] for a more detailed overview.

The reconfigurable functional units within these processors are additional units besides the standard functional units of a superscalar processor. The additional units have to be accessed explicitly by the software. In contrast, the GAP features a homogeneous array of reconfigurable units that are transparent to the software.

A processor using a small reconfigurable architecture to accelerate statically determined parts of a program is presented by Clark et al. [4]. More flexible is the VEAL system [5] where the VEAL virtual machine is responsible to deal with the configuration of the loop accelerator online. Hence, an overhead for the dynamic compilation occurs. The PACT XPP architecture [13] contains several small VLIW processor cores located besides the reconfigurable array. These processors are responsible for the control driven parts of an application while the array offers high-throughput data operations. The WARP processor presented by Lysecky et al. [12] contains a FPGA-like accelerator, an on-chip profiler, and an on-chip CAD module (a separate CPU). The profiler identifies critical parts of the software and they are translated into a FPGA configuration online by the CAD module. Santos et al. [15] presented the 2D-VLIW processor. It comprises a pipelined two-dimensional array of functional units, which are configured by VLIW instructions containing as many operations as FUs are available. A similar approach is the ADRES reconfigurable processor [1], [26]. It comprises a VLIW front-end with a subordinated coarse grained reconfigurable array.

The RAW microprocessor [18], [19] and the EDGE architecture [3] implemented by the TRIPS microprocessor seem to be comparable at first glance because of their tiled architectures. However, the EDGE architecture introduces a new type of instruction set architecture (ISA) as one of its core concepts. Instructions in this ISA also include information about how to map operations on the tiles. This placement has to be calculated statically by the tool-chain in advance. Hence, a special compiler is required and normal instruction streams cannot be executed. The RAW microprocessor focuses on scalability of the number of used tiles (and therefore the used number of transistors) and wire delays. But if more than one tile shall be used by a program, again a special compiler is needed to map the application. This architecture therefore has some properties of many-core systems but cannot speed up single threaded applications without a special compilation technique.

None of the above mentioned architectures is able to directly execute a complete conventional sequential instruction stream within a two-dimensional array of reconfigurable functional units without any additional tool. The presented GAP architecture [21] exactly provides this feature.

III. Basic Architecture of the GAP

This section describes the basic idea of the GAP architecture and demonstrates the execution of a sample code fragment. A more detailed description of the different parts of the GAP are given in the next section.

A. Architectural Overview

The basic aim of the GAP architecture is that execution of normal programs shall profit from reconfigurable architectures. The GAP architecture combines the two dimensional arrangement of functional units (FUs) from reconfigurable systems together with the instruction fetching and decoding logic of a general purpose processor leading to the architecture shown in figure 1. Together with some additional components for branch execution and memory accesses (at the west side and the east side of the array), the GAP is able to execute conventional program binaries within its two dimensional array of FUs. Small loops will be mapped into the array and profit from the special architecture. All other non-loop-based program sequences are executed also within the array, and thus, they can likewise profit from the asynchronous execution.

In contrast to conventional superscalar processors, the GAP is able to issue also data-dependent instructions to the corresponding FUs within the same clock cycle. As consequence, GAP as in-order issue processor, reaches an issuing throughput similar to that of an out-of-order processor but does not require any out-of-order logic like issuing buffers, reorder and commit stages, as well as register renaming. However, execution of instructions is more like data driven processing within a reconfigurable architecture. Additionally, instruction execution is not done in a simple clocked way, rather it is possible to execute several dependent operations in one machine clock cycle. Instruction execution and timing is described in Section IV-D.

The array is arranged in rows and columns of FUs (see Figure 2) that are described in Section IV-C in detail. At the basic GAP architecture each column is attached to an architectural register of the processor's instructions set. We chose the PISA (Portable Instruction Set Architecture) instruction set known from the SimpleScalar simulation tool set [2]. Hence, the basic array contains 32 columns of FUs for the general purpose registers and an additional special column

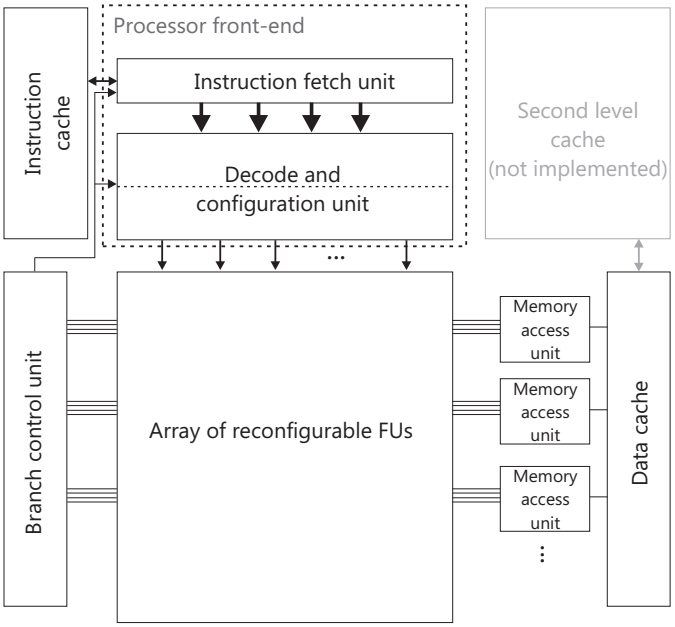


Fig. 1. Block diagram of the GAP core

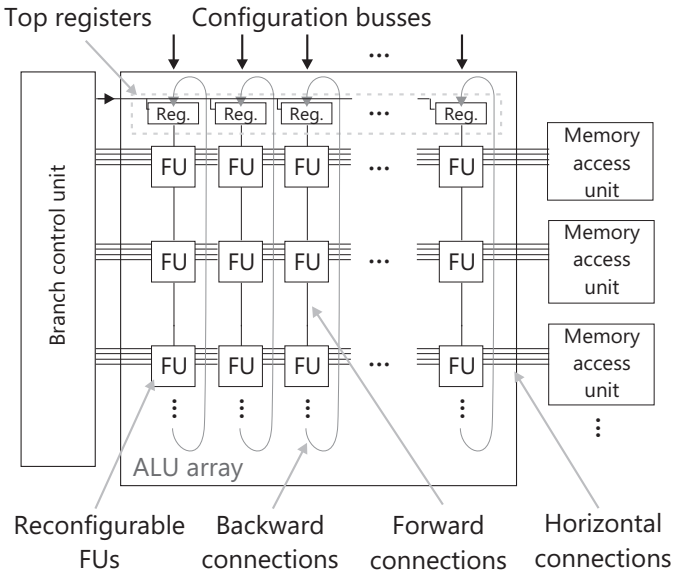


Fig. 2. General organization of the ALU array

for the multiply/divide registers (hi/lo) with the corresponding FUs (one multiply/divide unit per row). The number of rows depends on the maximum length of loop bodies that should be accelerated. An evaluation of the required number of rows is given in Section V.

The general data flow within the array is from the north to the south. A single 32 bit register is arranged at the top of each column and is called *top register*. This register contains the starting value of the corresponding architectural register of the column for the next execution phase. Each FU is able to read the output values of every FU from the previous row as inputs and its output is available for all FUs in the next row. So there is no data exchange within a single row and no data can be moved to the rows up. In general, each FU is configured by exactly one machine instruction or it is configured as route forward i.e., the FU bypasses the data from the previous FU in the same column to the next row.

Each array row is accompanied with a memory access unit that serves as communication interface to the data cache. The memory access units read the address from the previous row and forward data to a FU input of the same row in case of a load. A store receives the address as well as the store value from the previous row.

The single branch control unit on the west of the array has connections to all rows. Its task is to determine if conditional branches have to be taken. For this purpose, it checks the output value of a given FU against the branch condition. If the condition is true, the actual execution phase is stopped and the currently valid values in the corresponding row are copied from all columns to their top registers.

B. Code Execution Example

The placement of instructions into the array is demonstrated by the following simple code fragment. The pseudo machine instructions add fifteen numbers out of subsequent memory cells followed by negating the result.

```
1:    move R1,#15     ;15 values
2:    move R2,#addr   ;start address
3:    move R3,#0      ;Register for the sum loop:
4:    load R4,[R2]    ;load an element
5:    add R3,R3,R4    ;add
6:    add R2,R2,#4    ;inc address
7:    sub R1,R1,#1    ;dec counter
8:    jmpnz R1,loop   ;end of loop?
9:    sub R3,R1,R3    ;negate the result (R1=0)
```

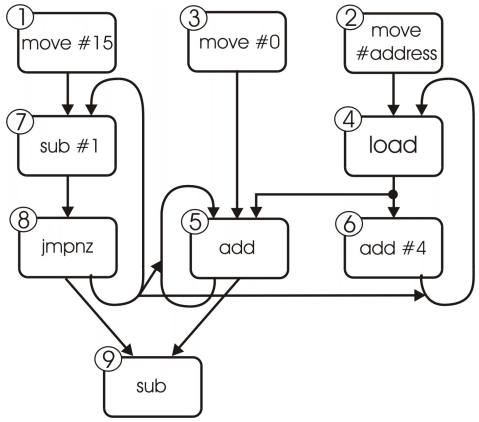


Fig. 3. Dependency graph of the example instructions.

Figure 3 shows the dependency graph of the 9 instructions, which can be recognized again at the placement of the instructions within the GAP back-end shown in Figure 4. The instructions 1 to 3 are placed within the first row of the array. Instruction 4 (`load`) depends on *R*2 that is written in the first row and therefore, it must be located in the second row. It reads the address as a result of instruction 2 and forwards the

data received from the memory into the column *R*4, which is the destination register of the load. The instructions 5 to 7 are placed in an analog way. Instruction 8 is a conditional branch that could change program flow. To sustain the hardware simplicity, conditional branches are placed below the lowest already arranged instruction. In this case, the branch has to be located after the third row. The last instruction is placed after the branch in the fourth row. Hence, if the branch is taken, the GAP takes a snapshot of the register contents at the corresponding row and copies the data into the top registers. In this case, instruction 9 is discarded. Otherwise, the *sub* is executed without any time loss.

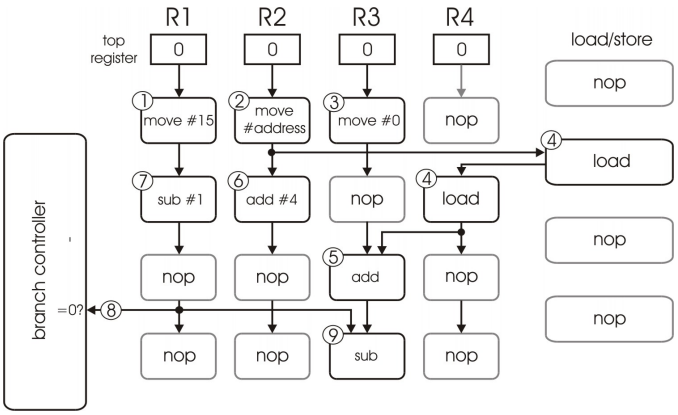


Fig. 4. Placement of the complete example fragment

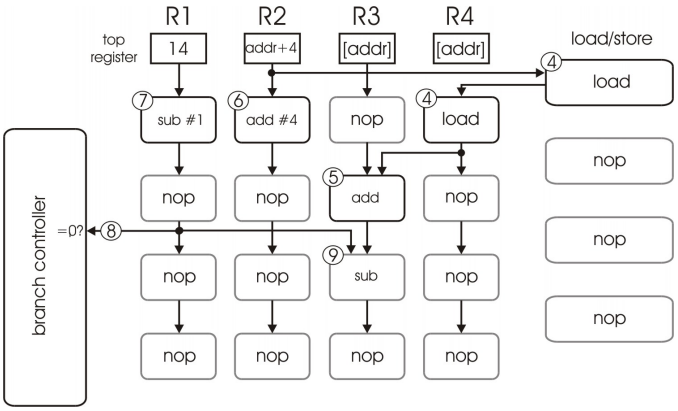


Fig. 5. Placement of the loop body (instructions 4 to 8) and the subsequent instruction (instruction 9)

After the first execution of the loop body, the backward branch to the label *loop* has to be performed. The branch target is already mapped into the array but it is placed among other instructions that must not be executed again. To keep the hardware simple, the GAP is only able to copy data into the top registers of the first row. The decoding starts again with the new address. The newly started decoding maps only the loop and the following instructions into the array (see Figure 5). Now, the loop target is the first instruction within the array, and hence, all subsequent loop iterations can be executed directly within the array without any additional fetch, decoding and configuration phases. Due to the placement of instructions following the loop body, the GAP does not suffer from any misprediction penalty at loop termination as other processors using branch prediction would do.

