VLSI Design

# International Conference on Electronics, Circuits, and Systems

Guest Editors: Jean-Baptiste Begueret and T. Taris

# International Conference on Electronics, Circuits and Systems

# International Conference on Electronics, Circuits and Systems

Guest Editors: Jean-Baptiste Begueret and T. Taris

# Contents

Hindawi Publishing Corporation
VLSI Design
Volume 2008, Article ID 629076, 1 page
doi:10.1155/2008/629076

*Editorial*

# International Conference on Electronics, Circuits, and Systems

**Jean-Baptiste Begueret and Thierry Taris**

*IMS Laboratory, University of Bordeaux, 351 Cours de la Libération, 33405 Talence Cedex, France*

Correspondence should be addressed to Jean-Baptiste Begueret, jb.begueret@ims-bordeaux.fr

Received 25 February 2008; Accepted 25 February 2008

This special issue of VLSI Design is devoted to topics from the 2006 IEEE International Conference on Electronics, Circuits, and Systems (ICECS2006) that was held in Nice, France, 10th–13th December 2006. As in the last year, the selected papers reflect continuing trends toward higher levels of analog and digital circuit techniques covering a wide variety of subjects within the analog and RF integrated circuits and signal processing fields, ranging from basic analog building blocks to system applications.

The major areas of electronics, circuits, and systems covered by ICECS2006 drew 553 submissions from 52 countries spanning the globe; 345 contributions were selected by more than 200 reviewers for presentations. Among the published contributions in the ICECS2006 proceedings, a preselection of 20 papers has been completed and corresponding authors were invited for contributions to this special issue. We received 10 full manuscripts and after the assessment of reviewers, 9 of them have been accepted for publication in this special issue.

The presented contributions in this special issue deal with digital circuits and algorithms dedicated for numerous building blocks of telecommunication applications.

Selecting only 9 papers out of 345 contributions of the conference presentations is a difficult and critical task. We are aware that we probably missed many excellent contributions, but we do know that we did our best to put together a special issue as complete as possible.

The Technical Program of ICECS2006 is the result of a truly international cooperation of experts. We feel very much indebted to Technical Program Committee for its valuable help to keep the level of ICECS at the high standard that our scientific community has grown accustomed to expect from it. We wish to thank our coresearchers from all around the world for choosing to submit their contributions to ICECS for review. Due to their many and important contributions, we managed to create a technical program of high scientific quality. In fact, it has been gratifying to learn more about the advances first described at ICECS2006. We would like to thank the numerous volunteers who helped to review the submitted papers. We also wish to express our deepest gratitude for the efforts of Hindawi personnel. This special issue is only possible with their expert help.

*Jean-Baptiste Begueret*
*Thierry Taris*

Hindawi Publishing Corporation
VLSI Design
Volume 2008, Article ID 160728, 7 pages
doi:10.1155/2008/160728

*Research Article*

# A Programmable Hardware Cellular Automaton: Example of Data Flow Transformation

**Samuel Charbouillot, Annie Pérez, and Daniele Fronte**

*Laboratoire Matériaux et Microélectronique de Provence (L2MP-POLYTECH), UMR CNRS 6137, IMT-Technopôle de Château Gombert, 13451 Marseille Cedex 20, France*

Correspondence should be addressed to Annie Pérez, perez@polytech.univ-mrs.fr

Received 13 April 2007; Accepted 9 December 2007

Recommended by Jean-Baptiste Begueret

We present an IP-core called PHCA which stands for programmable hardware cellular automaton. PHCA is a hardware implementation of a general purpose cellular automaton (CA) entirely programmable. The heart of this structure is a PE array with reconfigurable side links allowing the implementation of a 2D CA or a 1D CA. As an illustration of a PHCA program, we present the implementation of a symmetric cryptography algorithm called ISEA for Ising spin encryption algorithm. Indeed ISEA is based on a 2D Ising spin lattice presenting random series of disordered spin configurations. The main idea of ISEA is to use this disorder to encrypt data. Efficiency of ISEA and PHCA implementation results are given.

## 1. INTRODUCTION

Cellular automata (CA) were originally introduced by von Neumann for studying self-reproduction in biological systems [1]. Then they have been used for language recognition and modelling of physical systems [2]. The mathematical properties of cellular automata were also studied. Nowadays, CA automata are also used for high-quality random numbers generation [3] and implementations of reconfigurable hardware CA are proposed [4].

This paper proposes an intellectual property (IP) core for a programmable hardware cellular automaton (PHCA). PHCA is a powerful tool for the design and test of 1D or 2D cellular automata rule applications. An acyclic one-dimensional cellular automaton and a cyclic two-dimensional cellular automaton can be implemented on the PHCA. The architecture of the PHCA is a fine grained fully parallel structure inspired by a classic single instruction multiple data (SIMD) structure made of 1-bit processing elements (PEs) [5].

An example of PHCA program concerns a cryptography application. The cryptography field is still increasing nowadays. Electronic transactions become very important and require security since most of them are concerned with either payments or confidential data. Public key and secret key cryptographic algorithms provide a solution to this security problem. These algorithms are able to ensure data authenticity, integrity, and confidentiality [6]. Secret key algorithms are more suitable for hardware implementation.

In the context of the secret key algorithms, we propose a symmetric algorithm based on cellular automata rules. This algorithm is called Ising spin encryption algorithm (ISEA) because it uses a system of Ising spins. In this paper, we focus on a 2D Ising spin lattice [7, 8]. The time evolution of the spin configuration in this lattice is managed by local rules leading to disordered configurations in accordance with certain conditions. The configuration space is explored by a random walk imposed by a microcanonical Monte Carlo method [9]. ISEA uses the disordered spin configurations to encrypt data by combining the spin lattice and an array of data to be encrypted. This encryption process is rather fast. Moreover, the permanent exchanges between neighbor sites introduce a constant noise useful against the attacks based on power analysis.

The PHCA may be programmed according to 1D or 2D cellular automata (CA) rules. This work focuses on the PHCA with the 2D configuration and is programmed according to ISEA rules. Each site of the spin lattice system is updated by a PE. All the PEs apply the same rule concurrently. An example of resulting encrypted data array is given below. A first version of a PHCA provided with a 32 × 32 PE array has been implemented on an Xilinx FPGA xc3s5000. The throughput of the encrypted data stream is 16 Mbps.

This paper is divided into six sections. Section 2 shows how the PHCA architecture maps a CA. Section 3 introduces the microcanonical Monte Carlo methods and describes the local rules of the algorithm ISEA. Section 4 shows in detail the encryption process. In Section 5, we present and discuss the ISEA en-/decryption results and the PHCA implementation performances. Finally, Section 6 gives our conclusion.

## 2. PHCA ARCHITECTURE

The aim is to realize a programmable hardware tool suitable for CA rules implementation. The architecture of this tool is inspired by a classic SIMD structure [5].

### 2.1. Mapping a cellular automaton

A cellular automaton consists of several identical cells governed by simple rules. The cellular automaton is globally synchronized; that is, at each time step each cell updates its state according to some set of local rules.

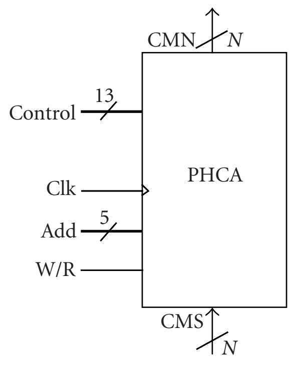
More precisely, the next state of each cell depends on the present state of the neighbor cells [10]. The cell itself may be included in its own neighborhood. A cellular automaton can be of any dimension and can be either cyclic or acyclic. Moreover, CAs are suitable for hardware implementation since they are simple, regular, locally interconnected, and modular.

This work focuses essentially on 2D CA with a north, east, west, and south (NEWS) array of cells that are synchronous, governed by local rules, uniform (i.e., all the cells obey the same rule), and with a von Neumann neighborhood. In this case, the next state $x_{i,j}(t+1)$ of the cell $(i; j)$ depends on its own present state and on the present state of its four nearest neighbors:

$$x_{i,j}(t+1) = f\left[x_{i,j}(t),\ x_{i-1,j}(t), x_{i+1,j}(t), x_{i,j-1}(t), x_{i,j+1}(t)\right]. \tag{1}$$

In order to design an IP-core mapping this definition, we chose to describe a multiprocessor fine-grained structure operating in fully parallel mode. For the instruction stream organization, we chose an SIMD scheme in order to avoid synchronization as well as connection problems.

The heart of this SIMD structure is an array of processing elements (PEs) controlled by the same instruction. The memory is distributed. At each clock cycle, all the PEs execute concurrently the same instruction on the data stored in their internal memory elements. We wanted to map one cell of the CA to one PE. We chose a one-bit architecture for each PE in order to integrate more PEs (more cells) in the array than in the case of more coarse-grained structures. Of course the consequence is that the computation performances slow down when multibit operands must be treated.



| Pin name | Function |
|---|---|
| Control | Control bus |
| Clk | Clock |
| Add | RAM address bus |
| W/R | RAM write(1) read(0) |
| CMS | Data input bus |
| CMN | Data output bus |

Figure 1: PHCA logic symbol and associated pin functions.

### 2.2. PHCA symbol and interconnections

The PHCA logic symbol for an $M \times N$ PE array is given in Figure 1. The external data enter through the $N$-bit south-data bus CMS and exit through the $N$-bit north-data bus CMN. The thirteen control lines bring the same instruction word to each PE. As we shall see below, each PE has a private 32 × 1 bit RAM controlled by the W/R input and addressed by the 5-bit Add input bus. All the registers of the PHCA are synchronized by the same clock Clk.

An example of a 4 × 4 PE array is shown in Figure 2. This regular processor square grid has fixed communication links between the nearest neighbors. Moreover, when all the switches of the west array side are in position 1, the PE array is wrapped around in a toric mode to implement a cyclic two-dimensional cellular automaton. Otherwise, when all the switches are in position 2, the PHCA becomes a chain of PEs to implement an acyclic one-dimensional cellular automaton. This last configuration is not explored in the present work.

### 2.3. PHCA processing element

The PHCA contains $M \times N$ single-bit processor elements. The structure of a PE is detailed in Figure 3. A PE is equipped with a 32×1-bit RAM, five multiplexers, one single-bit arithmetic and logic unit (ALU), four 1-bit registers (NS, EW, C, CM), and input/output ports on all four sides. The ALU is a full adder/subtractor. The result of an addition is given on the ALU outputs CY and SM, and the result of a subtraction on the ALU outputs BW and SM. These ALU outputs CY, SM, and BW correspond also to logic operations in accordance with certain conditions. The registers and RAM accept data from up to eight possible sources through the five multiplexers. The concatenation of these multiplexer's control bits gives the 13-bit instruction word. The instruction set of PHCA is given in Table 1. Up to five commands can be executed simultaneously during each instruction cycle.

N/S and E/W links connect a processor cell to its four neighbors. CMS/CMN links provide the PE array with a second vertical link system which is particularly useful because it does not communicate with the ALU. So these CMS/CMN links allow a south−north shift of the data stream through the whole array concurrently with other PE operations.

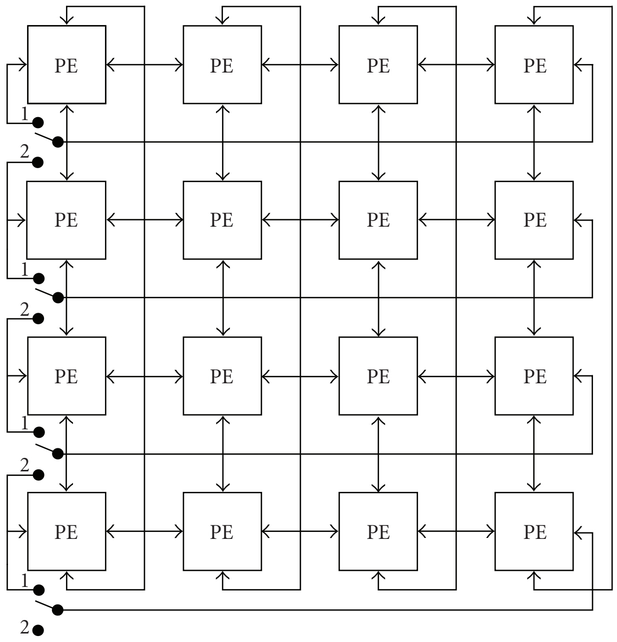


FIGURE 2: Fixed communication links between the nearest neighbors. Configurable links on the west side of the array.

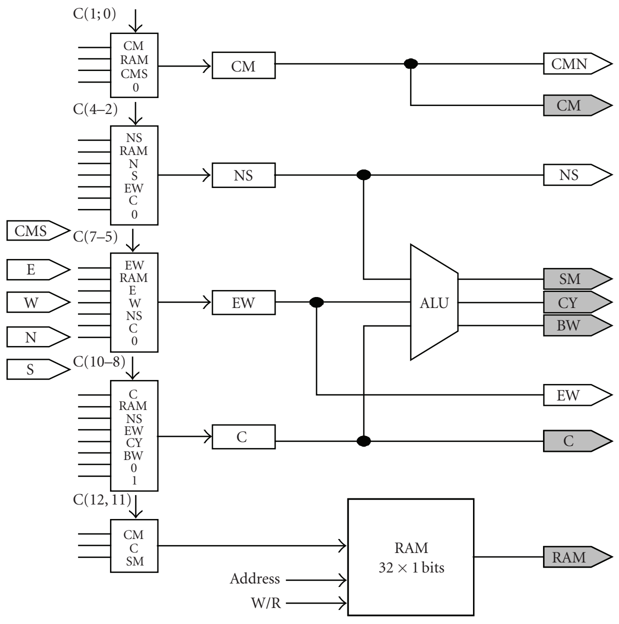


FIGURE 3: PE architecture.

The dark-grey outputs are reinjected as multiplexers inputs into the PE itself.

## 3. MICROCANONICAL MONTE CARLO METHOD

### 3.1. Main idea

Many processes in the nature include the randomness in themselves. This randomness can be used in order to generate long unpredictable key sequences needed by stream cipher schemes. Mathematical models which describe such physical phenomena are probability models.

Since a 2D Ising spin lattice presents a random series of disordered spin configurations, the main idea in the ISEA algorithm is to use this series of configurations to encrypt data. The associated probability model is implemented on a CA with determinist reversible rules.

Numerical simulations are powerful tools to simulate phase transitions on statistical systems. Monte Carlo and molecular dynamics represent two complementary schemes for such simulations. A microcanonical Monte Carlo (MMC) [9] method represents a simulation algorithm interpolating between the Monte Carlo and molecular dynamics techniques. The MMC method consists of taking a random walk on a surface of constant energy. This random walk will generate successive configurations of the statistical system.

In order to ensure a fast and secure encryption of sensitive data through the PHCA, we propose to use these configurations. The PHCA has to perform the three following actions:

(1) storing the successive rows of data to be encrypted coming from the south-input bus CMS and shifting these data through the PE array up to the north-output bus CMN;
(2) ensuring a permanent random walk by executing the microcanonical Monte Carlo local rules;
(3) combining the data flow and the lattice statistical system configurations in order to encrypt the data.

### 3.2. Microcanonical Monte Carlo method

The statistical system to simulate is the 2D Ising model. Let us consider a square lattice of $M \times N$ sites with one spin $S$ at each site. The spins may be up or down. With the MMC method, each site $i$ is also provided with a reservoir containing an energy $E_{ri}$.

Two kinds of energies are involved in this model. The first one is a magnetic interaction energy; for a link $(i, j)$, between two neighbor sites $i$ and $j$, the magnetic energy is expressed by

$$m_{ij} = S_i \text{ xor } S_j. \tag{2}$$

So $m_{ij} = 0$ if the two considered spins point towards the same direction, otherwise $m_{ij} = 1$. The second kind of energy is called "reservoir" energy; it is the sum of all the private site reservoir energies $E_{ri}$.

At each time step, all the spins try to flip. Nevertheless, the flip has a cost in terms of magnetic energy. Indeed, if the spin $S_i$ of site $i$ flips, the magnetic energy varies by

$$\Delta M_i = -2[\Sigma_j(S_i \text{ xor } S_j) - 2], \tag{3}$$

where $j$ refers to the four neighbors of the site $i$. An illustration is given in Figure 4. The local rule is that if $\Delta M_i$ is smaller than or equal to $E_{ri}$, the spin $S_i$ flips. Otherwise, $S_i$ does not change. In other words, if the site has enough reservoir energy to pay the flip, then the spin can flip effectively.

TABLE 1: Instruction set.

| Description | Code C12 · · · C0 | Description | Code C12 · · · C0 |
|---|---|---|---|
| NOP | XXXXXXXXXXX00 | EW← NS | XXXXX100XXXXX |
| CM← RAM | XXXXXXXXXXX01 | EW← C | XXXXX101XXXXX |
| CM← CMS | XXXXXXXXXXX10 | EW← 0 | XXXXX110XXXXX |
| CM← 0 | XXXXXXXXXXX11 | NOP | XX000XXXXXXXX |
| NOP | XXXXXXXX000XX | C← RAM | XX001XXXXXXXX |
| NS← RAM | XXXXXXXX001XX | C← NS | XX010XXXXXXXX |
| NS← N | XXXXXXXX010XX | C← EW | XX011XXXXXXXX |
| NS← S | XXXXXXXX011XX | C← CY | XX100XXXXXXXX |
| NS← EW | XXXXXXXX100XX | C← BW | XX101XXXXXXXX |
| NS← C | XXXXXXXX101XX | C← 0 | XX110XXXXXXXX |
| NS← 0 | XXXXXXXX110XX | C← 1 | XX111XXXXXXXX |
| NOP | XXXXX000XXXXX | RAM← RAM | 00XXXXXXXXXXX |
| EW← RAM | XXXXX001XXXXX | RAM← CM | 01XXXXXXXXXXX |
| EW← E | XXXXX010XXXXX | RAM← C | 10XXXXXXXXXXX |
| EW← W | XXXXX011XXXXX | RAM← SM | 11XXXXXXXXXXX |

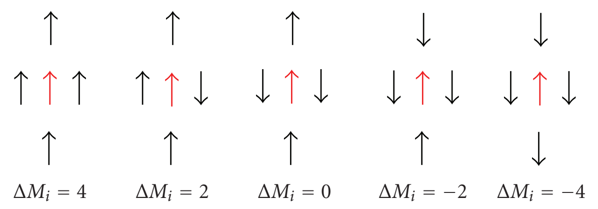


FIGURE 4: Magnetic energy costs $\Delta M_i$ for the central site $i$ spin flip.

## 4. ENCRYPTION PROCESS WITH ISEA

The three actions enumerated in Section 3.1 are quite suitable for cellular automata. Each PE of the PHCA updates one site. A spin-up is coded 0; a spin-down is coded 1. The reservoir energy is 4-bit coded. So two arrays of 1-bit values coexist simultaneously in the PHCA: the array of spins is updated at each time-step and the array of data shifts to the north. In order to encrypt the data, each PE xors the bit of data and the bit of spin.

During the initialization phase, the programmer has to choose the initial spin configuration and to distribute the reservoir energy. Then he has to choose the number of iterations of the MMC rules to compute before xoring the spin bit and the data bit. These choices constitute the key $S_k$ of the encryption process. This cryptography algorithm is symmetric and the key is secret. Let us detail how to store the initial values in the PE array and how to manage iterations of the MMC method on the spin array.

(i) During the loading phase, the spin and the reservoir energy values are presented to the southern side of the PE array through the CMS data bus (see Figure 1). Then these data are shifted to the north. When all the PEs receive the first bit to store through their CMS input (see Figure 3), they store it in their CM register and then transfer it from CM to the RAM. This process is iterated, in bit-serial mode, till all the initial values are stored in the array.

(ii) During the computation phase, according to the MMC method, the operations to be performed are rather simple: xor, shift, addition, subtraction.

## 5. RESULTS AND DISCUSSION

The efficiency of the ISEA algorithm and the results of our first FPGA implementation of PHCA are presented thereafter.

### 5.1. *Application to image en-/decryption*

An application example of our hardware CA programmed with the ISEA algorithm is the color image encryption/decryption system shown in Figure 5. The clear original $640 \times 853$ picture is given in Figure 6(a). Each pixel is coded with 3 bytes (red, green, and blue) so each line of this image can be divided into 120 128-bit words to fit with the PE array horizontal size.

In order to ensure a secure data exchange, both the sender and the receiver need a PHCA with, for instance, $128 \times 128$ PEs. The operations required to encrypt and decrypt are detailed thereafter.

(1) The sender imposes the initial spin values $S$ and distributes the total reservoir energy $R$. Then he programs the PHCA in order to perform $U$ initial spin lattice configuration updates. In the example leading to Figure 6(b) results, the initial configuration of the Ising lattice was all the spins pointing towards down. For the distribution of the reservoir energy $R$, an energy of 2 was distributed to each cell except for 3 cells (called "hot cells") which received an energy of 4. Hot cell coordinates constitute the information $R'$. Moreover, 2000 initial spin lattice configuration updates were carried on. The concatenation of $S$, $R'$, and $U$ constitutes

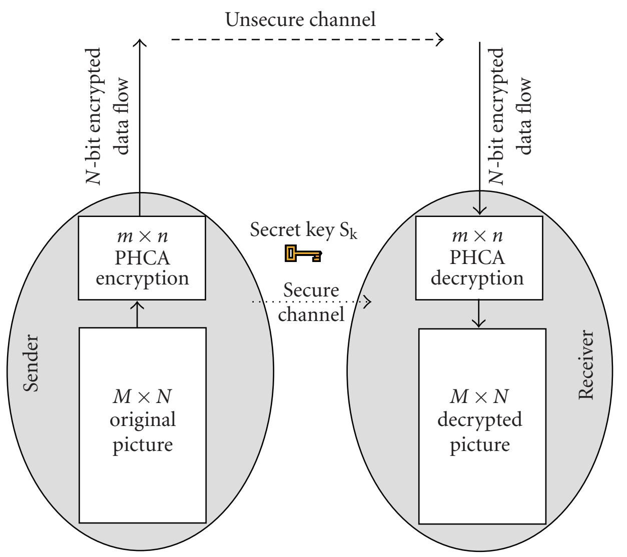


FIGURE 5: Complete PHCA-based encryption/decryption system.

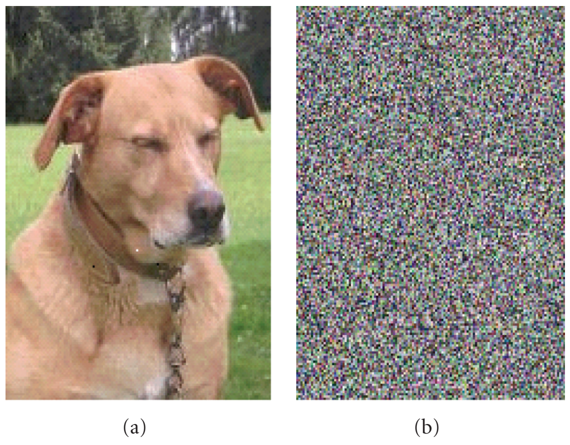


FIGURE 6: (a) Original picture. (b) Encrypted picture.

the secret key $S_k$ which must be transmitted to the receiver through a secure channel.

(2) The sender introduces the clear image through the south side of its PHCA, one word at a time. These data shift to the north and after each shift step, they are xored with the spin lattice configuration. The resulting encrypted image is shown in Figure 6(b). One can notice that the initial picture is completely scrambled at this step.

(3) The receiver gets the secret key $S_k$ through a secure channel. Then he initializes its PHCA with $S$ and $R$ and programs it to perform $U$ spin lattice configuration updates.

(4) The receiver introduces the encrypted message into the south side of its PHCA. These operations allow to exactly recover the initial data picture at the north side of the receiver PHCA.

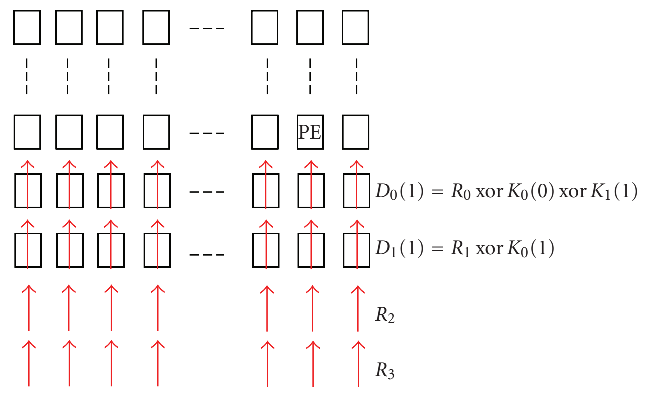

PE
$D_0(1) = R_0$ xor $K_0(0)$ xor $K_1(1)$
$D_1(1) = R_1$ xor $K_0(1)$
$R_2$
$R_3$

FIGURE 7: Random key sequence generation at time $t = 1$.

## 5.2. *Test of randomness*

The pixels in Figure 6(b) seem to be randomly distributed. In order to test the quality of the two-dimensional CA random number generator (RNG) produced by the ISEA algorithm, the Diehard tests [11] were used. The input file for the Diehard test program is a binary file resulting from the concatenation of the random keys $C_i$ generated by the Ising spin configuration.

How can we generate the long unpredictable key sequences $C_i$ necessary for the cipher? Let $K_i$ be the concatenation of all the spin values of a whole lattice row. At time $t = 0$, at the beginning of the encryption process illustrated in Figure 7, the first row $R_0$ of clear data is introduced through the south of the PE array and xored with $K_0(0)$. Then at time $t = 1$, the result $D_0(0)$ is shifted to the north and xored with $K_1(1)$, and so on.

At time $t = t_m$, the first encrypted data row $D_0(t_m)$ available at the north of the PE array is given as follows:

$$D_0(t_m) = R_0 \text{ xor } C(t_m), \tag{4}$$

where $C(t_m) = K_0(0)$ xor $K_1(1)$ xor$,\ldots,$xor $K_m(t_m)$ is the first encryption key of the random sequence.

The battery of the 17 Diehard tests was applied on a sequence of 70 M keys, $C(t_m), C(t_m+1), \ldots, C(t_m+a)$. Figure 8 gives the proportion of passed Diehard tests versus the total reservoir energy $R$. On one side, for low $R$, the spins are "frozen" because the sites have no sufficient $E_r$ to flip their spin. On the other side, for high $R$, all the spins flip simultaneously. These results show that $R$ must be chosen between 1000 and 3000 to obtain high-quality randomness.

Other energy-band values are found depending on the $S$ and $U$ parameter values, on the way of distributing the initial reservoir energy, and on the lattice size. A deeper investigation on the ISEA algorithm efficiency and a comparison with other RNGs are actually in progress.

One can notice other advantages of PHCA and ISEA. First, concerning PHCA, the permanent exchanges between neighbor sites introduce a constant noise useful against the attacks based on power analysis. Then, concerning the ISEA algorithm, the fact that the MMC method conserves the total

Table 2: Encryption cores resource and performance.

| Core | Std Helion | PHCA | ISM |
|---|---|---|---|
| Application | AES | ISEA $32 \times 32$ sites | ISEA $32 \times 32$ sites |
| Technology | Spartan 3–5 | Spartan xc3s5000 | Spartan xc3s5000 |
| Logic resource | 251 slices 3 block rams | 32589 slices | 14148 slices |
| Max clock frequency | 151 MHz | 161 MHz | 132 MHz |
| Max data rate | 402 Mbps | 16 Mbps | 2110 Mbps |
| Programmable | No (dedicated to AES) | Yes (with 1D or 2D AC rules) | No (dedicated to ISEA) |

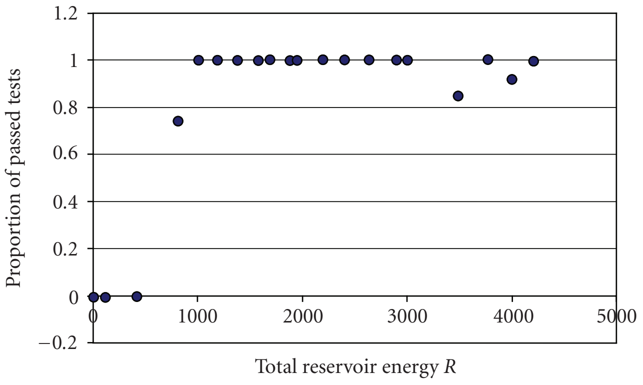


Figure 8: Diehard test results.

energy of the spin system can be used as a test to reveal some hardware anomaly.

### 5.3. *PHCA implementation performances*

Since the PHCA is programmable and has configurable interconnect switches, it is suitable for 1D or 2D CA rules. So it can constitute a powerful tool to elaborate and test cellular automata rules. It can also be used as a multialgorithm CA.

We implemented an Xilinx FPGA xc3s5000, a first version of the PHCA containing a $32 \times 32$ PE array. PHCA implementation results are reported in Table 2. The clock frequency of the PHCA is 161 MHz. The 309 PE instructions are necessary to update a spin array configuration. The throughput of the en-/decrypted data stream is 16.7 Mbps. The 1-bit architecture of the ALU and of the registers constitutes a throughput limitation. For instance, 25 clock cycles are necessary to perform an addition of two bytes.

In order to perform a faster en-/decryption process, we designed a machine called Ising spin machine (ISM). It is dedicated to the implementation of the ISEA algorithm with $32 \times 32$ sites. ISM performs one update every 2 clock-cycles. Targeting also a Spartan-3 device, the throughput is 2 Gbps (see Table 2). ISM goes 125 times faster than PHCA and uses twice less resources; in return it is not a multialgorithm CA.

Table 2 presents also the implementation result of the core Helion [12], a commercial implementation of the well-known secret-key AES algorithm [13]. Helion data rate performance is 5 times slower than ISM and 25 times faster than PHCA. However, Helion is only dedicated to the AES algorithm.

## 6. CONCLUSION

The IP-core PHCA proposed in this work has a fine grained SIMD architecture very suitable to implement cellular automata-based algorithms. The heart of the structure is a PE array with reconfigurable-side links allowing to get a cyclic 2D CA or an acyclic 1D CA.

An application of the 2D CA configuration of PHCA to data flow encryption/decryption using the proposed ISEA algorithm has been presented here and leads to two kinds of conclusions concerning the hardware and the algorithm, respectively.

Concerning the hardware, the implementation of ISEA on PHCA leads to a data rate of 16 Mbps which is 125 times lower than the performance obtained from a core that we designed to be dedicated to ISEA. Nevertheless, PHCA has the important advantage to be programmable. So it can be used as an experimentation platform to test the algorithms efficiency and their implementation on a 2D cell array architecture. If the test is successful, in a second step, macros dedicated to the chosen algorithms can be designed to improve the performances and get smaller area.

Concerning the ISEA algorithm, we saw that ISEA allows to code a data stream using a random walk on a surface of constant energy generated by the MMC method. The high quality of the random number generated by ISEA has been tested by the battery of the 17 Diehard tests.

The random numbers generated by ISEA are used as the long and unpredictable keys needed by the data stream encryption/decryption as presented in this work. Moreover, the elaboration of an experimentation platform for stream ciphers comparison is actually in progress. It uses two Virtex-II FPGA boards (one for encryption and one for decryption). Postimplementation Xilinx ISE simulation results already show that the encryption method using ISEA runs faster than the ciphers using the algorithms presented in [3, 4, 14].

Otherwise, the security of the whole encryption/decryption system compared to secret key security standards is also under investigation.

## REFERENCES

[1] P. Sarkar, "A brief history of cellular automata," *ACM Computing Surveys*, vol. 32, no. 1, pp. 80–107, 2000.

[2] F. Bagnoli and A. Francescato, "A cellular automata machine," in *Cellular Automata and Modeling of Complex Physical*

*Systems*, P. Manneville, N. Boccara, G. Y. Vichniac, and R. Bidaux, Eds., p. 312, Springer, Berlin, Germany, 1990.

[3] M. Tomassini, M. Sipper, and M. Perrenoud, "On the generation of high-quality random numbers by two-dimensional cellular automata," *IEEE Transactions on Computers*, vol. 49, no. 10, pp. 1146–1151, 2000.

[4] R.-J. Chen, Y.-T. Lai, and J.-L. Lai, "Architecture design and VLSI hardware implementation of image encryption/decryption system using re-configurable 2-D Von Neumann cellular automata," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '06)*, pp. 153–156, Island of Kos, Greece, May 2006.

[5] NCR GAPP Application Notes, *NCR Corporation*, Dayton, USA, 1985.

[6] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, Fla, USA, 1997.

[7] C. F. Baillie, *Lattice Spin Models and New Algorithms: A Review of Monte Carlo Computer Simulations*, World Scientific, River Edge, NJ, USA, 1990.

[8] M. Creutz, "Microcanonical Monte Carlo simulation," *Physical Review Letters*, vol. 50, no. 19, pp. 1411–1414, 1983.

[9] M. Creutz, "Deterministic Ising dynamics," *Annals of Physics*, vol. 167, no. 1, pp. 62–72, 1986.

[10] T. Toffoli and N. Margulus, "Programmable matter: concepts and realization," *Physica D*, vol. 47, no. 1-2, pp. 263–272, 1991.

[11] G. Marsaglia, "Diehard," 1998, http://www.stat.fsu.edu/pub/diehard/.

[12] http://www.heliontech.com/aes.htm.

[13] J. Daemen and V. Rijmen, "AES Proposal: Rijndael," September 1999, http://www.esat.kuleuven.ac.be/.

[14] M. D. Galanis, P. Kitsos, G. Kostopoulos, N. Sklavos, O. Koufopavlou, and C. E. Goutis, "Comparison of the hardware architectures and FPGA implementations of stream ciphers," in *Proceedings of the 11th IEEE International Conference on Electronics, Circuits and Systems (ICECS '04)*, pp. 571–574, Tel-Aviv, Israel, December 2004.

Hindawi Publishing Corporation
VLSI Design
Volume 2008, Article ID 351962, 10 pages
doi:10.1155/2008/351962

*Research Article*

# Enabling VLSI Processing Blocks for MIMO-OFDM Communications

**Barbara Cerato,[1] Guido Masera,[2] and Emanuele Viterbo[1]**

[1] *Dipartimento di Elettronica, Informatica e Sistemistica (DEIS), Università degli Studi della Calabria, via P. Bucci, 87036 Rende (CS), Italy*
[2] *Dipartimento di Elettronica, Politecnico di Torino, C.so degli Abruzzi 24, 10129 Torino, Italy*

Correspondence should be addressed to Barbara Cerato, bcerato@deis.unical.it

Received 30 April 2007; Revised 3 December 2007; Accepted 17 January 2008

Recommended by Jean-Baptiste Begueret

Multi-input multi-output (MIMO) systems combined with orthogonal frequency-division multiplexing (OFDM) gained a wide popularity in wireless applications due to the potential of providing increased channel capacity and robustness against multipath fading channels. However these advantages come at the cost of a very high processing complexity and the efficient implementation of MIMO-OFDM receivers is today a major research topic. In this paper, efficient architectures are proposed for the hardware implementation of the main building blocks of a MIMO-OFDM receiver. A sphere decoder architecture flexible to different modulation without any loss in BER performance is presented while the proposed matrix factorization implementation allows to achieve the highest throughput specified in the IEEE 802.11n standard. Finally a novel $E_8$ sphere decoder approach is presented, which allows for the realization of new golden space time trellis coded modulation (GST-TCM) scheme. Implementation cost and offered throughput are provided for the proposed architectures synthesized on a 0.13 $\mu m$ CMOS standard cell technology or on advanced FPGA devices.

## 1. INTRODUCTION

MIMO-OFDM (Multi-input multi-output—orthogonal frequency-division multiplexing) is a very promising communication technique that enables to establish very high throughput and reliable wireless links. In order to achieve this goal, space-time (ST) codes are used, since they can conjugate both transmission rate and reliability enhancement of the communication system. ST codes have been considered for some recently proposed standards such as IEEE 802.11n WLAN and 802.16e WMAN.

However, the computational complexity of MIMO-OFDM receivers is much higher than in the single-input single-output (SISO) OFDM approach; as a consequence the potentials offered by MIMO-OFDM are still far from being fully exploited in actual implementations.

Figure 1 depicts the structure of a 2 × 2 transmit-receive antenna MIMO-OFDM communication scheme. At the receiving side, after the RF/Analog front-end, multiple OFDM demodulation stages, implemented as FFT processors (one per antenna) are allocated, followed by the MIMO signal detector. The adoption of a full-rate and full-diversity ST code demands specific demapping and decoding capabilities, which are covered in Figure 1 by the "ST-code decoder and demapper" block. Finally, a trellis coded modulation (TCM) channel decoder implements forward error correction.