IV. Detailed GAP Description

A detailed description of the superscalar pipeline front-end, the back-end architecture, the execution and timing of the operations, the memory hierarchy, and several optimizations are presented in this section.

A. Superscalar Processor Front-end

Instructions are fetched from the instruction fetch unit out of a normal program binary generated by a standard compiler e.g., the GCC. A program counter determines the position of the next instructions to fetch. The current implementation fetches four instructions in parallel. During decoding, each instruction is treated in one of the following ways:

- **Unconditional direct branches:** These instructions are executed directly within the decode stage i.e., the program counter is set to the given value and the decoding of the subsequent instructions is cancelled.
- **Arithmetic-logic instructions:** All arithmetic and logic instructions are register-register instructions (a load/store instruction set architecture like PISA is assumed). All decoded arithmetic/logic instructions are placed into the array of FUs. The column in which an operation is placed is determined by the output register of the instruction and the row depends on the availability of the input data and the last read access to the output register. Input data is available after the last write access to the corresponding register i.e., the lowermost instruction placed within the column. Because instructions are only placed and not executed, it is possible to place independent as well as dependent instructions into the array at the same cycle.
- **Conditional branches and register indirect jumps:** The branch control unit is able to handle one branch or jump instruction per row of the array. Because both types of control flow changes (conditional branches and register indirect jumps) depend on the content of at least one register, branches/jumps have to be placed below the row in which the value of the corresponding register is calculated. Additionally, it is necessary for the loop acceleration to catch all register values out of a single row (after one loop iteration) and copy them into the top registers for the next iteration. Hence, branches/jumps are placed below the lowest operation of the loop body within the array.
- **Load/store instructions:** Memory instructions are executed by the memory access units. Each unit is able to execute one load or one store instruction. A load instruction is placed as high in the array as possible i.e., below the last write access to the address register and the last read access to the destination column. Store instructions additionally depend on preceding branches. All memory operations are equipped with an ordinal

number that prevents load instructions to be executed before a foregoing store instruction.

The front-end is active as long as the array is not filled up to the last row. If the array is full, the back-end is busy with executing the instructions already placed in the array. This could happen if some long latency operations like load instructions have to be executed or if a loop is executed. In both cases, the front-end can switch into sleep mode to save energy.

When the execution of the operations inside the array reaches the last row or when the branch controller signals a taken branch, the front-end resumes to fetch and decode instructions. If needed, the program counter is set to the new value given by the branch controller beforehand.

The processor front-end is connected to the back-end by several configuration lines. Because four instructions are decoded in a single cycle, also up to four operations could be issued to a column in the array. Hence, the maximum of four configuration busses per column, including the branch controller and the memory access units, are required. But, due to our evaluations we find out that only two busses per column are sufficient. If more than two instructions (out of four) must be issued to the same column, an additional cycle is required. The configuration unit is aware of this circumstance.

The only feedback signals from the back-end to the front-end are the signals from the branch controller to the fetch unit (a new program counter and a valid signal) and a *finished* signal. The *finished* signal indicates that the execution has reached the end of the array.

B. Timing Analysis at the Front-end

Besides the placement of the operations into the array, the front-end is also responsible to determine the timing. Because of the asynchronous execution of the operations, the time at which a valid data word arrives at a particular location is not known. To synchronize the synchronous units (see Section IV-C) with the asynchronous execution within the array, the configuration unit is aware of the timing of each operation. Therefore, the runtime of each operation is known by the configuration unit in terms of so-called *pico-cycles*. We have chosen one pico-cycle as $\frac{1}{4}$ of a machine clock cycle. At each configuration step i.e., the assignment of one operation, the configuration unit calculates the time at which the FU output is valid. Inside the configuration unit one pico-cycle counter is responsible for each architectural register i.e., each column. The number of pico-cycles required for the current operation is added to the longest path of the source values. For the evaluations, we assumed the execution times for the operations as shown in Table I. These assumptions are based on the fact that some operations require a carry chain (*add, sub, setlt, setgt*), require a shifter (*shl, shr*), or are flat logic (*and, or, xor, not*).

If the resulting pico-cycle counter exceeds one machine clock cycle, a so-called *timing token register* inside the FU is activated (see Section IV-D) and the counter is decreased by four. Otherwise, the timing token register of the FU is bypassed i.e., the timing token passes immediately. The pico-cycle counter is implemented as a two bit counter and the overflow indicates that the token register has to be activated. Due to the overflow, decreasing the counter is not required explicitly.

Operation	Pico-cycles
Add, Sub	3
Setlt, Setgt	3
And, Or, Xor, Not	1
Shl, Shr	2

TABLE I
PICO-CYLE TIMES ASSUMED FOR THE EVALUATION

In a real implementation of GAP the longest signal path depends on the technology and the placement. Hence, the pico-cycles required for an instruction class are not known at design time (before synthesis). To address this problem, they can be stored in an internal look-up table (RAM) that is initialized with the maximum pico-cycle value (four pico-cycles in this case) at system reset. A test software determines the actual value per instruction class and sets the entries in the table accordingly.

C. Back-end Architecture

The GAP back-end comprises the FU array, the branch controller, and the memory access units (see Figure 2). The branch controller and the memory access units are designed as synchronous units while the FU array is asynchronous (except for the configuration part and the timing token register). The three parts are described below.

- **Functional unit array:** The array consists of a two-dimensional arrangement of identical FUs. Each FU contains an ALU that is able to perform the following operations: *+, -, shl, shr, and, or, xor, not, neg, setlt, setgt*. Besides each ALU, a configuration register including a constant value, two input multiplexers, and the token logic for the timing is present. Moreover, a bypass multiplexer is available that allows the route-forward operation i.e., the output value of the previous FU in the same column is routed to the next row. The block diagram of a complete FU is shown in Figure 6. The input multiplexers are able to select the output of each FU in the row above or the top registers, respectively. Because of the asynchronous execution inside the array, no registers are located in the data path of the FU. In contrast, the configuration network is fully synchronous and the register for the timing token can be enabled by configuration.
- **Branch controller:** The branch controller comprises a configuration register, two input selectors, and a compare-to-zero comparator per row of the array. Additionally, the branch controller offers a bus for the new program counter (for taken branches) to the processor front-end together with a *valid*-signal. Besides controlling branches, this unit enables the top registers to store the values delivered by the feedback network. It is also aware of the time when all operations in the array are finished i.e., when all timing

tokens arrived at the last row. Furthermore, the branch controller manages the loop acceleration i.e., if the branch target is the first instruction that is already configured in the array, the configuration is left unmodified. Only the new data is copied into the top registers in this case and the timing tokens are activated at the top row.

- **Memory access units:** The memory access units are synchronous units. They are responsible for read and write operations of the array to the main memory. A read or write operation occurs if the software executes a load or store instruction, respectively. To reduce the access time, small local caches containing a single cache line are present in each memory access unit (see Section IV-E). Each memory access unit contains a configuration register, two input selectors, and an output bus. The input selectors can choose any of the FUs' output of the upper row as address and write data input. The output bus directs the read data to any FU of the same row. An operation of the memory access unit is triggered at the time the required timing tokens are available. The output token is synchronously generated if the operation is performed.

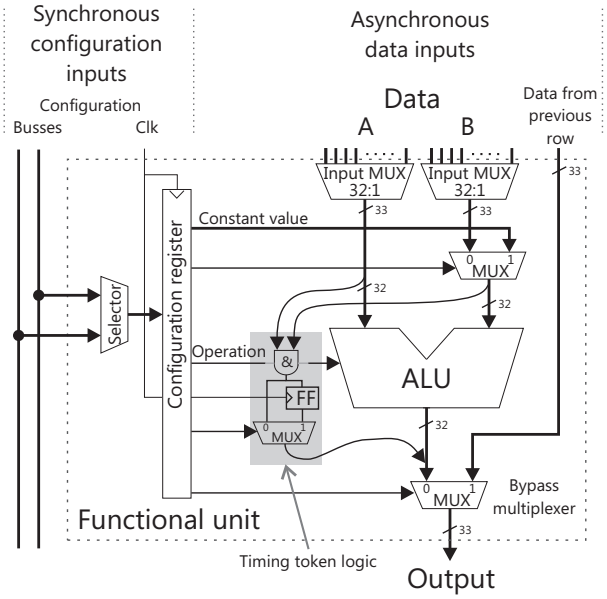


Fig. 6. Block diagram of a functional unit comprising an ALU, input selectors, bypass network, and a configuration register set by one of the two configuration busses

The units of the back-end are configured by different synchronous configuration busses. These busses range from the configuration unit of the front-end to the last row of the array. Because of a high number of rows, these busses could be comparatively long and, hence, have a negative impact on the maximum clock rate. To overcome this problem, a register can be integrated to pipeline the transportation of the configuration data and to shorten the maximum delay.

A much more interesting issue is the wire length of the horizontal connections. If the value of register zero (*R*0) is required at the memory access unit or the value of *R*31 is required at the branch controller, these values have to pass from the leftmost FU to the right side of the array or vice-versa, respectively. To get rid of these long wires together with the delay, we evaluated different array extensions described in Section IV-F. For our evaluations, we assumed that the maximum delay is taken into account by the pico-cycles required for an operation.

D. Instruction Execution and Timing

The front-end units of the GAP i.e., the instruction fetch, decode and configuration units, work in a synchronous way as well as the configuration paths within the array of FUs. The branch controller and the memory access units are also synchronous to a global clock signal. In contrast to these units, execution of the operations within the FUs is performed asynchronously.

So, the data stored within the top registers move to all FUs in the first row that are configured to use data out of at least one of these registers. The output of each FU in the first row moves to the inputs of the FUs in the second row corresponding to their configuration and so on.

To align synchronous parts with data from asynchronous execution, we introduced so-called *timing tokens*. These tokens propagate in parallel to each data word through the array. If a FU is configured for an operation with two registers as source operands, the FU sends a token if both source tokens are available. To generate the delay of the tokens, the FUs are equipped with an additional synchronous single bit register, which allows to bypass the token signal depending on its configuration. In general, these registers are in bypass mode. The idea is that a token moves ahead of the corresponding data till it reaches an active timing token register. Here, it has to wait until the next rising clock edge i.e., until the data at the associated FU is valid. The token registers are activated by the configuration unit if the longest data path from the current FU back to the FU with the last active token register or the top of the array is longer than one machine clock cycle. Hence, if a synchronous unit receives a token, the data at the input will be valid not later than the next rising edge of the clock signal.

Accordingly, the synchronous branch controller as well as the memory access units use the timing token of the incoming data as *data valid* signal. If it is set at the rising edge of the clock signal, the corresponding action must be performed. Additionally, an internal *done* state must be set to ensure that the action is done only once. Because of the nature of the timing token, it is not reset during a single execution iteration of one configuration. If a configuration is executed multiple times like a loop body or if the array is reconfigured, the *done* state must be reset.