The MIMO channel is modeled by its impulse response between each transmit-receive antenna pair. Assuming $h_{ij}$ represents the time-varying channel fading coefficient between the $j$th transmit antenna and the $i$th receive antenna, the MIMO channel with $M_t$ transmit and $M_r$ receive antennas is described through a $M_r \times M_t$ matrix $\mathcal{H}$, where $h_{ij} \sim \mathcal{N}_c(0,1)$. Transmitted space-time codewords $\mathbf{X}$ are $M_t \times L$ matrices, where $L$ is the number of channel uses required by the ST code. Assuming the "block fading" channel model, each transmitted $\mathbf{X}$ will be affected by an independently varying channel matrix $\mathcal{H}$. Then, the $M_r \times L$ received matrix is

$$\mathbf{Y} = \mathcal{H}\mathbf{X} + \mathbf{Z}, \qquad (1)$$

where $\mathbf{Z}$ is the additive white Gaussian noise matrix with entries $\sim \mathcal{N}_c(0, N_0)$.

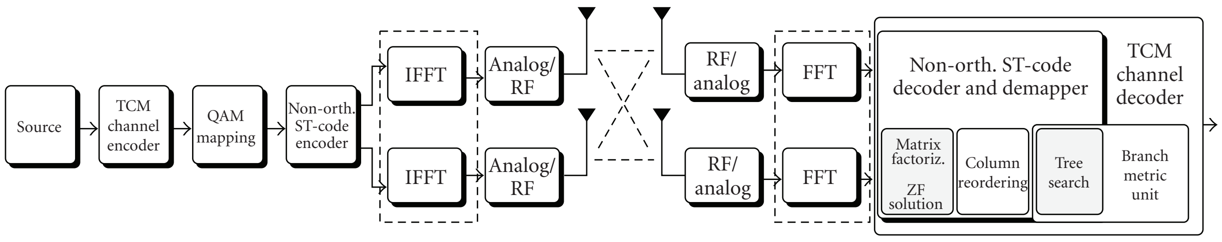


Figure 1: ST-Code MIMO System.

When data symbols belong to a $Q^2$-QAM modulation, it is convenient to represent the codewords $\mathbf{X}$ in vectorized form,where real and imaginary components of the $Q^2$-QAM are separated in two $Q$-PAM modulations, resulting in real component codewords $\mathbf{x}$. Consequently, the channel matrix $\mathcal{H}$ is rearranged in a real-valued matrix $\mathbf{H}$ and $\mathbf{Y}$ is replaced with the real-valued $\mathbf{y}$. For a linear ST block code, $\mathbf{x}$ can be obtained as $\mathbf{x} = \mathbf{Bs}$, where $\mathbf{B}$ is the ST-block code generator matrix and $\mathbf{s}$ is the vectorized data vector with entries in $Q$-PAM. Note that $(Q\text{-PAM})^N$ with $N = 2M_r \times L$ is a hypercubic-shaped constellation carved from a multidimensional integer grid $\mathbb{Z}^N$.

Provided that $\mathbf{H}$ is perfectly known at the receiver, the optimal detector, able to minimize the codeword error rate in a MIMO channel, is the maximum likelihood (ML) detector, which solves the problem

$$\hat{\mathbf{s}} = \arg \min_{\mathbf{s}\in(Q\text{-PAM})^N} ||\mathbf{y} - \mathbf{Ms}||^2, \tag{2}$$

where $\mathbf{M} = \mathbf{HB}$ and $N = 2 \times M_r \times L$. The cardinality of the search space, $Q^N$, depends on the number of receive antennas, the chosen modulation scheme, and the number of channel uses, while the factor 2 comes from the decomposition in real and imaginary components.

Hereinafter, in order to consider a currently practical situation, we will consider a two transmit and two receive antennas system, with a two-channel use ST block code ($M_t = M_r = L = 2$). An example of such a code is the golden code proposed in [1–3] and adopted by the IEEE 802.16e WMAN standard. We then have $N = 8$, $\mathbf{y}$, $\mathbf{x}$, and $\mathbf{s}$ are $8 \times 1$ real vectors and $\mathbf{H}$ is a $8 \times 8$ real-valued matrix. Thus, when using 16-QAM symbols, the direct computation of (2) results in the evaluation of $4^8 = 65,536$ possible solutions.

Due to the high complexity of the exhaustive search, more efficient methods were proposed. Most of these approaches rely on the rearrangement of (2). In particular, a linear transformation such as QR or Cholesky decomposition allows to rewrite $\mathbf{M}$ as the product of two matrices, one of which is upper triangular [4]. Imposing $\mathbf{M} = \mathbf{QR}$, (2) can be rewritten as

$$\begin{aligned}\arg \min_{\mathbf{s}\in(Q\text{-PAM})^N} ||\mathbf{y} - \mathbf{QRs}||^2 &= \arg \min_{\mathbf{s}\in(Q\text{-PAM})^N} ||\mathbf{Q}^T\mathbf{y} - \mathbf{Rs}||^2 \\ &= \arg \min_{\mathbf{s}\in(Q\text{-PAM})^N} ||\tilde{\mathbf{y}} - \mathbf{Rs}||^2,\end{aligned} \tag{3}$$

where we have exploited the orthogonality of $\mathbf{Q}$ and $\tilde{\mathbf{y}} = \mathbf{Q}^T\mathbf{y}$ .

One of the most interesting consequences of this interpretation is that the exploration of the constellation lattice can be thought as a tree traversal. This search tree has $N$ levels and each node in a level has exactly $Q$ sons, representing the points in one dimension of the $Q$-PAM's. This traversal can be done with polynomial complexity adopting the so-called *sphere decoder* (SD), [5].

Recently proposed concatenated ST coding schemes [6] offer a further reliability enhancement by adopting a combined forward error correction approach based on a high rate bandwidth-efficient trellis coded modulation (TCM) scheme. This Golden ST TCM (GST-TCM) scheme for $2 \times 2$ MIMO provides a reasonable ML decoding complexity solution by using Viterbi algorithm and a branch metric computer based on several parallel sphere decoders. A modified sphere decoder is required to support this kind of concatenated scheme, which is an unexplored subject of investigation, from the implementation point of view.

This paper deals with the implementation issues of main processing tasks that enable the development of MIMO receivers. A MIMO detector is organized in two key processing tasks, *matrix factorization* and *sphere decoding* (or *tree traversal*): we then propose efficient architectures for these two key functions. The latter function is the core function of a high performance MIMO detector and its hardware implementation tends to be critical in terms of both throughput and complexity, especially in high data rate systems.

The matrix factorization task operates on the lattice generator matrix $\mathbf{M}$. Since the code generator matrix is constant, the processing must be performed at the channel estimation update frequency, which can change significantly according to the scenario and is generally one or two orders of magnitude lower than the signaling rate. However, in a MIMO-OFDM scheme space, time decoding has to be carried out independently on each subcarrier, determining a dramatic growth of the throughput demand even for matrix factorization.

In Section 2, the sphere decoding algorithm is briefly overviewed, while Section 3 deals with the hardware design of three key building blocks: a sphere decoder, a matrix factorization architecture, and an enhanced sphere decoder for GST-TCM. Finally, Section 4 points out the implementation results achieved for the proposed architectures.

## 2. THE SPHERE DECODING ALGORITHM

Sphere decoding algorithms are a family of algorithms originally proposed to search the closest point to a given one in a lattice. Their use in wireless communications was suggested for the first time in [5], where the lattice structure of multidimensional constellation is exploited to find the closest point to the received vector.

When solving the minimization problem (2), sphere decoding algorithms achieve a polynomial average complexity by exploring only a subset of the solution space [4].

In particular, a hypersphere is constructed around the received vector $\mathbf{y}$ and only points inside it are taken into account. This constraint can be expressed as

$$||\mathbf{y} - \mathbf{M}\mathbf{s}||^2 \leq C_0, \tag{4}$$

where $C_0$ is the square radius of the hypersphere [5, 7, 8].

The upper triangular structure of the matrix $\mathbf{R}$ in (3) enables every component to be separately considered for the computation of the distance between the two points. The distance $d^2(\mathbf{s}) = \|\tilde{\mathbf{y}} - \mathbf{R}\mathbf{s}\|^2$ can also be computed recursively as follows. We consider the partial metrics

$$T_l(\mathbf{s}^{(l)}) = \begin{cases} 0 \quad \text{if } l = N+1 \\ T_{l+1}(\mathbf{s}^{(l+1)}) + \left|\tilde{y}_l - \sum_{j=l}^{n} R_{lj}s_j\right|^2 \\ \quad = T_{l+1}(\mathbf{s}^{(l+1)}) + \left|\tilde{y}_l - \sum_{j=l+1}^{N} R_{lj}s_j - R_{ll}s_l\right|^2 \\ \quad = T_{l+1}(\mathbf{s}^{(l+1)}) + \left|\psi_{l+1} - R_{ll}s_l\right|^2 \\ \quad \text{if } l = 1,\dots,N, \end{cases} \tag{5}$$

where $\mathbf{s}^{(l)} = [s_l, s_{l+1}, \dots, s_N]$, $\psi_{l+1} = \tilde{y}_l - \sum_{j=l+1}^{N} R_{lj}s_j$ with $l = 1,\dots,N$. Since the term $\sum_{j=l+1}^{N} R_{lj}s_j = 0$ for $l = N$, then we have $\psi_{N+1} = \tilde{y}_l$. After $N$ steps, the distance $d^2(\mathbf{s})$ is obtained as $d^2(\mathbf{s}) = T_1(\mathbf{s})$.

As an example, a three-level tree for a 4-PAM modulation is depicted in Figure 2. $T_l$ is the distance metric at level $l$ defined in (5). At every level, the radius constraint (4) must be verified and satisfied, otherwise the branch is pruned. In general, the radius is progressively reduced every time a leaf is reached at a distance that is smaller than current radius.

Several algorithms have been studied in order to make the tree traversal efficient. First algorithm, proposed by Fincke and Pohst in [7], needs to chose explicitly an initial radius.A more efficient solution was proposed by Schnorr and Euchner(SE) [9]. In this case, the initial radius is selected as the distance from the (ZF-DFE) solution and a "depth and best first" traversal of the tree is performed. Originally thought for infinite lattices, the SE algorithm was then adapted to finite lattices [4, 10].

The SE algorithm has intrinsically variable throughput and this makes it not very suitable for hardware implementation. The key to make this algorithm efficient or, at least, with a predictable throughput, is to make an effective pruning. Many theoretical studies in recent literature aim at reaching this goal [11]. A very interesting approach consists in an effective column reordering, which uses heuristic methods to reduce the search complexity with limited performance loss [12]. This technique results in very efficient tree search circuits but additional area is necessary for the preprocessing phase.

On the contrary, the approach proposed in this paper is based on the computational complexity reduction of the tree search algorithm with no column reordering. This solution is suitable for a flexible implementation that can adapt to different modulation sizes.

## 3. VLSI ARCHITECTURES

Implementation architectures for two key building blocks in MIMO detectors are presented in this section (tree search processing and matrix factorization). An enhanced sphere decoder is then described to be applied in the concatenated GST-TCM scheme.

### *3.1. Tree search processing block*

Given the choice of adopting a fully ML detection algorithm for (2), several implementation options have been proposed in the literature.

A first classification can be done with respect to the choice of real- or complex-valued tree construction. In the real case, the tree is twice deeper than the complex one. In complex trees, on the contrary, every node has the square of the number of sons with respect to the real tree. As an example, with $M_t = M_r = 4$ and 16-QAM modulation, a complex-valued tree construction would lead to a 4-level tree, where each node has 16 sons, while 8 levels and 4 sons per node appear in the corresponding real-valued tree. Although [13] demonstrates that a complex-valued tree results in a lower number of visited nodes, the construction of a real-valued tree allows for a more flexible solution, adaptable to different modulation schemes.

Another classification criterion is with respect to the implementation parallelism:

(i) parallelism at the level of tree exploration;
(ii) parallelism at the level of the metric computation for all sons of a given node and in the selection of the most probable son.

The first technique can be adopted only with suboptimal algorithms, while the second approach is not feasible with large cardinality QAM modulation schemes, as it implies a large number of concurrent multiplications. Hence, parallelism is not viable for the implementation of flexible architectures. A serial architecture, designed for high throughput, can achieve both flexibility and low area cost.

Detailed descriptions of the proposed architecture can be found in [14, 15]. The proposed architecture adopts a real-valued tree construction and a serial organization. This key advantage offered by this choice is the possibility of a run-time selection of the modulation scheme. The system is furthermore adaptable to different transmitting schemes including the golden code through the use of some

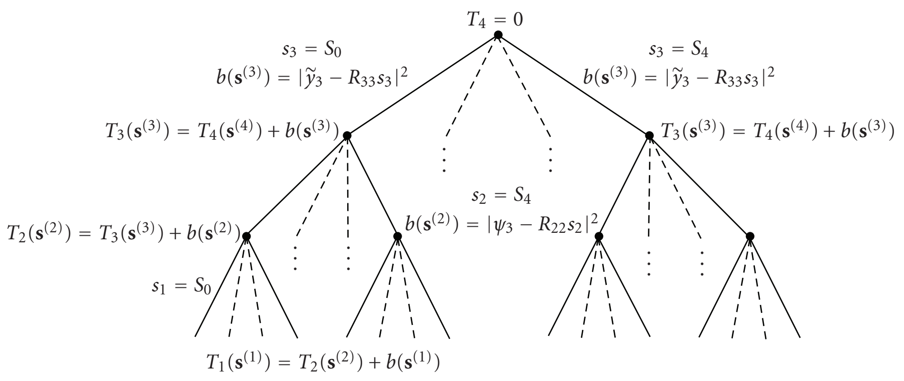

$T_4 = 0$
$s_3 = S_0$
$b(\mathbf{s}^{(3)}) = |\tilde{y}_3 - R_{33}s_3|^2$
$s_3 = S_4$
$b(\mathbf{s}^{(3)}) = |\tilde{y}_3 - R_{33}s_3|^2$
$T_3(\mathbf{s}^{(3)}) = T_4(\mathbf{s}^{(4)}) + b(\mathbf{s}^{(3)})$
$T_3(\mathbf{s}^{(3)}) = T_4(\mathbf{s}^{(4)}) + b(\mathbf{s}^{(3)})$
$s_2 = S_4$
$b(\mathbf{s}^{(2)}) = |\psi_3 - R_{22}s_2|^2$
$T_2(\mathbf{s}^{(2)}) = T_3(\mathbf{s}^{(3)}) + b(\mathbf{s}^{(2)})$
$s_1 = S_0$
$T_1(\mathbf{s}^{(1)}) = T_2(\mathbf{s}^{(2)}) + b(\mathbf{s}^{(1)})$

Figure 2: Tree organization for the sphere decoder. $S$ represent the vector of symbol value in 4-PAM, $[-3, -1, 1, 3]$.

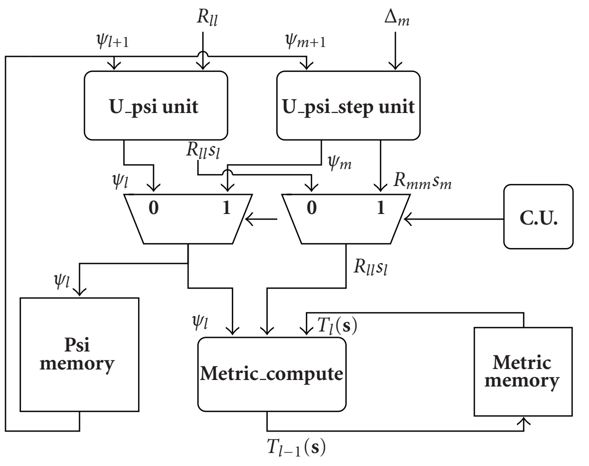


Figure 3: *Sphere decoder* block scheme (case of a node expanded in the depth-first mode, with no pruning).

instantiation parameters, which allow to choose the datapath width and the number of levels of the search tree.

The SE algorithm adopts the "depth and best first" traversal of the tree and the minimization of $|\psi_{l+1} - R_{ll}s_l|$ is required according to the problem formulation given in (5). The computation of the $|\psi_{l+1} - R_{ll}s_l|$ values for all possible $s_l$ tends to become infeasible when the order of the modulation increases due to the large number of required operations. Core of the proposed approach is the selection of the $s_l$ that minimizes $|\psi_{l+1} - R_{ll}s_l|$ by means of the division $\psi_{l+1}/R_{ll}$.

In particular, the iterative evaluation of (5) is rearranged in two steps. At the first step, the value $\psi_{l+1}$ is received as an input from the previous iteration and the desired $s_l$ for the analyzed node is directly obtained through the division $\psi_{l+1}/R_{ll}$; moreover, the output $\psi_l$ is calculated for the selected $s_l$ as $\psi_l = \tilde{y}_{l-1} - \sum_{j=l}^{N} R_{(l-1)j}s_j$. The second processing step receives $\psi_l$ and $T_l$, to actually compute $T_{l-1}$, according to (5). The two operations are performed by units **U_psi_Unit** and **Metric_Compute** in Figure 3, where memories required to store $\psi_l$ amounts and $T_l$ metrics are also shown.

It is worth noting that the result of the division $\psi_{l+1}/R_{ll}$ is rounded to the closest $Q$-PAM constellation points $\pm 1, \pm 3, \ldots$. As a consequence, a general purpose hardware divisor is not necessary and the required operation can be executed by means of the first $\log_2 Q$ steps of a successive subtraction divider [16]. This divider has a very simple architecture that employs only shifts and subtractions; although it tends to be very slow for a complete division, this solution can be effectively used when only a few shift and add elementary operations are required.

In a high throughput sphere decoder, a new $T_l$ metric must be evaluated at each clock cycle. In order to achieve this target, the two steps exploit a pipelined architecture. Additionally, an alternative metric must always be ready, also when a pruning of the tree occurs; therefore, in the proposed architecture, two "candidate" nodes are selected in parallel when processing a given father node. The first one is a direct son of the current node, selected by the **U_psi** unit of Figure 3 by descending along the tree. The "alternative" node, selected by **U_psi_Step Unit**, is placed at a higher level in the tree and it is chosen when the branch has to be pruned, that is when the current metric exceeds the best current metric evaluated in the tree traversal. The procedure adopted to select the alternative node is described below.

In the **U_psi Unit**, the evaluation of the direct son of the current node makes use of the division $\psi_{l+1}/R_{ll}$ and the result is approximated either by defect or by excess to the nearest PAM constellation point: the best choice for $s_l$ is given by (see Figure 4)

$$s_{l_{(1)}} = \frac{\psi_{l+1}}{R_{ll}} + \Delta_l, \qquad (6)$$

where $\Delta_l$ is the correction term. The sign of $\Delta_l$ is exploited to select the second (and following) nearest point in the PAM constellation, according to the following rule:

$$s_{l_{(k)}} = s_{l_{(k-1)}} - (-1)^k \operatorname{sign}(\Delta_l)\,(k-1)\,A, \qquad (7)$$

where $A$ is the distance between two consecutive points.

Thus, **U_psi_Step Unit** simply computes (7) to find the second most probable value of $s_l$. Figure 4 shows the sequence of alternative nodes selected at a given tree level, after the occurrence of pruning.

Summarily, we have the following.

(i) The division approach achieves low complexity and flexibility in terms of supported modulation schemes.

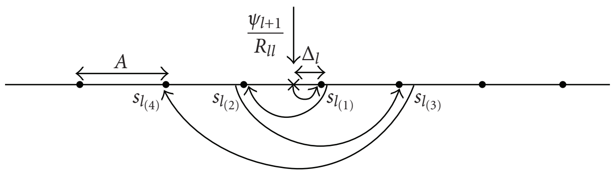


FIGURE 4: Method used to select alternative nodes in U_psi_step unit.

(ii) The concurrent evaluation of two "candidate" nodes provides a significant speed-up to the inherently serial SE sphere decoding algorithm and has a limited impact on complexity.

### 3.2. *Matrix factorization*

Understanding of throughput requirements is fundamental in the architectural study of this processing block. The IEEE 802.11n WLAN standard, which adopts space-time coding, implies that a new channel estimation is performed whenever a packet arrives; this means that the number of matrix factorizations ranges from a minimum of 64 in a time period of 36 microseconds, to a maximum of 128 in 28 microseconds.

In the design of the matrix factorization block, a first choice between householder transformations and Givens-rotations-based algorithms [17] has to be made. The latter approach results in a sequence of rotation operations that cancel elements under the main diagonal of the matrix. Givens rotations require a larger number of floating-point operations compared to householder transformations; nevertheless they may be implemented using parallel systolic arrays and for this reason they are usually preferred for hardware implementation.

Every single processing element (PE) of the systolic array must perform the angle calculation and the rotation to cancel the matrix elements. Several alternatives exist to accomplish these two tasks, and the most common ones are

(1) computation of *sine* and *cosine* of the angle by means of operations including square roots and divisions;
(2) direct angle calculation and rotation using CORDIC processors [18].

The main advantage of the *sine* and *cosine* approach is that primitives can be optimized resulting in an efficient, although expensive, implementation. The second technique is less expensive, but outputs are generated with longer latencies and data dependency between operations. The very high throughput required by this application can hardly be achieved by iterative CORDIC-based algorithms. Other alternatives have to be explored to reduce the latency of every single processor. Among the square root-free algorithms, the squared Givens rotations (SGR) proposed by Dölher [19] constitute a good compromise between complexity and speed [20, 21].

Let us indicate with $\mathbf{a} = (0, \ldots, 0, a_k, \ldots, a_n)$ the row of an $n \times n$ matrix, where a 0 must be introduced in the $k$th position and with $\mathbf{r} = (0, \ldots, 0, r_k, \ldots, r_n)$ another row having the same number of leading zeros; the standard Givensrotations (StdGR) algorithm employs this set of updating equations to cancel the element $a_k$:

$$\overline{\mathbf{a}} = q^{-1}(-a_k\mathbf{r} + r_k\mathbf{a}), \qquad \overline{\mathbf{r}} = q^{-1}(r_k\mathbf{r} + a_k\mathbf{a}), \qquad q = \sqrt{r_k^2 + a_k^2}. \tag{8}$$

The SGR algorithm takes advantage of the observation that $\overline{r}_k = q$ introduces the matrix $\mathbf{U} = \operatorname{diag}(\mathbf{R}) \cdot \mathbf{R}$ and exploits the relations $\mathbf{u} = r_k\mathbf{r}$ and $\overline{\mathbf{u}} = \overline{r}_k\overline{\mathbf{r}}$. Then, to simplify the notation, the new vectors $\mathbf{v} = \mathbf{a}/\sqrt{w}$ and $\overline{\mathbf{v}} = \overline{\mathbf{a}}/\sqrt{\overline{w}}$ are introduced for some $w, \overline{w} > 0$. After some algebra, we can express (8) with a new set of updating equations:

$$\overline{\mathbf{u}} = \mathbf{u} + wv_k\mathbf{v}, \qquad \overline{\mathbf{v}} = \mathbf{v} - \frac{v_k}{u_k}\mathbf{u}, \qquad \overline{w} = wu_k/\overline{u}_k. \tag{9}$$

When compared to StdGR, SGR algorithm shows half the number of multiplications and no square-root operation. The updating sequence can be arranged in a systolic array of PEs performing the aforementioned computations.

The PE array can be arranged according to different structures, namely the triangular (TA), square, and linear (LA) shapes: each of them shows a different percentage of PE reuse and a different throughput. Slightly different functions are then associated in the array organization to boundary and internal PEs.

Figure 5 pictures a generic systolic array layout, able to perform QR decomposition of a $4 \times 4$ matrix. The identity matrix must enter the systolic array immediately after the matrix to be processed, in order to produce the $\mathbf{Q}$ matrix. During the processing of the input matrix $\mathbf{M}$, the coefficients of $\mathbf{Q}$ are already computed and stored in the internal registers.

Depending on (9), boundary and internal processing elements must behave differently when a diagonal element of the matrix enters a node. In Table 1, the computations performed by the nodes in the different operating modes are listed. In the table, Reg and Reg2 are two registers needed to store the parameters between different steps. The subscript in indicate that a parameter takes origin from the preceding PE in accordance with the connections in Figure 5, while subscript out indicates that a parameter takes origin in the current PE. It must be also noted that the parameter $w_{\text{out}}$ is updated only in diagonal mode, while in the other modes it maintains the registered value.

The internal processing element (IPE) appears to be the most computationally intensive block of the entire system. Figure 6 depicts the architecture of the IPEs derived from

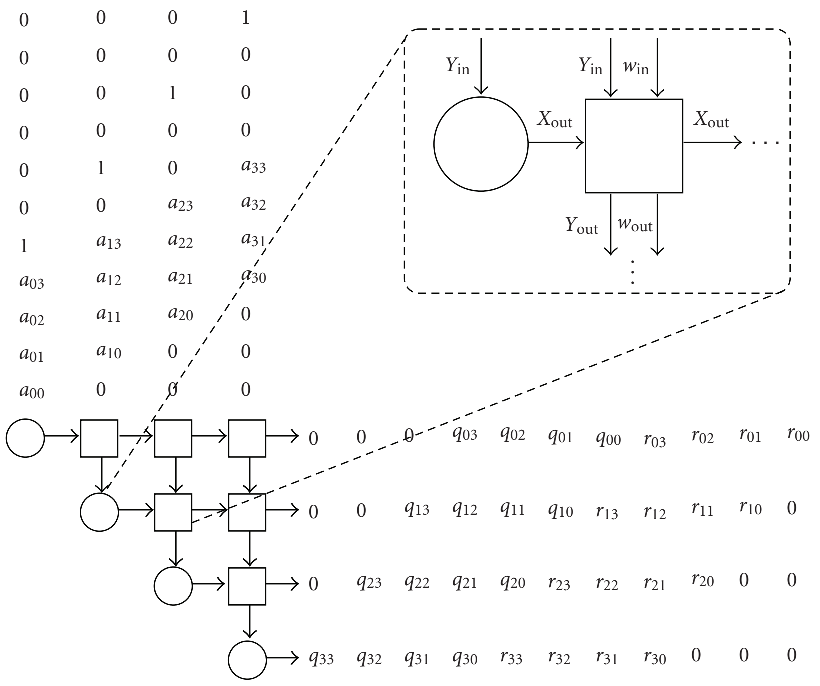


Figure 5: Systolic array for QR decomposition of a $4 \times 4$ matrix.

Table 1: Operations performed by the PE's.

| Mode | Operation |
|---|---|
| **Boundary PE** | |
| Diagonal | Reg $\Leftarrow Y_{in}$; $X_{out} \Leftarrow Y_{in}^2 \cdot w_{in}$ |
| Nondiagonal | Reg $\Leftarrow$ Reg; $X_{out} \Leftarrow Y_{in} \cdot \text{Reg} \cdot w_{in}$ |
| **Internal PE** | |
| Diagonal | Reg $\Leftarrow Y_{in}$; Reg2 $\Leftarrow \dfrac{Y_{in}}{X_{in}}$; $w_{out} \Leftarrow w_{in} \cdot \dfrac{X_{in}}{X_{in} + Y_{in}^2}$ <br> $X_{out} \Leftarrow X_{in} + Y_{in}^2 \cdot w_{in}$ <br> $Y_{out} \Leftarrow Y_{in} - \dfrac{Y_{in}}{X_{in}} \cdot X_{in}$ |
| Nondiagonal | Reg $\Leftarrow$ Reg; Reg2 $\Leftarrow$ Reg2, $w_{out} \Leftarrow w_{out}$ <br> $X_{out} \Leftarrow X_{in} + Y_{in} \cdot \text{Reg} \cdot w_{in}$ <br> $Y_{out} \Leftarrow Y_{in} - \text{Reg2} \cdot X_{in}$ |

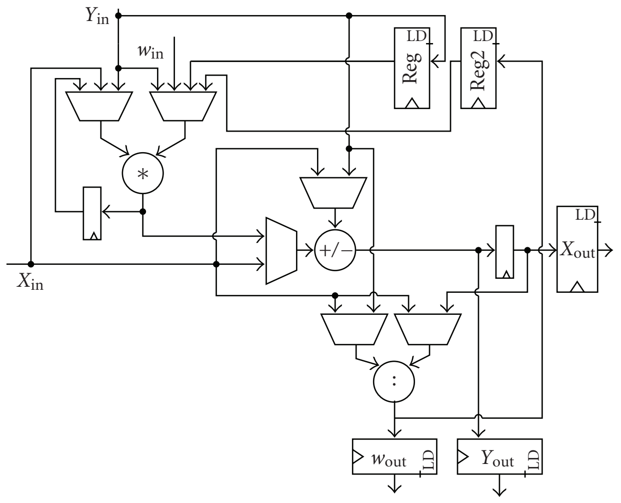


Figure 6: Block diagram of internal PE.

Table 1. Although the divisor has a latency of two clock cycles and two divisions are needed in the diagonal mode, a proper overlapping with the nondiagonal mode guarantees a total latency of three clock cycles.

The method proposed in [22] is adopted to realize the division operation. Using the a Taylor series, the divisor $(Y)$ expressed on $2m$ bits is decomposed into two $m$-bit groups, higher $(Y_H)$ and lower bits $(Y_L)$. Since $(Y_H)^2 \gg (Y_L)^2$, we can write

$$\frac{X}{Y} = \frac{X}{Y_H + Y_L} = \frac{X(Y_H - Y_L)}{Y_H^2 - Y_L^2} \simeq \frac{X(Y_H - Y_L)}{Y_H^2} \tag{10}$$

with maximum fractional error $< 2^{-2m}$. This divisor takes two clock cycles to complete the division on 16 bit fixed-point data [23]; it requires a multiplier, an adder/subtracter, and a 256 8-bit entries LUT to store the inverse of $Y_H^2$. The overall complexity of the internal PE is therefore given by two 16 bit multipliers, two adders/subtracters and a LUT.

In this paper, we considered $8 \times 8$ real matrices as required by the $2 \times 2$ MIMO system with two channel uses per codeword. With a plain triangular architecture, which allows to obtain the highest throughput, a new matrix can enter the array after 16 steps (8 for computing **R** matrix and 8 for **Q**), that is every 48 clock cycles. In order to factorize 64 matrices in 28 microseconds we need to maintain the clock period shorter than 9 nanoseconds, while a period of 4.5 nanoseconds is required to factorize 128 matrices.

### 3.3. Enhanced sphere decoder for $E_8$ lattices

In this section, we address concatenated bandwidth efficient coding schemes for MIMO channels, where a space-time code with nonvanishing determinant is used as inner code and an outer trellis code is concatenated to further increase the reliability of the communication [6].

This TCM exploits the basic idea of partitioning the inner constellation; at each channel use, a signal is selected from one of the partitions. In standard TCM for AWGN channels, the Euclidean distance between points in the same subset is made as large as possible [24]. Full rank ST code design is based on the maximization of the minimum determinant

$$\Delta_{\min} = \min_{\mathbf{X}\neq\hat{\mathbf{X}}} \det\left[(\mathbf{X}-\hat{\mathbf{X}})(\mathbf{X}-\hat{\mathbf{X}})^{\dagger}\right], \tag{11}$$

where $\mathbf{X}$, $\hat{\mathbf{X}}$ are distinct codeword matrices. This pseudo-distance replaces the role of the Euclidean distance. In [6] $\Delta_{\min}$ is optimized using set-partitioning that increases the minimum determinant with the partitions. The $\mathbb{Z}^8$ lattice structure of the inner golden code is used, so that sublattices and their cosets are used as partitions. The outer convolutional encoder guarantees that signals are selected properly from different cosets. Among the possible 8-dimensional sublattices considered in GST-TCM, we choose the Gosset lattice $E_8$ (the densest packing in 8 dimensions [25]).

Any received point has to be decoded to one of the 16 possible cosets of $E_8$ compounding $\mathbb{Z}^8$. The decoder needs to compute the branch metrics of the inner code to perform Viterbi ML decoding of the concatenated codeword. This is obtained by ML lattice decoding of the received vector in each coset of the $E_8$ sublattice.

In order to decode the $E_8$ lattice, we consider that $E_8 \subset \mathbb{Z}^8$ and adapt the classical sphere decoder (as that in [14]) operating on $\mathbb{Z}^8$.

Consequently, this decoding problem can be solved by thinking of $E_8$ as a punctured $\mathbb{Z}^8$ lattice and setting proper constraints to discriminate the relevant points $E_8$ within $\mathbb{Z}^8$. This means that at a given tree level, the integer signal vector cannot assume all values; actually it is constrained by the selections that have already been made at upper levels.

These constraints can be derived directly from the construction A of $E_8$ based on the (8,4,4) extended Hamming code [6]. Let $\mathbf{c} = [c_0,\ldots,c_7]$ denote one of the 16 binary codewords that are used as coset leaders of $2\mathbb{Z}^8$ to obtain $E_8$.

Taking into account that the tree must be traversed starting from the last dimension, we have

$$\mathbf{c} = \begin{cases} c_7 = \text{free}, \\ c_6 = \text{free}, \\ c_5 = \text{free}, \\ c_4 = c_7 \oplus c_6 \oplus c_5, \\ c_3 = \text{free}, \\ c_2 = (c_4 \oplus c_3)\cdot c_5, \\ c_1 = (c_4 \oplus c_3)\cdot c_6, \\ c_0 = (c_4 \oplus c_3)\cdot c_7. \end{cases} \tag{12}$$

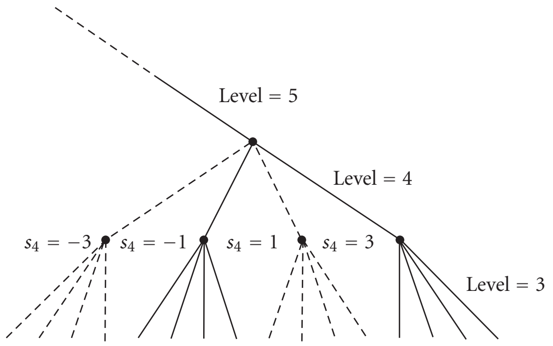


Figure 7: Cross-section at levels 4 and 3, assuming $c_7$ = "1," $c_6$ = "0," $c_5$ = "1" and $\bar{c}_4$ = "1" we obtain $c_4 = c_7 \oplus c_6 \oplus c_5$ = "0" while $c'_4 = \bar{c}_4 \oplus c_4$ = "1" where $c$ is the output of the convolutional encoder and represents a coset leader of $2\mathbb{Z}^8$ in $E_8$ and $\bar{\mathbf{c}}$ is a coset leader of $E_8$ in $\mathbb{Z}^8$.

Table 2: Synthesis results at 0.13 $\mu$m technology for SD and matrix factorization blocks.

| | SD | Matrix factor |
|---|---|---|
| Core Area [GE] | 61 k | 198 k |
| Max. Clock [MHz] | 213 | 223 |
| Throughput at SNR = 20 dB | 148.6 Mbps 16-QAM | 4.63 Mmat/s |

If, at level $i$, $c_i$ is free, then the signal can assume any value in the original QAM constellation, otherwise its value is constrained.

In order to perform the ML detection, we have to derive the proper evolution of the received signal among the different sublattices. In particular, we can define $\mathbf{c}$ as the output of the convolutional encoder, which is related to the current state of the encoder, and $\bar{\mathbf{c}}$ as one of the 16 coset leaders of $E_8$ in $\mathbb{Z}^8$. Combining $\mathbf{c}$ with the coset leader $\bar{\mathbf{c}}$, we obtain a binary vector $\mathbf{c}' = \bar{\mathbf{c}} \oplus \mathbf{c}$ that gives the 256 distinct coset leaders of $2\mathbb{Z}^8$ in $\mathbb{Z}^8$. Thus, all $\mathbf{c}'$ vectors identify the actual allowed points inside $\mathbb{Z}^8$. From the practical point of view, $\bar{\mathbf{c}}$ is fixed for the considered $E_8$ decoder, while the allowed and interdicted values of the signal $s_i$ depend on the value of $\mathbf{c}'$. If $c'_i$ = "0," then $s_i$ can take the values $[\ldots,-7,-3,1,5,\ldots]$, otherwise it can take the values $[\ldots,-5,-1,3,7,\ldots]$; the bounds of this sets depend on the constellation used for the transmission. It is worth noting that, when $c_i$ is free, $c'_i$ can assume both the values 0 and 1, leading $s_i$ to assume any value in the original PAM constellation.

Figure 7 shows levels 3 and 4 of a tree for the sphere decoding of 4-PAM systems: solid lines are practicable edges, while dashed lines correspond the interdicted ones. For this cross-section, we assume $c_7$ = "1," $c_6$ = "0," and $c_5$ = "1," resulting in $c_4$ = "0," $\bar{c}_4$ = "1," and $c'_4$ = "1." Therefore, values $[-1, 3]$ are allowed in this example. At level 3, instead, $c_3$ is free, and as a consequence $c'_3$ can assume both values "0" and "1" and the four branches are all admissible.

Table 3: Comparison results for SD building block.

| SD | Our | | [26] | [27] |
|---|---|---|---|---|
| Antennas | 2 × 2 per two channel uses | | 4 × 4 | 4 × 4 |
| Modulation | 4,16,64-QAM | 16-QAM | 16-QAM | 16-QAM |
| Detector | Depth-first sphere | | Depth-first sphere | $K$-best sphere |
| BER perf. | ML | | ML | Quasi-ML |
| Tech. $\mu$m | 0.13 | 0.25 | 0.25 | 0.35 |
| Core area GE | 61 k +preproc. | 56 k +preproc. | 117 k +preproc. | 91 k +preproc. |
| Max. clock | 213 MHz | 109 MHz | 51 MHz | 100 MHz |
| Throughput @ SNR = 20 dB | 148.6 Mbps 16-QAM | 83 Mbps | 73 Mbps | 52 Mbps |

Table 4: FPGA synthesis results for matrix factorization building block.

| Tech. $\mu$m | xc4vlx200 | xc2v1000 | xc4vlx200 |
|---|---|---|---|
| Handled matrices | 8 × 8 Real | 4 × 4 Complex | 4 × 4 Complex |
| Array | TA | LA | SE |
| no. of PEs | 32 | 4 | 2 |
| $f_{clk}$ MHz | 89 | 101 | 115 |
| Area | 8321(9%) 92 DSP48 | 1666 Slices (32%) 4 BRAM (10%) | 9117 Slices (10%) + 22 DSP48 (23%) + 9 BRAM (3%) |
| Throughput | 1.85 Mmat/s | 0.45 Mmat/s | 0.15 Mmat/s |

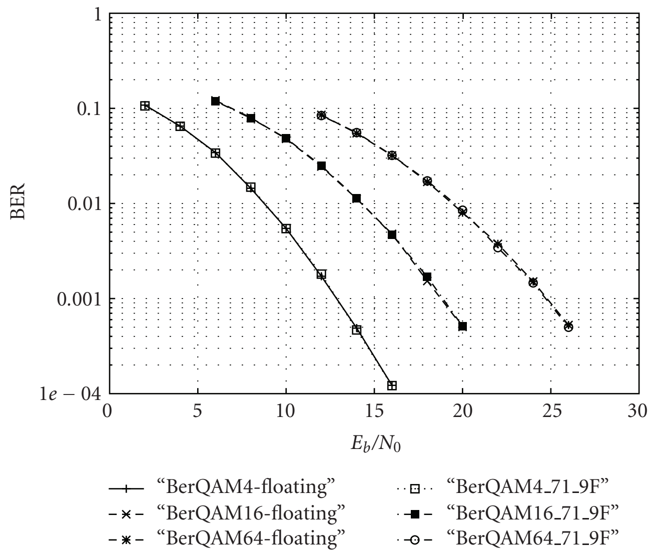


Figure 8: Proposed system performance with different modulations.

The proposed scheme allows to realize with a unique circuit the branch metric computer unit required in the Viterbi algorithm necessary for the decoding of the $\mathbb{Z}^8/E_8$ TCM transmission scheme in [6]. Note that, at each stage of the trellis, 16 different $E_8$ decoders are required.

The adopted architecture is very similar to the architecture described in Section 3.1 and in [14]. The only difference is the additional functional block, the "constraint maker," able to realize (12).

## 4. IMPLEMENTATION RESULTS

Sphere decoder performance in the 2 × 2 golden code scenario described in Section 1 is reported in Figure 8 in terms of bit error rate (BER) versus SNR, for 4-, 16-, and 64-QAM modulations. Fixed-point results are also plotted for the case of a 16-bit data representation (7 bits for integer and 9 for fractional parts): in accordance with [23], these results prove that, for this particular application, 16-bit representation is sufficient to achieve the floating-point performances and thus it has been adopted for all the processing blocks here described.

The proposed architectures have been synthesized on a 0.13 $\mu$m commercial CMOS standard cell technology with synopsys design compiler. The synthesis results are presented in Table 2: the sphere decoder synthesis results here listed are obtained with a flexible architecture able to decode 4 to 64 QAM modulations, while the matrix factorization block has been realized with a triangular array architecture (Mmat/s indicates millions of matrices processed in a second). It must be noticed that synthesis results differ from those in [14], although referred to the same implementation, due to the use of different synthesis libraries.

For comparison purposes, the tree search block has been also synthesized on 0.25 $\mu$m CMOS Standard cell technology (Table 3): we then compare our architecture to the ML implementation described in [26] and the quasi-ML implementation in [27]. It must be noted that two different implementations are presented in [26], one is ML, while the other has close to ML BER performance: as the latter implementation adopts a completely different approach and maps a suboptimal algorithm, only the first implementation figures are included in Table 3 for comparison purposes.

Analyzing data in Table 3, it can be observed that our rearranged approach for the sphere decoder with a single metric computation per cycle allows a significant complexity reduction (approx. 50% for 16 QAM modulation) with respect to parallel structures. At the same time, thanks to the pipelined architecture, we can achieve a remarkable average decoding throughput without any highly specialized structure. Moreover, our flexible decoder is not limited to a single modulation scheme, but it can adapt to different modulations (4-, 16-, and 64-QAM).

Fair comparisons to other implementations cannot be done for the matrix factorization block, as published solutions adopts completely different architectures. For the sake of completeness, we report here two FPGA developments, [21, 28], which implement the SGR algorithm. The main features of these architectures are summarized in Table 4, together with the synthesis results of our solution mapped onto a Xilinx Virtex4 (xc4vlx200) FPGA device. Both [21, 28] carry out the computation of $4 \times 4$ complex matrices, while we process $8 \times 8$ real-valued ones. This means that, while the single PE complexity will be greater in the complex scenario, the number of PE the data flow pass through is twice and with the basic TA topology, while for a $4 \times 4$ matrix there will be 8 PEs, and 32 PEs are required for $8 \times 8$ a matrix.

Another difference among these implementations is related to the processing topology; while our solution adopts a TA processing topology with 32 PEs, [28] makes use of a linear array (LA) organization with 4 PEs and two single PEs are used in [21], one for boundary processing and the second one for internal processing.

A further difference with respect to [28] is that in our implementation weight $w$ is updated according to (9) while in [28] it is fixed to a constant value.

In conclusion, the standard cell version fully reaches both the 64 matrices in 36 microseconds and the 128 matrices in 28 microseconds goals and the throughput of the proposed approach compares favourably to that of the other implementations showing high performances at a limited additional cost. On the contrary, the FPGA implementation enables only to reach the 64 matrices in 36 microseconds.

The $E_8$ decoder, instead, adopting the same architecture as the $\mathbb{Z}_8$ sphere decoder presents a comparable complexity. A little increase in area is due to the addition of the functional block "constraint maker," leading the overall complexity to 62 kGates, and the maximum achievable frequency to 196 MHz.

## 5. CONCLUSIONS

The hardware implementation of key building blocks in a MIMO-OFDM receiver has been presented. The analysis of the blocks shows their high level of complexity, which justifies the ASIC design approach. The sphere decoder architecture enables to manage different modulations without any loss in BER performance while the proposed matrix factorization algorithm and arrangement allow to achieve the highest throughput specified in the 802.11n standard. Finally, the design of an enhanced sphere decoder, capable of supporting $E_8$ decoding in a ST-TCM concatenated schemes, has been proposed.

## REFERENCES

[1] J.-C. Belfiore, G. Rekaya, and E. Viterbo, "The golden code: a $2 \times 2$ full-rate space-time code with nonvanishing determinants," *IEEE Transactions on Information Theory*, vol. 51, no. 4, pp. 1432–1436, 2005.

[2] H. Yao and G. Wornell, "Achieving the full MIMO diversity-multiplexing frontier with rotation-based space-time codes," in *Proceedings of the 41st Annual Allerton Conference on Communication, Control, and Computing*, Monticello, Ill, USA, October 2003.

[3] P. Dayal and M. K. Varanasi, "An optimal two transmit antenna space-time code and its stacked extensions," in *Proceedings of the 37th Asilomar Conference on Signals, Systems and Computers*, vol. 1, pp. 987–991, Pacific Grove, Calif, USA, November 2003.

[4] M. O. Damen, H. El Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Transactions on Information Theory*, vol. 49, no. 10, pp. 2389–2402, 2003.

[5] E. Viterbo and E. Biglieri, "A universal decoding algorithm for lattice codes," in *Proceedings of the 14th GRETSI Symposium on Signal and Image Processing*, pp. 611–614, Juan-les-Pins, France, September 1993.

[6] Y. Hong, E. Viterbo, and J.-C. Belfiore, "Golden space-time trellis coded modulation," *IEEE Transactions on Information Theory*, vol. 53, no. 5, pp. 1689–1705, 2007.

[7] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Mathematics of Computation*, vol. 44, no. 170, pp. 463–471, 1985.

[8] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Transactions on Information Theory*, vol. 45, no. 5, pp. 1639–1642, 1999.

[9] C. P. Schnorr and M. Euchner, "Lattice basis reduction: improved practical algorithms and solving subset sum problems," *Mathematical Programming*, vol. 66, no. 2, pp. 181–199, 1994.

[10] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Transactions on Information Theory*, vol. 48, no. 8, pp. 2201–2214, 2002.

[11] K.-W. Wong, C.-Y. Tsui, R. S.-K. Cheng, and W.-H. Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '02)*, vol. 3, pp. 273–276, Phoenix, Ariz, USA, May 2002.

[12] C. Hess, M. Wenk, A. Burg, et al., "Reduced-complexity MIMO detector with close-to ML error rate performance," in *Proceedings of the 17th Great Lakes Symposium on VLSI (GLSVLSI '07)*, pp. 200–203, Stresa, Lago Maggiore, Italy, March 2007.

[13] A. Burg, M. Borgmann, C. Simon, M. Wenk, M. Zellweger, and W. Fichtner, "Performance tradeoof in the VLSI implementation of the sphere decoding algorithm," in *Proceedings of the 5th IEE International Conference on 3G Mobile Communication Technologies*, pp. 93–97, London, UK, October 2004.

[14] B. Cerato, G. Masera, and E. Viterbo, "A VLSI decoder for the golden code," in *Proceedings of the 13th IEEE International Conference on Electronics, Circuits and Systems (ICECS '06)*, pp. 549–552, Nice, France, December 2006.

[15] B. Cerato, G. Masera, and E. Viterbo, "Decoding the Golden Code: a VLSI design," http://arxiv.org/abs/0711.2383v1.

[16] B. Parhami, *Computer Arithmetic. Algorithms and Hardware Designs*, Oxford University Press, Oxford, UK, 2000.

[17] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The John Hopkins University Press, Baltimore, Md, USA, 1996.

[18] B. Haller, J. Götze, and J. R. Cavallaro, "Efficient implementation of rotation operations for high performance QRD-RLS filtering," in *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP '97)*, pp. 162–174, Zurich, Switzerland, July 1997.

[19] R. Dölher, "Squared givens rotation," *IMA Journal of Numerical Analysis*, vol. 11, no. 1, pp. 1–5, 1991.

[20] R. W. G. Lightbody and R. Woods, "Design of parametrizable silicon intellectual property core for qr-based rls filtering," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 4, pp. 659–678, 2003.

[21] M. Karkooti, J. R. Cavallaro, and C. Dick, "FPGA implementation of matrix inversion using QRD-RLS algorithm," in *Proceedings of the 39th Asilomar Conference on Signals, Systems and Computers*, pp. 1625–1629, Pacific Grove, Calif, USA, October-November 2005.

[22] P. Hung, H. Fahmy, O. Mencer, and M. Flynn, "Fast division algorithm with a small lookup table," in *Proceedings of the 33th Asilomar Conference on Signals, Systems, and Computers*, vol. 2, pp. 1465–1468, Pacific Grove, Calif, USA, October 1999.

[23] L. M. Davis, "Scaled and decoupled Cholesky and QR decompositions with application to spherical MIMO detection," in *Proceedings of the IEEE Wireless Communications and Networking (WCNC '03)*, vol. 1, pp. 326–331, New orleans, La, USA, March 2003.

[24] S. Benedetto and E. Biglieri, *Principles of Digital Transmission With Wireless Applications*, Kluwer Academic/Plenum Publishers, New York, NY, USA, 1999.

[25] J. Conway and N. Sloane, *Sphere Packings, Lattices and Groups*, Springer, New York, NY, USA, 1992.

[26] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1576, 2005.

[27] Z. Guo and P. Nilsson, "A VLSI architecture of the Schnorr-Euchner decoder for MIMO systems," in *Proceedings of the 6th IEEE Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication*, vol. 1, pp. 65–68, Shanghai, China, May-June 2004.

[28] F. Edman, "Digital hardware aspects of multiantenna algorithms," Ph.D. dissertation, Department of Electroscience, Lund University, Lund, Sweden, February 2006.

Hindawi Publishing Corporation
VLSI Design
Volume 2008, Article ID 312614, 8 pages
doi:10.1155/2008/312614

*Research Article*

# Design and Implementation of a Hardware Module for MIMO Decoding in a 4G Wireless Receiver

**Alberto Jiménez-Pacheco,[1] Ángel Fernández-Herrero,[2] and Javier Casajús-Quirós[1]**

[1] *Departamento de Señales, Sistemas y Radiocomunicaciones, Escuela Técnica Superior de Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, Ciudad Universitaria s/n, 28040 Madrid, Spain*
[2] *Departamento de Ingeniería Electrónica, Escuela Técnica Superior de Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, Ciudad Universitaria s/n, 28040 Madrid, Spain*

Correspondence should be addressed to Ángel Fernández-Herrero, angelfh@die.upm.es

Received 18 May 2007; Accepted 26 October 2007

Recommended by Jean-Baptiste Begueret

Future 4th Generation (4G) wireless multiuser communication systems will have to provide advanced multimedia services to an increasing number of users, making good use of the scarce spectrum resources. Thus, 4G system design should pursue both higher-transmission bit rates and higher spectral efficiencies. To achieve this goal, multiple antenna systems are called to play a crucial role. In this contribution we address the implementation in FPGAs of a multiple-input multiple-output (MIMO) decoder embedded in a prototype of a 4G mobile receiver. This MIMO decoder is part of a multicarrier code-division multiple-access (MC-CDMA) radio system, equipped with multiple antennas at both ends of the link, that is able to handle up to 32 users and provides raw transmission bit-rates up to 125 Mbps. The task of the MIMO decoder is to appropriately combine the signals simultaneously received on all antennas to construct an improved signal, free of interference, from which to estimate the transmitted symbols. A comprehensive explanation of the complete design process is provided, including architectural decisions, floating-point to fixed-point translation, and description of the validation procedure. We also report implementation results using FPGA devices of the Xilinx Virtex-4 family.

## 1. INTRODUCTION

The aim of the 4MORE Project (4G MC-CDMA Multiple Antenna System-on-Chip for Radio Enhancements) is to complement worldwide research efforts on MIMO systems, MC-CDMA, and other advanced signal processing techniques that will provide the high data rates and spectral efficiencies expected from 4G wireless multiuser communication systems. In order to investigate the real performance and feasibility of implementation of these technologies, a complete hardware demonstrator of a broadband mobile terminal (MT) has been designed and is being constructed within the 4MORE project [1]. The demonstrator will focus on an MT with two antennas, but a base station (BS) emulator with four antennas will also be built, since it is required for validation of the MT.

Multi-carrier CDMA, based on the serial combination of direct sequence CDMA and OFDM, has been considered for the physical layer in the downlink because it derives benefits from both technologies: OFDM, with appropriate carrier spacing and guard interval, provides robustness against multipath, avoiding intersymbol interference; whereas the use of CDMA with orthogonal spreading codes provides frequency diversity and multiple-user flexibility [2].

The use of multiple antennas is another enabling technology for 4G systems, which helps to exploit spatial diversity, to increase capacity and to mitigate the effects of fading. In our system the space-time block code for two transmit antennas designed by Alamouti [3] is employed. This option has been favoured over other MIMO technologies, such as beam-forming or layered space-time coding (BLAST) because it provides the maximum attainable diversity order for the number of antennas employed using a simple decoding algorithm.

To achieve good bit error rate (BER) performance, state-of-the-art channel coding techniques, including duo-binary

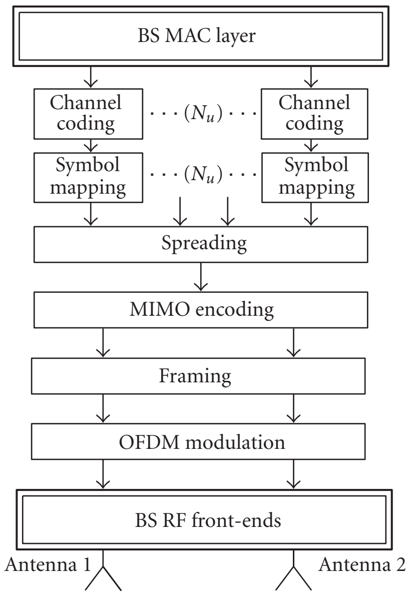


FIGURE 1: Simplified diagram of the BS transmitter.

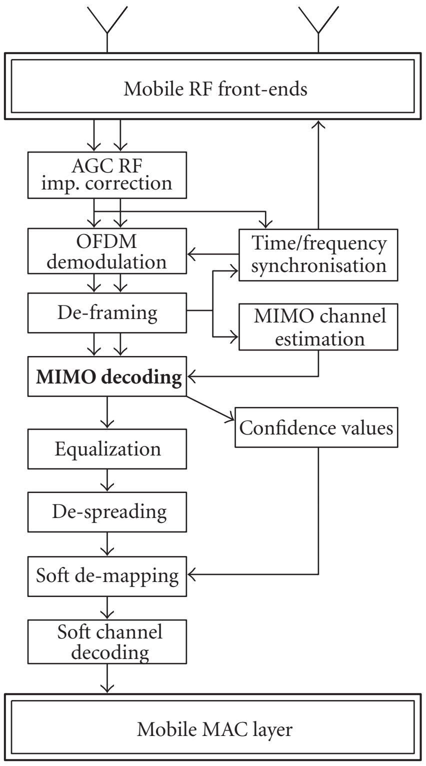


FIGURE 2: Simplified diagram of the MT receiver.

turbo codes [4] for the uplink, and convolutional and low density parity check codes [5] for the downlink, are employed in the 4MORE demonstrator.

The joint use of all these sophisticated technologies greatly increases the complexity of the transceiver. To deal with the constraints of VLSI design, the demonstrator includes ASICs as well as FPGAs. From the onset of the project it was clear that the demonstrator would make use of some well-established algorithms that could be implemented on ASICs, but the flexibility provided by FPGAs was required to accommodate to the more innovative algorithms to be investigated, bearing in mind that design and implementation tasks would partially overlap in time.

The rest of the paper describes the design and implementation in FPGAs of the hardware module that performs MIMO decoding in the MT, and is organized as follows. In Section 2 a brief overview of the complete downlink system is given, where focus is on the receiver. The basis of the Alamouti MIMO decoding scheme is reviewed in Section 3. Sections 4 and 5, respectively, describe the architecture of the MIMO decoder and detail its fixed-point translation. We discuss implementation details and results in Section 6, before we finally draw our conclusions.

## 2. OVERVIEW OF THE DOWNLINK SYSTEM

### 2.1. Transmitting base station

A simplified diagram of the transmitting BS is shown in Figure 1. Data bits to be transmitted to each active user are independently channel encoded and mapped onto symbols of the appropriate constellation (QPSK, 16-QAM or 64-QAM).

Each modulated symbol is multiplied by the spreading code of the corresponding user, and the spread symbols of the $N_u$ active users are added together to be simultaneously transmitted over the same set of $S_f = 32$ subcarriers, which constitutes an MC-CDMA symbol. In our system, the spreading factor in frequency is $S_f = 32$, and the number of users must be in the range of $1 \leq N_u \leq S_f$.

An OFDM symbol consists of $N_s = 21$ contiguous MC-CDMA symbols, so that information is simultaneously transmitted over $N_d = N_s \times S_f = 672$ subcarriers.

Data is prepared for multiantenna transmission by the MIMO encoding module. According to the Alamouti scheme [3], a pair of OFDM symbols $\{\mathbf{x}(n), \mathbf{x}(n+1)\}$, also known as a space-time block, is transmitted employing two antennas over two consecutive symbol periods. During the first symbol period, $\mathbf{x}(n)$ is transmitted from the first antenna, and simultaneously $\mathbf{x}(n+1)$ is transmitted from the second one. During the next symbol interval, the first antenna outputs $-\mathbf{x}^*(n+1)$, while the second one transmits $\mathbf{x}^*(n)$, with $(\cdot)^*$ standing for complex conjugate and $n$ for the symbol epoch. Small bold letters denote vectors with $N_d$ elements, corresponding to the number of data subcarriers in an OFDM symbol.

Before OFDM modulation, the framing module interleaves pilot symbols in the data stream, in order to aid channel estimation at the receiver. One IFFT operation per transmit antenna is required for OFDM modulation, to convert data to the time domain. The IFFT size is 1024, and the sampling rate is 61.44 MHz.

Each stream of complex OFDM symbols is finally IQ-modulated, power amplified by independent RF front-ends, and radiated in the 5-GHz band.

### 2.2. *Receiving mobile terminal*

A simplified diagram of the MT receiver is depicted in Figure 2. Analog signals received by the two antennas of the MT are downconverted to baseband by twin zero-IF RF front-ends, and then sampled at 61.44 MHz. After automatic gain control (AGC) and correction of RF impairments caused by the zero-IF architecture of the front-ends, time and frequency synchronization must be performed in order to minimize misalignments with the transmitting BS.

One FFT operation per antenna branch is required to recover the symbols in the frequency domain (OFDM demodulation).

Next, pilots are split from information symbols by the deframing module. By interpolation of pilot symbols in time and frequency, the MIMO channel estimator provides the MIMO decoder with channel state information (CSI), which is combined with two contiguously received OFDM symbols to build the improved signal from which to estimate the modulated symbols.

However, the output stream of the MIMO decoder further requires module equalization [6] and despreading (separation of users by correlation with their spreading codes) before detection of the desired user can take place. The output of the soft demapper is finally sent to the channel decoder to make decisions about the transmitted information bits.

## 3. MIMO DECODING PRINCIPLE

The fact that during each symbol period both antennas simultaneously transmit different information implies that a linear combination of symbols, affected by the channel frequency response of the different paths, will be received at each antenna of the MT. Due to the intelligent way in which spatial diversity is introduced, a simple linear processing of the signals received by the two antennas during a space-time block eliminates the co-antenna interference (CAI) artificially created by MIMO transmission.

For each space-time block, the MIMO decoder must perform the following linear combination:

$$\begin{aligned}\tilde{x}(n,l) &= \sum_{j=1}^{2}\left[h_{1,j}^{*}(n,l)y_j(n,l)+h_{2,j}(n+1,l)y_j^{*}(n+1,l)\right],\\ \tilde{x}(n+1,l) &= \sum_{j=1}^{2}\left[h_{2,j}^{*}(n,l)y_j(n,l)-h_{1,j}(n+1,l)y_j^{*}(n+1,l)\right],\end{aligned}\tag{1}$$

where $h_{i,j}(n,l)$ is the estimated frequency response of the channel between transmit antenna $i$ and receive antenna $j$ at the $l$th subcarrier ($1 \le l \le N_d$) during the $n$th OFDM symbol period, $y_j$ is the signal obtained after OFDM demodulation at antenna branch $j$, and $\tilde{x}$ is the combined output signal. Assuming ideal channel estimation, and a constant channel response during one space-time block, it can be shown that this combining scheme provides full diversity order and cancels CAI [3], leading to this simple model for the combined signal:

$$\tilde{x}(n,l) = \mathcal{H}(n,l)x(n,l)+\mathcal{N}(n,l),\tag{2}$$

where $x(n,l)$ is the $l$th element of vector $\mathbf{x}(n)$, and $\mathcal{N}(n,l)$ is a Gaussian noise term. Equation (2) is valid for all $n$, but the equivalent channel $\mathcal{H}(n,l)$ has slightly different expressions for even and odd $n$:

$$\begin{aligned}\mathcal{H}(n,l) &= \sum_{j=1}^{2}\left[|h_{1,j}(n,l)|^2+|h_{2,j}(n+1,l)|^2\right],\\ \mathcal{H}(n+1,l) &= \sum_{j=1}^{2}\left[|h_{1,j}(n+1,l)|^2+|h_{2,j}(n,l)|^2\right].\end{aligned}\tag{3}$$

According to (2), information $x(n,l)$ could be now recovered from $\tilde{x}(n,l)$ by zero-forcing equalization (dividing by the real factor $\mathcal{H}(n,l)$) or by MMSE equalization [6].

## 4. ARCHITECTURE OF THE MIMO DECODER

The MIMO decoder must implement (1) to obtain the MIMO-combined signal $\tilde{x}$, and (3) to obtain the equivalent channel $\mathcal{H}$, required by the equalizer.

The memory of the Alamouti scheme is one OFDM symbol. Throughout the paper we have used the pair $(n,l)$ to refer to the OFDM symbol and subcarrier indices. After OFDM demodulation, information received on all subcarriers is converted from parallel to serial, so we recover a single (complex) stream per antenna branch, that is, the $(n,l)$ pair of indices is equivalent to a single-time index $(n-1)N_d+l$. Hence, a straightforward implementation of the decoder would require the storage of a whole OFDM symbol for every input and output signal (real and imaginary parts of the received signal on each antenna, those of the estimates for the $2\times 2$ MIMO channel, those of the combined output signal, and the equivalent channel), making a total of $15\times N_d$ samples. However, if all complex signals in (1) are split in their real and imaginary parts (superscripts $(\cdot)^r$ and $(\cdot)^i$), after some algebra and intelligent grouping of terms, we arrive to expressions that suggest a much more efficient implementation. For example, for the real part of $\tilde{x}$ we get:

$$\begin{aligned}\tilde{x}^r(n,l) &= s_2(n,l)+s_1(n+1,l),\\ \tilde{x}^r(n+1,l) &= s_1(n,l)-s_2(n+1,l),\end{aligned}\tag{4}$$

where we have defined:

$$\begin{aligned}s_i(n,l) &= \sum_{j=1}^{2}s_{i,j}(n,l),\\ s_{1,j}(n,l) &= h_{2,j}^r(n,l)y_j^r(n,l)+h_{2,j}^i(n,l)y_j^i(n,l),\\ s_{2,j}(n,l) &= h_{1,j}^r(n,l)y_j^r(n,l)+h_{1,j}^i(n,l)y_j^i(n,l).\end{aligned}\tag{5}$$

Equation (5) is valid for all $n$, and corresponds to memoryless arithmetic operators that will run continuously, while all memory effects have been included in (4). The architecture inferred from these equations is shown in Figure 3, where all signals are real. All arithmetic resources are disposed so as to make a 100% utilization of them, including the programmable adder/substractor A3 at the output of the module. The whole structure works as a pipeline running at

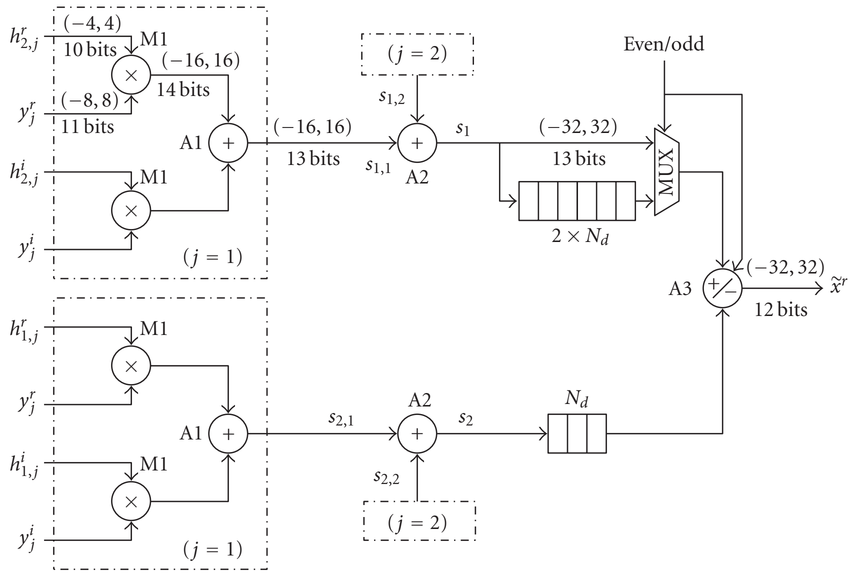


FIGURE 3: Architecture for the MIMO decoder (real part $\tilde{x}^r$). Signal ranges and wordlengths displayed are for the fixed-point implementation option Q2 (see Section 5 and Table 2).

TABLE 1: Parameters of the modes implemented in the demonstrator.

| Modulation | Channel coding rate ($R_{cc}$) | Number of users ($N_u$) |
|---|---|---|
| QPSK ($b = 2$) | 1/2 | 1 to 32 |
| 16-QAM ($b = 4$) | 2/3 | 1 to 32 |
| 64-QAM ($b = 6$) | 3/4 | 1 to 32 |

clock speed and, although not explicitly shown in Figure 3, adders and multipliers have registered outputs. The even/odd signal indicates whether the current OFDM symbol is even or odd, and is used to control the multiplexer and to change between addition and substraction in the programmable adder A3. Slotted rectangles are used to represent multibit shift-registers, which do not need to be resettable. We observe that memory requirements for evaluation of $\tilde{x}^r$ are $3 \times N_d$ samples, and that the total latency is equal to $N_d + 4$ clock periods.

We do not show the full details of the architectures used to evaluate $\tilde{x}^i$ and $\mathcal{H}$ because they are very similar to that shown in Figure 3, just placing the appropriate signals at the inputs. For evaluation of $\tilde{x}^i$, the major difference is that first-level adders A1 are replaced by subtractors, while for $\mathcal{H}$, the programmable adder/substractor A3 is replaced by a simple unsigned adder, the rest of the adders being unsigned as well. Thus, the MIMO decoder comprises three submodules very much like the one shown in Figure 3, and we therefore reduce the total memory requirements of the complete module to $9 \times N_d$ samples.

This architecture can be easily and efficiently adapted to a different number of antennas at the receiver. To this end, the arithmetic blocks surrounded by dotted lines in Figure 3 should be replicated, both in the upper and lower branches of the architecture, and the two-input adders A2 should be replaced by cascaded adders to handle more than two inputs. While deploying more than two antennas at the MT is unpractical, this architecture could also be used for MIMO decoding in the uplink, where a BS with four or more receive antennas is feasible.

## 5. FIXED-POINT TRANSLATION

The fixed-point translation of the architectural design described in the previous section was accomplished following three steps.

(a) Determine the range of each input, output, and intermediate signal involved in the MIMO decoder.
(b) Obtain the number of bits (precision) required for each signal.
(c) Test the robustness of the design by performing BER simulations.

Following this process, similar to that described in [7], we seek to obtain a low-cost, performance-effective implementation for the hardware module.

### 5.1. Estimation of signal ranges

This task was accomplished with the help of the SystemC-based floating-point software simulator that has been developed within the 4MORE Project, which accurately models the behaviour of all the modules in the demonstrator and includes a realistic MIMO channel model. It is possible with this simulator to obtain traces of the signals at any point in the communication link.

We show in Table 1 the most important parameters of the different working modes that have been implemented in the demonstrator. While the range for the channel estimates $h_{i,j}$ is independent of the mode, the range for the

Table 2: Fixed-point quantization rules.

| Signal | | Q1 | | Q2 | | Q3 | |
|---|---|---|---|---|---|---|---|
| | | Range | Bits | Range | Bits | Range | Bits |
| Inputs | $y_j^r$, $y_j^i$ | $(-8.0, 8.0)$ | 12 | $(-8.0, 8.0)$ | 11 | $(-8.0, 8.0)$ | 10 |
| | $h_{i,j}^r$, $h_{i,j}^i$ | $(-8.0, 8.0)$ | 12 | $(-4.0, 4.0)$ | 10 | $(-4.0, 4.0)$ | 9 |
| Output of . . . (combined signal path) | M1 | $(-16.0, 16.0)$ | 14 | $(-16.0, 16.0)$ | 14 | $(-16.0, 16.0)$ | 13 |
| | A1 | $(-16.0, 16.0)$ | 15 | $(-16.0, 16.0)$ | 13 | $(-16.0, 16.0)$ | 12 |
| | A2 | $(-32.0, 32.0)$ | 16 | $(-32.0, 32.0)$ | 13 | $(-32.0, 32.0)$ | 12 |
| Output of . . . (equivalent channel path) | M1 | $(0.0, 16.0)$ | 14 | $(0.0, 16.0)$ | 12 | $(0.0, 16.0)$ | 11 |
| | A1 | $(0.0, 16.0)$ | 15 | $(0.0, 16.0)$ | 11 | $(0.0, 16.0)$ | 10 |
| | A2 | $(0.0, 32.0)$ | 16 | $(0.0, 16.0)$ | 10 | $(0.0, 16.0)$ | 9 |
| Global outputs | $\widetilde{x}^r$, $\widetilde{x}^i$ | $(-32.0, 32.0)$ | 14 | $(-32.0, 32.0)$ | 12 | $(-32.0, 32.0)$ | 11 |
| | $\mathcal{H}$ | $(0.0, 32.0)$ | 14 | $(0.0, 32.0)$ | 10 | $(0.0, 32.0)$ | 9 |

received signals $y_j$ depends on the modulation type and on the number of users. The widest signal range will be attained when 64-QAM modulation is combined with the maximum number of users. By careful examination of histograms of large records of data obtained running the SystemC simulator with these parameters, we found that the range for the real and imaginary parts of the received signals $y_j$ lied with high probability in the interval $(-4.0, 4.0)$ while for the channel estimates $h_{i,j}$ the range was found to be $(-3.0, 3.0)$. The histograms observed for all signals were almost Gaussian in shape. To be on the safe side we decided to include an extra margin, and considered the ranges for $y_j$ and $h_{i,j}$ to be $(-4.0, 4.0)$ for the design. By doing so we try to take outliers into account, and some of the variability of the channel which might have not been captured in our data records. Bear in mind that the channel variability greatly affects the amplitude of the received signals, and that the MIMO channel model is quite complex, its behaviour being influenced by many physical and statistical parameters.

Once the ranges for input signals were known, those of intermediate and output signals could be obtained taking into account the theoretical margins that result when operating with inputs whose range is already known. Nevertheless, this would lead to an overdimensioned module, due to the existence of hidden correlations between the inputs. After all, each of the received signals $y_j$ is a linear combination of the data $x$ multiplied by the channel paths $h_{i,j}$. Therefore, we resorted to histogram observation to determine those ranges. The results are all shown in parentheses in Figure 3 and also in Table 2.

### 5.2. *Word-length optimization*

To ease this task we developed a simple software model of the MIMO decoder, identical to the module included in the floating-point SystemC simulator of the whole chain, but much faster and practical, since all unnecessary burdens were removed. This new software model can be quickly modified to include fixed-point conversion effects in any of its parts.

As performance metric we used the signal-to-quantization noise ratio (SQNR) at the outputs of the MIMO decoder, measured by comparison of the outputs of the floating-point version of the module with that obtained after including quantization effects in some signal, or in all of them. By doing so we seek to keep the power of quantization noise much lower than that of additive white Gaussian (AWGN) noise, hence guaranteeing a negligible effect of the first one on performance.

Fixed-point conversion effects were introduced one signal at a time, and simulations were run in parallel with both versions of the MIMO decoder. The number of bits assigned to the fractional part of the signal under study was then adjusted and simulations repeated until a target value for the SQNR was reached.

Next, fixed-point effects were removed from that point, and we proceeded to optimize the word-length of another signal in the module.

Nevertheless, for those signals that share the same statistics, quantization effects were simultaneously analysed. For instance, optimization of the number of bits at the output of all multipliers M1 in Figure 3 was done simultaneously, running simulations with all multipliers substituted by their fixed-point counterparts, all of them with the same number of bits. For the same reason, all first-level adders A1 were simultaneously optimized, as well as all second-level adders A2.

Following this procedure we obtained, three sets of quantization rules, to which we will refer as Q1, Q2, and Q3 from now on, each of them established aiming at a different goal. The final parameters for these quantization rules are shown in Table 2 (and for Q2, they are also embedded in Figure 3). The number of bits displayed for all signals includes integer plus fractional part.

Quantization rule Q1 was conceived overdimensioned to ensure that it would work with every mode of the demonstrator. Quantization rule Q2, slightly less resource-consuming than Q1, was tried for 64-QAM, but final results were not good enough. As it will be shown in next section, the 64-QAM constellation is very sensitive to even small noise increments. Finally, Q3 was designed to work only with QPSK modulation, using the minimum number of resources.

Signal traces to run the tests were obtained from the complete SystemC simulator, always setting $N_u = 1$, since in this case the range of the inputs is the smallest and therefore the required precision is the highest. We used 64-QAM signals for Q1 and Q2, and QPSK for Q3. The target value for SQNR was set to be greater than 55 dB when designing Q1, 45 dB with Q2, and 35 dB with Q3.