E. Memory Hierarchy

Reaching a high memory throughput is a main requirement to get a high processor performance. The GAP back-end comprises one memory access unit per row. Hence, assuming

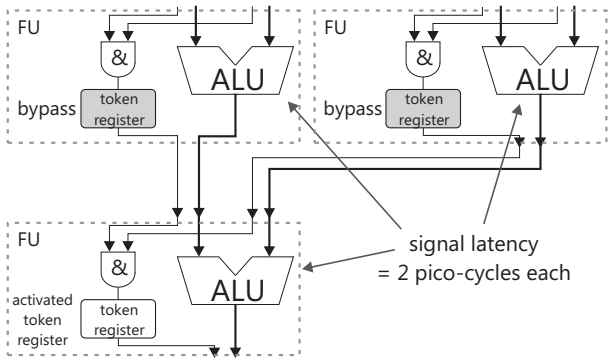


Fig. 7. Three instructions, each requires 2 pico-cycles. The token register of the upper FUs are bypassed and the one of the lower FU is activated

n rows also *n* memory accesses can occur simultaneously. To reduce the concurrency at the memory interface, each memory access unit contains its own small private cache. Because of hardware simplicity, the current evaluations assume only a single cache line within each memory access unit. These caches serve only for read accesses, writes go directly to the next memory hierarchy i.e., the data cache. For cache consistency, writes are also directed to all other memory access units, which compare the received write address to their own cache and update it, if necessary.

The memory write sequence is guaranteed by the back-end by allowing only a single write access in one cycle. This technique enables the memory access units to distribute writes through a shared write bus to the other memory access units and the data cache in parallel. Additionally, writes are executed in the original program order and take care of preceding taken branches. Hence, no unintended writes can occur. Between two consecutive writes all loads that are in the program order in between these writes can be executed simultaneously.

Memory throughput is increased by the data cache, which is a non-blocking cache [16]. All read accesses from the memory access units go to the data cache. If one request cannot be satisfied it is redirected to the next level cache (which is not implemented in our current simulator). Meanwhile, requests from other memory access units can be served by the data cache.

F. Architectural Optimizations

A problem of GAP arises from branches that can be resolved only at the execution time inside the array. These branches lead to a flush and reconfiguration of the array if they do not form a loop. Another shortcoming of the basic GAP architecture is the huge amount of chip area required for the 32 columns for 32 architectural registers. Additionally, the wire delay is a very important issue in modern chip design and the long horizontal wires have a big impact on the operating performance (not the clock frequency because the concerned parts are asynchronous). The following optimizations of the GAP architecture consider these drawbacks.

1) Branch Prediction: Like most other processors the GAP can profit from branch prediction. A taken conditional branch inside the array that does not implement a loop leads to a flush of the array and a new configuration phase. For this purpose, a branch prediction tries to increase the length of the dataflow graph mapped into the array by predicting the direction of conditional branches. A misprediction results in an array flush and starts a new configuration phase. Hence, the number of speculation levels of the GAP predictor is only limited by the number of rows because we allow only one branch per row.

In contrast to conventional speculative superscalar processors, the GAP does not require any commit logic that is aware of speculative execution. Instead, the register values are taken out of the row of the mispredicted branch and are copied into the top registers. Now, they are available for the further correct execution of the program.

2) Predicated Execution: Another possibility to reduce the number of flushes is to get rid of some conditional branches. Therefore, the optimized GAP is able to use predicated execution. In the case of a predicated instruction, it will be placed in the array like a normal instruction. An additional logic inside the FUs allows to use predication for all instructions. A flag is set in the configuration register of the corresponding FU, which indicates that the operation is predicated. If the predicate is true, the operation is executed but if it is false, the FU switches to route-forward mode. The predicate itself is determined by the branch control unit i.e., the original branch preceding the predicated instruction is not executed as a branch but it sets the corresponding predication signal appropriately. This technique is only useful for forward branches with very small jump distances because the predicated instructions have to be taken into account completely during the calculation of timing tokens.

3) Horizontal Array Dimension: The horizontal extension of the basic functional unit array is as high as the number of architectural registers. Using a standard RISC instruction set architecture like ARM, PISA, or openRISC (ORBIS32) requires 16 or 32 columns. The high number of columns leads to a high wire delay between the leftmost and the rightmost FUs of the GAP. Additionally, the input selectors of each FU must be able to select each of the outputs of the previous row resulting in 33-to-1 33 bit width multiplexers (32 registers plus one input for the memory access unit * 32 data bits plus the timing token).

Reducing the number of columns has been reached by decoupling the architectural registers from the array columns. The processor front-end supports a technique like the well-known register renaming [20]. But, in contrast to the standard register renaming that maps less architectural registers to a higher number of physical registers, GAP does it vice-versa: a high number of architectural registers is mapped to a lower number of physical registers i.e., columns.

Mapping is done by a mapping table with one entry per architectural register and a column counter. At an array flush, each entry in the mapping table is set to invalid and the counter is set to zero. Every time the result of an instruction should be written into an architectural register, the column is determined by the value in the corresponding entry of the mapping table.

If it is invalid, the content of the counter is written into the mapping table and the counter is increased. If the counter reaches the maximum value (the number of columns) no instructions writing to further architectural registers can be placed inside the array. The configuration stops at that time.

As a result of the lower number of columns, some configurations have to be split into multiple configurations for execution. In Section V-C we have evaluated how many configurations can be satisfied by a smaller array dimension.

V. Evaluation

The evaluation of the GAP architecture concerns the array size with different optimization steps. In all evaluations, the GAP features branch prediction unless otherwise noted. Besides the pure performance, we also evaluated the utilization of the FU array and the memory access units.

A. Evaluation Methodology

To evaluate the performance of the basic GAP architecture, we compared it to a superscalar processor simulated by the SimpleScalar (SS) tool set with the parameters given in Table II. Therefore, the GAP architecture is implemented as a cycle accurate simulator written in C++. The pipeline stages, the FU array, the branch controller, the memory access units and the cache are modelled in the GAP simulator. Both simulators execute the identical binary files.

Parameter	SimpleScalar	Basic GAP
L1 I-Cache	128:64:1	128:64:1
L1 D-Cache	1024:32:1	1024:32:1
Fetch, decode, and issue width	8-way	4-way
Multipliers	1	1 per row
Integer ALUs	8	31 per row
Branch prediction	bimod	bimod
Return address stack	8	none
Branch target buffer	512*4	none
Load/store queue	8 entries	one L/S unit per row
Memory latency	24	24

TABLE II
Parameters of both simulators

At first glance, it seems not to be fair to compare the SimpleScalar architecture with 8 ALUs to the GAP with up to 992 ALUs (31 columns x 32 rows, *R0* is fixed to 0). But we first evaluated the IPC rates of the SimpleScalar configured by the given parameters and with different numbers of ALUs to identify its peek performance. Therefore, we used the same benchmarks as for the evaluation of the GAP. The results presented in figure 8 show only a marginal increase in average IPC using more than 4 ALUs. The reason is that the issue unit is not able to find enough independent instructions to execute in the same cycle. To overcome this bottleneck, the fetch width and the decode width have to be increased together with the size of the Register Update Unit (RUU). But increasing the issue complexity would influence the maximum frequency dramatically as stated by Cotofana et al. [7]. To be conservative, we used twice the fetch and decode width for the SimpleScalar than for the GAP and an RUU size of 128 for the SimpleScalar while the GAP does not need any issue queue.

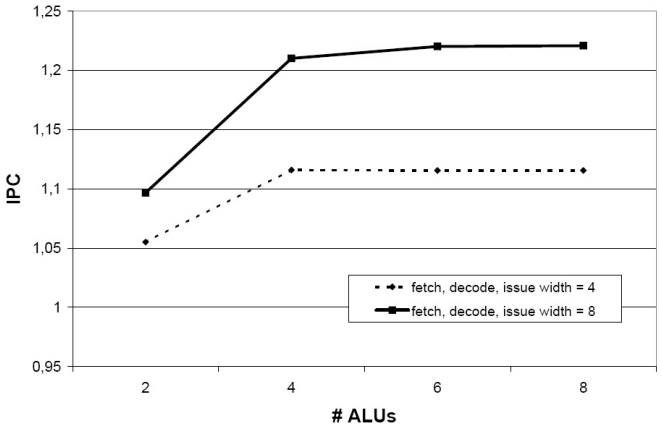


Fig. 8. Average IPC rates of all benchmarks using the SimpleScalar with different number of ALUs and a fetch/decode width of 4 and 8, respectively.

Several benchmarks out of the MiBench [8] benchmark suite without floating point operations are selected to determine the GAP's performance. As input data, the *small* data sets are used. The *stringsearch* benchmark is an adapted version of the original benchmark reduced to the central algorithm without any output. Unless otherwise stated, the identical binary files are used for the GAP simulation as well as for the SimpleScalar.

B. Performance of the GAP Architecture With Branch Prediction and Predicated Execution

The first performance evaluation concerns two different optimization steps of the GAP: the basic architecture with integrated branch prediction and the basic architecture with branch prediction and additional predicated execution. The latter requires a slight modification of the execution binary because of the predication bits. This modification is done only by an analysis of the executable and not by compiler options. As GAP array size we have chosen 4, 8, 16, and 32 rows and always 31 columns of FUs, one multiplier and one memory access unit per row.

Figure 9 shows the IPC rates of the benchmarks using the GAP compared to the SimpleScalar. Using 16 or 32 rows, the GAP outperforms the SS in all cases, except for *bitcount*. The benchmarks *adpcm*, *dijkstra*, and *stringsearch* result in a higher IPC even with only 4 rows. On average the GAP with 4 rows reaches a slightly higher IPC than the SS. The top performance is reached by the *SHA* benchmark with 32 rows: The IPC is 2.56 times the IPC of the SimpleScalar.

After adapting the executable binary files with predication bits, we simulated the same benchmarks with enabled predicated execution. Figure 10 shows the results of these measurements. The predicated execution achieves a slightly better IPC on average with 32 rows. But, in particular the *adpcm* obtains a significantly higher performance in all row configurations

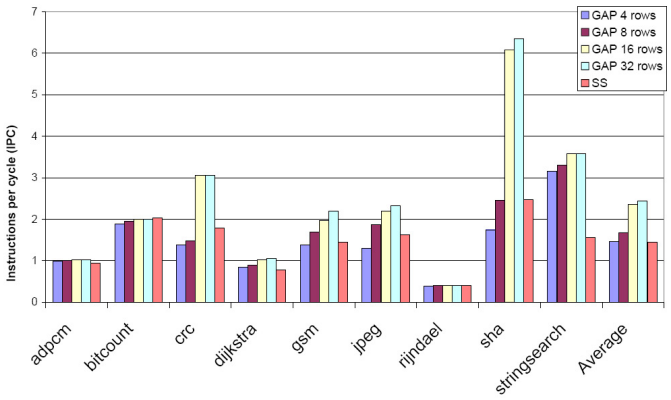


Fig. 9. IPC rate of the GAP with branch prediction compared to the SimpleScalar.

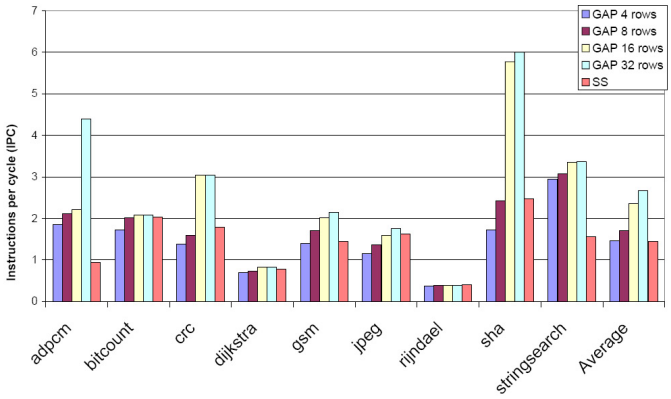


Fig. 10. IPC rate of the GAP with branch prediction and predicated execution compared to the SimpleScalar.

(nearly 2-fold with 4 rows up to 4.65-fold with 32 rows) while *dijkstra* and *rijndael* show a performance loss. The *bitcount* demonstrates an unexpected behavior: The performance of the 4-row version is decreased while the configurations with 8, 16, and 32 rows show a performance gain. All other benchmarks reach at least the same performance with enabled predicated execution than without.