As will be shown later (see Figure 4), the demonstrator may require values of the signal-to-noise ratio (SNR) per information bit ($E_b/N_0$) at the input of the receiver as high as 13 dB to obtain a low BER, the limiting case being that of 64-QAM modulation with 32 users. This is tantamount to a value of the per-carrier signal-to-noise ratio ($\mathrm{SNR}_c$) of approximately 20 dB, since $E_b/N_0$ and $\mathrm{SNR}_c$ are related by [6] by the following equation:

$$\mathrm{SNR}_c(\mathrm{dB}) = E_b/N_0 + 10 \log_{10}\left(b \cdot R_{cc} \cdot \frac{N_u}{S_f}\right). \quad (6)$$

Measurements with signal traces obtained running the simulator in this limiting case resulted in the higher value $\mathrm{SNR}_c = 22.1$ dB at the ouput of the MIMO decoder, the increase being due to the combining process.

At the end of the word-length optimization process we ran a final simulation to compare the floating-point version with the optimized fixed-point one, including all quantization effects simultaneously. The measured SQNR value was about 48 dB for Q1, safely bigger than 20 dB, and output $\mathrm{SNR}_c$ fell only from 22.11 dB to 22.10 dB when including quantization effects. For Q2, the final SQNR was about 40 dB, while $\mathrm{SNR}_c$ fell to 22.05 dB. For Q3, losses in $\mathrm{SNR}_c$ were negligible.

### 5.3. Validation in terms of BER performance

As final step, the SystemC simulator was used to validate in terms of BER performance the final decisions concerning signal ranges and word-length optimization. For this purpose a complete fixed-point software model of the MIMO decoder was developed, which is bit-accurate with the VHDL source code to be implemented in the FPGAs. By substitution of the original floating-point MIMO decoding module by its fixed-point counterpart in the complete SystemC simulation chain, and including appropriate floating/fixed-point interfaces to the neighbouring modules, we verified the degradation in BER performance introduced by the fixed-point MIMO decoder. This can be checked in Figures 4–6, where the BER versus $E_b/N_0$ performance has been evaluated for different modes of the demonstrator.

As it can be seen in Figure 4, quantization Q1 is suitable for every mode, with a maximum loss of about 0.14 dB at BER $= 10^{-4}$ for 64-QAM (negligible with 16-QAM and QPSK). From Figure 5, quantization Q2 can be considered for 16-QAM with a loss up to 0.14 dB, but not for 64-QAM, where losses reach 1 dB. Finally, according to Figure 6, Q3 is suitable for QPSK with negligible losses, while it worsens by 0.3 dB for 16-QAM, a loss double than that obtained using Q2.

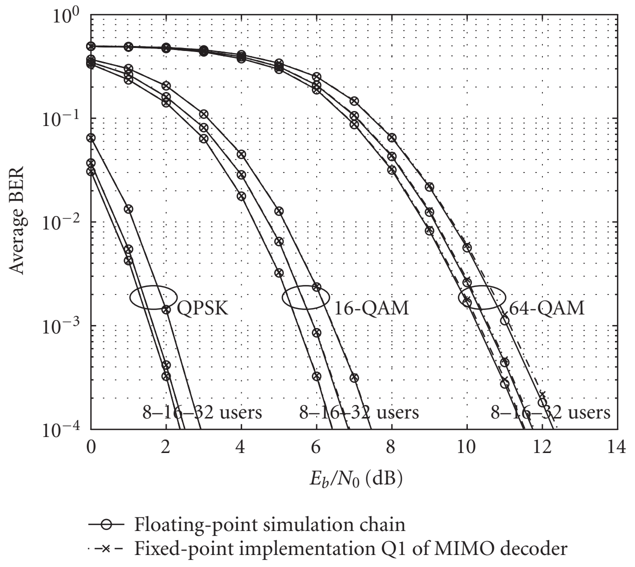


FIGURE 4: BER degradation comparing the floating-point version of the MIMO decoder (solid lines with marker "o") and its fixed-point counterpart implementation Q1 (dashed lines with marker "x").

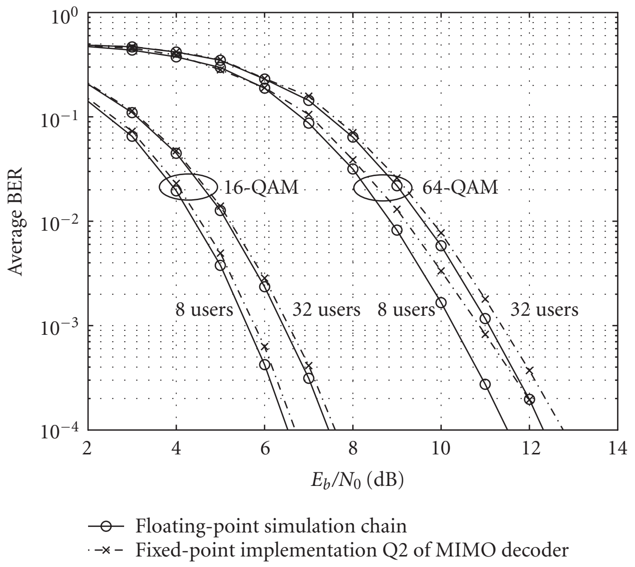


FIGURE 5: BER degradation comparing the floating-point version of the MIMO decoder (solid lines with marker "o") and its fixed-point counterpart implementation Q2 (dashed lines with marker "x").

## 6. IMPLEMENTATION AND RESULTS

The following tools were used during the design: Xilinx ISE 7.1 and the XST engine were used for VHDL synthesis and place-and-route, while Mentor ModelSim SE 6.0d was used to run functional and post place-and-route simulations. The target FPGAs considered for the implementation are Xilinx Virtex-4, since they are most suitable for implementation of wireless systems [8]. Specifically, model XC4VLX100-12 units are included in the demonstrator.

Table 3: Synthesis results for the MIMO decoding module.

| | DSP48 | Flip-flops | Slices | LUTs | Logic | Route-through | Shift registers | DSP slices | Min. clock cycle (ns) |
|---|---|---|---|---|---|---|---|---|---|
| Q1 | Auto | 599 | 3245 | 6337 | 704 | 5 | 5628 | 24 | 7.965 |
| Q1 | Yes | 651 | 3405 | 6321 | 105 | 0 | 6216 | 49 | 9.554 |
| Q2 | Yes | 419 | 2435 | 4544 | 92 | 0 | 4452 | 49 | 9.985 |
| Q2 | Auto | 423 | 2495 | 4946 | 489 | 5 | 4452 | 24 | 6.577 |
| Q2 | No | 759 | 3963 | 7628 | 3163 | 13 | 4452 | 0 | 5.524 |
| Q3 | Auto | 390 | 2308 | 4515 | 436 | 5 | 4074 | 24 | 6.956 |

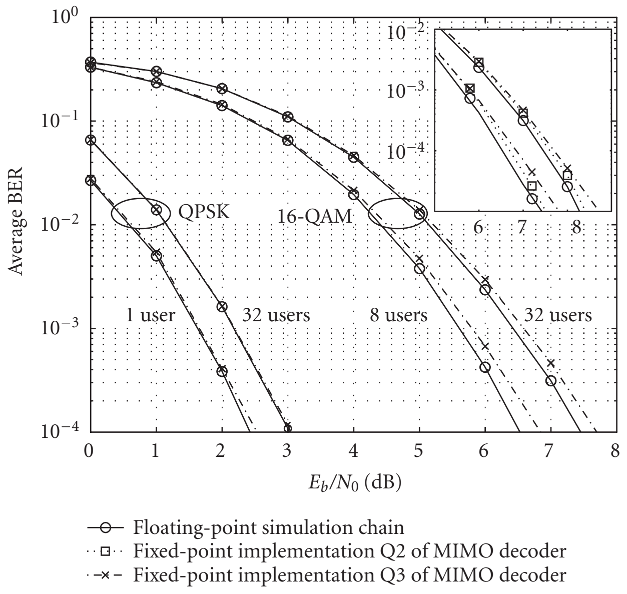


Figure 6: BER degradation comparing the floating-point version of the MIMO decoder (solid lines with marker "o") and its fixed-point counterpart implementation Q3 (dashed lines with marker "x"). In the zoomed area, results for the fixed-point implementation Q2 are also shown for comparison (dotted lines with marker "⊡").

Table 3 shows the synthesis results for the MIMO decoder using the three different fixed-point implementations discussed in Section 5 and summarized in Table 2.

The second column, labelled "DSP48," refers to an option of the synthesis tool which can take three different values: "no" means that no DSP blocks are allowed; "yes" tells the synthesis tool to use as many of them as required; and "auto" triggers a free use of the DSP blocks, depending on the best trade-off found by the tool.

The value of that option has a very significant effect on the column "DSP slices" since the architecture of MIMO decoder needs 24 multipliers. When using "auto" for the "DSP48" option, these are made available as DSP blocks by the synthesis tool, whereas when the "yes" option is selected, the tool also maps the 21 adders (including 15 adders, 4 substractors, and 2 programmable adders/substractors) and other elements in DSP blocks, finally getting 49 DSP slices used, and consequently reducing the number of LUTs in the column "Logic" (from 3163 to 92 for Q2, while shift registers keep the same size).

The column "LUTs" can be obtained by adding the following three: "Logic," LUTs used for logic functions and arithmetic; "Route-through" for routing paths between slices; and "Shift registers." The data in this last column are very relevant for our design, since shift registers are large components in the architecture and consume the greatest part of the resources (except in the case of value "no" for "DSP48"). They affect the slice count, since the width of the registers is reduced when changing to more severe quantizations (from Q1 to Q3).

Considering the total number of slices, there is a reduction of 23% from quantization Q1 to Q2 ("auto"), while it is only 7.5% from Q2 to Q3.

The column "Flip-flops" includes the registers needed in the control unit and also those used for the pipeline. This excludes the registers that follow the arithmetic units mapped to DSP blocks, since they are directly taken from the blocks, and not from the slices.

The last column is the minimum clock cycle inferred by the synthesis tool with a timing constraint of 100 MHz, which is the clock frequency available in the demonstrator. It can be emphasized that the use of DSP blocks results in a slower design, due to the additional routing needed to reach the (fixed) positions of those components in the FPGA. In this regard, the fastest implementation (and also the largest in area) is the one using quantization rules Q2 selecting "no" for the "DSP48" option.

Quantized outputs of the deframing and channel estimation modules (see Figure 2) obtained from the floating-point SystemC simulator were used as realistic input test patterns to perform the functional validation of the hardware implementation. The outputs of the VHDL simulations driven by these patterns were compared for equality with those obtained by the bit-accurate fixed-point software model of the MIMO decoder, when driven by those same input patterns.

## 7. CONCLUSIONS

We have presented the design methodology used in the implementation of a MIMO decoder within a 4G radio system. The architecture of the system has been optimized to comply with the throughput requirements while reducing implementation area.

Given the random nature of the inputs, the design of wireless systems demands a simulation-based fixed-point translation approach for word-length optimization. A robust simulation framework, able to deal both with floating-point and fixed-point descriptions, has proven to be essential in the design.

Several quantization versions have been developed, synthesized with different options, in order to check the trade-offs between accuracy and use of resources in different conditions.

Our implementation results using Xilinx Virtex-4 devices show that the MIMO decoder requires a limited number of FPGA resources, while achieving high performance.

## ACKNOWLEDGMENTS

This work has been supported by European FP6 IST 2002 507039 Project 4MORE and by the Spanish Ministry of Science and Technology under Project TEC2006-13067-C03-03.

## REFERENCES

[1] 4MORE IST project website, http://ist-4more.org.
[2] S. Hara and R. Prasad, "Overview of multicarrier CDMA," *IEEE Communications Magazine*, vol. 35, no. 12, pp. 126–133, 1997.
[3] S. M. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 8, pp. 1451–1458, 1998.
[4] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Transactions on Communications*, vol. 44, no. 10, pp. 1261–1271, 1996.
[5] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, 1999.
[6] A. Fernández-Herrero, A. Jiménez-Pacheco, G. Caffarena, and J. Casajús-Quirós, "Design and implementation of a hardware module for equalisation in a 4G MIMO receiver," in *Proceedings of International Conference on Field Programmable Logic and Applications (FPL '06)*, pp. 1–4, Madrid, Spain, August 2006.
[7] W. Sung and K.-I. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *IEEE Transactions on Signal Processing*, vol. 43, no. 12, pp. 3087–3090, 1995.
[8] "Virtex-4 user guide," March 2006, http://www.xilinx.com/support/documentation/user_guides/ug070.pdf.

Hindawi Publishing Corporation
VLSI Design
Volume 2008, Article ID 892370, 12 pages
doi:10.1155/2008/892370

*Research Article*

# A Time-Consistent Video Segmentation Algorithm Designed for Real-Time Implementation

**M. El Hassani,[1] S. Jehan-Besson,[2] L. Brun,[2] M. Revenu,[2] M. Duranton,[3] D. Tschumperlé,[2] and D. Rivasseau[1]**

[1] *NXP Semiconductors, 2 Rue de la Girafe, B.P. 5120, 14079 Caen, Cedex 5, France*
[2] *Laboratoire GREYC, 6 Boulevard du Maréchal Juin, 14050 Caen, France*
[3] *NXP Semiconductors, High Tech Campus 60, 5656 AE Eindhoven, The Netherlands*

Correspondence should be addressed to S. Jehan-Besson, jehan@greyc.ensicaen.fr

Received 30 April 2007; Revised 13 November 2007; Accepted 17 January 2008

Recommended by Jean-Baptiste Begueret

We propose a time-consistent video segmentation algorithm designed for real-time implementation. Our algorithm is based on a region merging process that combines both spatial and motion information. The spatial segmentation takes benefit of an adaptive decision rule and a specific order of merging. Our method has proven to be efficient for the segmentation of natural images with few parameters to be set. Temporal consistency of the segmentation is ensured by incorporating motion information through the use of an improved change-detection mask. This mask is designed using both illumination differences between frames and region segmentation of the previous frame. By considering both pixel and region levels, we obtain a particularly efficient algorithm at a low computational cost, allowing its implementation in real-time on the TriMedia processor for CIF image sequences.

## 1. INTRODUCTION

The segmentation of each frame of a video into homogeneous regions is an important issue for many video applications such as region-based motion estimation, image enhancement (since different processing may be applied on different regions), 2D to 3D conversion. These applications require two main features from segmentation: accuracy of regions boundaries in the spatial segmentation and temporal stability of the segmentation from frame to frame.

As far as spatial segmentation is concerned, it can be classified into two main categories, namely, contour-based and region-based methods. In the first category, edges are computed and connected components are extracted [1]. One of the drawbacks of such an approach is that the computation of the gradient is prone to large errors especially on noisy images. Moreover, the closure of the edges in order to create connected regions is a difficult task and an efficient resolution of such a problem may induce cumbersome computations. Finally, such an approach cannot take benefit of statistical properties of the considered image regions. The region-based segmentation methods avoid these drawbacks by considering regions as basic elements. Among region-based segmentation methods [2–6], we are interested here in a bottom-up segmentation approach where regions are grown using a merging process. In such approaches, similar neighbouring regions are merged according to a decision rule [7, 8]. The initial regions can be the pixels or an over-segmentation of the image which can be obtained by a watershed algorithm [9, 10]. As mentioned by [11], bottom-up algorithms rely on three notions: a model for the description of a region, a merging predicate, and a merging order. This gives rise to numerous heuristics according to the different choices performed on these three steps [4, 7, 12–14]. Compared to other classical approaches, for example, [7, 12, 13], the authors of [4] have proposed recently an adaptive threshold justified by statistical inequalities. They obtain good results with few parameters to tune. However, in the context of a real-time implementation, their merging predicate still requires too many computations. Moreover, their algorithm is dedicated to the segmentation of still images and so, it does not take into account the temporal dimension of video sequences.

When dealing with video segmentation, various algorithms have been tested in the literature. The first class of approaches proposes to perform a 3D segmentation by considering the spatiotemporal data as a volume. We can cite the work of [15] that takes benefit of the 3D structures tensor for segmentation. Some other recent works propose 3D approaches using a mean-shift-based analysis

[16, 17]. Let us note that if each shot is segmented as a 3D volume, the number of frames to store for each segmentation may be unbounded. On the other hand, if the number of stored frames is artificially limited by the available memory, some 3D regions may be artificially split on long shots. Therefore, 3D approaches require the storage of several frames in memory and necessitate a high bandwidth which is a drawback for the design of electronic devices.

The second class of methods concerns frame-by-frame algorithms. In these approaches, the spatial segmentation of the second frame is deduced from the spatial segmentation of the first frame using motion estimation [13, 18–20]. Regions from adjacent frames are then merged according to motion similarity, colour similarity, or localisation similarity. In such approaches, a matching is performed between regions of the different frames. All the regions are then linked and video objects tracking algorithms [20] may then take benefit of such a correspondence between regions.

On the other hand, some applications, such as image enhancement or video compression, may need a coherent segmentation between frames without requiring an exact tracking of each region from frame to frame. In this paper, we propose a segmentation algorithm devoted to such applications. The first aim of our algorithm is thus not to match the regions of two consecutive frames but only to take benefit of the spatial segmentation of the first frame in order to construct a coherent spatial segmentation of the second one.

Our contributions may be divided in three points.

(i) Spatial segmentation: our spatial segmentation takes benefit of both an adaptive decision rule and an original order of merging. As in [4], the adaptive threshold is computed using a statistical modelisation of the region combined with the statistical inequality of McDiarmid [21]. However, in our approach, each pixel is modelled as a single random variable (in [4], the authors model each pixel as a sum of $M$ random variables). This method gives a simpler predicate that is more adapted to real-time implementation. Good results are obtained for spatial segmentation with few parameters to be set.

(ii) Temporal consistency: another contribution is the design of a region segmentation that does not encounter strong variations over time. We propose to simply take benefit of scene-change detection, that is widely used in video segmentation [22–24], rather than motion estimation that remains a real bottleneck for real-time implementation. We construct a coherent segmentation from frame to frame by combining both pixel and region information through the use of an improved change detection mask (CDM) that takes benefit of the region segmentation of the previous frame. Experimental results conducted on real video sequences demonstrate a good temporal consistency.

(iii) Hardware implementation: as far as the implementation is concerned, we exploit the data level parallelism (DLP) by processing some basic treatments in parallel. Moreover, the classical union-find data structure [25] is improved by using local registers to reduce the access time of find operations. We obtain an efficient algorithm for video segmentation at a low computational cost. Our method runs in real time on the TriMedia processor for CIF image sequences.

The paper is organised as follows. The spatial segmentation method is detailed in Section 2. The temporal consistency improvement is explained in Section 3. In Section 4, we discuss the implementation of the algorithm. Experimental results and measures are given in Section 5.

## 2. SPATIAL SEGMENTATION

Let us consider an image $I$, the notation $|\cdot|$ represents the cardinal and $I(p,n)$ the pixel intensity at position $p = (x,y)^T$ in the frame $n$.

A region-based segmentation problem aims at finding a relevant partition of the image domain in $m$ regions $\{S_1, S_2, \dots, S_m\}$. We focus here on region-merging algorithms where a decision criterion determines whether two regions must be merged or not. In this paper, we first introduce a statistical model for the regions. We then detail how these statistical tools are used for the computation of the merging predicate. We finally explain the whole merging algorithm and especially the order of merging.

### *2.1. Statistical model*

Images are corrupted by noise which gives random values (r.v.) to pixel intensities. Due to this random part in image acquisition systems, an image $I$ is classically considered to be an observation of a perfect statistical image $I^*$. The intensity $I(p)$ of a pixel $p = (x,y)^T$ is then modelled as the observation of a random vector $X_i$ whose values belong to the interval $[0,g]$ (e.g., $g = 255$ for 8 bits images). An ideal region $S^*$ is then represented by a vector of independent r.v. $(X_1, X_2, \dots, X_n)$, where $n = |S^*|$. Let us denote by $S$ the real region associated to $S^*$, that is, composed of the same set of pixels as $S^*$. The intensity of the $i$th pixel of $S$ within $I$ is then considered as an observation of the r.v. $X_i$. Following [4], we define a partition of $I^*$ into homogeneous regions $\{S_1^*, \dots, S_m^*\}$ by the following requirements:

(1) all the pixels of any statistical region should have the same expectation

$$\forall i \in \{1,\dots,m\}, \quad \forall (p,q) \in (S_i^*)^2, \\ E(I^*(p)) = E(I^*(q)); \tag{1}$$

(2) two adjacent pixels belonging to different statistical regions should have different expectations

$$\forall (i,j) \in \in \{1,\dots,m\}^2, \quad \forall (p,q) \in S_i^* \times S_j^*, \\ E(I^*(p)) \neq E(I^*(q)) \tag{2}$$

Such a definition may be easily extended to multichannel images [4] by requiring that the pixel expectations are equal on each channel within one region and that the expectation of at least one channel differs between pixels belonging to different regions.

Note that according to our definition, all the pixels of one region should have the same expectation. The regions extracted by a segmentation algorithm based on this definition should thus be composed of pixels with a nearly constant intensity (we thus assume an underlying flat facet model). This criterion may be justified by the reflective properties of surfaces. Indeed, the reflection of light under a surface is determined by a Lambertian and a specular component [26]. The specular component produces specular spikes often characterised by regions with a nearly maximal intensity. The specular component decreases abruptly and may be neglected, within a segmentation scheme, outside the specular spikes. The intensity of a Lambertian surface varies slowly according to its normals. A region of the image with a nearly constant value correspond thus either to a specular spike or to a Lambertian surface with an almost constant normal. Such a segmentation scheme provides thus a partition which resumes the main physical and geometrical properties of a 3D scene. Higher-level processes such as the segmentation of the image into objects or the segmentation of textured objects [27] would require to input within the algorithm a priori knowledge about what are the expected objects of the scene or what a textured area is.

In order to be selfcontent, let us now introduce the very useful statistical inequality proposed by [21] and introduced within the region segmentation framework by [4]. We take benefit of this inequality for the computation of the merging predicate.

**Theorem 1** (McDiarmid's inequality). *If $\{X_l\}$ are $N$ independent random variables whose observations $x_l$ take their values in a measurable space $A$, and $f : A^N \longmapsto \mathbb{R}$ is a function that satisfies the following constraint for $1 \le l \le N$:*

$$\sup \left| f(x_1,\dots,x_N) - f(x_1,\dots,x_{l-1},x'_l,x_{l+1},\dots,x_N)\right| < c_l, \tag{3}$$

*where $x_l$ and $x'_l$ are two different possibilities for the lth component of an observation vector $(x_1,\dots,x_N) \in A^N$. Then for every $\epsilon > 0$,*

$$P\left(\left| f(X_1,\dots,X_N) - E(f(X_1,\dots,X_N))\right| > \epsilon\right) \le 2\exp\left(\frac{-2\epsilon^2}{\sum_{l=1}^N c_l^2}\right). \tag{4}$$

### 2.2. Merging predicate

In order to compute a merging predicate, we consider two regions $S_1$ and $S_2$ of a current partition. The associated vectors of r.v. in the ideal image $I^*$ are respectively denoted by $\mathbf{Y}_1$ and $\mathbf{Y}_2$. The r.v. $\mu_1(\mathbf{Y}_1)$ and $\mu_2(\mathbf{Y}_2)$ denote respectively the means of $\mathbf{Y}_1$ and $\mathbf{Y}_2$. We suppose that $\mathbf{Y}_1$ and $\mathbf{Y}_2$ belong to the same homogeneous region of $I^*$. Our default decision rule consists thus to merge the two regions $S_1$ and $S_2$, respectively associated to $\mathbf{Y}_1$ and $\mathbf{Y}_2$. However, under the hypothesis that $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are included in the same homogeneous region of $I^*$, the probability that $|\mu_1(\mathbf{Y}_1) - \mu_2(\mathbf{Y}_2)|$ is greater than a given value is bounded by Theorem 1. If this probability falls under a given threshold, we refuse the hypothesis and thus do not merge the two regions $S_1$ and $S_2$.

More precisely, let us consider the vector

$$\mathbf{Y} = (\mathbf{Y}_1,\mathbf{Y}_2) = (\underbrace{I^*(p_1),\dots,I^*(p_{|S_1|})}_{\mathbf{Y}_1},\underbrace{(I^*(p'_1),\dots,I^*(p'_{|S_2|})}_{\mathbf{Y}_2}) \tag{5}$$

and the mean functions

$$\mu_i(\mathbf{Y}_i) = \frac{1}{|S_i^*|}\sum_{k=1}^{k=|S_i|} I_i^*(p_k), \quad i = 1,2. \tag{6}$$

Our merging decision rule is based on the following theorem.

**Theorem 2.** *Let one consider two vectors of r.v. $\mathbf{Y}_1$ and $\mathbf{Y}_2$ encoding the intensities of two connected regions of an ideal image $I^*$. Under the hypothesis that $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are included into the same homogeneous region and using the previously defined notations, one has*

$$P(|\mu_1(\mathbf{Y}_1) - \mu_2(\mathbf{Y}_2)| > \epsilon) \le 2\exp\left(\frac{-2\epsilon^2|\mathbf{Y}_1||\mathbf{Y}_2|}{g^2(|\mathbf{Y}_1| + |\mathbf{Y}_2|)}\right), \tag{7}$$

*where $(|\mathbf{Y}_j|)_{j\in\{1,2\}}$ denotes the size of vector $\mathbf{Y}_j$ (i.e. the cardinal of the associated region $S_j$).*

*Proof.* Let us consider the vector $\mathbf{y} = (x_1,\dots,x_N)$ in $[0,g]^N$. This vector may be considered as an outcome of the r.v. $\mathbf{Y}$. In order to apply the McDiarmid theorem we define the following function:

$$f(\mathbf{y}) = f(x_1,\dots,x_N) = (\mu_1(\mathbf{y_1}) - \mu_2(\mathbf{y_2})), \tag{8}$$

where $N = |\mathbf{Y}_1| + |\mathbf{Y}_2|$, $\mathbf{y_1} = (x_1,\dots,x_{|\mathbf{Y}_1|})$ and $\mathbf{y_2} = (x_{|\mathbf{Y}_1|+1},\dots,x_N)$.

Let us compute the variation of the function. If we make a variation on the intensity of one $x_l$ with $l \le |S_1|$. We have

$$\sup|f(x_1,\dots,x_n) - f(x_1,\dots,x'_l,\dots,x_n)| \le \frac{g}{|S_1|}. \tag{9}$$

This gives us the value of the bounding coefficients $c_l = g/|\mathbf{Y}_1|$ for the $|\mathbf{Y}_1|$ first variables. Similarly, if we make a variation on the intensity of $x_l, l \in \{[\mathbf{Y}_1| + 1,\dots,N\}$, we obtain $c_l = g/|\mathbf{Y}_2|$. We then compute the sum over all the variables:

$$\sum_{l=1}^N c_l^2 = g^2\left(\frac{1}{|\mathbf{Y}_1|} + \frac{1}{|\mathbf{Y}_2|}\right). \tag{10}$$

Moreover, according to our hypothesis, if $\mathbf{Y}_1$ and $\mathbf{Y}_2$ belong to the same homogeneous region of $I^*$, all the pixels of $\mathbf{Y}_1$ and $\mathbf{Y}_2$ have the same expectation. We have thus, $E(f(\mathbf{Y})) = E(\mu_1(\mathbf{Y}_1) - \mu_2(\mathbf{Y}_2)) = 0$ and we obtain the expected result using conjointly Theorem 1 and (10). □

Note that the bounds on the probability provided by Theorem 2 may be equivalently represented by

$$P(|\mu_1(\mathbf{Y}_1) - \mu_2(\mathbf{Y}_2)| > F^{-1}(\delta)) \le \delta$$
$$\text{with } \delta = F(\epsilon) = 2\exp\left(\frac{-2\epsilon^2|\mathbf{Y}_1||\mathbf{Y}_2|}{g^2(|\mathbf{Y}_1| + |\mathbf{Y}_2|)}\right). \tag{11}$$

After some basic calculus we find that, under the assumption that $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are included into the same homogeneous region of $I^*$, we have with a probability at most $\delta$

$$|\mu_1(\mathbf{Y}_1) - \mu_2(\mathbf{Y}_2)| > gQ\sqrt{\frac{|\mathbf{Y}_1| + |\mathbf{Y}_2|}{|\mathbf{Y}_1||\mathbf{Y}_2|}} \tag{12}$$

with $Q = \sqrt{(1/2)\ln(2/\delta)}$.

Below the probability $\delta$, which is supposed to be low, we consider that the event $|\mu_1(\mathbf{Y}_1) - \mu_2(\mathbf{Y}_2)| > F^{-1}(\delta)$ is not probable. In this case, we refuse the initial hypothesis stating that $\mathbf{Y}_1$ and $\mathbf{Y}_2$ belong to the same homogeneous region of $I^*$ and thus do not merge the two regions. Our merging predicate may thus be stated as follows:

$$P(S_1, S_2) = \begin{cases} \text{true} & \text{if } |\mu_1 - \mu_2| \le Qg\sqrt{\dfrac{|S_1| + |S_2|}{|S_1||S_2|}}, \\ \text{false} & \text{otherwise,} \end{cases} \tag{13}$$

where $\mu_1$ and $\mu_2$ denote respectively the values of $\mu_1(\mathbf{Y}_1)$ and $\mu_2(\mathbf{Y}_2)$ for the observation $I$. These two terms represent the mean value of the two regions $S_1$ and $S_2$. The term $g$ denotes the maximum level of $I$ ($g = 255$ for gray-scale images).

Note that our merge criterion is equivalent to

$$\frac{|S_1||S_2|}{|S_1| + |S_2|}(\mu_1 - \mu_2)^2 \le (Qg)^2. \tag{14}$$

The left member of this last equation corresponds to the difference between the squared error of $S_1 \cup S_2$ and the sum of the squared errors of $S_1$ and $S_2$ [28]. Our merge criterion may thus be also interpreted as a bound on the increase of the squared errors of the regions.

Our criterion may be adapted to multichannel images as follows:

$$P(S_1, S_2) = \begin{cases} \text{true} & \text{if } \max\limits_{c \in \{a,b,c\}} \dfrac{|\overline{c_1} - \overline{c_2}|}{g_c} \le Q\sqrt{\dfrac{|S_1| + |S_2|}{|S_1||S_2|}}, \\ \text{false} & \text{otherwise,} \end{cases} \tag{15}$$

where $\overline{c_i}$ represents the mean value of the region $S_i$ for the channel $c$ taken in the set of channels $\{a, b, c\}$ and $g_c$ denotes the maximum value on channel $c$. We take the maximum of the values obtained for each channel as a criterion. Indeed, if the predicate is true, it will be true for all the channels and so the merge hypothesis is accepted. In this paper, we have chosen the YUV space which is the native colour space of video sequences.

Both our method and the one of Nock [4] are based on the McDiarmid inequality. However, Nock models each pixel of the ideal image $I^*$ as a sum of $M$ random variables whereas our method only uses one r.v. per pixel. The approach proposed by Nock consists to fix the probability $\delta$ and to use $M$ in order to vary the merge threshold. To our point of view, the probability $\delta$ below which we refuse the merge hypothesis has a more straightforward interpretation than the variable $M$. The resulting criteria are slightly different, our criterion differs by a factor $1/\sqrt{M}$ from the one first proposed by Nock. Our criterion is also significantly different from the second Nock criterion which uses an estimate of the number of final regions whose cardinal is equal to a given value. However, both our criterion and the final Nock criterion may be related, our one being more strict than the one of Nock [4] for a given probability $\delta$.

Let us note finally that the way we derived our criterion provides an alternative explanation to the eventual over-merging produced both by our algorithm and the one of Nock. Indeed, our basic hypothesis consists to suppose that $\mathbf{Y}_1$ and $\mathbf{Y}_2$ belong to the same homogeneous region of $I^*$. As in a contrario approaches first introduced by [29], we refuse this hypothesis only when we observe an event which has a low probability (according to $\delta$) to occur under this hypothesis. We may thus merge regions corresponding to different homogeneous regions of $I^*$ if our observation does not contradict our hypothesis.

### *2.3. Merging order*

An edge $e$ denotes a couple of adjacent pixels $(p, p')$ in a 4-connectivity scheme. The set of edges of an image is denoted by $A_e$ and the number of edges by $N_e$. The order of merging is built on the edges weights as in [4, 12]. The idea behind this order of merging is to merge first similar regions rather than different ones. The similarity between pixels is measured by computing the distance between two pixel colours as follows:

$$w(p, p', n) = |I(p, n) - I(p', n)|. \tag{16}$$

For colour images, the edge weight becomes

$$w(p, p', n) = \sqrt{\sum_{I \in \{a,b,c\}} (I(p, n) - I(p', n))^2}, \tag{17}$$

where $a$, $b$, $c$ denote the three channels of a particular colour space.

Note that alternative weight may be designed. For example, one may balance the distance along each axis of a color space by some weight (or equivalently scale each axis according to its weight). Numerous colour space with different properties may be chosen in (17). For our algorithm, we consider the YUV colour space which is the native colour space of CIF sequences. The colour space $(L^*a^*b^*)$ provides partitions with a little greater subjective quality but with a higher computational cost.

The edges are sorted in an increasing order of their weights and corresponding couples of pixels are processed in this order for merging. This sorting step only requires two traversals of the image: the first traversal allows to compute the histogram of edge weights. The second traversal stores

```
for $i := 1$ to $N_e$ do
    Read the $i$th edge: $(p_1, p_2)$
    $S_1 = \text{FIND}(p_1)$
    $S_2 = \text{FIND}(p_2)$
    if $P(S_1, S_2) = \text{True}$ then
        $\text{UNION}(S_1, S_2)$
    end if
end for
```

Algorithm 1: Merging regions algorithm.

each edge in an array associated to its weight. The amount of memory required for each array is deduced from the histogram of edge weights. This sorting step is similar to the one usually used within the watershed algorithm [9].

### 2.4. *Merging algorithm*

Our spatial segmentation could be divided in three steps. In the first one, we compute the weights of edges and their histogram. In the second step, we sort edges increasingly according to their weights. In the last step, we merge pixels or regions connected by edges following their order. Algorithm 1 describes more particularly the merging loop.

The term $N_e$ represents the number of edges within the image $I$ in the 4-connectivity. In the merging process, we use the union-find data structure [25]. The union function merges two disjoint regions into one region, and the find function identifies the region to which a certain pixel belongs. Implementation details are given in Section 4.

## 3. TIME CONSISTENCY IMPROVEMENT

In video segmentation, the quality of the spatial segmentation is not the only requirement, time consistency is also a very important one. If, in two successive frames, one region is segmented very differently because of noise, occlusion or deocclusion, results of segmentation would be very difficult to exploit for any application like image enhancement, depth estimation, and motion estimation. Many works, see for example [19], use motion estimation to improve time consistency in video segmentation. However, motion estimation [30] is a real bottleneck for real-time implementation and is even sometimes unreliable. In this paper, we combine an improved change detection mask (CDM) with spatial segmentation in order to improve the temporal consistency of our segmentation.

### 3.1. *Change detection mask*

The CDM is designed using both illumination differences between frames and region segmentation of the previous frame.

We first detect changing pixels using the frame difference. Then, we take benefit of the region segmentation of the previous frame in order to classify the pixels not only at a pixel level but also at a region level.

Given the current frame $I(:, n)$ and the previous one $I(:, n-1)$, the frame difference FD is given by

$$\text{FD}(n, p) = |I(p, n) - I(p, n-1)|. \tag{18}$$

Classically, FD is thresholded in order to distinguish changing pixels from noise. The pixel label is given by

$$L(n, p) = \begin{cases} 0 & \text{if } \text{FD}(n, p) \leq \text{tr}_1, \\ 1 & \text{otherwise,} \end{cases} \tag{19}$$

where $\text{tr}_1$ is a positive constant chosen according to the noise level of the image. This threshold may be set experimentally (Section 5) or estimated according to any measure of the image noise. A pixel $p$, with $L(n, p) = 1$, is considered as a changing pixel. We then use the previous segmentation in order to convert the CDM from the pixel level to a region level which is more reliable [23]. For each region $S_i$ in the previous segmentation, we compute $N_{i,\text{changing}}$:

$$N_{i,\text{changing}} = |\{p \in S_i, L(n, p) = 1\}| \tag{20}$$

which denotes the number of changing pixels of the current image whose $(x, y)$ coordinates belong to $S_i$ in the previous segmentation. We then compute $\tau(S_i) = N_{i,\text{changing}}/|S_i|$ which represents the ratio of changing pixels between the previous and the current image in the region $S_i$. Pixels are then classified using three categories:

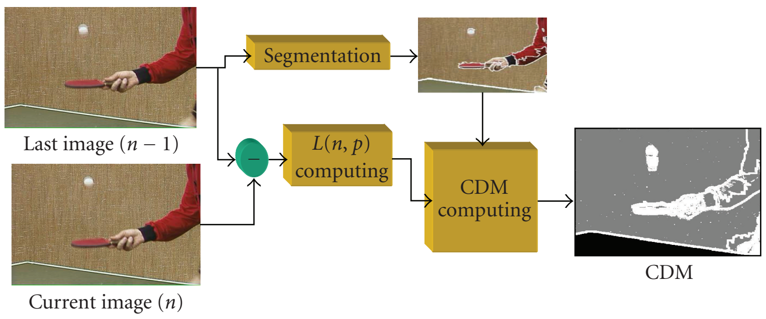
$$\text{CDM}(n, p) = \begin{cases} 0 & \text{if } (\tau(S_i) \leq \text{tr}_2), \\ 1 & \text{if } (\tau(S_i) > \text{tr}_2), (L(n, p) = 0), \\ 2 & \text{if } (\tau(S_i) > \text{tr}_2), (L(n, p) = 1), \end{cases} \tag{21}$$

where $\text{tr}_2$ is a positive constant. In the experiments, we take $\text{tr}_2 = 0.01$ (i.e. a region is a changing region when it contains at least 1% of changing pixels). The value of the threshold is chosen so that we do not miss any changing region.