The performance loss of some configurations/benchmarks can be explained by the timing of the execution inside the array. If a short forward branch, which is mostly taken, is substituted by predicated execution, the predicated instructions increase the longest path inside the array without being executed. Hence, the time to execute a complete configuration is also increased, which in turn reduces the IPC. In the case of the dramatic performance increase of the *adpcm*, which reaches with 32 rows an IPC of 4.65 times the IPC of the SS, the predicated execution shows its positive aspects: branches that are resolved by predicated execution are hard to predict, which leads to a better performance by using predicated execution. The following evaluations are performed only with branch prediction without predicated execution.

C. Utilization of the GAP Back-end

The basic architecture of the GAP consists of one column per architectural register and one memory access unit per row. At the maximum configuration evaluated in Section V-B this results in 992 FUs, 32 multiplier/dividers, and 32 memory access units. The second evaluation step concerns the number of FUs and memory access units that are really required to reach the performance mentioned above. Because of the direct relationship of the destination register as well as the dependencies of an instruction to its placement inside the array, we expect a very low utilization of the FUs.

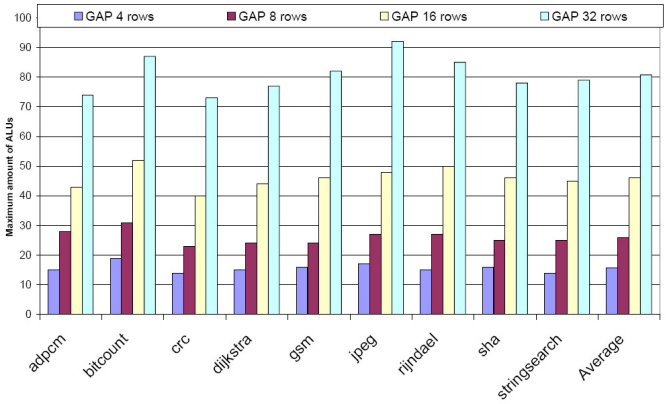


Fig. 11. Maximum number of used FUs per configuration with branch prediction.

Figure 11 shows the amount of FUs, which are used for real operations (not as route forward) with enabled branch prediction. The average overall utilization using 32 rows is about 81 FUs. That means, the utilization with 32 rows ranges from 6.7% up to 8.2%. Using only four rows, the GAP needs just 15 FUs on average i.e., 12% of its overall FUs. Although more than 64 FUs are used by the configuration with 32 rows in every benchmark, a maximum array with 64 FUs is reasonable. This is because the presented values show the maximum utilization by the benchmarks; the average utilization is much lower.

The number of allocated memory access units is shown in Figure 12. In the four and eight row configurations, every benchmark requires all available memory access units. At the 16 row version, *crc* uses only 14 and *jpeg* as well as *rijndael* and *dijkstra* all 16 memory access units. The other benchmarks allocate 15 memory access units. A saturation of memory access units is reached at the 32 row version of GAP: The maximum number of memory access units (31) is allocated by *jpeg* while *stringsearch* requires only 23.

These measurements indicate that a high number of memory access units seems to be required to get a good performance result. But, the presented values are only the maximum number of allocated memory access units i.e., they do not figure out if this particular configuration is relevant for the overall performance.

To clarify this question, the maximum number of rows that are used to accelerate loops is presented in Figure 13. Because

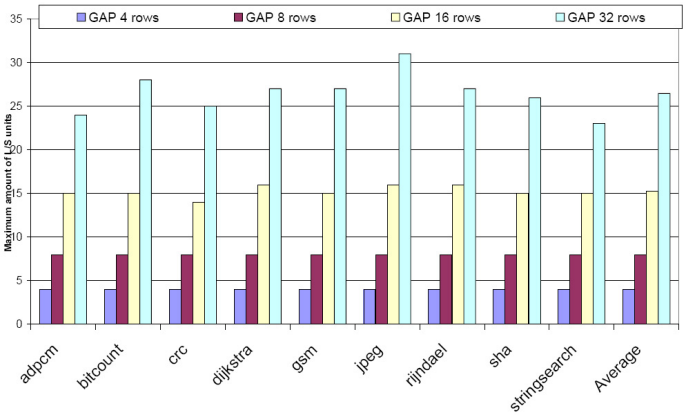


Fig. 12. Maximum number of used load/store units per configuration with branch prediction.

the GAP architecture supports only one memory access per row, the number of memory access units required for loop acceleration cannot be higher than the number of used rows. Hence, with only 17 memory access units most benchmarks would perform well if 32 rows are available. With less rows only 3, 6, resp. 12 units are required on average.

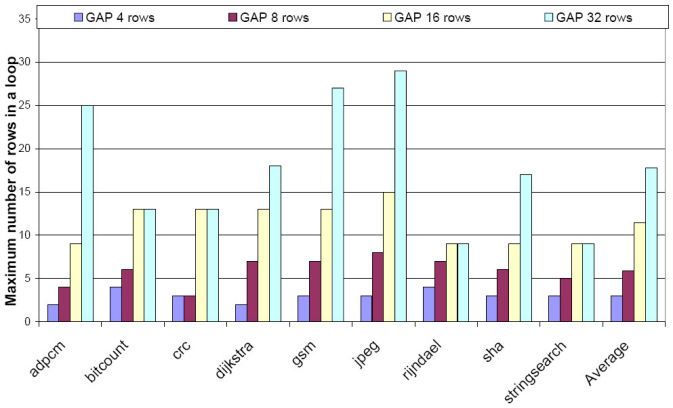


Fig. 13. Maximum number of used rows at loop acceleration with branch prediction.

Another area consuming resource is the multiply/divide unit. Although each row of the array contains a multiply/divide unit, the evaluations show a clear result: only one multiply/divide unit is required for all benchmarks and all configurations. Hence, a single multiply/divide unit, which can be accessed by all rows would be sufficient.

Besides the number of rows the number of columns also dictates the hardware effort. To find out a minimum array width we measured the number of used columns per configuration i.e., the number of columns with at least a single issued instruction. The histogram in Figure 14 shows how much configurations could be satisfied by a certain number of columns. Therefore we conservatively assumed a maximum array length of 32 rows i.e., using a shorter array would also lead to smaller configurations. As a result we can notice that only 8 columns are required to satisfy 95% of all configurations emerging in

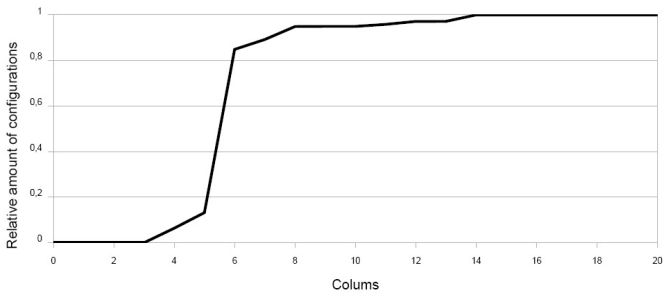


Fig. 14. Ratio of the satisfied configurations depending on the number of columns.

all applied benchmarks. With 14 columns, 99% and with 20 columns all of the configurations can be mapped completely into the array.

Besides the lower number of FUs also the input multiplexers of the FUs shrink if the array width is decreased. Choosing an array width where not all configurations can be issued completely does not mean that some applications cannot be executed at all. Rather these configurations require an additional array flush resulting in an additional configuration phase.

VI. Conclusion and Future Work

We presented the new GAP architecture comprising a processor back-end similar to a two-dimensional reconfigurable array. In contrast to well-known reconfigurable architectures the GAP is able to execute a conventional instruction stream generated by a standard compiler. The comparison to an out-of-order superscalar processor simulator shows a maximum performance factor of 2.56 when using the GAP with branch prediction and 4.65 when using it with additional predicated execution.

The great advantage of the GAP architecture is the improvement of sequential program execution that cannot be provided by modern multithreaded and multicore architectures. Besides the performance, we also evaluated different sizes of the FU array and its utilization to find an optimal array size. The measurements show that the number of allocated FUs is much less than the total amount of FUs (about 6-12%).

In future, we will further improve the performance by implementing multiple configuration layers, which speed up frequently used configurations and allow to decrease the maximum number of rows and columns. Multiple configuration layers would support especially function calls and configurations that do not fit into a single configuration layer. Hence, the layers will at least compensate the drawback of a smaller array. Moreover, a possible misprediction penalty can be eliminated completely by having the target instruction stream already configured within another layer.

Further optimizations concern the horizontal connection network. Because the current implementation allows to transport data from any FU of one row to any FU of the next row, complex multiplexers are required at each data input of the FUs. To reach simpler horizontal connections, we will reduce

the number of horizontal busses. This reduction will restrict the number of output values that can be transported to the inputs of the next row's FUs. Hence, the configuration unit must be modified to take care of that circumstance. If more data must be transferred to succeeding FUs than busses are available, the concerned operations must be placed in the following row.

REFERENCES

[1] F. Bouwens, M. Berekovic, A. Kanstein, and G. Gaydadjiev. Architectural exploration of the adres coarse-grained reconfigurable array. In *ARC*, pages 1–13, 2007.

[2] D. Burger and T. Austin. The simplescalar tool set, version 2.0. *ACM SIGARCH Computer Architecture News*, 25(3):13.

[3] D. Burger, S. W. Keckler, K. S. McKinley, M. Dahlin, L. K. John, C. Lin, C. R. Moore, J. H. Burrill, R. G. McDonald, and W. Yode. Scaling to the end of silicon with EDGE architectures. *IEEE Computer*, 37(7):44–55, 2004.

[4] N. Clark, J. Blome, M. Chu, S. Mahlke, S. Biles, and K. Flautner. An architecture framework for transparent instruction set customization in embedded processors. In *Proc. of the International Symposium on Computer Architecture*, pages 272–283, 2005.

[5] N. Clark, A. Hormati, and S. Mahlke. VEAL: Virtualized execution accelerator for loops. In *Proc. of the International Symposium on Computer Architecture*, pages 389–400, 2008.

[6] K. Compton and S. Hauck. Reconfigurable Computing: A Survey of Systems and Software. *ACM Computing Surveys*, 34(2):171–210, June 2000.

[7] S. Cotofana and S. Vassiliadis. On the design complexity of the issue logic of superscalar machines. *EUROMICRO Conference*, 1:10277, 1998.

[8] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and T. Brown. Mibench: A free, commercially representative embedded benchmark suite. *4th IEEE International Workshop on Workload Characteristics*, pages 3–14, December 2001.

[9] S. Hauck, T. Fry, M. Hosler, and J. Kao. The Chimaera Reconfigurable Functional Unit. In *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 1997)*, pages 87–96, 1997.

[10] J. Hauser and J. Wawrzynek. Garp: a MIPS processor with a reconfigurable coprocessor. In *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 1997)*, pages 12–, 1997.

[11] M.-H. Lee, H. Singh, G. Lu, N. Bagherzadeh, F. J. Kurdahi, E. M. Filho, and V. C. Alves. Design and implementation of the morphosys reconfigurable computingprocessor. *Journal of VLSI Signal Processing Systems*, 24(2–3), March 2000.

[12] R. Lysecky, G. Stitt, and F. Vahid. Warp processors. *ACM Trans. Des. Autom. Electron. Syst.*, 11(3):659–681, 2006.

[13] PACT XPP Technologies, July 2006. *http://www.pactxpp.com/main/download/XPP-III_overview_WP.pdf*.

[14] R. Razdan and M. D. Smith. A High-Performance Microarchitecture with Hardware-Programmable Functional Units. In *Proceedings of the 27th Annual International Symposium on Microarchitecture*, pages 172–80, 1994.

[15] R. Santos, R. Azevedo, and G. Araujo. 2d-vliw: An architecture based on the geometry of computation. In *ASAP '06: Proceedings of the IEEE 17th International Conference on Application-specific Systems, Architectures and Processors*, pages 87–94, Washington, DC, USA, 2006. IEEE Computer Society.

[16] J. Sicolo. A multiported nonblocking cache for a superscalar uniprocessor. Master's thesis, Department of Computer Science, University of Illinois, Urbana IL, 1990.