Every pixel of regions qualified as static is labelled using $\text{CDM}(n, p) = 0$. The two other labels concern pixels within changing regions. Depending on the value of the frame difference, the pixel is qualified as a changing one $(\text{CDM}(n, p) = 2)$ or as a one $(\text{CDM}(n, p) = 1)$. Such a classification is then used to segment the current frame. An example of classification is given in Figure 1 for the video sequence "Table".

### 3.2. *Merging process*

The merging process is now divided in three main steps. Firstly, static regions are kept as they were segmented in the previous frame. Secondly, we apply a connected component labelling (CCL) algorithm [31] to extract connected components of pixels with $\text{CDM}(n, p) = 1$. This second step builds seeds from the segmentation of the previous frame. These seeds link the current segmentation to the previous one in a time-consistent way. Thirdly, we apply the spatial segmentation only on edges $(p, p')$ connecting a changing pixel within a changing region $(\text{CDM}(n, p) = 2)$ to a pixel



■ Static pixel within a static region (CDM = 0)
▨ Static pixel within a changing region (CDM = 1)
□ Changing pixel within a changing region (CDM = 2)

Figure 1: Computation of the CDM using the difference between the current image and the previous one and the region segmentation of the previous frame.

belonging to a changing region. This last pixel may be either changing or static ($\text{CDM}(n,p) \in \{1,2\}$). Note that static pixels within changing regions have been connected in the second step by a CCL algorithm.

The whole process can be formalised as follows. Considering an edge $(p_i, p_i')$ between two pixels, we define the following function:

$$\varphi(n,(p_i,p_i')) = \begin{cases} 0 & \text{if } \text{CDM}(n,p_i)\text{CDM}(n,p_i') = 0, \\ 1 & \text{if } \text{CDM}(n,p_i)\text{CDM}(n,p_i') = 1, \\ 2 & \text{if } \text{CDM}(n,p_i)\text{CDM}(n,p_i') \geq 2. \end{cases} \tag{22}$$

The $\varphi$ function allows us to classify the edges in the following three categories (a brief summary is provided by Figure 2).

(i) The first category ($\varphi(n,a) = 0$) (Figure 2(a)) corresponds to the edges which have at least one pixel belonging to a static region. These edges are not considered for the segmentation of the current image $n$. Static regions are then segmented in the same way between two successive images $n-1$ and $n$.

(ii) The second category ($\varphi(n,a) = 1$) (Figure 2(b)) corresponds to the edges that connect two non changing pixels in changing regions. For these edges, we simply apply a connected component labelling (CCL) algorithm [31].

(iii) The third category ($\varphi(n,a) \geq 2$) (Figure 2(c)) corresponds to the edges which have at least one pixel that is considered as a changing one (i.e. $\text{CDM}(n,p_i) = 2$). These edges are processed using the merging order and the merging predicate defined in Section 2.2. Edges belonging to this category are denoted by $A_u$.

Figure 3 describes the three steps corresponding to the process of the three categories of edges.

In Section 5, we propose the computation of an objective measure for temporal consistency. The measures obtained on real video sequences demonstrate a real improvement

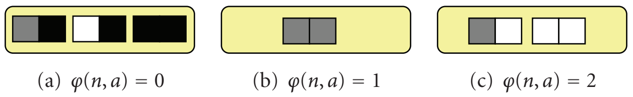


Figure 2: The figure gives the different combinations of pixels available for each category. The pixels are designed as follows : black pixel ($\text{CDM}(n,p) = 0$), gray pixel ($\text{CDM}(n,p) = 1$), white pixel ($\text{CDM}(n,p) = 2$).

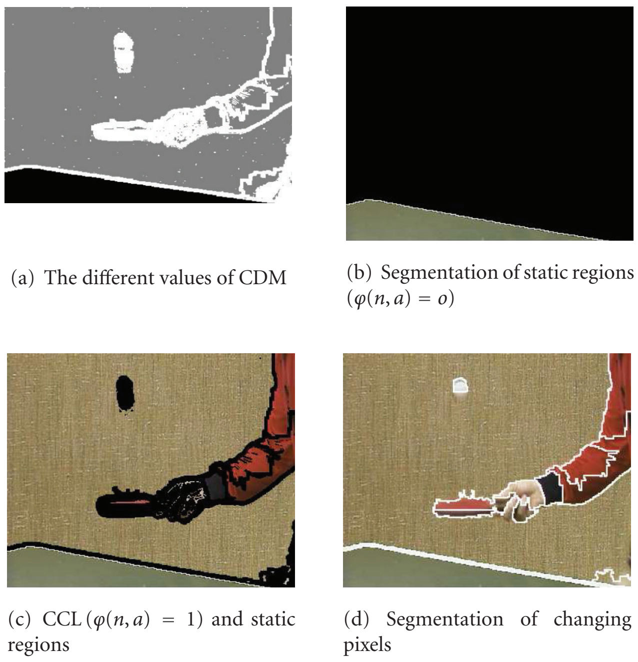

(a) The different values of CDM
(b) Segmentation of static regions ($\varphi(n,a) = o$)
(c) CCL ($\varphi(n,a) = 1$) and static regions
(d) Segmentation of changing pixels

Figure 3: Description of the three steps of the segmentation process for the video "Table". (a) Gives the different values of the CDM. (b), (c), and (d) describe the evolution of the process of the edges $a$ with respectively $\varphi(n,a) = 0, \varphi(n,a) = 1$, and $\varphi(n,a) = 2$. In these three last figures, black pixels are pixels that have not yet been classified, whereas white pixels correspond to region boundaries found at each step.

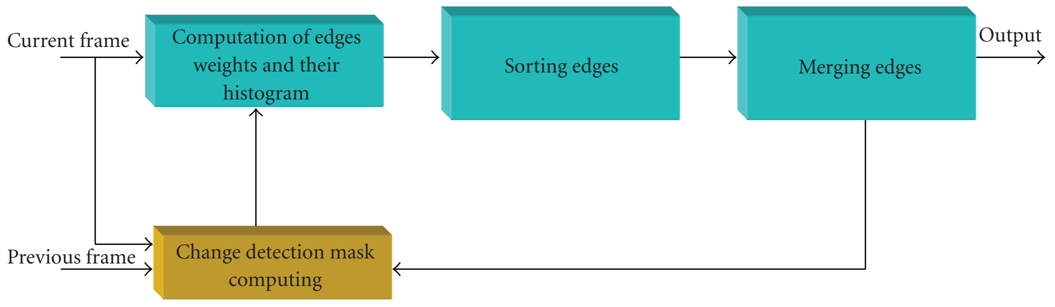

Current frame
Computation of edges weights and their histogram
Sorting edges
Merging edges
Output
Previous frame
Change detection mask computing

Figure 4: The general diagram of video segmentation.

of temporal consistency. Moreover, the way we exploit the CDM decreases also the computational cost of the algorithm since the edges in static area are not reconsidered, and those linking the "no changing pixels" in changing area are simply processed by a CCL algorithm.

When successive images are not correlated (in the case of a scene cut, e.g.), the set $A_u$ contains most of the edges of the image which leads to a new spatial segmentation as shown in the example of a shot cut given in Figure 11. Our algorithm handles, thus, naturally the shot cuts and does not need to be combined with a shot cuts detection algorithm.

## 4. IMPLEMENTATION CONSIDERATIONS

In this section, we propose to describe optimisations that have been made to allow a real-time treatment. The whole algorithm of video segmentation is summarised in Figure 4.

Apart from the merging loop, all other functions access pixels data in a predictable way (e.g., from top to bottom left to right). The cache memory benefits from this regularity, since it exploits spatial and temporal locality of data, and consequently causes less cache misses. In the merging loop, the union-find data structure is unpredictable, and consequently causes an important data cache stalls. To reduce the data cache stalls cycles, we investigate some optimisations that are detailed in the following sections and we take benefit of the TriMedia processor to exploit the data level parallelism (DLP) and instruction level parallelism (ILP) of our algorithm.

### *4.1. Organisation of data*

Our organisation of data should allow an efficient computation of both our merge criterion (13) and our union and find operations. Let us recall that when using an union-find merging scheme each region of the image is encoded by a spanning tree whose vertices are the pixels of the region. These tree data structures are usually encoded by storing for each pixel the index of its parent within the spanning tree. The information about the region are associated to the root of the trees and both the roots and the region information are updated during an union operation.

Since our merge criterion only uses the mean color $(\overline{y}, \overline{u}, \overline{v})$ and the cardinal $|S|$ of the regions, one simple organisation of our data would consist in associating each pixel $p$ with the fields $(\overline{y}, \overline{u}, \overline{v}, |S|, \text{father})$, where father denotes the father of $p$ within the tree.

However, grouping the region data and the father index would require to manipulate the whole vector $(\overline{y}, \overline{u}, \overline{v}, |S|, \text{father})$ within find operations. Since only the father field is required by the find operation such an organisation of the data would induce the storage of useless data within the cache memory.

We thus decided to store into two separate arrays the data required for the merge operations (namely the vector $(\overline{y}, \overline{u}, \overline{v}, |S|)$) and the encoding of the trees. More precisely, our organization of data is as follows:

(1) one array Data which stores for each created region its $(\overline{y}, \overline{u}, \overline{v}, |S|)$ fields;
(2) one array Father which encodes our sequence of union operations;
(3) one array Label of size $|I|$ initialised to a special flag indicating that each pixel is initially its own father.

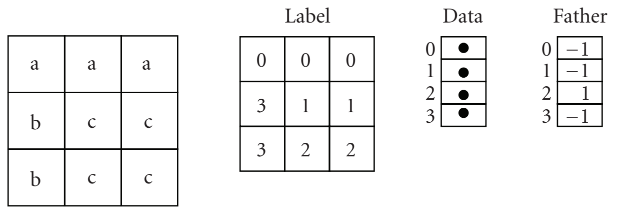
If a region is reduced to a single pixel $p$, Label $(p)$ is set to a special flag and the data of the region retrieved from the image $I$. We thus decide to create a new entry within the array Data only if the associated region is composed of at least 2 pixels. More precisely, if a merge of two pixels $p_1$ and $p_2$ is decided by our merge criterion,

(1) a new entry $l$ is created within the array Data and initialised according to $I(p_1)$ and $I(p_2)$;
(2) label $(p_1)$ and Label $(p_2)$ are set to $l$;
(3) father $(l)$ is set to a special flag indicating that $l$ has yet no father.

Our data structure is further updated in the two following cases.

(1) One pixel $p$ is aggregated to an already created region labelled by $l$. In this case, Label $(p)$ is set to $l$ and Data $(l)$ is updated according to $I(p)$. The array Father remains unchanged.
(2) Two already created regions with respective labels $l_1$ and $l_2$ are merged. In this case, one of the labels (say $l_1$) survives, Data $(l_1)$ is updated according to Data $(l_2)$ and Father $(l_2)$ is set to $l_1$.

Figure 5(b) illustrates the state of our different data structures after the segmentation of Figure 5(a). Two pixels in Figure 5(a) are merged if they have the same label. In this



(a) I (b) Data structures

Figure 5: The data structures used to compute union-find operations and our merge criterion.

example, we first considered horizontal edges between pixels and then vertical ones. Both horizontal and vertical edges have been considered using a scan line order. Note that the array Data is completely filled by the four regions created during the union operations. We only get three final regions as encoded by the array Father where all labels, except label 2, are their own father.

Since all regions encoded by the array Data are composed of at least 2 pixels, the maximal number of entries within this array is equal to $|I|/2$. Moreover, the vertices of the trees encoded by the array Father correspond to regions composed of at least 2 pixels. The maximal size of the array Father is thus also equal to $|I|/2$. Note that this upper bound may be reached if we first decompose the image into regions made of 2 adjacent pixels and then order the merges in such a way that the tree encoding the union of all these elementary regions is linear.

Note that when using such an organisation of data, all the required memory is allocated before union and find operations. We thus avoid the risk of a memory overflow.

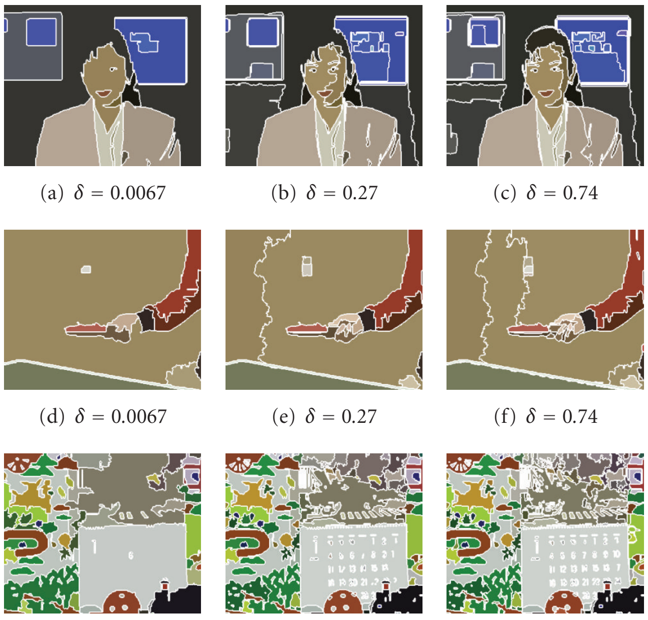
### 4.2. *TriMedia processor*

We experimented this data organisation on the TriMedia processor [32]. The cache memory of this particular TriMedia is 128 KByte, 4 way associative, with block of 128 Byte. The replacement algorithm used is LRU.

In order to increase the computational efficiency, we propose to take benefit of the data level parallelism (DLP) provided by our algorithm (computation of edge's weight, frame difference, classification of pixels in CDM). This allows to increase the throughput (i.e., amount of pixels processed per unit time) by processing data in parallel when it is possible. The core of TriMedia is a VLIW architecture with 5 issues slots. Each slot has some functional unit, and each functional unit could process 4 bytes in parallel (SIMD mode). The instruction level parallelism (ILP) is extracted by the compiler, while the DLP could be exploited through the use of custom operations, loop unrolling, and grafting. So we use these optimisations to exploit the DLP available in our algorithm.

## 5. EXPERIMENTAL RESULTS

In this section, we present experimental results of our algorithm run on TriMedia with many very known CIF video sequences.



(a) $\delta = 0.0067$ (b) $\delta = 0.27$ (c) $\delta = 0.74$
(d) $\delta = 0.0067$ (e) $\delta = 0.27$ (f) $\delta = 0.74$
(g) $\delta = 0.0067$ (h) $\delta = 0.27$ (i) $\delta = 0.74$

Figure 6: Segmentation of one frame of the video sequences "Akiyo", "Table", "Mobile" with $\delta = 0.0067$, $\delta = 0.27$, $\delta = 0.74$.

### 5.1. *Spatial results*

The probability $\delta$ tunes the coarseness of the segmentation. In Figure 6, we show the influence of this parameter on the level of details obtained. This parameter is highly correlated to the number of segmented regions. A value of this parameter around 0.74 provides a sufficient level of details for most of the video sequences we have considered. However, the chosen value and the associated level of details are highly dependent on the application. We can remark that this algorithm is able to segment very precisely small regions of interest such as the mouth or the eyes of "Akiyo". It can also segment the different numbers of the calendar in the sequence "Mobile". However, we can observe an over-segmentation of some textured regions such as the wall in the sequence "Table". This is mainly due to the fact that assumption (1) is more adapted to the segmentation of flat regions. Our ongoing research is directed towards the design of a new merging criterion for the segmentation of textured regions.

As a comparison, we propose here some results obtained with two other well-known algorithms: algorithm EGBIS [12] and the statistical region merging (SRM) algorithm of Nock and Nielsen [4]. These two algorithms are based on region merging schemes with the same merging order than our method. The main difference between the three methods lies in the merging predicate. The results are displayed in Figure 7. For each algorithm, we have tuned the parameters in order to reach a segmentation that allows a good subjective representation of the elements of the image (numbers of the calendar, eyes of the woman, etc.). We can see on these examples that our real-time algorithm gives comparable

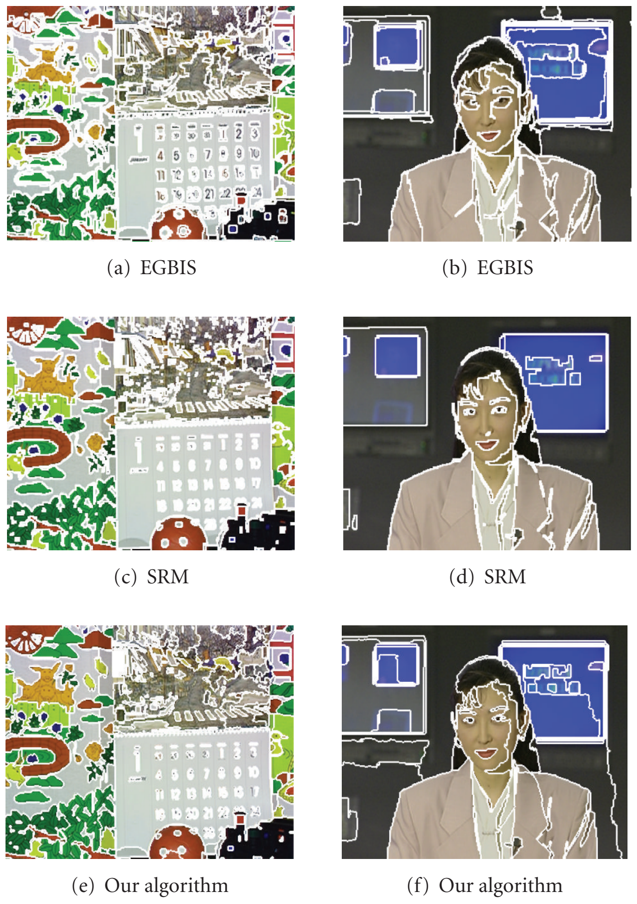

(a) EGBIS (b) EGBIS
(c) SRM (d) SRM
(e) Our algorithm (f) Our algorithm

Figure 7: Comparison of our segmentation results with those obtained using the algorithms EGBIS [12] and SRM [4].

results than the two other algorithms. This last point has been confirmed by other experiments that are not reported here. Our real-time implementation is thus achieved without detriment to the subjective quality of the results.

### 5.2. Spatiotemporal results

In the experiments, we take $\mathrm{tr}_1 = 6$ and $\mathrm{tr}_2 = 0.01$ (i.e., a region is a changing region when it contains at least 1% of changing pixels). The values of these thresholds are the same for all the video sequences.

In order to see the influence of our temporal process, we show here an example of segmentation results with and without time consistency in Figures 8(c) and 8(b). We can see that the segmentation of the wall is the same for the two frames 1 and 9 of the video sequence "Table" when we use the time-consistency improvement.

We then propose to display the segmentation results along the video sequence "Akiyo" in Figure 9 and the video sequence "Paris" in Figure 10. We can observe that the method gives satisfying and stable results for these sequences.

We have also tested the robustness of our method in the case of a shot cut. The video sequence "Football" is followed by the video "BBC Disc". Experimental results are given in

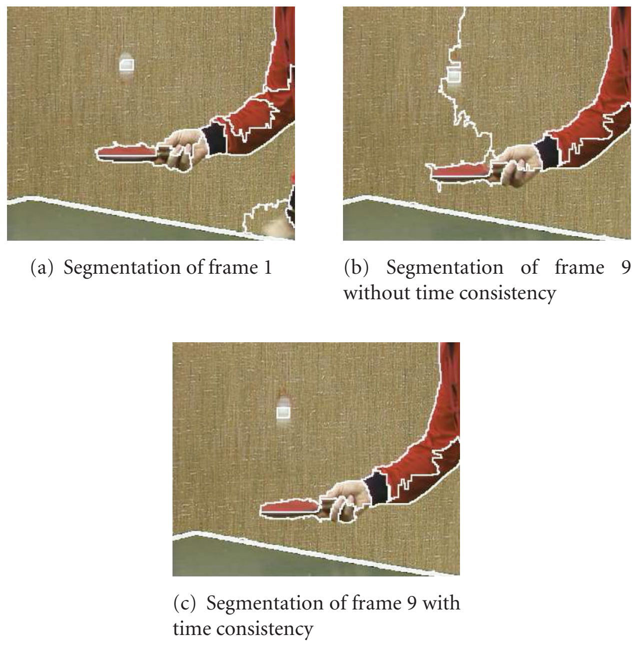

(a) Segmentation of frame 1 (b) Segmentation of frame 9 without time consistency
(c) Segmentation of frame 9 with time consistency

Figure 8: Comparison of the segmentation results obtained with and without time consistency on the video sequence "Table".

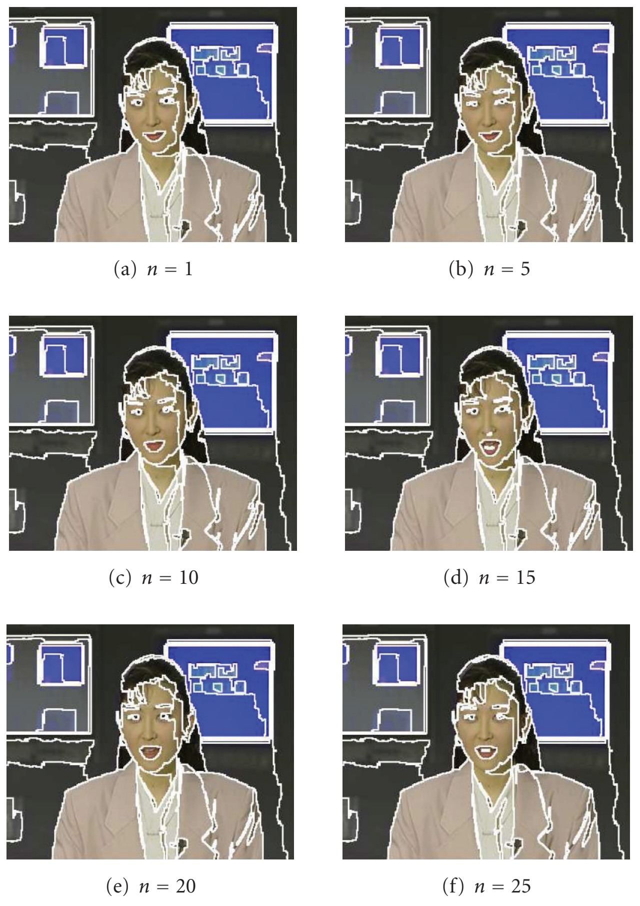

(a) $n = 1$ (b) $n = 5$
(c) $n = 10$ (d) $n = 15$
(e) $n = 20$ (f) $n = 25$

Figure 9: Results for the spatiotemporal segmentation of two video sequences "Akiyo" ($\delta = 0.81$).

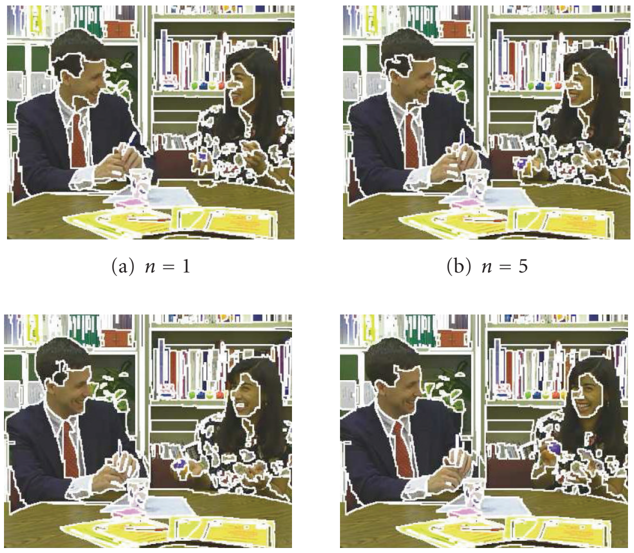

(a) $n = 1$ (b) $n = 5$ (c) $n = 10$ (d) $n = 15$

FIGURE 10: Results for the spatio-temporal segmentation of two video sequences "Paris" ($\delta = 0.81$).

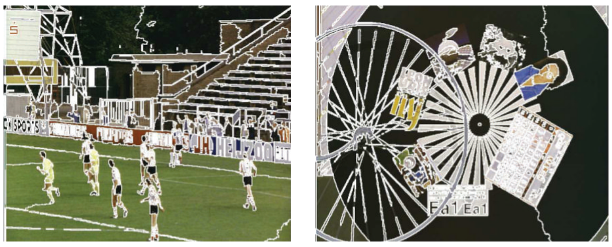
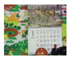

(a) (b)

FIGURE 11: Experimental results in the presence of a video scene cut. (a) Segmentation of the last frame of the video "Football". (b) Segmentation of the first frame of the first image of the video"BBCDisc".

TABLE 1: Experimental measures of time consistency.

| Sequence | Akiyo | Table | Paris | Mobile  |
|---|---|---|---|---|
| Time consistency (SRM) | 0.95 | 0.8 | 0.86 | 0.79 |
| Time consistency (our approach without CDM) | 0.88 | 0.73 | 0.89 | 0.84 |
| Time consistency (our approach with CDM) | 0.98 | 0.92 | 0.97 | 0.92 |

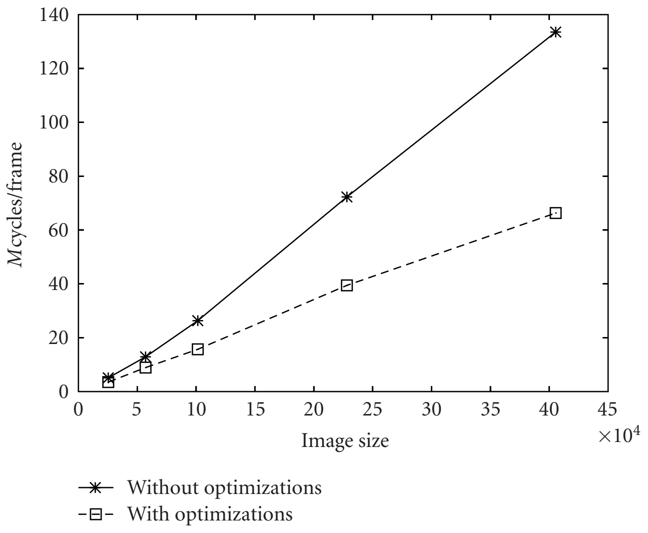


FIGURE 12: Evaluation of the computational cost regarding the image size (with one image of the video Akiyo, $\delta = 0.74$).

Figure 11. We can observe that the spatial segmentation of the first frame of the video "BBCDisc" is not influenced by the spatial segmentation of the previous frame that belongs to the video "Football". Indeed, in this case, most of edges belong to the third category of edges ($\varphi(n,a) = 2$) where the predicate is recomputed.

### 5.3. *Evaluation of time consistency*

We use a classical measure to evaluate time consistency. Given the segmentation of the previous frame SEG($n-1$) and the segmentation of the current one SEG($n$), we find a correspondence between regions in SEG($n-1$) and SEG($n$). For each region $S_{i,n-1} \in \text{SEG}(n-1)$, we choose the region $S_{j,n} \in \text{SEG}(n)$ that produces the most overlapping area

$$\text{Overlap}(i, n-1) = \max_j |S_{i,n-1} \cap S_{j,n}|. \qquad (23)$$

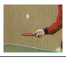
We then sum the overlap measures for all the regions in SEG($n-1$). The consistency measure is the percentage of this number to the size of the image. The results for this measure are given in Table 1 for the video sequences "Akiyo", "Table Tennis", "Paris", and "Mobile". When enforcing consistency through the CDM, time consistency is higher, and visually, segmentation is more stable from frame to frame and still fit very well regions boundaries as shown in Figures 9 and 10. We can also see that the time consistency of the spatial segmentation algorithm SRM [4] is roughly equivalent to the time consistency of our spatial algorithm without computation of the CDM.

### 5.4. *Evaluation of the computational cost*

In this section, we propose to give the number of Mcycles the algorithm takes on TriMedia for different resolutions and different versions of our algorithm. We propose to compare the spatial computational cost with the one obtained using the Nock algorithm [4].

The computational cost has been evaluated as a function of the image size in Figure 12. In this figure, the computational cost (in Mcycles/frame) has been computed for one image of the video "Akiyo" at different resolutions (QCIF,

Table 2: Compuational cost.

| Sequence | Akiyo | Table  | Paris | Mobile |
|---|---|---|---|---|
| *M*cycles/frame (SRM) | 34.57 | 66.53 | 38.03 | 25.41 |
| *M*cycles/frame (without CDM, without optimizations) | 26.33 | 32.21 | 28.18 | 24.73 |
| *M*cycles/frame (without CDM, with optimizations) | 15.68 | 15.84 | 16.59 | 16.2 |
| *M*cycles/frame (with CDM) | 9.87 | 11.37 | 11.02 | 10.06 |

CIF, SD, and two other resolutions). This computation has been performed with and without the optimisations described in Section 4.2. First, the results given in Figure 12 show that the complexity is approximatively linear regarding the image size. Indeed, the spatial computational cost is principally induced by the union-find algorithm and the edges sorting. As explained in Section 2.3, the sorting step is performed in a linear time $0(|I|)$. As far as the union-find algorithm is concerned, the complexity is given by $0(\alpha(n_u, n_f)n_f)$ where $n_u$ is the number of union operations and $n_f$ is the number of find operations ($n_u < n_f$). The function $\alpha$ is a very slowly growing function [25]. Since the number of find operations can be upper-bounded by $c|I|$ where $c$ is a constant, the complexity at worst can be approximated by $0(\alpha(n_u, n_f)|I|)$ which gives an almost linear complexity. This assessment is confirmed by the experimental results given in Figure 12.

We then propose to compare the computational cost of our algorithm to the SRM algorithm [4]. The main difference between the two spatial algorithms lies in the computation of the predicate. The predicate of SRM leads to higher computational cost as demonstrated in Table 2. Our algorithm gives a lower computational cost even without optimisations. When including these improvements, the computational cost decreases. In Table 2, we also give the number of Mcycles the algorithm takes on TriMedia when enforcing the temporal consistency. The exploitation of the CDM reduces the computational cost. This reduction depends on the correlation between two successive frames.

With a 450 MHz TriMedia, we are able to process more than 25 frames per second. We can then conclude that our algorithm is avalaible in real time for QCIF or CIF sequences.

## 6. DISCUSSION

Designing usable algorithms for video processing requires low-computational methods. Directed by this constraint, we propose here an efficient time-consistent algorithm for video segmentation. Let us discuss the strengths and limitations of our algorithm regarding the three main points of this work.

(i) Spatial segmentation: we propose here an alternative statistical modelisation to the work of Nock and Nielsen [4]. This leads to a simpler predicate for merging that is more adapted to a real-time implementation and gives good results for the spatial segmentation. However, as in [4], such a statistical model is dedicated to the segmentation of flat regions and may produce an over-segmentation on textured area of an image.

(ii) Temporal consistency: the proposed algorithm allows to obtain both stable segmentation results and a reduction of the computational cost. This method is based on the use of a CDM and of region information deduced from the first frame. Regions are not linked from one frame to another leading to a video segmentation algorithm that is robust to scene cut and occlusion. However, if never this algorithm has to be exploited for video object tracking, region matching will be useful. It can be obtained by comparing regions of two consecutive frames using statistical inequalities.

(iii) Hardware implementation : our algorithm runs in real time for CIF sequences. For standard definition (SD) or high definition (HD) sequences some further efforts are needed. In order to obtain a real-time implementation, we have directed our attention to the parallelisation by blocks of the spatial segmentation. However, we still investigate this part and notably the merging of the different spatial segmentations obtained for the different blocks. This last step remains delicate.

We finally want to outline that such a real-time video segmentation algorithm would help many video algorithms by leading to a better comprehension of the image content. Among applications, we can think of time conversion, peaking (also named unsharp masking), video compression, or deinterlacing. The region segmentation algorithm can be exploited directly using regions boundaries and region color properties or as a source of information on the image content (level of noise, complexity of the scene, main colors) which can be exploited to better design existing algorithms [33]. Our on-going research is also directed to the design of such region-based algorithms for electronic devices (e.g., : set-top box).

## ACKNOWLEDGMENTS

The authors would like to thank Patrick Meuwissen, O. P. Gangwal, and Zbigniew Chamski for their constructive suggestions. We also would like to thank the reviewers for their very useful comments and suggestions.

## REFERENCES

[1] G. Iannizzotto and L. Vita, "Fast and accurate edge-based segmentation with no contour smoothing in 2-D real images," *IEEE Transactions on Image Processing*, vol. 9, no. 7, pp. 1232–1237, 2000.

[2] Y. Haxhimusa, A. Ion, W. G. Kropatsch, and T. Illetschko, "Evaluating minimum spanning tree based segmentation algorithms," in *Proceedings of the 11th International Conference on Computer Analysis of Images and Patterns (CAIP '05)*, A. Gagalowicz and W. Philips, Eds., vol. 3691 of *Lecture Notes in*

*Computer Science*, pp. 579–586, Versailles, France, September 2005.

[3] L. Brun, M. Mokhtari, and F. Meyer, "Hierarchical watersheds within the combinatorial pyramid framework," in *Proceedings of the 12th International Conference on Discrete Geometry for Computer Imagery (DGCI '05)*, vol. 3429 of *Lecture Notes in Computer Science*, pp. 34–44, Poitiers, France, April 2005.

[4] R. Nock and F. Nielsen, "Statistical region merging," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1452–1458, 2004.

[5] S. Lallich, F. Muhlenbach, and J.-M. Jolion, "A test to control a region growing process within a hierarchical graph," *Pattern Recognition*, vol. 36, no. 10, pp. 2201–2211, 2003.

[6] S. Pateux, "Spatial segmentation of color images according to the MDL formalism," in *Proceedings of the International Conference on Color in Graphics and Image Processing (CGIP '00)*, vol. 2, pp. 89–93, Saint Étienne, France, October 2000.

[7] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[8] E. Sharon, A. Brandt, and R. Basri, "Fast multiscale image segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '00)*, vol. 1, pp. 70–77, Hilton Head Island, SC, USA, June 2000.

[9] L. Vincent and P. Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 583–598, 1991.

[10] M. Couprie, L. Najman, and G. Bertrand, "Quasi-linear algorithms for the topological watershed," *Journal of Mathematical Imaging and Vision*, vol. 22, no. 2-3, pp. 231–249, 2005.

[11] P. Salembier and L. Garrido, "Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval," *IEEE Transactions on Image Processing*, vol. 9, no. 4, pp. 561–576, 2000.

[12] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[13] Y. Deng and B. Manjunath, "Unsupervised segmentation of colour-texture regions in images and video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 8, pp. 800–810, 2001.

[14] C. Fiorio and R. Nock, "Sorted region merging to maximize test reliability," in *Proceedings of the International Conference on Image Processing (ICIP '00)*, vol. 1, pp. 808–811, Vancouver, BC, Canada, September 2000.

[15] H.-Y. Wang and K.-K. Ma, "Automatic video object segmentation via 3D structure tensor," in *Proceedings of the International Conference on Image Processing (ICIP '03)*, vol. 1, pp. 153–156, Barcelona, Spain, September 2003.

[16] D. DeMenthon, "Spatio-temporal segmentation of video by hierarchical mean shift analysis," in *Proceedings of the Statistical Methods in Video Processing Workshop*, Copenhagen, Denmark, June 2002.

[17] L.-Y. Duan, M. Xu, Q. Tian, and C.-S. Xu, "Mean shift based video segment representation and applications to replay detection," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '04)*, vol. 5, pp. 709–712, Montreal, Quebec, Canada, May 2004.

[18] F. Moscheni, S. Bhattacharjee, and M. Kunt, "Spatio-temporal segmentation based on region merging," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 9, pp. 897–915, 1998.

[19] D. Wang, "Unsupervised video segmentation based on watersheds and temporal tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 5, pp. 539–546, 1998.

[20] I. Patras, E. A. Hendriks, and R. L. Lagendijk, "Video segmentation by MAP labeling of watershed segments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 326–332, 2001.