[17] F. Sun, S. Ravi, and N. K. Jha. A scalable application-specific processor synthesis methodology. In *Proceedings of the International Conference on Computer-Aided Design*, pages 9–13, 2003.

[18] M. B. Taylor, J. S. Kim, J. E. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffmann, P. Johnson, J.-W. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. P. Amarasinghe, and A. Agarwal. The Raw microprocessor: A computational fabric for software circuits and general-purpose programs. *IEEE Micro*, 22(2):25–35, 2002.

[19] M. B. Taylor, W. Lee, J. E. Miller, D. Wentzlaff, I. Bratt, B. Greenwald, H. Hoffmann, P. Johnson, J. S. Kim, J. Psota, A. Saraf, N. Shnidman, V. Strumpen, M. Frank, S. P. Amarasinghe, and A. Agarwal. Evaluation of the Raw microprocessor: An exposed-wire-delay architecture for ILP and streams. In *ISCA*, pages 2–13. IEEE Computer Society, 2004.

[20] R. M. Tomasulo. An efficient algorithm for exploiting multiple arithmetic units. pages 13–21, 1995.

[21] S. Uhrig. Processor with internal grid of execution units. Patent pending, WO 2007/143972 A3.

[22] S. Uhrig, S. Maier, G. Kuzmanov, and T. Ungerer. Coupling of a reconfigurable architecture and a multithreaded processor core with integrated real-time scheduling. In *13th Reconfigurable Architectures Workshop (RAW 2006), Rhodos, Greece*, Apr. 2006.

[23] S. Uhrig, B. Shehan, R. Jahr, and T. Ungerer. A two-dimensional superscalar processor architecture. In *The First International Conference on Future Computational Technologies and Applications, FUTURE COMPUTING 2009, Athens, Greece*, November 2009.

[24] S. Vassiliadis, S. Wong, and S. D. Cotofana. The molen pmu-coded processor. In *In in 11th International Conference on Field-Programmable Logic and Applications (FPL), Springer-Verlag Lecture Notes in Computer Science (LNCS) Vol. 2147*, pages 275–285, August 2001.

[25] S. Vassiliadis, S. Wong, G. Gaydadjiev, K. Bertels, G. Kuzmanov, and E. M. Panainte. The molen polymorphic processor. *IEEE Transactions on Computers*, 53(11):1363–1375, 2004.

[26] F.-J. Veredas, M. Scheppler, W. Moffat, and B. Mei. Custom implementation of the coarse-grained reconfigurable ADRES architecture for multimedia purposes. *International Conference on Field Programmable Logic and Applications*, pages 106–111, 2005.

Neuro-PID Control of Speed and Torque of Electric Vehicle

Sigeru Omatu, Michifumi Yoshioka, Toshihisa Kosaka, Hidekazu Yanagimoto
Department of Computer Science and Intelligent Systems
Osaka Prefecture University
Sakai, Osaka Japan
Email: omatu, yoshioka, kosaka, hidekazu@cs.osakafu-u.ac.jp

Jamal Ahamad Dargham
Department of Computer Engineering
University of Malaysia Sabah
Kota Kinabalu, Malaysia
Email: jamalad@ums.edu.my

***Abstract*—In this paper we consider the neuro-control method and its application to control problems of an electric vehicle. The neuro-control methods adopted here are based on proportional-plus-integral-plus-derivative (PID) control, which has been adopted to solve process control or intelligent control problems. In Japan about eighty four per cent of the process industries have used the PID control. After deriving the self-tuning PID control scheme (neuro-PID) using the learning ability of the neural network, we will show the control results by using the speed and torque control of an electric vehicle.**

***Keywords*-neuro-PID; electric vehicle control; speed control; torque controlcomponent;**

I. Introduction

In applying conventional control theory to practical problems, we have to model the plant or system. The modeling is done by using a set of linear differential or difference equations, in which unknown parameters are included. But the range of applicability is not so wide to cover real control problems. In real world, the plant and its environment are too complex to describe them by such linear models. For example, in a robotic control system, it may have many sensors providing inputs that cannot necessarily be interpreted as state variables. Furthermore, the models of the system may be unknown and interact with unknown changing environment.

Therefore, it is necessary to adopt new methods of control. They may not be so rigorous mathematically so that it can work in a wide range of domains and under more dynamic and more realistic conditions. One of the powerful methods is neuro-control based on the neural networks since the neural networks have preferable properties to overcome the difficult problems stated above. Some of them are 1) learning by experience (training), i.e., human-like learning behavior, 2) generalization ability, i.e., mapping ability of similar inputs to similar outputs, 3) nonlinear mapping ability, 4) parallel distributed processing, allowing fast computation for large scale systems, 5) robustness for noise and environmental change, 6) self-organizing property, etc. These properties make neuro-control suitable for applications to real control problems.

In this paper, we will adopt the proportional-plus-integral-plus-derivative (PID) control and tune the PID gains based on neural networks, which will be called as neuro-PID control. After deriving the neuro-PID controller, we will show the real application to torque control and speed control problems of an electrical vehicle [1].

II. Historical Review of Neuro-Control

The first neuro-control was discussed by Widrow and Smith [2] who used ADALINE to stabilize and control the pole balancing act. Other early research on neuro-control could also found in Waltz and Fu [3], Michie and Chambers [4], and Barto et al. [5].

Neuro-control research has begun sharp increase around 1987 when the first IEEE Conference on Neural Networks has held in San Diego. These papers have demonstrated that neuro-control methods can be applied successfully to control unknown nonlinear systems while conventional control approaches based on linear dynamical system theory could not solve such control problems. Many neuro-control structures were also proposed. Typical neuro-control methods are 1) feedback error learning by Kawato et al. [6], 2) neuro-internal model control by Hunt and Sbarbaro [7], 3) neuro-predictive control by Willia et al. [8], 4) Forward and inverse modelling by Jordan et al. [9]), 5) generalized and specialized learning by Psaltis et al. [10], 6) Self-tuning neuro-control by Omatu [11]. More information on neuro-control could be obtained by the books by D.A. White and D.A. Sofge [12], W. T. Miller III et al. [13], S. Omatu et al. [14], P.M. Mills et al. [15], and N.W. NG [16].

III. Error Back-Propagation Algorithm

The error back-propagation (BP) algorithm has been well-known since it was proposed by Rumerhart et al. [17] in 1985. As a neuro- PID control being described in detail later is derived based on the similar method of BP algorithm, we will explain the derivation of the BP algorithm in compact way.

The form of a neural network described by Fig. 1 is called a layered neural network since they have more than three layers which are called input layer, hidden layer, and output layer. Outputs of neurons in the input layer are the input data which should be processed. We assume that numbers of neurons in the input, hidden, and output layers are I, J,

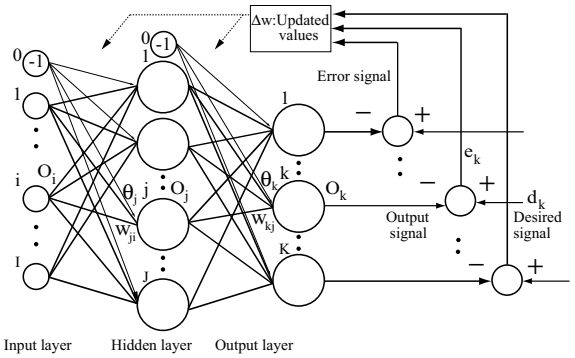


Figure 1. Structure of a layered neural network

and K, respectively. In Fig. 1, large circles denote neurons and each neuron, for example, neuron j can be described by the following nonlinear input-output relation:

$$O_j = f(\text{net}_j), \quad (1)$$

$$\text{net}_j = \sum_{i=1}^{I} w_{ji} O_i - \theta_j, \quad (2)$$

$$f(x) = \frac{1}{1+\exp(-x)} = sigmoid(x). \quad (3)$$

where O_j denotes the output of neuron j, w_{ji} denotes the connection weight from a neuron i to a neuron j, θ_j is a threshold value of neuron j.

Note that the output of a neuron is limited within 0 to 1 since $f(x) \in [0,1]$. If we assume that $O_0 = -1$ and $w_{j0} = \theta_j$, then we can rewrite net_j as follows:

$$\text{net}_j = \sum_{i=0}^{I} w_{ji} O_i. \quad (4)$$

From now on, we assume that threshold θ_j has been included in the weighting function and use the expression Eq.(4) instead of Eq.(2).

When the input data $\{O_i, i = 0, 1, \ldots, I\}$, connection weights w_{ji} from a neuron i in the input layer to a neuron j in the hidden layer where $\{j = 1, 2, \ldots, J, i = 0, 1, \ldots, I\}$, and connection weights w_{kj} from a neuron j in the hidden layer to a neuron k in the output layer where $\{k = 1, 2, \ldots, K, j = 0, 1, \ldots, J\}$, we can get the output values of the neural network by the following equation:

$$O_k = f(\text{net}_k),$$

$$\text{net}_k = \sum_{j=0}^{J} w_{kj} O_j.$$

Then we will compare the output $\{O_k\}$ with the desired value $\{d_k\}$ for each $k, k = 1, 2, \ldots, K$ and if there are large discrepances, we will correct the weighting functions, w_{ji} and w_{kj} such that the following error function E will be decreased.

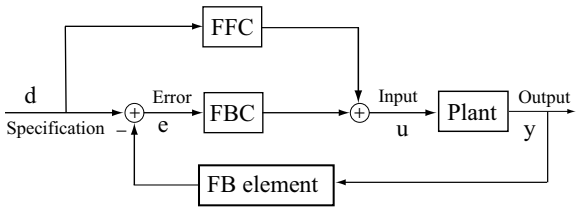


Figure 2. General structure of control system

$$E = \frac{1}{2} \sum_{k=1}^{K} e_k^2, \qquad e_k = d_k - O_k.$$

Using the gradient search, the minimizing cost of E is given by the following relation(the error back-propagation algorithm):

$$\Delta w_{kj} = w_{kj}(\text{new}) - w_{kj}(\text{old}) = \eta \delta_k O_j$$

$$\delta_k = e_k O_k (1 - O_k).$$

$$\Delta w_{ji} = w_{ji}(\text{new}) - w_{ji}(\text{old}) = \eta \delta_j O_i$$

$$\delta_j = \sum_{k=1}^{K} \delta_k w_{kj} O_j (1 - O_j)$$

$$k = 1, 2, \ldots, K, \quad j = 0, 1, \ldots, N.$$

Since the output O_k is limited within [0,1], we should modify the form when we need the value of (-∞, ∞), for example, $f(x) = x$, $f(x) = A(\frac{1}{2} - sigmoid(x))$, etc. Furthermore, to speed up the convergence of the gradient algorithm, we use an additional term as follows:

$$\Delta w_{kj}(\text{new}) = \eta \delta_k O_j + \alpha \Delta w_{kj}(\text{old}) \quad (5)$$
$$j = 0, 1, \ldots, N, \quad k = 1, 2, \ldots, K$$

$$\Delta w_{ji}(\text{new}) = \eta \delta_j O_i + \alpha \Delta w_{ji}(\text{old}) \quad (6)$$
$$i = 0, 1, \ldots, M, \quad j = 1, 2, \ldots, N$$

where the first term and second terms of (5) and (6) are called the learning term and the momentum terms, respectively and η and α are called learning rate and momentum rate, respectively.

IV. Feedback Control System Algorithm

We will consider the neuro-control scheme. The general control system can be described in Fig. 2 where FFC and FFB stand for feed-forward controller and feedback controller, respectively and FB is feedback. The aim of the controller is to find the suitable plant input u in order to follow the plant output y to the plant specification by adjusting the FFB and FFD.

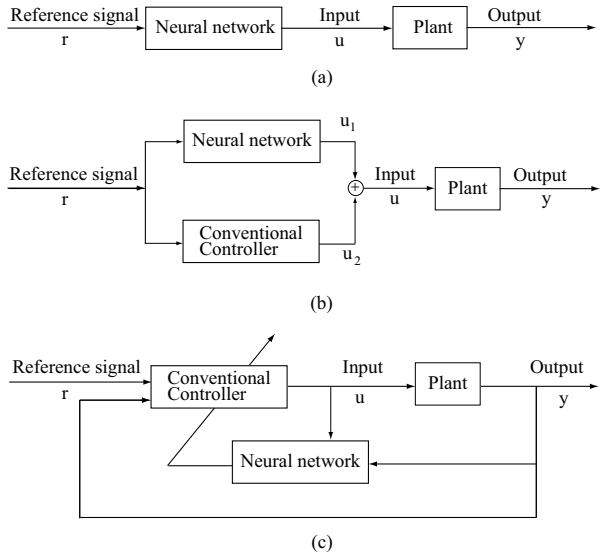


Figure 3. Three types of neuro-control system

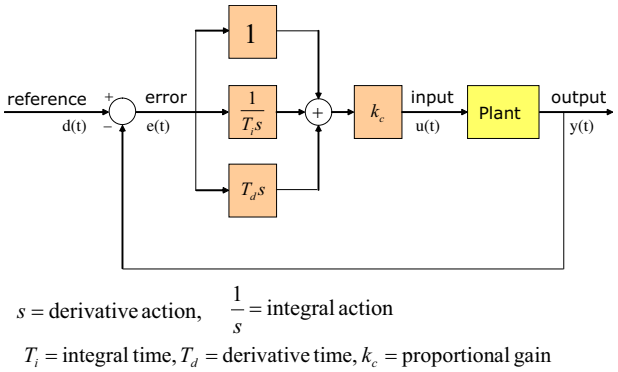


Figure 4. PID control system

The neuro-control is to determine the control input by using neural networks. Three types of neuro-controllers were proposed [14], [18]. They are 1) series type, 2) parallel type, and 3) self-tuning type as shown in Fig. 3.

Among them, we consider the third type. As a conventional controller we adopt the PID controller as shown in Fig. 4 and find the PID gains by using the BP algorithms.

A. Series Type Neuro-Control

This is to use the neural network directly such that the plant output will approach to reference signals as much as possible. The basic configuration is shown in Fig. 5 where (a) is the original structure, (b) is the series neuro-controller with an emulator, and (c) is the inverse dynamical structure. More detail algorithms have been explained in [19], [20], [21].

This approach is direct application of the layered neural network to find the control input and it is powerful for process control without so many fluctuations. But we need

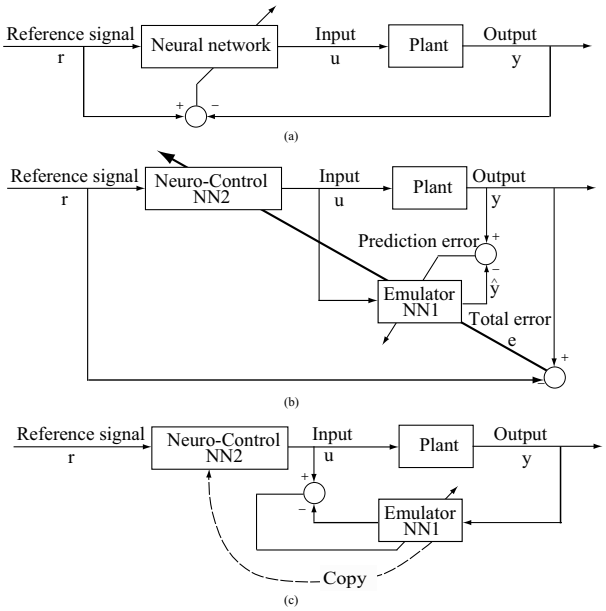


Figure 5. Series type neuro-control structure.

the emulator of the plant and it takes much time to find a stable parameter set of the neural network.

B. Parallel Type Neuro-Control

A parallel neuro-control architecture is shown in Fig. 3(b). For any conventional control scheme, we can use this type and the neural network works as the compensator of the adopted control scheme. If we take a feedback controller, this control becomes to the feedback error learning structure proposed by Kawato et al. [6].

Control engineers design an excellent controller at the laboratory or factory which is given by u_1 but when it is set at the real working place in an industrial factory, the control engineers must adjust the control input level such that it is suitable for real production under several environments. The adjustment is u_2 given by neuro-control in Fig. 3(b). This means that a well-trained cook at the restaurant could provide a delicious dinner for customers but on each table there are pepper and salt to be added to suit the taste of each individual dish. For detail algorithms see [14].

C. Self-Tuning Type Neuro-Control

The self-tuning neuro-control scheme is illustrated in Fig. 3(c) where a neural network is used to tune the parameters of a conventional control method like a human operator in the factory. The transfer function of PID controller is given by the following eauation.

$$G_c(s) = \frac{U(s)}{E(s)} = k_c\left[1 + \frac{1}{T_i s} + T_d s\right] \qquad (7)$$

where $U(s)$and $E(s)$ are input and error between the desired value and output. Here, k_c, T_i, and T_d are called as propor-

tional gain, integral time, and derivative time, respectively. In time domain, it can be written as follows:

$$u(t) = k_c \left[e(t) + \frac{1}{T_i} \int_{-\infty}^{t} e(\tau) \mathrm{d}\tau + T_d \frac{d}{dt} e(t) \right] \quad (8)$$

$$e(t) = d(t) - y(t) \quad (9)$$

Therefore, in the PID control it is essential to find a suitable PID gains. Many researchers have tried to determine them as precise as possible. The most famous method was proposed by Ziegler-Nichols and to determine them by the following relations(Ziegler-Nichols method).

$$k_c = \frac{1.2}{RL}, \qquad T_i = 2L, \qquad T_d = \frac{L}{2}$$

where R and L are maximum slope of the step response and the equivalent delay of the step response, respectively.

By rapid progress of computer, digital control has become common approach in control method and discrete PID control is also discussed. By discritizing Eq.(8) using trapezoidal rule for numerical integration, we obtain the following relation.

$$\begin{aligned} u(t) &= u(t-1) + K_p \big((e(t) - e(t-1) \big) + K_i e(t) \\ &+ K_d \big(e(t) - 2e(t-1) + e(t-2) \big) \end{aligned} \quad (10)$$

$$K_p = k_c - \frac{1}{2} K_i, \;\; K_i = k_c \frac{T}{T_i}, \;\; K_d = k_c \frac{T_d}{T}$$

As in the continuous-time case, Ziegler-Nichols method in the discrete-time case has become as follows:

$$\begin{aligned} K_p &= k_c - \frac{K_i}{2}, \\ K_i &= \frac{1.2}{RL} \frac{T}{2L} = \frac{0.6}{(\frac{L}{T})^2 (RT)} = \frac{0.6}{G_0 L_0^2}, \\ K_d &= k_c \frac{T_d}{T} = \frac{0.6}{G_0}, \\ G_0 &= \max_n \big(y(t) - y(t-1) \big), \qquad L_0 = \frac{L}{T} \end{aligned}$$

Ziegler-Nichols method is helpful to find the rough estimation of PID gains, it is not so good in any case. Therefore, in the process control the operators are adjusting these gains based on their experience and knowledge in trial and error as shown in Fig. 6.

We have developed a self-tuning PID controller. The control structure is shown in Fig. 7.

V. Derivation of Neuro-PID control

Using the learning ability of the neural networks, we will derive the neur-PID controller. Using (10), the output $y(t+1)$ of the plant is produced. Then the total error $E(t+1)$ is defined by

$$E(t+1) = \frac{1}{2} e(t+1)^2 \quad (11)$$

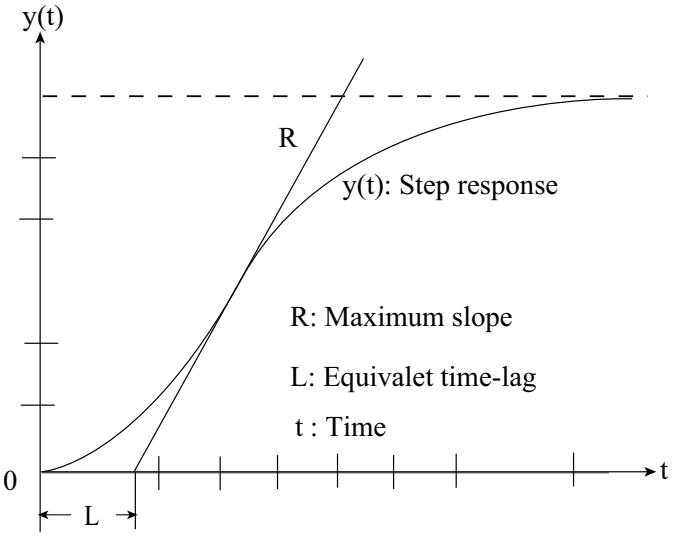


Figure 6. Ziegler-Nichols method

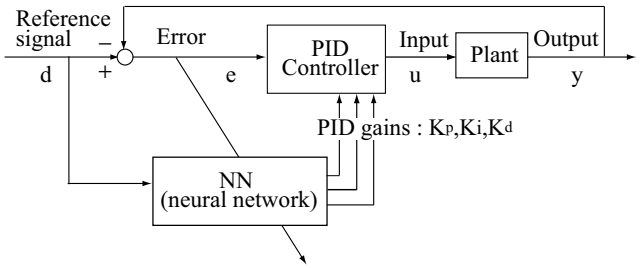


Figure 7. Self-tuning PID control system

where

$$e(t+1) = d(t+1) - y(t+1).$$

Let us consider the three layered neural network as shown in Fig. 8 where we assume that the output function is $f(x) = x$ in the output layer to extend the range of the PID gains over [0,1]. Using the three output neurons of the neural network, we denote the PID gains by

$$K_p = O_k(1), \qquad K_i = O_k(2), \qquad K_d = O_k(3) \quad (12)$$

$$O_k(m) = \mathrm{net}_k(m) = \sum_{j=0}^{N} w_{kj} O_j, \qquad m = 1, 2, 3 \quad (13)$$

$$O_k(m) = \mathrm{net}_k(m) = \sum_{j=0}^{N} w_{kj} O_j, \qquad m = 1, 2, 3 \quad (14)$$

where O_k shows the output of the neuron in the output layer.

By the similar way to the derivation of the BP algorithm,

we define the incremental values of w_{kj}, w_{ji} as

$$\Delta w_{kj}(t+1) = w_{kj}(t+1) - w_{kj}(t) = -\eta \frac{\partial E(t+1)}{\partial w_{kj}}\bigg|_{w_{kj}=w_{kj}(t)} + \alpha \Delta w_{kj}(t) \tag{15}$$

where $k = 1, 2, 3, \quad j = 0, 1, \ldots, N$.

$$\Delta w_{kj}(t+1) = w_{kj}(t+1) - w_{kj}(t) = -\eta \frac{\partial E(t+1)}{\partial w_{kj}}\bigg|_{w_{kj}=w_{kj}(t)} + \alpha \Delta w_{kj}(t) \tag{16}$$

where $j = 1, 2, \ldots, N, \quad i = 0, 1, \ldots, M, \quad .$

Let define δ_k as follows:

$$\delta_k = -\frac{\partial E(t+1)}{\partial \text{net}_k} \tag{17}$$

Then we have

$$-\frac{\partial E(t+1)}{\partial w_{kj}} = -\frac{\partial E(t+1)}{\partial \text{net}_k}\frac{\partial \text{net}_k}{w_{kj}} = \delta_k O_j. \tag{18}$$

By using the chain rule of the derivative, we have

$$\delta_k = -\frac{\partial E(t+1)}{\partial y(t+1)}\frac{\partial y(t+1)}{\partial u(t)}\frac{\partial u(t)}{\partial O_k}\frac{\partial O_k}{\partial \text{net}_k}. \tag{19}$$

Taking into account of the definition of $E(t+1)$ and $e(t+1)$ and the assumption of $f(x) = x$ and using (10) and $K_p = O_1, \ K_i = O_2, \ K_d = O_3$, we obtain

$$\frac{\partial u(t)}{\partial O_k} = \begin{cases} e(t) - e(t-1) & (k=1) \\ e(t) & (k=2) \\ e(t) - 2e(t-1) + e(t-2) & (k=3) \end{cases} \tag{20}$$

If we define the system Jacobian as

$$\text{Jac}(t) = \frac{\partial y(t+1)}{\partial u(t)}, \tag{21}$$

we have

$$\delta_k = e(t+1)\text{Jac}(t+1)\frac{\partial u(t)}{\partial O_k}, \qquad k = 1, 2, 3. \tag{22}$$

Therefore, we obtain the following relation.

$$\Delta w_{kj}(t+1) = w_{kj}(t+1) - w_{kj}(t) = \eta \delta_k O_j. \tag{23}$$

Similarly, we obtain the folowing relation.

$$\Delta w_{ji}(n+1) = \eta \delta_j O_i \tag{24}$$

$$\delta_j = \sum_{i=1}^{K} \delta_k w_{kj} O_j (1 - O_j). \tag{25}$$

From the above relation, we can adjust the PID gains step by step such that the minimum $E(t+1)$ could be achieved.