[21] C. McDiarmid, "Concentration," in *Probabilistic Methods for Algorithmic Discrete Mathematics*, pp. 195–248, Springer, New York, NY, USA, 1998.

[22] A. M. Tekalp, "Video segmentation," in *Handbook of Image and Video Processing*, Elsiever, Oxford, UK, 2005.

[23] A. A. Alatan, L. Onural, M. Wollborn, R. Mech, E. Tuncel, and T. Sikora, "Image sequence analysis for emerging interactive multimedia services-the European COST 211 framework," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 7, pp. 802–813, 1998.

[24] A. Caplier, L. Bonnaud, and J. Chassery, "Robust fast extraction of video objects combining frame differences and adaptative reference image," in *Proceedings of the International Conference on Image Processing*, vol. 2, pp. 785–788, Thessaloniki, Greece, October 2001.

[25] C. Fiorio and J. Gustedt, "Two linear time Union-Find strategies for image processing," *Theoretical Computer Science*, vol. 154, no. 2, pp. 165–181, 1996.

[26] G. J. Klinker, S. A. Shafer, and T. Kanade, "A physical approach to color image understanding," in *Color*, G. E. Healey, S. A. Shafer, and L. B. Wolff, Eds., pp. 134–165, Jones And Bartlett, Sudbury, Mass, USA, 1992.

[27] L. Wolf, X. Huang, I. Martin, and D. Metaxas, "Patch-based texture edges and segmentation," in *Proceedings of the 9th European Conference on Computer Vision (ECCV '06)*, vol. 3952 of *Lecture Notes in Computer Science*, pp. 481–493, Graz, Austria, May 2006.

[28] L. Brun and M. Mokhtari, "Two high speed color quantization algorithms," in *Proceedings of Computer Graphics, and Image Processing (CGIP '00)*, pp. 116–121, Cépaduès, Saint Étienne, France, October 2000.

[29] A. Desolneux, L. Moisan, and J. Morel, "Meaningful alignments," *International Journal of Computer Vision*, vol. 40, no. 1, pp. 7–23, 2000.

[30] A. Mitiche and P. Bouthemy, "Computation and analysis of image motion: a synopsis of current problems and methods," *International Journal of Computer Vision*, vol. 19, no. 1, pp. 29–55, 1996.

[31] K. Wu, E. Otoo, and A. Shoshani, "Optimizing connected component labeling algorithms," in *Medical Imaging 2005: Image Processing*, vol. 5747 of *Proceedings of SPIE*, pp. 1965–1976, San Diego, Calif, USA, April 2005.

[32] "pnx1500 databook," http://www.tcshelp.com/public_files.html.

[33] M. El Hassani, M. Duranton, and S. Jehan-Besson, "Dynamic peaking," 2007, submitted patent NXP.

Hindawi Publishing Corporation
VLSI Design
Volume 2008, Article ID 610420, 9 pages
doi:10.1155/2008/610420

*Research Article*

# An FFT Core for DVB-T/DVB-H Receivers

**A. Cortés, I. Vélez, I. Zalbide, A. Irizar, and J. F. Sevillano**

*Department of Electronic and Communication, CEIT and Tecnun, Univerisity of Navarra, 20018 Donostia-San Sebastian, Spain*

Correspondence should be addressed to A. Cortés, acortes@ceit.es

Received 27 April 2007; Revised 15 November 2007; Accepted 23 January 2008

Recommended by Jean-Baptiste Begueret

This paper presents the design and implementation of a 2K/4K/8K multiple mode FFT core for DVB-T/DVB-H receivers. The proposed core is based on a pipeline radix-$2^2$ SDF architecture. The necessary changes in the radix-$2^2$ SDF architecture to achieve an efficient FFT implementation are detailed. Quantization effects and timing design parameters are analyzed for DVB-T/DVB-H. Area and power results are provided for the proposed core.

## 1. INTRODUCTION

DVB-H adapts the successful DVB-T standard for digital terrestrial television to the specific requirements of mobile, handheld, and battery-powered receivers [1–3]. Both standards rely on an orthogonal frequency division multiplexing (OFDM) modulation scheme to achieve high-data rates in multipath environments. OFDM uses an inverse fast Fourier transform (IFFT) to modulate the signal and a fast Fourier transform (FFT) to demodulate it. One of the main differences between both standards is the number of points of the FFT: DVB-H that proposes an additional mode (4K) to the two DVB-T modes (2K and 8K). This new mode is a tradeoff between reception quality in movement and network size.

As DVB-H is an extension to DVB-T, it is possible to introduce DVB-H services in the bandwidth of DVB-T. One operator can offer two DVB-T services and one DVB-H service to its subscribers. Therefore, digital terrestrial television receivers should be able to receive both DVB-T and DVB-H signals. In these receivers, the FFT processor must be able to work in 2K/4K/8K multiple mode. Moreover, this module must have a high throughput in order to achieve the high-data rates required by both standards. Some 2K/4K/8K pipelined FFT architectures have been proposed in the literature [4].

The algorithm and architecture for the FFT core should be chosen trading off its processing speed, area, and power. Monoprocessor architectures such as [5] have to be discarded, as they are not able to fulfill the timing specifications. The throughput can be increased by using either parallel [6, 7] or pipeline architectures [8–14]. Pipeline architectures present smaller latency and lower-power consumption [8, 9], which makes them suitable for mobile devices such as DVB-T/DVB-H receivers.

Basically, two types of pipeline architectures can be distinguished: single-path delay feedback (SDF) architectures [10–12] and multipath delay commutator (MDC) architectures [9]. SDF architectures use registers more efficiently, since the outputs of the butterflies can be stored in shift registers. In MDC architectures, the input sequence is divided into several parallel lines that feed the butterfly applying the appropriate delays to the data. SDF architectures use less memory than MDC. On the other hand, MDC architectures obtain a slightly higher throughput than SDF architectures. The optimal choice depends on the application [9]. In the case of DVB-T/DVB-H receivers, the SDF architecture can achieve the required throughput and needs less area than the MDC architecture.

In addition to the pipeline structure, the radix of the algorithm also influences the complexity of the implementation. A radix-2 algorithm needs more products and gets a lower throughput than a radix-4 algorithm. However, a radix-4 algorithm can only process FFTs with a number of points that is a power of 4, and the butterfly is more complex.

In order to maintain the simplicity of the radix-2 butterfly, radix-$2^2$ ($r2^2$) algorithms have been proposed [11, 14]. The $r2^2$ algorithm is well-suited for DVB-T/DVB-H applications since it can work both with a number of points that is a power of 4 and with a number of points that is a power of 2.

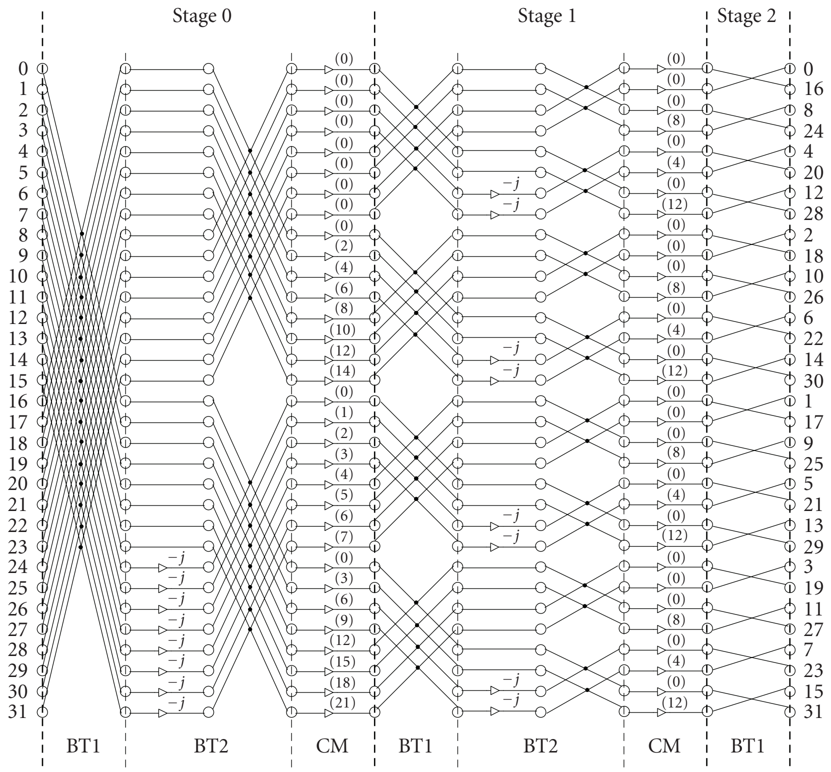


Figure 1: Flow graph of the r2$^2$ SDF 32-point FFT algorithm.

However, the implementation of the r2$^2$ FFT in [11] should be modified to achieve a multiple mode operation.

This paper presents a new 2K/4K/8K FFT core for DVB-T/DVB-H receivers. The r2$^2$-SDF pipeline architecture in [11] is adapted to achieve a multiple mode 2K/4K/8K FFT. The effect of the quantization errors of the FFT processor is studied, and design parameters are analyzed according to the DVB-T/DVB-H requirements.

This paper is organized as follows. Section 2 explains the modifications done to the r2$^2$ SDF algorithm so that it can operate in multiple mode. Section 3 describes the pipeline r2$^2$-SDF architecture proposed for a DVB-T/DVB-H receiver. Section 4 gives signal-to-noise-ratio (SNR), area, timing, and power results of the proposed FFT core. Finally, Section 5 summarizes the conclusions of this work.

## 2. r2$^2$ -SDF ALGORITHM

In this section, the r2$^2$-SDF algorithm presented in [11] will be modified by us to enable it to work in a multiple mode operation.

Figures 1 to 3 show the flow graph of the radix-2$^2$ algorithm when the length of the FFT is 32, 16, and 8, respectively. In these flow graphs, the index of the input and output data is shown. The black circles represent butterflies, and the arrows are the twiddle factor multiplications, where the number in brackets next to each arrow is the index $m$ of the twiddle factor, $W_N^{(m)} = e^{j2\pi m/N}$, of an FFT of length $N$.

Three different operations can be distinguished in each stage: BT1, which is the butterfly of type 1 defined in [11]; BT2, which is the butterfly of type 2 defined in [11]; and CM, which performs complex multiplications. We can observe that, for BT2, some input data of the butterflies are multiplied by $j$. The multiplication by $j$ can be implemented by exchanging the real and imaginary parts.

The twiddle factors of the CM at the $k$th stage, with $k = 0, 1, \ldots, \text{ceil}(\log_4 N) - 2$, are given by the vector $A_k = \{a_i\},\ i = 1, 2, \ldots, N/2^{2k}$, where $a_i = e^{-j2\pi p/N}$. The index $p$ is defined as

$$p = \begin{cases} 0, & 0 \le i < m, \\ 2 \cdot 2^{2k} \cdot (i - m), & m \le i < 2m, \\ 2^{2k} \cdot (i - 2m), & 2m \le i < 3m, \\ 3 \cdot 2^{2k} \cdot (i - 3m), & 3m \le i < 4m, \end{cases} \tag{1}$$

where $m$ is equal to $N/2^{2+2k}$.

The r2$^2$ algorithms reduce considerably the number of arithmetical operations performed by the FFT. The only restriction to process the r2$^2$-FFT algorithm is that the length of the FFT is a power of 2. Thus, the last stage of the algorithm is different according to the size of the FFT. If the number of points of the FFT is a power of 4, the last stage is composed of BT1 and BT2, as it can be seen in Figure 2. However, if the number of points of the FFT is only a power of 2, the last stage will be formed by BT1, as can be observed in Figures 1 and 3.

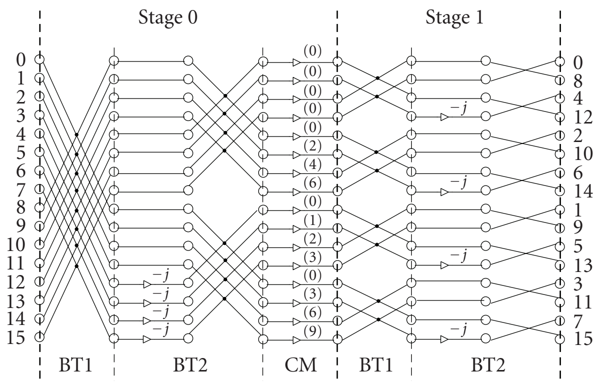


Figure 2: Flow graph of the r2$^2$ SDF 16-point FFT algorithm.

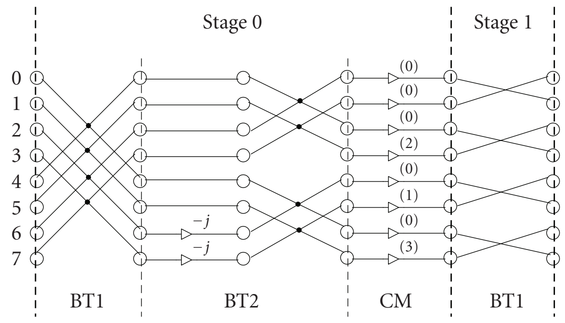


Figure 3: Flow graph of the r2$^2$ SDF 8-point FFT algorithm.

### 2.1. *Multiple mode operation*

In this section, the $2^{2(a-1)-1}/2^{2(a-1)}/2^{2a-1}$ multiple mode of the r2$^2$-SDF algorithm will be derived where $a$ represents the number of stages of an FFT of $N_{\max} = 2^{2a-1}$ points. The resources needed to process the FFT of the largest number of points, $N_{\max}$, will be implemented.

An FFT of $2^{2(a-1)-1}$ points can be easily obtained from a $2^{2a-1}$ points FFT. The first stage of the $N_{\max}$ points FFT does not need to be processed in order to calculate the $2^{2(a-1)-1}$ points FFT. Additionally, in the following stages of the $N_{\max}$ points FFT, the twiddle factors are the same as the ones needed for the $2^{2(a-1)-1}$ points FFT, as $W^{(4i)}_{N_{\max}} = W^{(i)}_{N_{\max}/4}$. This multiple mode implementation can be deduced by analyzing the flow graphs of a 32-points FFT and an 8-points FFT in Figures 1 and 2, respectively, where we have considered $a = 3$.

Using the resources of an FFT of length $N_{\max}$, a $2^{2(a-1)}$ points FFT can be obtained. In the latter FFT, $a-1$ stages are needed. The first $a-1$ stages of the $N_{\max}$ points FFT can be reused to process the $2^{2(a-1)}$ points FFT, if only the operations in the even positions are carried out. Thus, half the operations are done in each BT1, BT2, and CM. Moreover, the twiddle factors of the CM in stage $a-1$ of the $2^{2(a-1)}$ points FFT are always one. Thus, the operations of the last CM can be omitted, and the final stage of the $2^{2(a-1)}$ FFT will only contain a BT1 and a BT2. This multiple mode implementation can be easily concluded by inspection of Figures 1 and 2, taking into account that $W^{(2i)}_{N_{\max}} = W^{(i)}_{N_{\max}/2}$.

## 3. PIPELINE r2$^2$ -SDF ARCHITECTURE IN MULTIPLE MODE

The proposed FFT core receives the input data DATA_IN in natural order, and it generates the output DATA_OUT in bit-reversed order. This is not a problem as the reordering can be performed by subsequent modules of the DVB-T/DVB-H receiver (e.g., deinterleaver) with no additional cost. Input data arrives at clock rate. All the data needed to compute each FFT arrives as a block. In order to ease the integration of the FFT core within a DVB-T/DVB-H receiver, a validation signal, DATA_IN_VALID, will be set high during the arrival of valid data at the input of the FFT core. Similarly, when valid output data are ready at the output of the FFT core, a validation signal DATA_OUT_VALID is set high.

The FFT processor employs fixed-point arithmetic. Input and output data are represented using *dbw* bits. The twiddle factors have been quantized with *tbw* bits. The core scales data appropriately during internal operations to avoid overflow.

In the following, the basic building blocks of the r2$^2$-SDF architecture [11] are described and their implementation detailed. Then, the required modifications to achieve a multiple mode 2K/4K/8K FFT are explained. The section finishes with a summary of the main features of the proposed multiple mode FFT core.

### 3.1. *Basic building blocks*

The FFT processor is a distributed system where every module generates the control signals for the next module in the pipe, as shown in Figure 4. Shadowed lines represent data, and white lines control signals. The FFT core has ceil ($\log_4 N$) stages, where $N$ is the number of FFT points. A typical stage of the architecture consists of three processing elements: B1, B2, and CM; and three memory elements: ROM, FIFO1, and FIFO2. B1 and B2 carry out the processing of the two types of butterflies of the r2$^2$ algorithm (BT1 and BT2). FIFO1 and FIFO2 are used to achieve the required data shuffling for proper operation of the butterflies. In stage $k$, the depth of FIFO1 is $N/2^{k+1}$, and the depth of FIFO2 is $N/2^{k+2}$. CM computes the complex multiplications between the outputs of B2 and the twiddle factors stored in the corresponding ROM memory. The last stage of the FFT processor does not need the complex twiddle factor multiplication.

As explained before, the r2$^2$-SDF architecture can work with a number of points that is a power of 4 and with a number of points that is only a power of 2. When $N$ is just a power of 2, in the last stage, data are only processed by B1.

#### 3.1.1. *Module B1*

Figure 5 shows the structure of B1. The DATA_IN input port comes from the previous component in the pipe, normally a CM module. The DATA_OUT output port is connected to the next component in the pipe, usually a B2 module. The

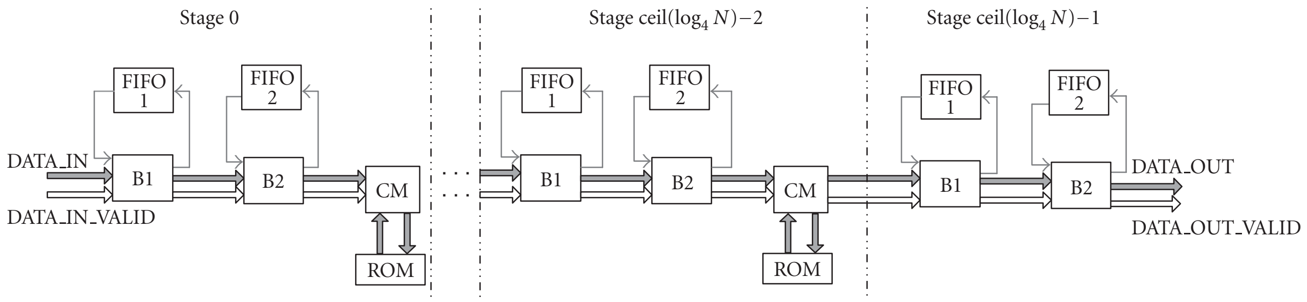


FIGURE 4: Typical pipeline r$2^2$-SDF architecture.

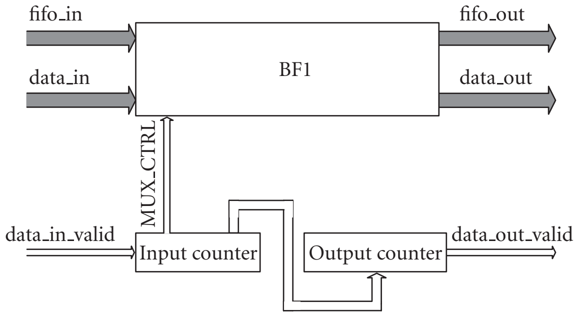


FIGURE 5: B1 module structure.

FIFO_IN and FIFO_OUT ports connect B1 with FIFO1. The size of the two counters of B1 is $2(\log_4 N - k)$, where $k$ is the stage. The implementation of BF1, the butterfly of type 1, is detailed in Figure 6.

Initially, the multiplexers MUX are in position 0, and FIFO1 is empty. During the arrival of the first $N/2^{k+1}$ data, FIFO1 is filled. Then, the multiplexers change to position 1, and the butterfly operations can be performed using the input data at port DATA_IN and the data stored in FIFO1. One of the butterfly outputs, X1, is output, whereas the other one, X2, is stored in FIFO1. After other $N/2^{k+1}$ cycles, multiplexers switch back to position 0. Data for the next computation is stored in FIFO1, and the results X2 of the previous butterfly operations are sent out.

The selection signal of the multiplexers, MUX_CTRL, is generated by the input counter. This counter increments its value when DATA_IN_VALID is high. When the input counter arrives to half of its count, DATA_OUT_VALID is set high, and the output counter is started. The output counter will count until FIFO1 is emptied of valid output data. When the output counter finishes its count, DATA_OUT_VALID is set to zero.

### 3.1.2. Module B2

Figure 7 shows an internal diagram of component B2, which is formed by a butterfly of type 2, BF2, and some control logic. The DATA_IN input port of B2 comes from the previous component in the pipe, a B1 module. The DATA_OUT output port is connected to the next component in the pipe,

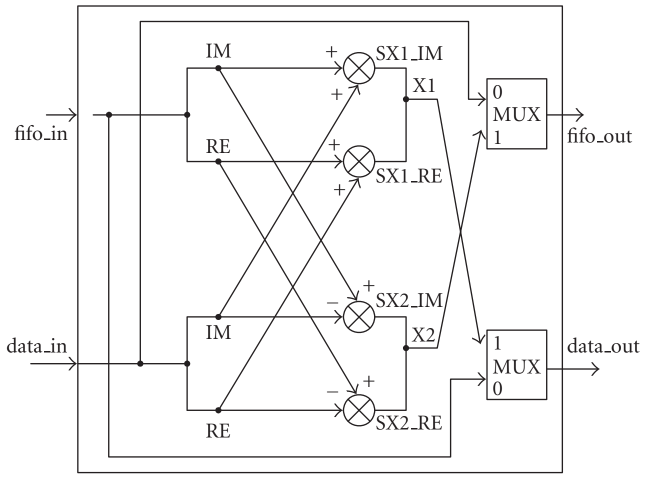


FIGURE 6: Arithmetic operations in the butterfly of type 1 (BF1).

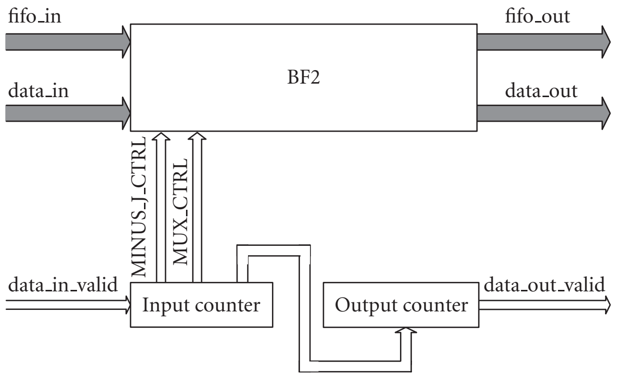


FIGURE 7: B2 module structure.

usually a CM module. The FIFO_IN and FIFO_OUT ports are connected to FIFO2.

Figure 8 details the implementation of BF2. Its structure is similar to BF1. FIFO2 is filled with the first $N/2^{k+2}$ data. Then, the multiplexers MUX change to position 1, and the input data at port DATA_IN and the data stored in FIFO2 are used to perform the required butterfly operations. The butterfly output X1 is sent out, and X2 is stored in FIFO2. After $N/2^{k+2}$ cycles, the multiplexers MUX switch back to position

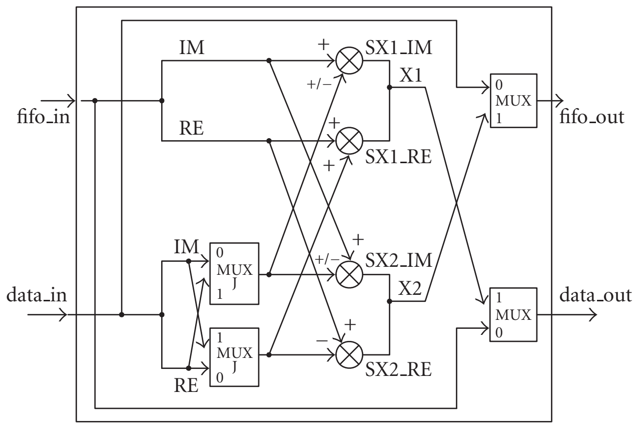


FIGURE 8: Arithmetic operations in the butterfly of type 2 (BF2).

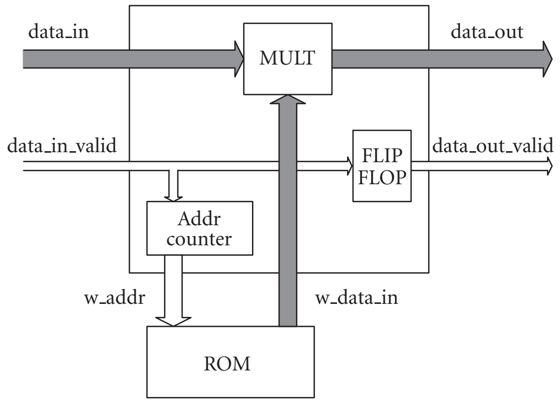


FIGURE 9: CM module structure.

0. Data for the next computation is stored in FIFO2, and the results X2 of the previous butterfly operations are output.

The multiplexers MUX J are used to handle efficiently the multiplications by $-j$ needed in a butterfly of type 2. Whenever a multiplication by $-j$ must be carried out, the multiplexers MUX J are set to position 1. The signal MINUS_J_CTRL controls the behavior of the multiplexers MUX J.

The input counter generates the signals that control the multiplexers MUX and MUX J. This counter increments its value when DATA_IN_VALID is high. Its size is $2(\log_4 N - k)$. The second most significant bit of this counter's value is used to generate MUX_CTRL. When the input counter is making the last quarter of its count, MINUS_J_CTRL is set to high.

When the input counter arrives to a quarter of its count, DATA_OUT_VALID is set to high, and the output counter starts to count. The output counter will count until FIFO2 is emptied of valid output data. When the output counter finishes its count, DATA_OUT_VALID is set to zero. The size of the output counter is $2(\log_4 N - k) - 1$.

### *3.1.3. Module CM*

The internal structure of CM is shown in Figure 9. This component carries out the twiddle factor multiplications. A one clock cycle complex multiplier has been implemented to perform the complex multiplications between the input data and the twiddle factors. The twiddle factors are read from a synchronous ROM. A counter of size $2(\log_4 N - k)$, addr counter, is used to generate the ROM addresses appropriately. A flip-flop is used to synchronize the DATA_OUT_VALID signal with DATA_OUT.

## *3.2. Multiple mode operation*

In order to accommodate the 2K/4K/8K multiple mode, some extra elements are needed in the r$2^2$ SDF architecture. The proposed architecture is depicted in Figure 10. As can be seen, the resources needed to process the FFT of the largest number of points, $N_{\max} = 8192$, have been implemented. Thus, there are $a = 7$ stages, and the twiddle factors have been calculated for $N_{\max}$.

When an 8K point FFT is to be calculated, the multiplexers shown in Figure 10 are configured so that the core works as described above. Multiplexers M4K and M2K are in position 0. When multiplexers are in position 0, they select the signal connected to the upper port.

In order to calculate a 2K point FFT, the first stage of the 8K FFT, stage 0 in Figure 10, does not have to be processed. Thus, multiplexers M2K are set to position 1 to bypass stage 0. Multiplexers M4K remain in position 0.

For the 4K points FFT, six complete stages (with both types of butterflies) are needed. In order to reuse the existing hardware, B1 of stage $k$ uses FIFO2 of stage $k$, and B2 of stage $k$ uses FIFO1 of stage $k+1$. Additionally, CM of stages 5 and 6 is bypassed. The former is achieved by setting M4K to position 1 and M2K to position 0. In each stage, CM needs half the twiddle factors of the 8K points FFT: those with an even address in the ROM memory. A control signal configures CM for proper operation according to the number of points of the FFT to be calculated.

TABLE 1: Memory requirements.

| FIFO | ROM |
|---|---|
| $2dbw \cdot (N_{\max} - 1)$ | $2dbw \cdot \sum_{k=0}^{\mathrm{ceil}(\log_4 N_{\max})-2} \frac{N_{\max}}{2^{2k}}$ |

## *3.3. Features of the proposed FFT core*

Table 1 summarizes the memory requirements of the core. The table shows the total number of memory bits used in the FIFOs and in the twiddle factor ROMs. In order to achieve more compact memories, the real and imaginary parts of each complex number are stored in the higher and lower part of the same memory position. Table 2 is a summary of the arithmetic operators needed in the core. It can be noted that the memory and arithmetic modules needed in the proposed multiple mode architecture are the same as those needed in a single mode 8K FFT.

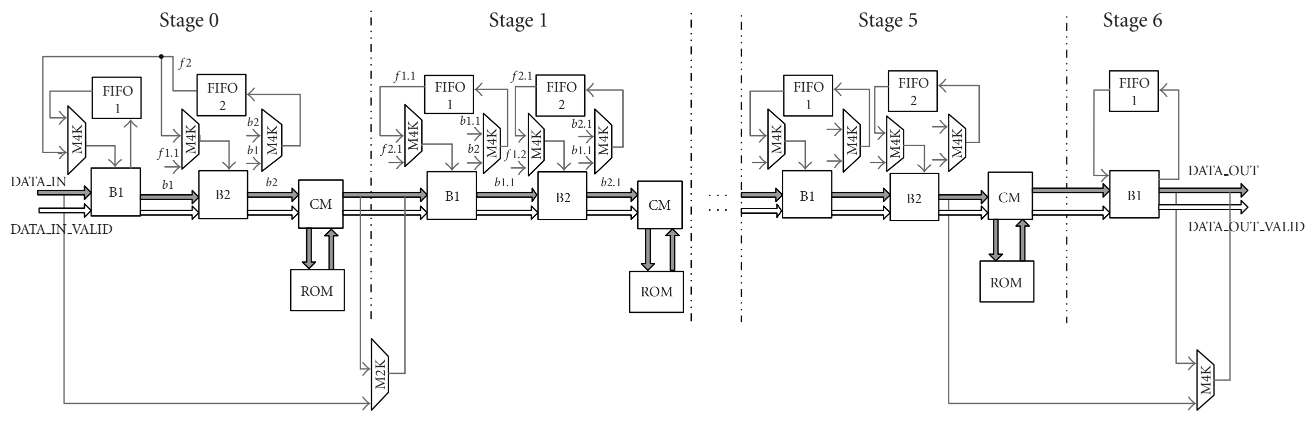


Figure 10: Architecture of the proposed 2K/4K/8K FFT.

Table 2: Arithmetic operators.

| | Input bitwidth | Output bitwidth | Quantity per module | Quantity in the core |
|---|---|---|---|---|
| Adders (B1) | *dbw* | *dbw* | 2 | $2 \cdot \text{ceil}(\log_4 N_{\max})$ |
| Subtractors (B1) | *dbw* | *dbw* | 2 | $2 \cdot \text{ceil}(\log_4 N_{\max})$ |
| Adders (B2) | *dbw* | *dbw* | 2 | $2 \cdot \text{floor}(\log_4 N_{\max})$ |
| Subtractors (B2) | *dbw* | *dbw* | 2 | $2 \cdot \text{floor}(\log_4 N_{\max})$ |
| Multipliers (CM) | *dbw* and *tbw* | *dbw+tbw* | 4 | $4 \cdot \text{ceil}(\log_4 N_{\max} - 1)$ |
| Adders (CM) | *dbw+tbw* | *dbw* | 1 | $\text{ceil}(\log_4 N_{\max}) - 1$ |
| Subtractors (CM) | *dbw+tbw* | *dbw* | 1 | $\text{ceil}(\log_4 N_{\max}) - 1$ |

Once the clock frequency $f_{\text{clk}}$ has been selected, the processing time $t_{\text{proc}}$ of the FFT module can be determined using

$$t_{\text{proc}} = \frac{1}{f_{\text{clk}}} \cdot \left(\frac{N}{2} + 3 \cdot \text{ceil}(\log_4 N) - 2\right). \qquad (2)$$

## 4. RESULTS

### 4.1. Analysis of the signal-to-noise ratio of the FFT

For an appropriate operation of the receiver, the degradation introduced in the signal due to the fixed-point computation of the FFT must be controlled. Monte Carlo simulations have been carried out comparing a fixed-point model of the proposed architecture with a floating-point FFT. The signal-to-noise-ratio (SNR) has been used to measure the degradation.

Figure 11 shows the SNR for the 2K/4K/8K FFT when the data are quantized, and the twiddle factors are left at floating point. It can be observed that both the data bitwidth and the number of points of the FFT influence the SNR. As $N$ increases, the number of arithmetical operations grows and, thus, the SNR decreases. The SNR increases in 6 dB per bit of *dbw*.

The effect of the quantization of the twiddle factors in the SNR has been studied for 2K (Figure 12(a)), 4K (Figure 12(b)), and 8K (Figure 12(c)) FFTs. These figures show the SNR for different values of *dbw* and *tbw*. It can be seen that for a given value of *dbw* and of $N$, increasing *tbw* above a certain value does not improve the performance.

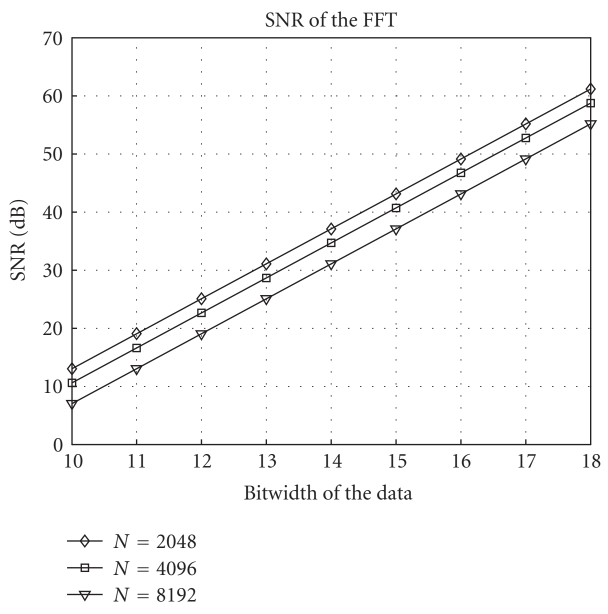
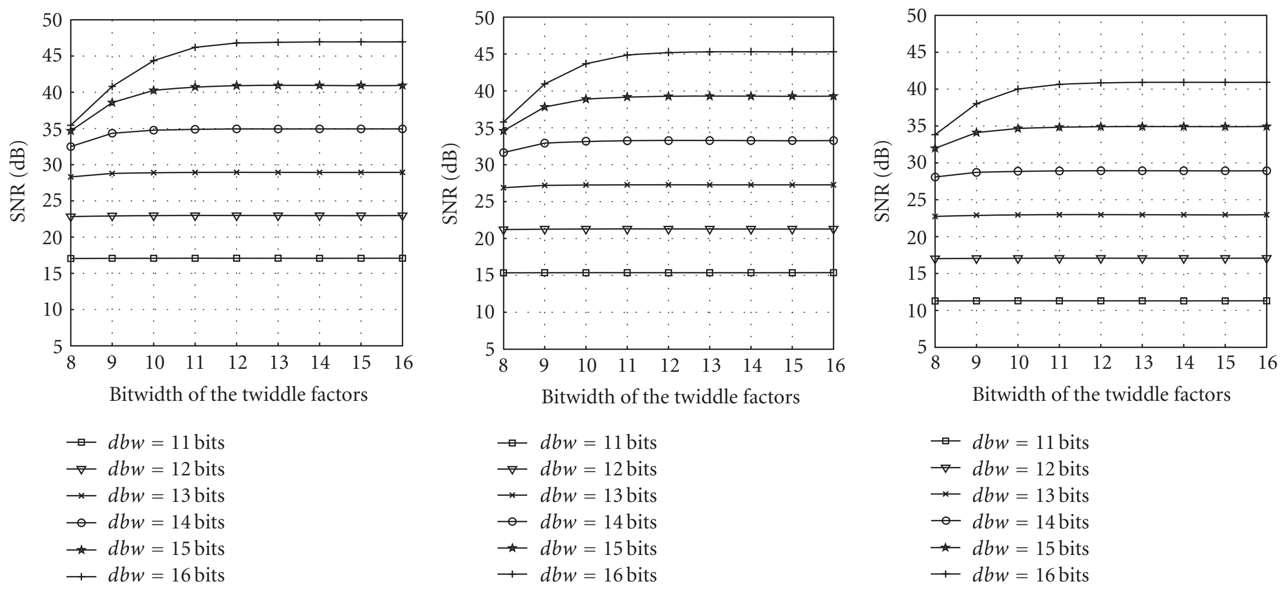


Figure 11: SNR for different data bitwidths (*dbw*) and number of points of the FFT ($N$). The twiddle factors are not quantized.