In order to caluculate the system Jacobian given by (21), we will propose the following approach by using the

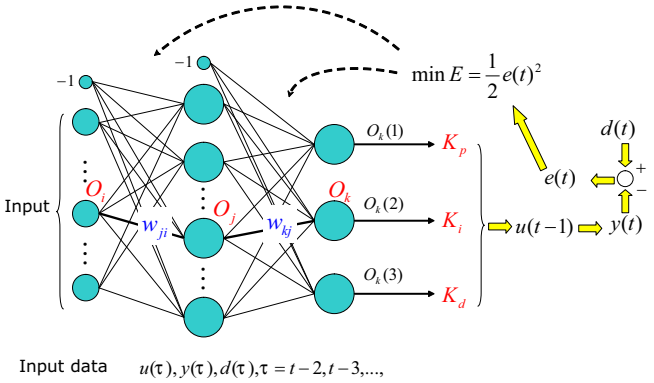


Figure 8. Neuro-PID structure

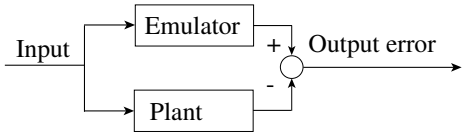


Figure 9. Emulator structure.

emulator. The emulator is constructed such that the output will become like the plant output for any input data as shown in Fig. 9.

The emulator can be designed by using a layred neural network as shown in Fig. 10. Using the neuro-emulator, we can approximate the system Jacobian. If the error of the neural network becomes small enough, the system Jacobian is given by the following equation:

$$\text{Jac}(t+1) \approx \sum_{j=1}^{N} W_{kj} O_j (1 - O_j) W_{j1}. \tag{26}$$

Here, W_{j1} denotes the connection weight from the input $u(n)$ to the neuron j.

VI. Application to Electric Vehicle Control

Due to environmental problems the automobile industry is currently venturing into producing electric vehicles. At the Shikoku Electric Power Company, Japan, a new type of electric car which is called PIVOT has been developed in 1993. The specification is shown in Table I and the overview and specific characteristics are illustrated in Figs. 11 and 12.

This PIVOT has equipped four wheels and each wheel has been made with in-wheel motor. Therefore, the wheels

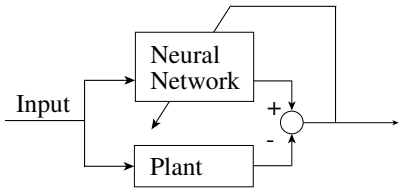


Figure 10. Neuro-emulator structure.

can be steered more than 90 degrees opposed to the body. This newly developed function accounts for universal drive performance such as lateral drive and rotation at a point. Another advantage is high-accuracy residual battery capacity indicator based on neural networks. A small and high accurate indicator has been developed. The residual battery capacity is calculated by a computer using voltage and current while driving.

The third one is an automatic battery exchange system. By the development of an automatic battery exchange system, the battery, having little residual capacity, is removed and a charged battery is installed within approximately five minutes, making refueling as easy as a gasoline-engine vehicle.

The fourth one is an energy-saving technology. Development of a regenerative braking system to convert kinetic energy to electrical energy and charge the battery during deceleration. Adoption of a lightweight frame/body and low air resistance body configuration and development of a lightweight heat-pump type air conditioning system are also equipped.

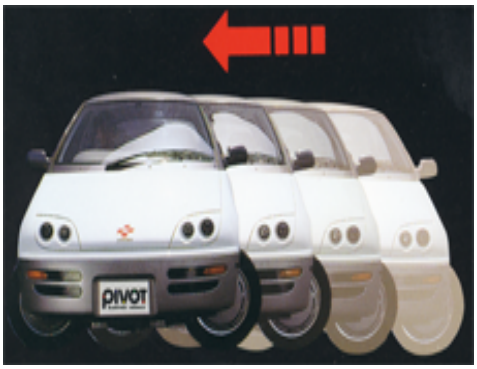
Table I
SPECIFICATION OF PIVOT.

Specification	Performance
length	4,126 mm
width	1,671 mm
height	1,603 mm
dry weight	2,200 Kg
passengers	4 persons
maximum speed	100 Km/h
range	200 Km(at a constant cruising spped of 40km/h
acceleration	Approximate 20 secs. from 0 m to 400 m
grand climb ability	30%
battery type	lead battery
equipment	power steering, heat-pump type sir conditioning

In 1993 when PIVOT was completed in Japan, there was no permission to drive any electric vehicle on the road in law and it is difficult to do the real driving experiment under various load change or load conditions, we have made experimental simulator as shown in Fig. 13. This can be written in Fig. 14 where DDC is direct digital controller which has been equipped with PID controllers, ACM is an alternative current motor which produces torque of OIVOT, DC is a direct current motor which produces any load with various specifications, T is a torque meter, and UFAS denoted a universal factory automation system.

For training the neural networks for various loads and various speeds, we have obtained the input and output data using the physical simulator illustrated in Fig. 13. As mentioned above, this simulator can be modeled as shown in Fig. 14 where DCM produces any kinds of loads and ACM outputs the corresponding control inputs by an AC motor. From our many experiences, we have used the neuro-control structure as shown in Fig. 15 where NNC means neuro-controller to adjust the PID gains and NNM was used to

Figure 11. PIVOT system and its driving mechanism.



Lateral drive

Rotation at a point

Figure 12. Feature of PIVOT system.

Figure 13. Experimental simulator.

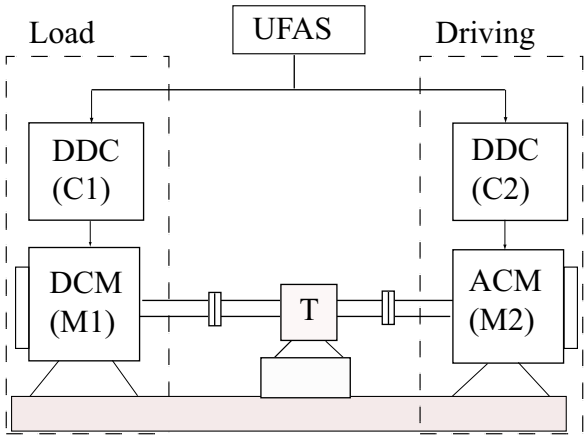


Figure 14. Experimental simulator model.

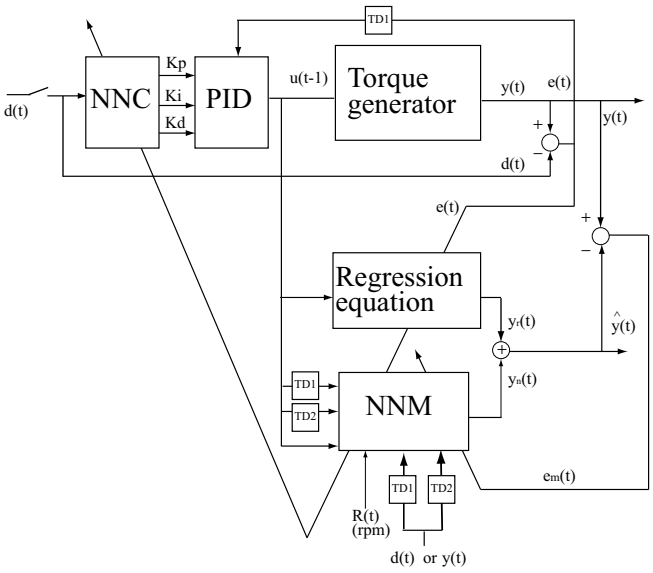


Figure 15. Experimental simulator where TD1 and Td2 are time-delay elements with one and two steps delays, respectively.

model the system emulator which is necessary to find the PID gains in NNC. Here, we use the parallel type emulator with regression model in order to speed up the modeling convergence and also used rotation number of motors. The notation $y\hat{(}t)$ means the estimated value of $y(t)$, $y_r(t)$ and $y_n(t)$ are estimated value of $y(t)$ by regression method and neural networks, respectively, $e(t) = d(t) - y(t)$, and $e_m(t) = y(t) - \hat{y}(t)$.

Figs. 16 and 17 are simulation results for conventional PID control case and proposed neuro-PID case. Fig. 18 is the PID gains for various initial values of PID gains. From these results we can see that the proposed neuro-PID torque control is better that the conventional PID controller based on the Ziegler-Nichols method. Furthermore, even if we start any initial values of PID gains, the excellent control results

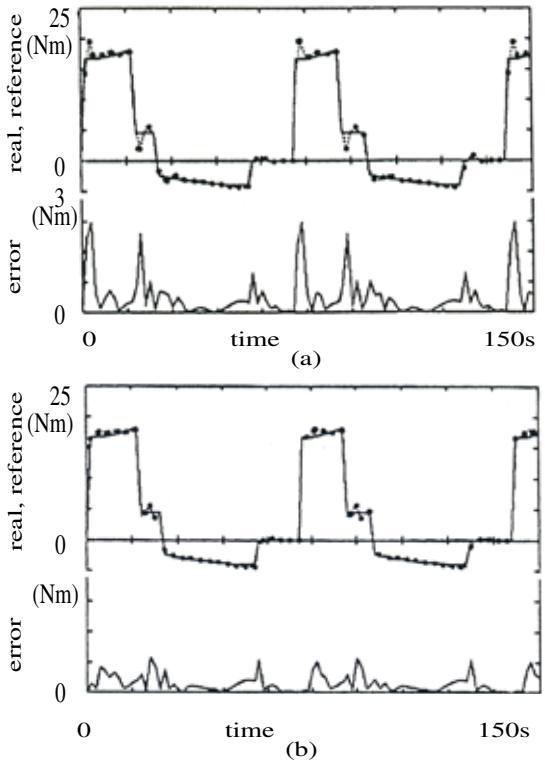


Figure 16. Conventional control results.

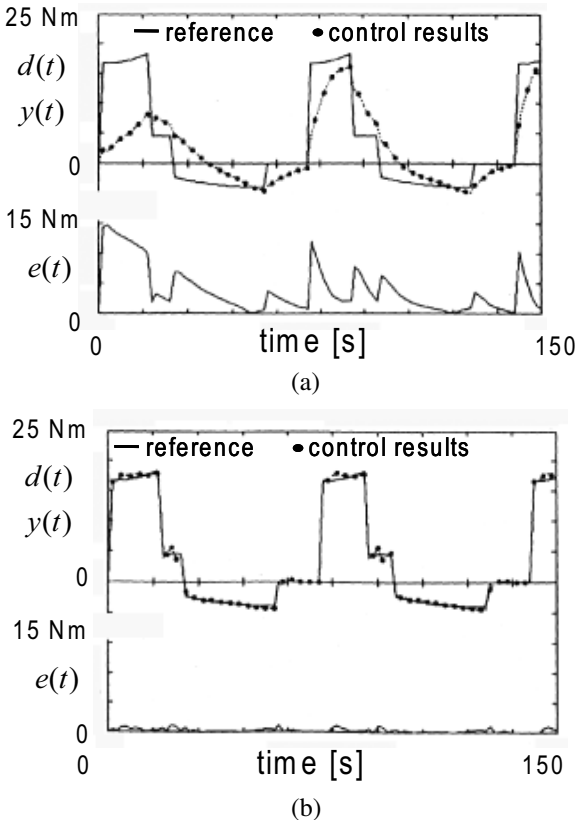


Figure 17. Torque control results by neuro-PID control

could be achieved compared with the values recommended by the company made of the physical simulator.

Figs. 19 and 20 are simulation results for speed control results in case of four modes and eleven mode speed control problems on-load. From these results we can see that the control results in the final stages are almost perfect even for any mode even if control results in the initial stage are not good. In these simulations, learning parameters are $\eta = 0.001 \sim 0.05$.

VII. Conclusions

In this paper we have proposed the neuro-PID control algorithms based on the neural network of layered type. This control method is robust with parameter change and noise as well as the environmental condition. Therefore, we can apply the neuro-PID controller to many kinds of control problems, for example, process control, regulator problem, serbo-problem etc. Among them we have applied the torque and speed control problems of PIVOT electrical vehicle (PIVOT) which has been develped by Shikoku Electric power Ltd, in Japan.

References

[1] Omatu, S., Yoshioka, M., Kosaka, T.: PID Control of Speed and Torque of Electric Vehicl. 2009 Third International Conference on Advanced Engineering Computing and Applications in Sciences Proceedings, Slima, Malta, pp. 157–162(2009)

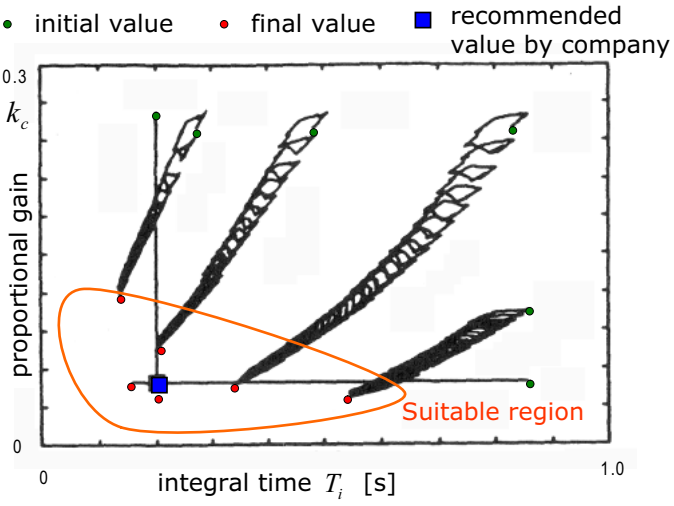


Figure 18. PID control gains.

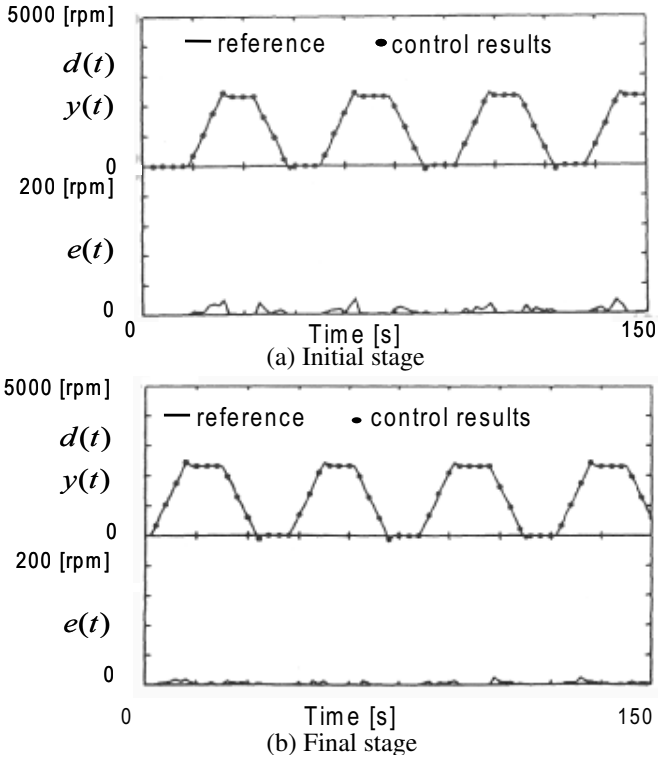


Figure 19. Speed control for four modes case.

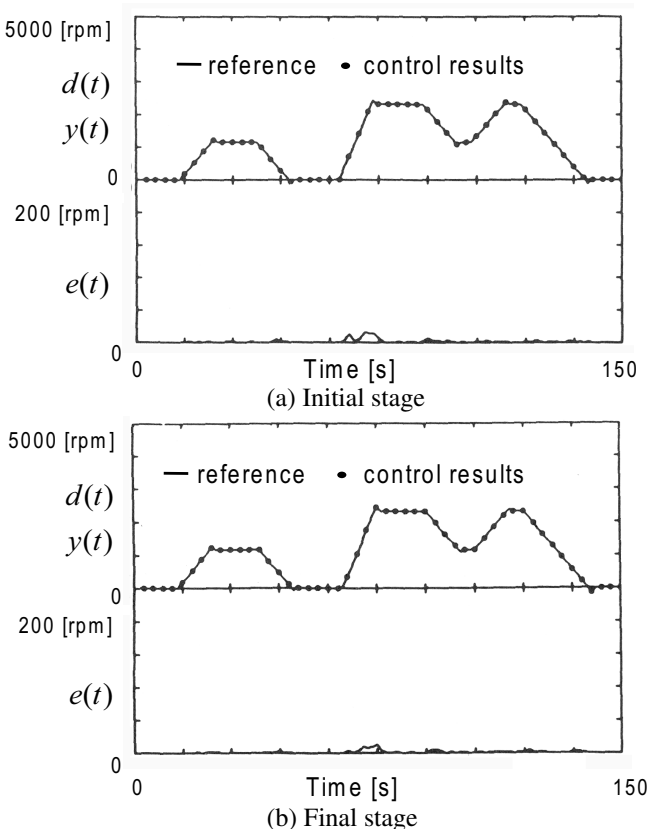


Figure 20. Speed control for eleven mode case.

[2] Widrow, B., F. W. Smith: Pattern-Recognizing Control Systems. Computer and Information Sciences Symposium Proceedings, 288– 317, Spartan, Washington, DC. (1963)

[3] Walt, M.D., Fu, K.S.: A Heuristic Approach to Reinforcement Learning Control Systems. IEEE Transactions on Automatic Control, AC-10, 4, 390–398 (1965)

[4] Michie, D., Chambers, R.A.: An Experiment in Adaptive Control. In Tou J.T. and Wilcox R.H. (Eds.): Machine Intelligence, Edinburgh, Oliver and Boyd, pp. 137-152 (1968)

[5] Barto, A.G., Sutton R.S., Anderson, C. W.:Neuronlike Adaptive Elements that Cn Solve Difficult Learning Control Problems. IEEE Transactions on Systems Man and Cybernetics, 13, 5, 834–846 (1983)

[6] Kawato, M., Furukawa, K., Suzuki, R.: A Hierarchical Neural Network Model for Control and Learning of Voluntary Movement. Biological Cybernetics, 57, 169–185 (1978)

[7] Hunt, K.J., Sbarbaro, R.: Neural Networks for Non-Linear Internal Model Control. IEE Proceedings Control Theory and Applications, 138, 5, 431–438 (1991)

[8] Willis, M.J., Montague, G.A., Dimassimo, C. Tham, M.T., Morris, A.J.: Artificial Neural Networks in Process Estimation and Control, Automatica, 28, 6, 1181–1187 (1992)

[9] Jordan M.I., Jacobs, R.A.: Learning to Control an Unstable System with forward Modelling, In Lippmann R.P., Moody S.E., and Touretzky D.S. (Eds.), Advances in Neural Information Processing Systems, San Mateo, Morgan Kaufmann, San Francisco (1990)

[10] Psaltis, D., Sideris A., Yamamura A.: A Multilayered Neural Network Controller, IEEE Control Systems Magazine, 8, 2, 17–21 (1988)

[11] Omatu, S.: Learning of Neural-Controllers in Intelligent Control Systems, In Zurada, J.M., Marks II, R.J. , and Robinson, C.J. (Eds.): Computational Intelligence Imitating Life, IEEE Press, New York (1994)

[12] White, D.A., Sofge, D.A. (Eds.): Handbok of Intelligent Control, Van Nostrand Reinhold, New York (1992)

[13] Miller, III, W.T., Sutton, R.S., Werbos, P.J. (Eds.): Neural Netoworks for Control, MIT Press, Massachusetts (1990)

[14] Omatu, S., Maruzuki, K., Rubiyah, Y.: Neuro-Control and Its Applications, Springer, London (1996)

[15] Mills, P.M., Zomaya, A.Y., Tade, M.O.: Neuro-Adaptive Process Control, John Wiley & Sons, Chichester, England (1996)

[16] Ng, G.W.: Application of Neural Networks to Adaptive Control of Nonlinear Systems, Research Studies Press, New York (1997)

[17] Rumelhart, D.E., McClelland, J.L., PDP Group: Parallel Distributed Processing, Explorations in the Microsteucture of Cognition Volume 1, MIT Press, Massachusetts, (1987)

[18] Omatu, S.: Neuro-Control Applications in Real-World Problems, Proceedings of the 10th Yale Workshop on Adaptive and Learning Systems, 92– 97, Yale University, New Haven (1998)

[19] Tanomal, J., Omatu, S.: Process Control by On-Line Trained Neural Controllers: IEEE Transactions on Industrial Electronics, 39, 6, 511–521 (1992)

[20] Maruzuki, K., Omatu, S., Rubiyah, Y.: Temperature Regulation with Neural Networks and Alternative Control Schemes: IEEE Transactions on Neural Networks, 6, 3, 572–582 (1992)

[21] Maruzuki, K., Omatu, S., Rubiyah, Y.: MIMO Furnace Control with Neural Networks: IEEE Transactions on Control Systems Technology, 1, 4, 238–245 (1993)

www.iariajournals.org

International Journal On Advances in Intelligent Systems
ICAS, ACHI, ICCGI, UBICOMM, ADVCOMP, CENTRIC, GEOProcessing, SEMAPRO, BIOSYSCOM, BIOINFO, BIOTECHNO, FUTURE COMPUTING, SERVICE COMPUTATION, COGNITIVE, ADAPTIVE, CONTENT, PATTERNS, CLOUD COMPUTING, COMPUTATION TOOLS
issn: 1942-2679

International Journal On Advances in Internet Technology
ICDS, ICIW, CTRQ, UBICOMM, ICSNC, AFIN, INTERNET, AP2PS, EMERGING
issn: 1942-2652

International Journal On Advances in Life Sciences
eTELEMED, eKNOW, eL&mL, BIODIV, BIOENVIRONMENT, BIOGREEN, BIOSYSCOM, BIOINFO, BIOTECHNO
issn: 1942-2660

International Journal On Advances in Networks and Services
ICN, ICNS, ICIW, ICWMC, SENSORCOMM, MESH, CENTRIC, MMEDIA, SERVICE COMPUTATION
issn: 1942-2644

International Journal On Advances in Security
ICQNM, SECURWARE, MESH, DEPEND, INTERNET, CYBERLAWS
issn: 1942-2636

International Journal On Advances in Software
ICSEA, ICCGI, ADVCOMP, GEOProcessing, DBKDA, INTENSIVE, VALID, SIMUL, FUTURE COMPUTING, SERVICE COMPUTATION, COGNITIVE, ADAPTIVE, CONTENT, PATTERNS, CLOUD COMPUTING, COMPUTATION TOOLS
issn: 1942-2628

International Journal On Advances in Systems and Measurements
ICQNM, ICONS, ICIMP, SENSORCOMM, CENICS, VALID, SIMUL
issn: 1942-261x

International Journal On Advances in Telecommunications
AICT, ICDT, ICWMC, ICSNC, CTRQ, SPACOMM, MMEDIA
issn: 1942-2601