Figures 11 and 12 can help the designer in the selection of *dbw* and *tbw*. In a multiple mode FFT processor, *dbw* and *tbw* must be selected for the maximum number of points of the FFT (8K in a DVB-T/DVB-H receiver). A SNR of at least

Table 3: DVB-T/DVB-H timing specifications and processing time of the FFT core.

| | 6 MHz ($f_s$ = 48/7 MHZ) | | | 7 MHz ($f_s$ = 8 MHZ) | | | 8 MHz ($f_s$ = 64/7 MHZ) | | |
|---|---|---|---|---|---|---|---|---|---|
| | 2K | 4K | 8K | 2K | 4K | 8K | 2K | 4K | 8K |
| $t_{\text{OFDM-SYMBOL}}$ ($\mu s$) | 308 | 616 | 1232 | 264 | 528 | 1056 | 231 | 462 | 924 |
| $t_{\text{proc}}$ ($\mu s$) | 151.6 | 301 | 600.1 | 148.6 | 294.8 | 587.8 | 113.7 | 225.7 | 450 |



(a) SNR of the 2K FFT (b) SNR of the 4K FFT (c) SNR of the 8K FFT

Figure 12: SNR for different values of data bitwidth (*dbw*) and twiddle factor bitwidth (*tbw*) for (a) $N = 2\text{K}$, (b) $N = 4\text{K}$, and (c) $N = 8\text{K}$.

40 dB is sufficient for terrestrial TV broadcasting [15]. In order to guarantee a SNR of 40 dB for the 8K FFT, $dbw = 16$ and $tbw = 11$ are needed.

## 4.2. Analysis of the timing of the FFT for DVB-T/DVB-H

For low-power applications, such as a DVB-T/DVB-H receiver, a slow-clock frequency is preferable: for example, equal to the sample rate of the FFT. For DVB-T/DVB-H, the minimum sample rate at the FFT ($f_s$) can be 48/7 MHz for a 6 MHz channel, 8 MHz for a 7 MHz channel, and 64/7 MHz for an 8 MHz channel [2]. However, the FFT processor shall be able to compute the FFT within the duration of an OFDM symbol plus the duration of the guard interval ($t_{\text{OFDM-SYMBOL}}$).

Table 3 presents the maximum allowed time to compute the FFT in DVB-T/DVB-H and the processing time of the proposed FFT core. Results are given for the different bandwidth channels (6 MHz, 7 MHz, and 8 MHz) and for the three lengths of the FFT (2K, 4K, and 8K). The value of $t_{\text{OFDM-SYMBOL}}$ requirement given in the table considers the worse case scenario: a guard interval with duration of 1/32 of the OFDM symbol period. The processing time of the FFT core, $t_{\text{proc}}$, has been calculated for a clock frequency equal to the corresponding sampling frequency $f_s$. It can be observed that the proposed FFT core is able to meet the timing requirements in all cases.

## 4.3. Area and timing comparison

Table 4 compares the proposed core with reported FFTs that could be used within DVB-T/DVB-H applications. The FFTs presented in [6, 10] have been designed for DVB-T, and they do not implement the 4K mode. The work in [7] only provides results for 8K.

In [4], a 2K/4K/8K FFT architecture is proposed. The twiddle factor multiplication is carried out using a CORDIC, and no ROM is needed for the twiddle factors. The CORDIC carries out 17 iterations to guarantee good performance. For comparison, we will relate the number of iterations to the precision of the twiddle factors. Following [16], we can estimate the precision in the rotated angle as $\delta_\theta \approx 2^{-(N_i-1)}$, where $N_i$ represents the number of iterations. A quantization error in the twiddle factors can be seen as an error in rotating an angle. It can be shown that max $(\delta_\theta) \approx 2^{-tbw}/(1-2^{-tbw})$. Thus, we can perform the following approximation:

$$tbw \approx \log_2\left(\left(1+2^{-(N_i-1)}\right)/2^{-(N_i-1)}\right) \approx N_i - 1. \quad (3)$$

Table 4: Comparison with other FFTs in the literature.

| | *dbw* | *tbw* | $f_{\text{clk}}$ (MHz) | Area (mm$^2$) | $t_{\text{proc}}$ ($\mu$s) | AT ($\mu$s $\cdot$ mm$^2$) | *N* |
|---|---|---|---|---|---|---|---|
| [4] | 16 | 16$^{(*)}$ | 30 | 66 | 273 | 18018 | 2/4/8K |
| [6] | 8 | 8 | 64 | 28.39 | 897 | 25465 | 2/8K |
| [7] | 11 | 11 | 20 | 18.29 | 717.35 | 13120.33 | 8K |
| [10] | 8 | 8 | 16 | 33.75 | — | — | 2/8K |
| Ours | 16 | 11 | 9.143 | 18.7 | 450 | 8415 | 2/4/8K |

$^{(*)}$This value is an approximation.

Table 5: Hardware complexity of FFT cores for DVB-T/H.

| Architecture | Radix | Multipliers | Adds/Subs |
|---|---|---|---|
| Parallel [6] | 2 | 64 | 96 |
| Pipeline-SDF | 2 | 48 | 76 |
| (Ours) | $2^2$ | 24 | 64 |

The area and timing results shown in Table 4 for the proposed multiple mode FFT core are given for the selected *dbw* and *tbw*. They have been obtained using the 0.35 $\mu$m XFAB 4-ML technology. The area value given for the proposed FFT processor is an estimation of the core area after layout, making the assumption that the layout area is twice the cell area. For a fairer comparison, the area of [6, 7] has been normalized to 0.35 $\mu$m using the same approach as [7]. In [4], the number of equivalent gates is provided. The value given in Table 4 has been estimated for a 0.35 $\mu$m technology using that number.

Table 4 shows the parameter AT as well. AT is the product between the area and the processing time $t_{\text{proc}}$. This parameter can be used to assess the efficiency of different cores. The table shows that the proposed core is the most efficient.

In addition, Table 5 presents a comparison of the computational complexity of [6], a pipeline-SDF r2 FFT design, and our proposal. As can be observed, our design requires less multipliers and adders. Thus, our FFT core presents a more efficient implementation.

### 4.4. *Area and timing results for FPGA implementation*

The FFT core has been prototyped in an FPGA virtex 2V6000FF1517. Table 6 presents a comparison of our core with other FFT cores for DVB-T in the literature. Only those proposals that give data about an FPGA implementation have been considered in the comparison. The table shows the working clock frequency, the total number of occupied slices, the necessary block RAMs, and the number of multipliers. One can observe that our FFT core presents the most efficient implementation for an FPGA.

### 4.5. *Area, timing, and power results for ASIC implementation*

The layout of the FFT core, with *dbw* = 16 and *tbw* = 11, has been carried out for the 0.35 $\mu$m AMS 3-ML technology. The FIFOs of stages 0 to 3 have been implemented using single port RAMs and some additional control logic, whereas the FIFOS of stages 4 to 6 have been implemented using standard cells. A detailed summary of the features of the proposed FFT core is presented in Table 7. The power consumption of the proposed FFT processor has been calculated at synthesis level. The switching activity has been extracted from simulations operating in the 8K mode. A chip photo of the layout of the FFT core for 0.35 $\mu$m AMS 3-ML technology is shown in Figure 13.

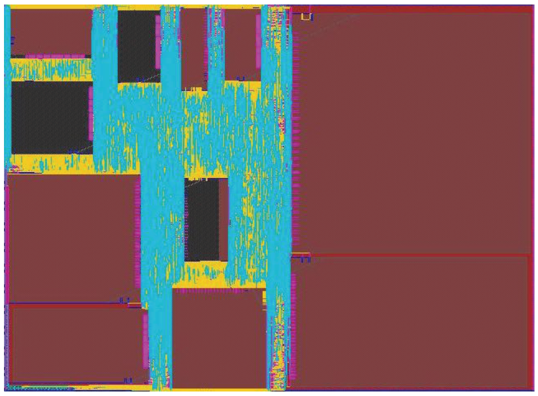


Figure 13: Layout of the 2K/4K/8K complex-point FFT core fabricated in a 0.35 $\mu$m technology, 4-ML CMOS process. The core size is 18.7 mm$^2$.

To sum up, our core simplifies the twiddle factors and, thus, reduces the number of multiplications. Therefore, our FFT proposal for a DVB-T/DVB-H system results in a more efficient ASIC and FPGA implementation than the proposals found in the literature.

## 5. CONCLUSION

An FFT core for DVB-T/DVB-H receivers has been designed and implemented. The core implements a pipeline r$2^2$ SDF architecture. This architecture can be adapted to achieve an efficient 2K/4K/8K multiple mode FFT processor. The extra hardware needed for multiple mode operation is minimal. In order to guarantee a SNR of 40 dB in all modes of operation, 16 bits and 11 bits are needed for the data bitwidth and the twiddle factor bitwidth, respectively. The architecture of the proposed FFT processor makes it possible to achieve the FFT processing time requirements of DVB-T/DVB-H working at the lowest-possible clock frequency. The proposed core is an

Table 6: Area and timing results in an FPGA virtex 2V6000FF1517.

| | *dbw* | *tbw* | $f_{\text{clk}}$ (MHz) | Occupied slices | BRAMs | Multipliers |
|---|---|---|---|---|---|---|
| [6] | 16 | 11 | 64 | 17305 (51%) | 96 (66%) | 16 (11%) |
| Ours | 16 | 11 | 9.143 | 6066 (17%) | 19 (13%) | 24 (16%) |

Table 7: Chip summary of our FFT processor.

| Items | Specification |
|---|---|
| FFT size | 2K/4K/8K |
| Clock frequency | 64/7 MHz |
| Data bitwidth (*dbw*) | 16 bits |
| Twiddle factor bitwidth (*tbw*) | 11 bits |
| Signal-to-quantization-noise ratio (SQNR) for $N_{\max} = 8\text{K}$ | 40.6 dB |
| Process technology | 0.35 $\mu$m XFAB 4-ML |
| Supply voltage | 3.3 V |
| Execution time (clock cycles) for $N = 2\text{K}$ | 1040 clock cycles |
| Execution time (clock cycles) for $N = 4\text{K}$ | 2064 clock cycles |
| Execution time (clock cycles) for $N_{\max} = 8\text{K}$ | 4115 clock cycles |
| Core power consumption for $N_{\max} = 8\text{K}$ | 114.65 mW |
| Core size | 18.7 mm$^2$ |

efficient implementation well suited for DVB-T/DVB-H receivers.

## ACKNOWLEDGMENTS

This research is supported in part by the *Ministerio de Industria, Turismo y Comercio* Grant no. FIT330100-2006-43 and by the Basque Government. A. Cortés holds the Torres Quevedo Grant no. PTQ05-02-02455, which was awarded by the Spanish Ministry of Education and Science, by the European Regional Development Fund and by the European Social Fund.

## REFERENCES

[1] G. Faria, J. A. Henriksson, E. Stare, and P. Talmola, "DVB-H: digital broadcast services to handheld devices," *Proceedings of the IEEE*, vol. 94, no. 1, pp. 194–209, 2006.

[2] ETSI EN 300744, "Digital video broadcasting (DVB); framing structure, channel coding and modulation for digital terrestrial television," 2004.

[3] U. H. Reimers, "DVB-The family of international standards for digital video broadcasting," *Proceedings of the IEEE*, vol. 94, no. 1, pp. 173–182, 2006.

[4] S. Y. Park, N. I. Cho, S. U. Lee, K. Kim, and J. Oh, "Design of 2K/4K/8K-point FFT processor based on cordic algorithm in OFDM receiver," in *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM '01)*, vol. 2, pp. 457–460, Victoria, BC, Canada, August 2001.

[5] J. F. Sevillano, A. Mtz de Gereñu, M. Leyh, P. Nagel, and A. Irizar, "An FFT parametrizable core," in *Proceedings of the 15th Conference on Design of Circuits and Integrated Systems (DCIS '00)*, pp. 230–234, Montpellier, France, November 2000.

[6] A. Cortés, I. Vélez, J. F. Sevillano, and A. Irizar, "An approach to simplify the design of IFFT/FFT cores for OFDM systems," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 1, pp. 26–32, 2006.

[7] Y.-W. Lin, H.-Y. Liu, and C.-Y. Lee, "A dynamic scaling FFT processor for DVB-T applications," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 11, pp. 2005–2013, 2004.

[8] T. Sansaloni, A. Pérez-Pascual, V. Torres, and J. Valls, "Efficient pipeline FFT processors for WLAN MIMO-OFDM systems," *Electronics Letters*, vol. 41, no. 19, pp. 1043–1044, 2005.

[9] Y. Jung, H. Yoon, and J. Kim, "New efficient FFT algorithm and pipeline implementation results for OFDM/DMT applications," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 1, pp. 14–20, 2003.

[10] C.-C. Wang, J.-M. Huang, and H.-C. Cheng, "A 2K/8K mode small-area FFT processor for OFDM demodulation of DVB-T receivers," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 1, pp. 28–32, 2005.

[11] S. He and M. Torkelson, "A new approach to pipeline FFT processors," in *Proceedings of the 10th International Parallel Processing Symposium (IPPS '96)*, pp. 766–770, Honolulu, Hawaii, USA, April 1996.

[12] J.-Y. Oh and M.-S. Lim, "New radix-2 to the 4th power pipeline FFT processor," *IEICE Transactions on Electronics*, vol. E88-C, no. 8, pp. 1740–1746, 2005.

[13] T. J. Ding, J. V. McCanny, and Y. Hu, "Rapid design of application specific FFT cores," *IEEE Transactions on Signal Processing*, vol. 47, no. 5, pp. 1371–1381, 1999.

[14] C.-P. Hung, S.-G. Chen, and K.-L. Chen, "Design of an efficient variable-length FFT processor," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '04)*, vol. 2, pp. 833–836, Vancouver, BC, Canada, May 2004.

[15] E. Bidet, D. Castelain, C. Joanblanq, and P. Senn, "A fast single-chip implementation of 8192 complex point FFT," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 3, pp. 300–305, 1995.

[16] Y. H. Hu, "The quantization effects of the CORDIC algorithm," *IEEE Transactions on Signal Processing*, vol. 40, no. 4, pp. 834–844, 1992.

Hindawi Publishing Corporation
VLSI Design
Volume 2008, Article ID 218565, 8 pages
doi:10.1155/2008/218565

*Research Article*

# Delay Efficient 32-Bit Carry-Skip Adder

**Yu Shen Lin and Damu Radhakrishnan**

*Department of Electrical and Computer Engineering, State University of New York, 1 Hawk Dr, New Paltz, NY 12561-2443, USA*

Correspondence should be addressed to Damu Radhakrishnan, damu@engr.newpaltz.edu

Received 27 April 2007; Accepted 9 December 2007

Recommended by Jean-Baptiste Begueret

The design of a 32-bit carry-skip adder to achieve minimum delay is presented in this paper. A fast carry look-ahead logic using group generate and group propagate functions is used to speed up the performance of multiple stages of ripple carry adders. The group generate and group propagate functions are generated in parallel with the carry generation for each block. The optimum block sizes are decided by considering the critical path into account. The new architecture delivers the sum and carry outputs in lesser unit delays than existing carry-skip adders. The adder is implemented in 0.25 $\mu$m CMOS technology at 3.3 V. The critical delay for the proposed adder is 3.4 nanoseconds. The simulation results show that the proposed adder is 18% faster than the current fastest carry-skip adder.

## 1. INTRODUCTION

The ever-increasing demand for mobile electronic devices requires the use of power-efficient VLSI circuits. Computations in these devices need to be performed using low-power, area-efficient circuits operating at greater speed. Addition is the most basic arithmetic operation; and adder is the most fundamental arithmetic component of the processor. Depending on the area, delay and power consumption requirements, several adder implementations, such as ripple carry, carry-skip, carry look-ahead, and carry select, are available in the literature [1, 2]. The ripple-carry adder (RCA) is the simplest adder, but it has the longest delay because every sum output needs to wait for the carry-in from the previous full-adder cell. It uses $O(n)$ area and a delay of $O(n)$ for an n-bit adder. The carry look-ahead adder has $O(\log n)$ delay and uses $O(n \log n)$ area. On the other hand, the carry-skip adder and carry-select adders have $O(\sqrt{n})$ delay and use $O(n)$ area [3].

In this paper, we present the design of a low-power adder with less delay while using minimum hardware. The standard carry generate-propagate logic is used to reduce the critical delay of the adder while blocks of RCAs are used for lesser power consumption. In our design, the generate-propagate logic balances the delay and the number of inputs to the skip logic limits the critical path delay. By applying our design procedure, we speed up the adder by 18% when compared to the current fastest 32-bit adder [4]. In Section 2, we will discuss the previous work done in the area of high-performance adders. In Section 3, we present the design of our adder. Section 4 presents the design of a few basic CMOS cells used in the adder. In Section 5, we present the simulation results for our adder and compare it to other fast adders.

## 2. THEORETICAL BACKGROUND AND PREVIOUS WORK

The design of a carry-skip adder is based on the classical definition of generate and propagate signals as follows [1, 2]:

$$\begin{aligned} p_i &= X_i \oplus Y_i, \\ g_i &= X_i \cdot Y_i, \end{aligned} \tag{1}$$

where $p_i$ is the propagate signal and $g_i$ is the generate signal, and $X_i$ and $Y_i$ are the input operands to the $i$th adder cell. The carry out from the $i$th adder cell is expressed as

$$C_{i+1} = g_i + p_i C_i, \tag{2}$$

where $C_i$ is the carry input to the $i$th cell.

Two signals, group generate and group propagate, are also defined in [1, 2] and are given by

$$G_{j:i} = g_j + p_j g_{j-1} + p_j p_{j-1} g_{j-2} + \cdots + p_j p_{j-1} p_{j-2} \cdots p_{i+1} g_i, \qquad P_{j:i} = p_j p_{j-1} p_{j-2} \cdots p_i, \tag{3}$$

where $G_{j:i}$ and $P_{j:i}$ are group generate and group propagate signals from $i$th cell to $j$th cell, respectively. Then, the expression for carry out from the whole group is given by

$$C_{j+1} = G_{j:i} + P_{j:i} C_i. \tag{4}$$

Different adder implementations have been developed to optimize various design parameters. Most adder implementations tend to trade off performance and area. One of the earliest adder implementations of this kind was a regular parallel adder layout also known as the Brent-Kung adder '82 [5]. It is a variation of the basic carry look-ahead adder . They emphasized the need for regularity in VLSI circuits to reduce design and implementation costs. They use two types of processor cells: white processor and black processor. The black processor performs the associative concatenation defined in [5] and the white processor simply transmits the data. The adder delay was calculated in terms of the number of exclusive-or (XOR) operations performed while treating each XOR delay as one unit time. For an n-bit adder, the Brent-Kung adder has a delay of $O(\log n)$ and uses $O(n \log n)$ area.

Wei-Thompson'85 [6] proposed an area-time optimal adder design using three types of adder cells: black cells, white cells, and driver cells. The black and white cells are quite similar to the ones used in Brent-Kung adder. They divided the n-bit adder into ascending and descending halves so as to limit the number of bits in the final stage. The concentration of the maximum number of bits was in the middle of the adder and was defined as the height of the adder. The algorithm ends up in an unbalanced binary tree with a delay of $O(\log n)$ consuming an area $O(n \log n)$.

The ELM-adder design presented in [7] computes the sum bits in parallel; thereby reducing the number of interconnects. It implements an n-bit adder as a tree of processors to directly compute the sums in $O(\log n)$ time. The area used is $O(n \log n)$. The adder design was expressed in terms of standard cells, which do not compute carry for each stage. Instead, partial sums were computed for each stage.

Kantabutra'93 [8] presents the design of a one-levelcarry-skip adder using an approach that is very similar to that of Wei-Thompson. In contrary to Wei-Thompson's approach, this design ends up in a symmetrical binary tree of adders. The fan-in to the carry-skip logic increases linearly towards the middle of the adder. A two-level carry-skip adder is presented in [9], where the whole adder stage is divided into a number of sections, each consisting of a number of RCA blocks of linearly increasing length. These adders reduce the delay at the cost of an increase in area and less regular layout.

Nagendra'96 [3] did a survey of various adder designs and concluded that the ELM adder was superior in terms of area, power, delay, and power-delay product. RCA was concluded to have utilized the least power, but has the highest delay due to its carry chain. A variable-width carry-skip adder was shown to be superior to constant-width carry-skip adder, the advantage being greater at higher precisions.

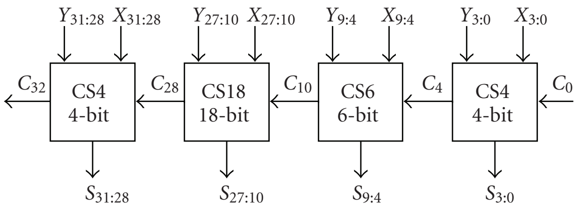


FIGURE 1: The 32-bit adder divided into 4 blocks [4].

A fully static carry-skip adder designed by Chirca'04 [4] achieved lower-power dissipation and higher performance. To reduce delay and power consumption, the adder is divided into variable-sized blocks that balance the inputs to the carry chain. The main principle behind this design was to utilize the lower blocks and make them work in parallel with higher blocks. This paper is a deviation from the tree approach presented in the ELM adder. A 32-bit adder implementation with a delay of 7 logic levels using carry-skip adders and ripple-carry adders was presented in [4]. This is shown in Figure 1. The logic-level delay defined in the paper is equivalent to the delay of a complex CMOS gate. Efficient and-or-invert (AOI) and or-and-invert (OAI) CMOS gates were used to reduce delay and power.

The 32-bit adder is divided into 4 adder blocks as shown in Figure 1. Carry-select adders were used in the final CS4 block, which significantly increases the hardware. The paper claims that the output will be ready with a delay of 7 logic levels, with the assumption that the critical delay path is the carry propagation path of $C_{32}$ bit. But a closer examination of the previous block CS18 reveals that the 27th bit of the sum output will be available only after a delay of 9 logic levels.

## 3. NEW DESIGN FOR THE 32-BIT CARRY-SKIP ADDER

The 32-bit carry-skip adder design presented in this paper uses a combination of RCAs together with carry-skip logic (SKIP), carry-generate logic (CG), and group generate-propagate logic (PG). The complete adder is divided into a number of variable-width blocks. Both the carry generation and skip logic use AOI and OAI circuits. The width of each block is limited by the target delay T.

Each block is further divided into subblocks. A subblock may contain additional levels of subblocks in a recursive manner. The lowest-level subblock is formed by a number of variable width RCAs. The adder structure is described as follows:

$$\begin{aligned} &\text{Block} \longrightarrow \langle\text{Block}\rangle * \langle\text{Block}\rangle \mid \langle\text{subblock}\rangle \mid \langle\text{RCA}\rangle, \\ &\langle\text{subblock}\rangle \longrightarrow \langle\text{subblock}\rangle * \langle\text{subblock}\rangle \mid \langle\text{RCA}\rangle. \end{aligned} \tag{5}$$

The 32-bit adder is divided into four blocks. A block diagram of the first three blocks ($A_0$, $A_1$, and $A_2$) is shown in Figure 2. The first block $A_0$ (LSB) is a full adder by itself.

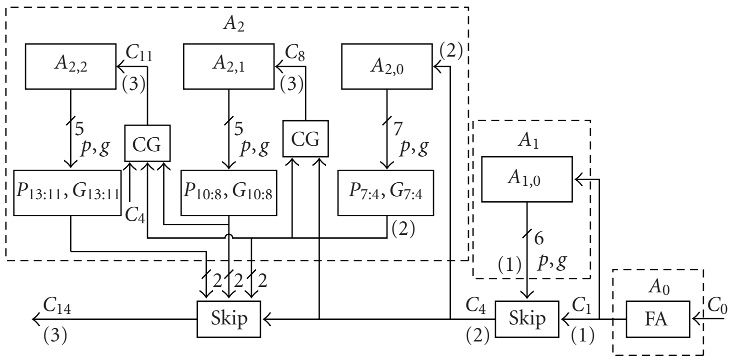


FIGURE 2: Block schematics for first three blocks of 32-bit adder.

The carry from the first block C1 is fed into the second block $A_1$ and is also fed into the skip logic. The generate and propagate functions $(p, g)$ are generated separately for each full adder in one unit time, where one unit time is defined as the delay of a complex CMOS gate with at most three transistors connected in series from the output node to any supply rail. In Figure 2, the numbers shown in parenthesis represent the number of unit delays of the signal arrival times at the appropriate signal leads. Since the delay of a complex CMOS gate is quadratic on its stack height, in our design, the stack height is limited to 3. This implies that the maximum number of transistors (NMOS or PMOS) in any series connected path is 3. This also restricts the maximum number of inputs to the carry-skip logic to 7. On the other hand, when the generate-propagate outputs are used for group generation and group propagation (3) outputs, a stack height of 3 in the CMOS implementation will allow a 4-bit RCA.

The carry-generation delay from the skip logic is minimized by alternately complementing the carry outputs. Hence, the carry signals generated are $\overline{C_1}, C_4, \overline{C_{14}}$, and so forth. For the very first 1-bit block $(A_0)$, the carry-generation logic is more important than the sum-generation logic since the overall delay of the adder is dependent on the carry from this block. Hence, this block is designed by minimizing the carry out delay as much as possible. The simplest expression of carry out from the LSB full adder is given by

$$\overline{C_1} = \overline{X_0 Y_0 + X_0 C_0 + Y_0 C_0}, \tag{6}$$

where $X_0$ and $Y_0$ are the operand bits and $C_0$ is the input carry. An AOI gate implements this.

The block $A_1$ in Figure 2 is implemented as a k-bit RCA. For any k-bit RCA, the total number of propagate and generate $(p, g)$ outputs would be 2 k. These 2 k outputs together with the carry from the previous block are fed into carry-skip logic to generate the new carry signal. The fan-in restriction of 7 to the carry-skip logic therefore limits the number of bits in the RCA to 3. The carry out $C_4$ from skip logic for block $A_1$ is given by

$$C_4 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 C_1. \tag{7}$$

Since $g$'s and $p$'s can be best implemented in complementary form, we can rewrite $C_4$ as

$$C_4 = \overline{\overline{g}_3 (\overline{p}_3 + \overline{g}_2)(\overline{p}_3 + \overline{p}_2 + \overline{g}_1)(\overline{p}_3 + \overline{p}_2 + \overline{p}_1 + \overline{C}_1)}. \tag{8}$$

By inspection, $C_4$ can be implemented by an or-and-invert (OAI) gate and is available in 2 time units. The final Sum output $S_3$ from this 3-bit RCA will be available in 4 time units. The sum outputs for this RCA are generated either as $S_i = p_i \oplus C_i$ or $S_i = \overline{\overline{p_i} \oplus \overline{C_i}}$ depending on the carry signal value ($C_i$ *or* $\overline{C_i}$). The carry out $C_2$ and $C_3$ are implemented as $C_2 = \overline{\overline{g_1} \cdot (\overline{p_1} + \overline{C_1})}$ and $C_3 = g_2 + p_2 C_2$, respectively.

Now consider block $A_2$ in Figure 2. The delay of carry signal arriving at the input of the skip logic is 2 time units. This implies that the group generate-propagate $(P, G)$ logic outputs feeding the skip logic must also be available in 2 time units. Hence, the inputs to the $(P, G)$ logic must be available in 1 time unit. This implies that the inputs to the $(P, G)$ logic must be the propagate and generate signals of the full adders. Block $A_2$ is divided into three subblocks $A_{2,0}$, $A_{2,1}$, and $A_{2,2}$ (in this case, each subblock is an RCA). The maximum width of each RCA is limited to 4 bits due to the fan-in restrictions imposed on the $(P, G)$ block. The width of each RCA is also limited by the target delay $T$ of the 32-bit adder. The width W of the first RCA is given as

$$W = T - D, \tag{9}$$

where $D$ is the arrival delay of the carry output from the previous block. The width of all remaining higher order RCAs in the same block will be 1 bit less because of the delayed arrival times of their carry input by an additional time unit. The carry inputs $C_8$ and $C_{11}$ to RCAs $A_{2,1}$ and $A_{2,2}$ are generated using AOI logic as follows:

$$\overline{C_8} = \overline{G_{7:4} + P_{7:4} C_4}, \tag{10}$$

$$\overline{C_{11}} = \overline{G_{10:8} + P_{10:8} G_{7:4} + P_{10:8} P_{7:4} C_4}. \tag{11}$$

For a target delay of 6 time units, the width of the first RCA in $A_2 (A_{2,0})$ is 4 bits and the widths of the remaining

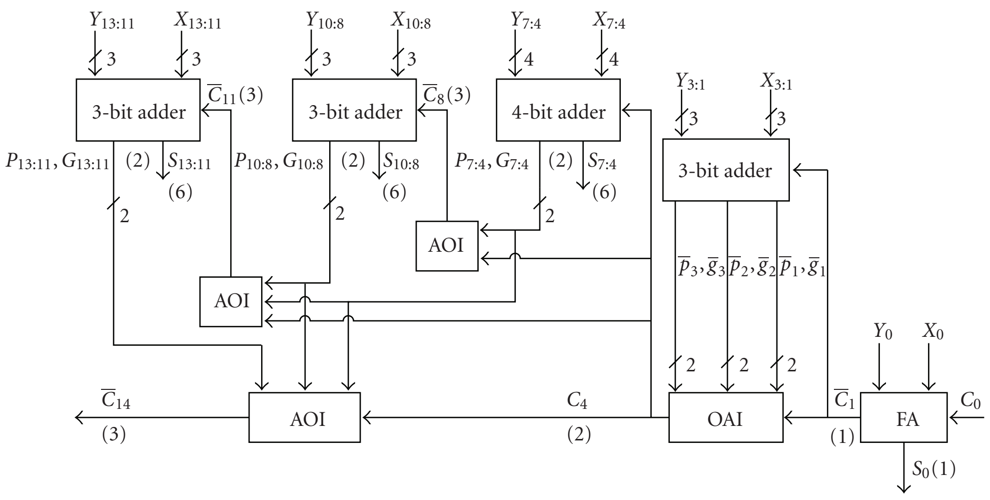


FIGURE 3: Detailed view of the first three blocks of 32-bit adder.

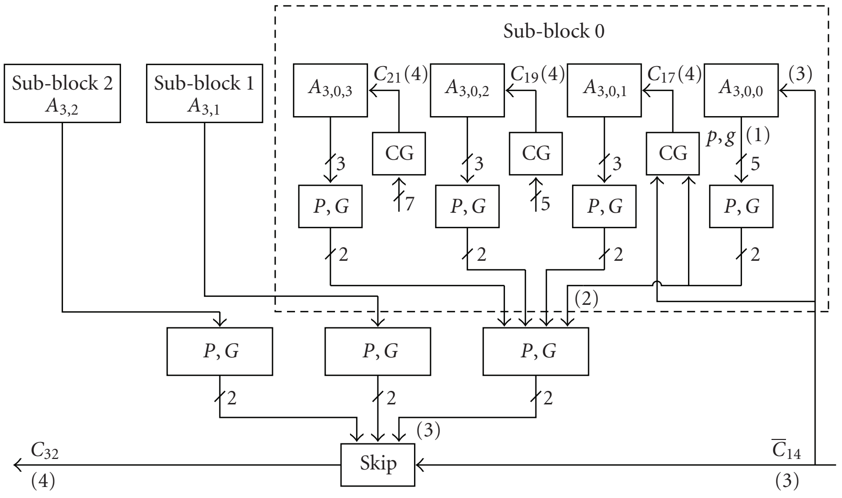


FIGURE 4: Block-3 of 32-bit adder with an expanded view of sub-block 0.

RCAs ($A_{2,1}$ and $A_{2,2}$) are each 3 bits. The number of RCAs in $A_2$ is limited to 3 due to the fan-in restriction of 7 on the skip logic. Each RCA in block $A_2$ also represents a subblock of $A_2$. The carry out $C_{14}$ from the skip logic is implemented using AOI logic as

$$\overline{C_{14}} = \overline{G_{13:11} + P_{13:11} \cdot G_{10:8} + P_{13:11} \cdot P_{10:8} \cdot G_{7:4} + P_{13:11} \cdot P_{10:8} \cdot P_{7:4} \cdot C_4}. \tag{12}$$

A detailed block diagram of the first three blocks of the 32-bit adder (an expanded view of Figure 2) is shown in Figure 3. The three blocks together form a 14-bit adder.

Next let us consider the final block $A_3$ of the 32-bit adder. Block $A_3$ is divided into a number of subblocks. The maximum number of subblocks is again limited to 3 due to the fan-in restrictions on the skip logic. A block diagram of $A_3$ with an expanded view of subblock $0(A_{3,0})$ is shown in Figure 4. The subblock 0 is further divided into RCAs. The number of inputs to the CG logic increases, successively, by 2 for each RCA and is limited to a maximum of 7 in any subblock. Hence, the number of RCAs in any subblock is limited either by the number of inputs to the CG block or by the number of inputs to the $(P, G)$ block. Therefore, subblock 0 can accommodate 4 RCAs. The carry input to the skip logic, as well as, to the first RCA ($A_{3,0,0}$) arrives in 3 time units. The propagate and generate signals ($p$ and $g$) from each RCA will be available with a delay of 1 time unit. This implies that we can have two levels of $(P, G)$ logic inside the block while satisfying the time delay constraints. Using (9), the width of the first RCA ($A_{3,0,0}$) is 3 bits, and the widths of the remaining RCAs are 2 bits each. Hence, the total width of subblock $0(A_{3,0})$ is 9 bits.

Figure 5 shows block $A_3$ with an expanded view of subblock $1(A_{3,1})$. The number of RCAs in $A_{3,1}$ is limited to 3 due to the condition stated earlier. The carry input $C_{23}$ to the first RCA ($A_{3,1,0}$) of this subblock is given by

$$C_{23} = \overline{\overline{G_{22:14}}\left(\overline{P_{22:14}} + \overline{C_{14}}\right)}. \tag{13}$$

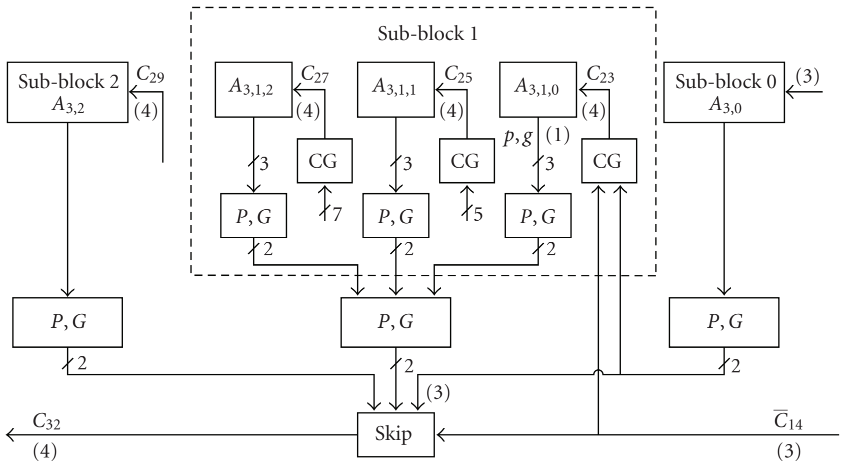


FIGURE 5: Block-3 of 32-bit adder with an expanded view of sub-block 1.

With an AOI logic implementation, $C_{23}$ will be available in 4 time units, thereby limiting the length of the first RCA to 2 bits. The carry inputs $C_{25}$ and $C_{27}$ to the remaining RCAs in subblock $1(A_{3,1})$ are also available in 4 time units. Thus, the maximum width of subblock $A_{3,1}$ is 6 bits. The carry input $C_{29}$ to the final subblock $2(A_{3,2})$ is given by

$$C_{29} = \overline{\overline{G_{28:23}}\,(\overline{P_{28:23}} + \overline{G_{22:14}})\,(\overline{P_{28:23}} + \overline{P_{22:14}} + \overline{C_{14}})}. \quad (14)$$

The maximum width of subblock $A_{3,2}$ can be calculated as 4 bits. This subblock can accommodate only 2 RCAs due to the fan-in limits of the CG blocks. Hence, the total width of block $A_3$ is 19 bits. By combining the 4 blocks $A_0$, $A_1$, $A_2$, and $A_3$ a 33-bit adder can be implemented. The width of subblock $A_{3,2}$ can be shortened to 3-bits for a 32-bit adder. The carry out $C_{32}$ from the skip logic is given by

$$C_{32} = \overline{G_{31:29}(\alpha)(\beta)(\gamma)}, \quad (15)$$

where, $(\alpha) = (\overline{P_{31:29}} + \overline{G_{28:23}})$, $(\beta) = (\overline{P_{31:29}} + \overline{P_{28:23}} + \overline{G_{22:14}})$, $(\gamma) = (\overline{P_{31:29}} + \overline{P_{28:23}} + \overline{P_{22:14}} + \overline{C_{14}})$.

An OAI logic implementation generates $C_{32}$ in 4 time units. A detailed block diagram of block $A_3$ is shown in Figure 6. The final breakdown of the 32-bit adder into 4 blocks is shown in Figure 7. A reduction in hardware can be achieved by moving subblock $A_{3,2}$ from block $A_3$ and placing it as another block $A_4$. This will eliminate 1 carry generate logic (OAI) and $1(P, G)$ logic.

Although our adder has already achieved the 32-bit requirement, we still have room to extend the width further, while keeping the target delay the same. The schemes for the 5th and 6th blocks are shown in Figure 8. The fifth block $A_4$ is divided into three subblocks. The subblocks ($A_{4,0}, A_{4,1}$, and $A_{4,2}$) have the same structure as block $A_3$. Since the carry fed into the 5th block has 4 unit delays, the maximum width of the first RCA will be 2 bits. The remaining RCAs will be 1 bit each. Thus, the maximum width for the fifth block will be 20 bits. The first subblock $A_{4,0}$ (11 bits) is divided into subblocks of 5, 3, 2, and 1 bit. The subblock $A_{4,1}$ (6 bits) is divided into 3 subblocks of 3, 2, and 1 bit. Similarly, the final subblock $A_{4,2}$ (3 bits) is divided into subblocks of 2 and 1 bit. The first 5-bit subblock ($A_{4,0,0}$) consists of a 2-bit RCA and 3 individual full adders. Individual full adder cells form all other subblocks. The 6th block $A_5$ is a single bit full adder. Thus, the total width of the adder becomes 54.

Based on the adder design procedure, we can derive a formula for calculating the maximum number of full adders in every block. The following notations are used in the derivation. $T$ Target delay of the $n$-bit adder in time units, $N(i)$ The number of RCAs in block $i$, $W(i, j)$ The width of RCA "$j$" in block $i$.

For any block $i$ $(i \geq 2)$, the number of RCAs is defined by a recursive function $N(i)$. The recursive function is not valid for the blocks $A_0$ and $A_1$, and the values for $N(0)$ and $N(1)$ when used in the recursive function are assumed to be zero

$$N(i) = \sum_{1}^{i} i + N(i-1), \qquad N(0) = N(1) = 0. \quad (16)$$

The width of an RCA is defined in terms of the target delay. The width $W(i, j)$ of the RCA"$j$" in any block"$i$" is defined as

$$W(i,j) = \begin{cases} \min\ (4, T-i), & \text{for } j = 0, \\ \min\ (4, T-i-1), & \text{for } 1 \leq j \leq N(i)-1, \end{cases} \quad (17)$$

where min $(a, b)$ is the minimum value among $a$ and $b$.

The carry input to the first RCA of the block can be obtained directly from the previous carry-skip stage. Hence, the calculation of width for the first block is done differently from the others.

The maximum number of full adders $FA(i)$ in block $i$ is given by

$$\begin{aligned} FA(i) &= \sum_{j=0}^{N(i)-1} W(i,j) \\ &= \sum_{j=1}^{N(i)-1} W(i,j) + W(i,0). \end{aligned} \quad (18)$$

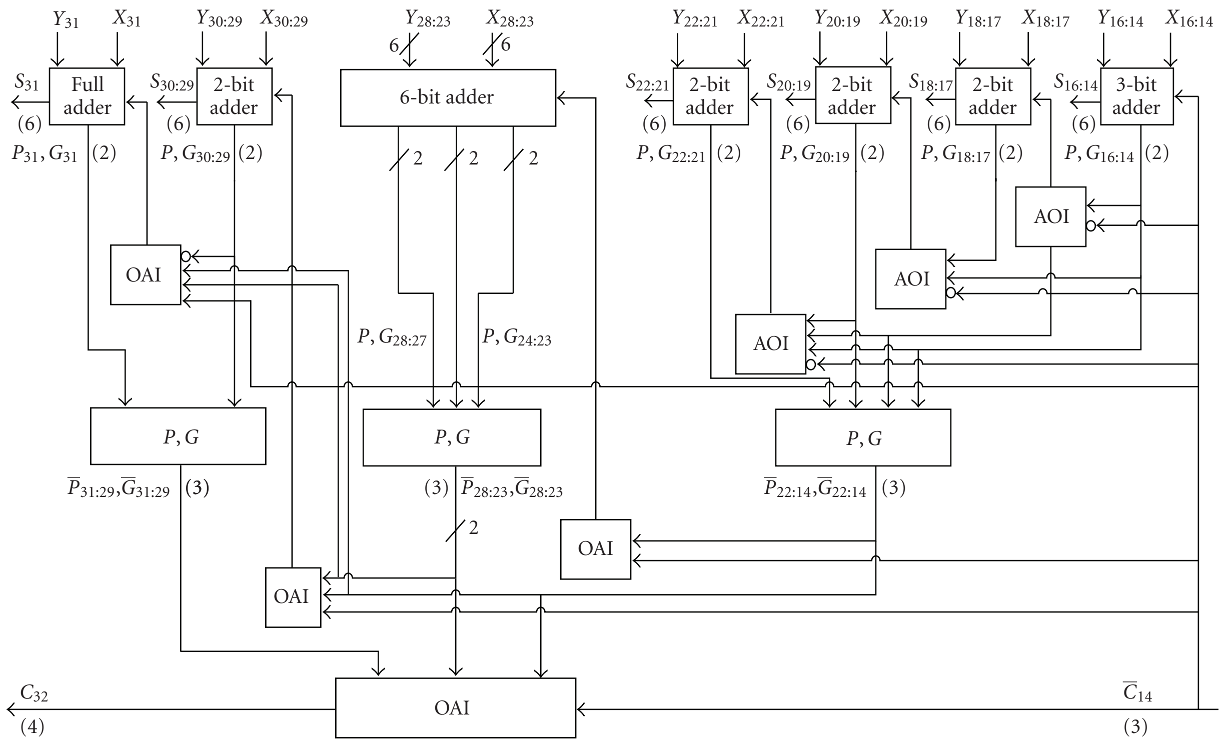


FIGURE 6: Detailed scheme for block $A_3$.

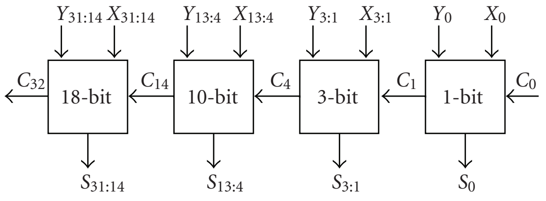


FIGURE 7: The proposed 32-bit adder.

TABLE 1: Maximum size of adders.

| Target delay ($T$)-time units | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|
| Adder size ($n$)-bits | 9 | 22 | 54 | 119 | 237 |

Table 1 lists the maximum adder size for a given target delay using our design procedure.

## 4. DESIGN OF BASIC CMOS CELLS

A few basic CMOS cells are used for the design of the adder stage. They are: AOI, OAI, and FA cells. Three different cells are used for AOI and OAI (3-input, 5-input, and 7-input). These cells are labeled as AOIn and OAIn, where n refers to the number of inputs to the cell. The 3-input and 5-input cells are implemented in a straightforward manner, and are given by the following Boolean expressions:

$$\text{AOI3: Out} = \overline{A + B \cdot C}, \tag{19}$$

where $A$, $B$, and $C$ are the inputs for the gate. The 3-input OAI is expressed as

$$\text{OAI3: Out} = \overline{A \cdot (B + C)}. \tag{20}$$

The expressions for 5-input AOI and OAI are given as

$$\begin{aligned}\text{AOI5: Out} &= \overline{A + B \cdot (C + D \cdot E)},\\ \text{OAI5: Out} &= \overline{A \cdot (B + C \cdot (D + E))},\end{aligned} \tag{21}$$

where $A, B, C, D$, and $E$ are the inputs to the cells. When the 7-input AOI and OAI cells are implemented in the above manner, the delay is prohibitive and hence we decided to implement them as a cascade connection of a number of smaller modules. Their corresponding Boolean expressions are given by

$$\begin{aligned}\text{AOI7: Out} &= \overline{A + B \cdot C + \overline{\overline{B \cdot D}} + \overline{\overline{(E + F \cdot G)}}},\\ \text{OAI7: Out} &= \overline{A \cdot (B + C) \cdot \left[\overline{\overline{(B + D)}} \cdot \overline{\overline{(E \cdot (F + G)}}\right]},\end{aligned} \tag{22}$$

where $A$, $B$, $C$, $D$, $E$, $F$ and $G$ are the inputs to the cells. Since we reduce the stack height of the transistors connected in series from 4 to 3, the 7-input AOI and OAI cells will be speeded up and the propagation delay will be almost the same as the 5-input AOI and OAI. The full adder cell used in our design is the low-energy CMOS adder cell presented in [10].

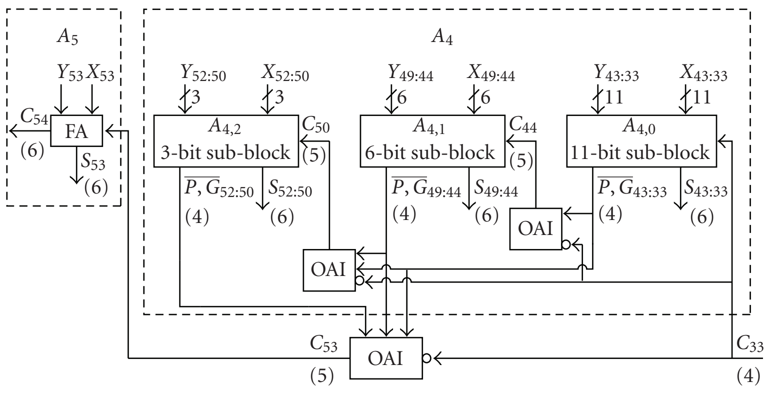


Figure 8: The schemes for the fifth and sixth blocks.

Table 2: Cell Characteristics.

| | Delay (ns) | Average power (mw) |
|---|---|---|
| AOI3 | 0.267 | 0.128 |
| OAI3 | 0.264 | 0.127 |
| AOI5 | 0.527 | 0.15 |
| OAI5 | 0.541 | 0.16 |
| AOI7 | 0.578 | 0.33 |
| OAI7 | 0.579 | 0.30 |
| FA | 0.604 | 0.34 |

Table 3: Adder comparison for delay, power, and power-delay product.

| | Delay (ns) | Power (mw) | PDP (pJ) |
|---|---|---|---|
| 32-bit adder (Chirca) | 4.15 | 4.68 | 19.4 |
| 32-bit adder (Gayles) | 4.39 | 3.26 | 14.3 |
| Our adder (32-bit) | 3.4 | 4.2 | 14.28 |
| Our adder (54-bit) | 4.3 | 8.6 | 36.98 |

## 5. SIMULATION

The adder was implemented using Tanner tools pro 11.03. L-edit was used to generate the layout and T-spice was used for performing the simulation. The generic 0.25 $\mu$m CMOS technology was used with 3.3 volts supply voltage. The different CMOS cells (AOI, OAI, and FA) were simulated for worst-case delays and the delays are tabulated in Table 2. From Table 2, it may be noted that the 5 and 7-input cell delays are comparable to that of the FA, while the 3-input cells have a much less delay. The average power was measured by feeding 10,000 random vectors at a frequency of 500 MHz and is also shown in Table 2.

For comparison purposes, we selected two other types of adders. They are (i) 32-bit carry skip-adder proposed in [4] and (ii) 32-bit multilevel carry-skip adder proposed in [11]. The first one is referred here as Chirca adder and the second one is referred as Gayles adder. These adders were compared with our 32-bit adder by measuring the critical path delays. To get a more realistic estimation of the delays involved, we laid out the complete 32-bit adder stages and performed TSPICE simulation. The simulation was carried out at a frequency of 100 MHz. The simulation results are shown in Table 3. These results show that our 32-bit adder has the minimum delay of 3.4 nanoseconds while Gayles adder exhibited a maximum delay of 4.39 nanoseconds. The Chirca adder had a delay of 4.15 nanoseconds. Thus, our design has a speedup of 18% and 22% compared to those of Chirca and Gayles adders, respectively. Our 32-bit adder was then extended to a 54-bit adder with marginal delay increase, and these simulation results are also included in Table 3. Even this 54-bit adder is found to be faster than the 32-bit Gayles adder.

The power consumption showed a marginal increase of power for our adder compared to Gayles adder while outperforming Chirca adder. Overall, our 32-bit adder achieved the lowest power-delay product.

## 6. CONCLUSIONS

In this paper, we presented a new 32-bit adder using carry-skip logic. The adder was implemented by dividing the adder into several blocks. The size of each block is limited by the delay of the carry-in signal and the final target delay. An algorithm is used to calculate the maximum size of the adder satisfying the target delay. The delay of a full adder is used as the unit of measurement in our analysis. The adder has been implemented by generating the layout with Generic 0.25 $\mu$m CMOS technology. The TSPICE simulations carried out at a frequency of 100 MHz and supply voltage of 3.3 V showed a critical path delay of 3.4 nanoseconds. The comparison results show that our adder is faster than Chirca and Gayles carry-skip adders. Overall our proposed adder is 18% and 22% faster compared to the Chirca and Gayles adders, respectively. Furthermore, a 54-bit adder implemented using our approach can operate almost at the same speed as a 32-bit Chirca adder or Gayles adder. Even though our adder has a marginal increase in power consumption compared to the Gayles adder, overall, we achieved the lowest power-delay product.

## REFERENCES

[1] I. Koren, *Computer Arithmetic Algorithms*, A. K. Peters, Natick, Mass, USA, 2nd edition, 2002.

[2] B. Parhami, *Computer Arithmetic Algorithms and Hardware Designs*, Oxford University Press, Oxford, UK, 2000.

[3] C. Nagendra, M. J. Irwin, and R. M. Owens, “Area-time-power tradeoffs in parallel adders,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 43, no. 10, pp. 689–702, 1996.

[4] K. Chirca, M. Schulte, J. Glossner, et al., “A static low-power, high-performance 32-bit carry skip adder,” in *Proceedings of the EUROMICRO Symposium on Digital System Design (DSD ’04)*, pp. 615–619, Rennes, France, August-September 2004.

[5] R. P. Brent and H. T. Kung, “A regular layout for parallel adders,” *IEEE Transactions on Computers*, vol. 31, no. 3, pp. 260–264, 1982.

[6] B. W. Y. Wei, C. D. Thompson, and Y. F. Chen, “Time optimal design of a CMOS adder,” in *Proceedings of the 19th Annual Asilomar Conference on Circuits, Systems, and Computers*, pp. 186–191, Pacific Grove, CA, USA, November 1985.

[7] T. P. Kelliher, R. M. Owens, M. J. Irwin, and T.-T. Hwang, “ELM-A fast addition algorithm discovered by a program,” *IEEE Transactions on Computers*, vol. 41, no. 9, pp. 1181–1184, 1992.

[8] V. Kantabutra, “Designing optimum one-level carry-skip adders,” *IEEE Transactions on Computers*, vol. 42, no. 6, pp. 759–764, 1993.

[9] V. Kantabutra, “Accelerated two-level carry-skip adders-a type of very fast adders,” *IEEE Transactions on Computers*, vol. 42, no. 11, pp. 1389–1393, 1993.

[10] S. Goel, S. Gollamudi, A. Kumar, and M. Bayoumi, “On the design of low-energy hybrid CMOS 1-bit full adder cells,” in *Proceedings of the 47th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS ’04)*, vol. 2, pp. 209–212, Hiroshima, Japan, July 2004.

[11] E. Gayles, R. M. Owens, and M. J. Irwin, “Low power circuit techniques for fast carry-skip adders,” in *Proceedings of the 39th IEEE Midwest Symposium on Circuits and Systems*, vol. 1, pp. 87–90, Ames, Iowa, USA, August 1996.

Hindawi Publishing Corporation
VLSI Design
Volume 2008, Article ID 753043, 6 pages
doi:10.1155/2008/753043

*Research Article*

# High-Performance Timing-Driven Rank Filter

**Péter Szántó,[1] Gábor Szedő,[2] and Béla Fehér[1]**

[1] *Department of Measurement and Information Systems, Budapest University of Technology and Economics, Magyar tudósok krt. 2, 1117 Budapest, Hungary*
[2] *Xilinx Inc., 2100 Logic Drive, San Jose, CA 95124, USA*

Correspondence should be addressed to Péter Szántó, szanto@mit.bme.hu

Received 27 April 2007; Accepted 2 November 2007

Recommended by Jean-Baptiste Begueret

This paper presents an FPGA implementation of a high-performance rank filter for video and image processing. The architecture exploits the features of current FPGAs and offers tradeoffs between complexity and performance. By maximizing the operating frequency, the complexity of the filter structure can be considerably reduced compared to previous 2D architectures.

## 1. INTRODUCTION

Rank-order filtering is a nonlinear filtering technique, which selects an element from an ordered list of *TAP* number of samples. In the two-dimensional (2D) case, filtering takes place on the contents of a rectangular window (or more generally, an arbitrary shape), which slides across the image. Every time the window is moved by one pixel column, a set of obsolete elements is discarded and a set of new elements is inserted. The samples within the window are sorted and the element with the specified rank replaces the output element of the window. Most typical ranks are median, minimum, and maximum, but the selection can be easily tailored to the needs of any application. Compared to other filters, such as FIR, Laplacian, or blur filters, rank filters can effectively remove impulses like noises while preserving the edges of the original image. This can be very useful for various applications, for instance, removing certain types of transmission noises or preprocessing for edge detection. This paper presents a hardware architecture that is tailored for high-performance color video processing, but it can be used in various applications such as IP block by taking advantage of design time parameterization. The paper concentrates on the timing-driven architecture selection which exploits the high operating frequency of recent FPGA and ASIC technologies, thus reducing hardware resource requirements.

## 2. PREVIOUS WORK

The successful adaptation of rank filters in different applications catalyzed research activities for new algorithms and implementations.

Bit-serial approaches [1, 2] provide the lowest complexity, but they do not lend themselves well to high sample rate implementations as filtering performance is proportional to the precision of the input data. However, the processing rate typically does not depend on the number of samples which changes between processing cycles.

Insert/delete or sorting network-based architectures [3, 4] explicitly order incoming samples. In every cycle, the least recent sample is discarded and the most recent input is inserted into the magnitude sorting structure at the appropriate location. While these solutions require relatively few comparators, the feedback nature of the algorithm hinders pipelining.

Another set of applications stores the samples in the order of arrival and selects the appropriate output sample by calculating the location of the output sample dynamically. These architectures are easier to pipeline and they still require few comparators.

## 3. PROPOSED ARCHITECTURE

On filtering images or videos, the filter window is sliding horizontally across the input image, as illustrated in Figure 1. In

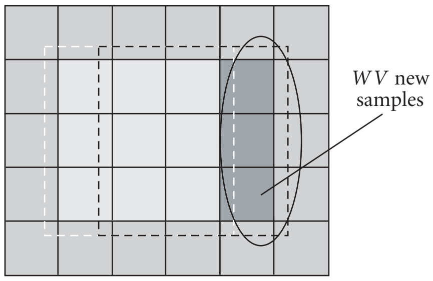


Figure 1: 2D image filtering.

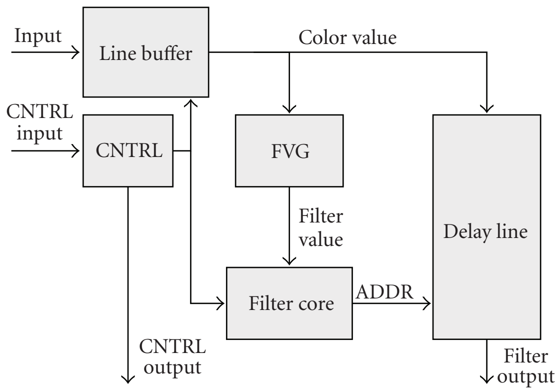


Figure 2: Global filter architecture.

case of a simple rectangular window, to generate a valid output, *WV* (vertical size of the filter window) new input samples should be processed. Word-serial architectures can process one input sample per clock cycle. When comparing different solutions, an important classification criterion is the level of input parallelization. In the 2D filtering case, the filter should operate at *WV* times of the input pixel frequency and generate a valid input sample every *WV*th clock cycle.

Fully parallel filters can generate a valid output sample every clock cycle, irrespective of the number of input samples required to achieve this process. Consequently, such filters process *WV* new samples in a single clock cycle, and the required operating frequency is equal to the input pixel frequency. At the same time, hardware resource requirements are greatly increased. Previous papers typically considered fully parallel architectures such as 2D filters; however, as this paper proves, using recent FPGA technologies, this solution is suboptimal due to the inefficient resource utilization.

Multiword architectures are hybrid solutions; in one cycle, they can handle more than one input sample, but less than the fully parallel implementation. This solution allows finding an optimal balance between operating frequency and hardware complexity. Using given filter window and input pixel frequency, with *NI* defining the number of new input samples in a single cycle, the required operating frequency can be computed as

$$FO_{\max} = FS\frac{WV}{NI}. \qquad (1)$$

On processing color images, using the full per-pixel information (e.g., full RGB or YCbCr values) is not an efficient solution. Filtering these components independently not only increases computational requirements but may also introduce blur effects, as it may generate new color values which did not exist on the input image. A better solution is to use a magnitude-like value, such as luminosity. If the input format does not contain such a component, it can be generated within the filter.

### 3.1. Global filter architecture

The proposed architecture consists of five main components (as illustrated in Figure 2): the line buffer (LB), the optional filter value generator (FVG), the delay line (DL), the filter core (FC), and the control unit (CNTRL).

The LB stores *WV-1* lines of the original input frame in the internal memory. The FVG is only required if the input format does not contain a magnitude-like component. For YCbCr or YUV input representations, this module can be omitted as the Y component lends itself well to magnitude ordering. For RGB input (luminance), a typical magnitude value can be calculated. The DL is an addressable FIFO which stores the full per-pixel information of the pixels residing inside the FC. The FC itself uses the values computed by the FVG and generates the appropriate address for the DL. CNTRL generates properly delayed synchronization signals and output valid signals. As the rest of the architecture is independent of the FC solution, further discussion will focus on the FC and its extensions.

### 3.2. Word-serial filter core

The operation of the FC is based on observations introduced in [5]. As a first assumption, the filter contains *TAP* number of different samples. For each sample, an index value is generated, which is equal to the number of samples which are smaller than the given sample.

This results in *TAP* distinct values for the *TAP* samples which range from 0 (the smallest sample) to *TAP*-1 (the largest sample). The ranked sample is the one which has the index value equal to the required rank. The block diagram in Figure 3 illustrates the hardware implementation of the algorithm for *TAP* = 5. The *D*[3:0] data registers store older filter values, while the new data value is saved into the *ND* register. In every cycle, these registers shift their data to the left. Older values are compared with the new value (the result is "1" if the new value is smaller than the older ones, and "0" otherwise), and the comparison result is saved into the LSB position of *TAP*-1, *TAP* wide registers (*CR*[3:0]). The MSB positions of these *CR* registers are updated with the value of the previous *CR* register. So, the full content of the *CR*[] registers is

$$CR[k] = \{CR[k-1](TAP-2:0), C[k]\}, \qquad (2)$$

where (:) denotes bit selection, {} denotes concatenation, and $C[k]$ denotes the *k*th comparison result. The comparison result of a given value is shifted to the left together with

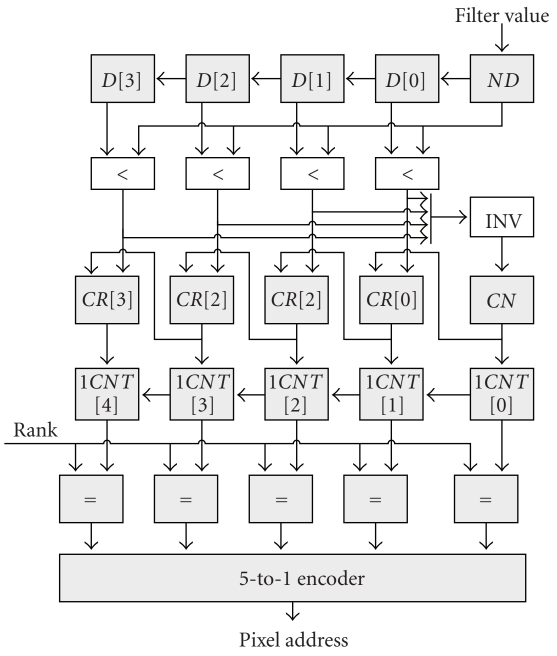


Figure 3: Filter core.

Table 1: Filtering example.

| | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|
| $D[\,]$, $ND$ | 0 | 25 | 37 | 12 | 12 |
| $CR[\,]$, $CN$ | 00000 | 10011 | 11011 | 10000 | 10010 |
| $1CNT[\,]$ | 0 | 3 | 4 | 1 | 2 |

the filter value. Therefore, at any given time, $CR[k]$ stores the comparison results of $D[k]$ with all the other values within the filter. The *TAP* wide register for the new value (*CN*) is computed differently; it is generated using the negated result of the comparators; namely, the *k*th bit is updated with the $(k+1)$th comparison result. The 0th bit (self-comparison) is set to "0." Counting the "1"s in the $CR[]$ and *CN* registers gives a number of values which are smaller than the given value. These bit summing operations are carried out by the 1CNT modules. The straightforward way is to use an adder tree with *TAP* one-bit inputs. For the *CN* register, this is the only solution, as its content can change arbitrarily from clock to clock. Generation of $CR[k]$ can be optimized taking into consideration the fact that only two bits change from $CR[k-1]$: the MSB (comparison result with the discarded sample) and the LSB (comparison result with the new value). Therefore, bit summing can be implemented using an incrementer/decrementer. The results of the bit summing blocks are compared with the required rank, generating a *TAP* bit wide vector of results containing exactly one "1" at the position of the cell which contains the required output. An encoder passes this position to the DL as an address. Table 1 shows an example with the data registers ($D[]$, *ND*), $CR[]$, *CN* and the output of the 1CNT blocks.

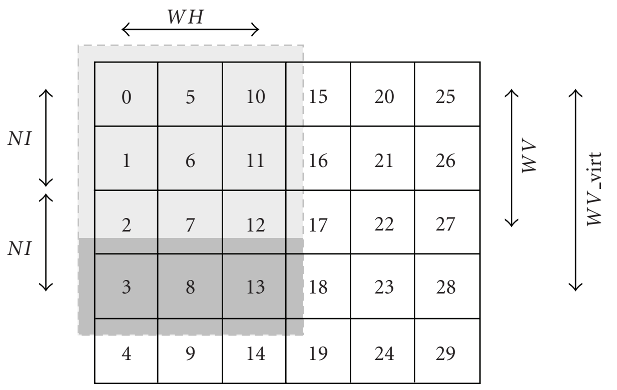


Figure 4: Virtual filter kernel.

### 3.3. Multiword filter core

The architecture presented in the previous section can be easily extended to process more than one new filter value per clock cycle. Instead of one, the data registers ($D[]$) and the comparator result shift registers ($CR[]$) should shift by *NI* data positions. The yet single *CN* and *CR* registers become register arrays with *NI* elements. The number of comparators is increased, as all old samples should be compared with all new samples and new samples should be compared with each other. The required number of comparators for a *TAP* sized filter with *NI* new samples is

$$C = (TAP - NI)^{*}NI + \frac{NI^{*}(NI-1)}{2}. \tag{3}$$

If *WV* is not an integer multiply of *NI*, the bandwidth of the filter core input supersedes that of the input stream. So in some clock cycles, the number of valid new data is going to be less than *NI*. The simplest solution to make the filter capable of processing different number of new samples is to insert multiplexers into the appropriate data paths, in front of $D[]$, $ND[]$, $CR[]$, and $CN[]$ registers. Two-to-one multiplexers always suffice as the number of valid new inputs is either *NI* or *WV mod NI* (see Figure 4). Still, for large apertures, numerous multiplexers may be required.

Another solution is to insert padding samples as necessary such that in every clock cycle *NI* new samples can be entered, thus creating a virtual filter kernel (VK). Figure 4 illustrates such kernel for $WV = 3$ and $NI = 2$ case. Valid samples in the window are marked with light grey; padding samples are marked with dark grey (the actual value of the padding samples are irrelevant). Obviously, this method makes the size of the VK larger than that of the real filter window, hence requiring more hardware resources as parts of the FC scale with the size of the VK.

Figure 5 presents the contents of the data registers clock by clock, using the example in Figure 4, as new inputs are inserted and the filter window is moved horizontally. Background shading of valid and invalid (padding) samples corresponds to Figure 4. Samples on the right are the input samples. As any given register may contain valid or invalid samples during operation, comparisons are done using all data

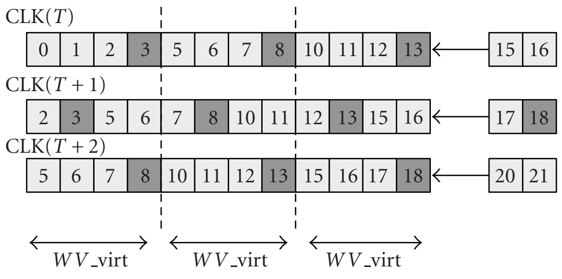


Figure 5: Masked data register.

registers, irrespective of the validity. Therefore, the number of comparators required scales with the size of the VK.

Padding samples are masked after the comparator result registers ($CR[\,]$, $CN[\,]$), but before the 1CNT blocks. For each older sample, masking is done for $2*NI$ bits. $NI$ bits mask the comparison results with the $NI$ new samples, and other $NI$ bits mask the comparison results of the oldest $NI$ samples. The output ranking part is the same as in the single-word case. The number of required equality comparators is proportional to the size of the real filter window as it is sufficient to select the appropriate output when all samples in a new column have been inserted into the filter. In these cycles, the locations of the valid samples are well defined.

### 3.4. *Multiword filter with multiple outputs*

In case valid samples are used for padding, the virtual filter kernel can be viewed as $NP + 1$ filter windows processed together, where $NP$ is the number of padding lines added to the filter window to form the VK. For example, the 3×4 virtual kernel in Figure 4 can be viewed as two 3×3 partially overlapping filter windows. The FC presented in the previous section already computes all the required comparison results to generate valid outputs for both of the 3×3 filter windows. However, to come up with 2 separate outputs, the mask generator, the one-counters, and the output address generator should be replicated. The advantage is that the relation between the operating frequency and the number of new inputs processed in a single cycle becomes even better, significantly improving efficiency:

$$FO = \frac{\left\lceil \dfrac{WV}{NI} \right\rceil}{\left\lceil \dfrac{WV}{NI} \right\rceil * NI - WV + 1} * FS. \qquad (4)$$

The drawback is that the LB should store $WV$ lines of the input image instead of $WV-1$. In case of real-time video filtering, an output buffer may also be required.

### 3.5. *Nonrectangular filter window*

The mask-based filtering architecture allows for the easy implementation of nonrectangular (convex and nonconvex) filter windows. The most significant difference compared to the multiworld implementation described above is that the valid

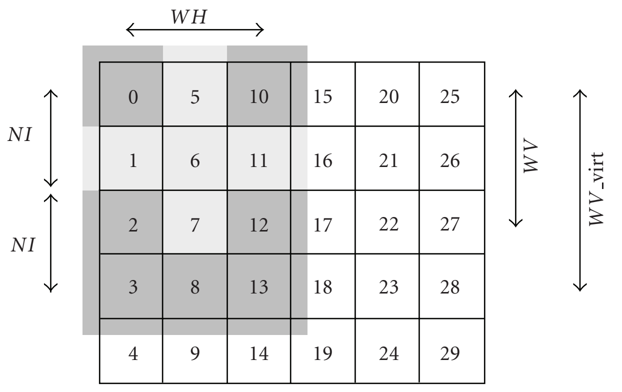


Figure 6: Nonrectangular mask.

or invalid status of a given filter value may change as the filter window slides across the input image. For example, in Figure 6, pixel 10 is invalid in the first computation cycle it is used, but as the window slides one pixel to the right, it becomes valid.

Consequently, bit summing becomes more complex as the number of possible transitions between the masked $CR[\,]$ and $CN[\,]$ registers is increased. Nonrectangular windows typically increase the number of invalid samples within the VK. Therefore, using the bit summing block for the valid samples only may reduce resource requirements. Practically, in the latter implementation, only the number of $ND[\,]$ and $D[\,]$ registers scales with the virtual filter window; all other processing units are implemented only for the valid data.

### 3.6. *Weighted rank filtering*

Some applications require the use of weighted filter windows, rendering some input samples more significant than others when determining the output of the filter. The proposed method allows for the application of integer weights. The comparison result bits ($CR[\,]$ and $CN[\,]$ registers' outputs) are replicated as many times as determined by the corresponding weight factor. However, the bit summing blocks become increasingly complex as their inputs become wider due to bit replication. Also, the $TAP$ bit summing operation results in $TAP$ different values, which are in the range of $0 \cdots W-1$, where $W$ is the summation of all the weights. As $TAP$ is smaller than $W$, not all integer values will be presented at the outputs of the bit summing units. Therefore, a simple equality comparator is no longer adequate to determine the ranked sample. Instead, the filter has to find the sample which has the closest bit summing value to the required rank (which is in the range of $0 \cdots W-1$).

To facilitate the correct selection, the proposed architecture (see Figure 7) employs several difference computing units and a selection tree. The difference computing units process the required rank and the outputs of the bit summing units. The two input minimum calculators select the smaller of their inputs together with a binary flag which shows whether the left or the right input was selected. At the root of the tree, the concatenated tag bits determine the

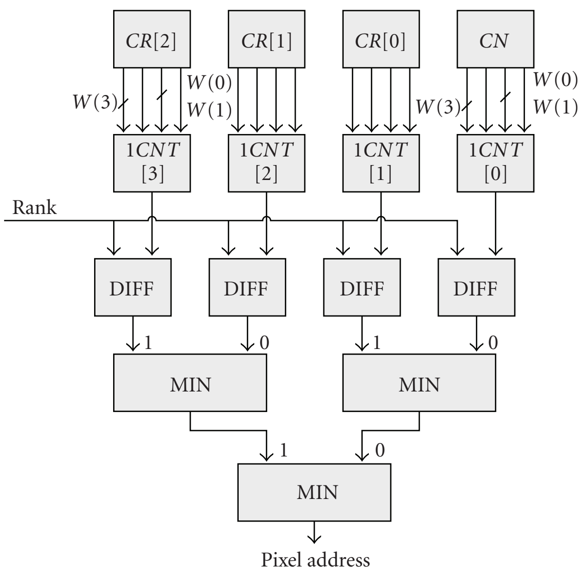


Figure 7: Filter core for weighted rank filtering.

Table 2: Word-serial operating frquency.

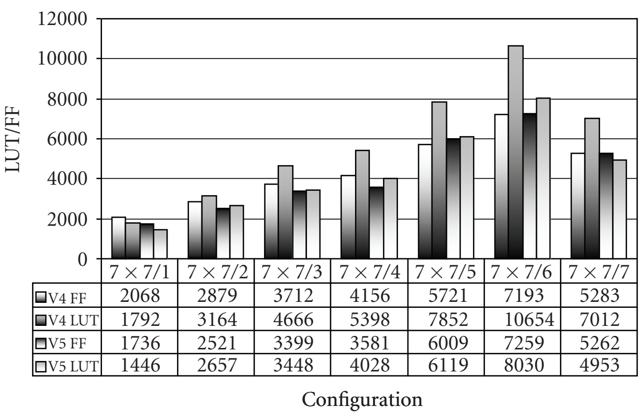
| Family | Number of taps (*TAP*) | | |
|---|---|---|---|
| | 9 | 25 | 49 |
| XC5V-3 | 460 MHz | 420 MHz | 400 MHz |
| XC4V-10 | 400 MHz | 375 MHz | 355 MHz |
| XC3S-4 | 245 MHz | 195 MHz | 175 MHz |

location of the sample which has the closest bit summing value to the required rank. This value can be used to address the DL.

## 4. IMPLEMENTATION RESULTS

The following implementation results were obtained using 24-bit RGB input, with an FVG that sums the three color components and outputs a 10-bit result. Table 2 summarizes the operating frequencies obtained for the word-serial architecture for different Xilinx FPGA families and different *TAP* numbers. These values can be used as a reference to help determine the required parallelization level of the FC, depending on the input pixel frequency and the filter window size.

Table 2 offers different solutions even for one of the most demanding commercial video format, HDTV1080p, which has a pixel frequency of 75 MHz. For example, a Virtex-4 device can perform real-time filtering on HDTV source using a 49-tap filter by employing a multiword FC configuration with 2 input samples per clock cycle. Figure 8 summarizes the resource requirements of a 49-tap rank filter using different FC configurations (configuration *WVxWH/NI*). LUT and FF denote the number of lookup tables and flip flops in Virtex-4 and Virtex-5 devices, respectively. Figure 6 demonstrates that some multiword configurations (such as 7×7/5, 7×7/6) may require more resources than the full parallel architecture (7×7/7). The reason for this is that the VK becomes much larger than the valid filter window due to the enormous number of padding samples.



| | 7 × 7/1 | 7 × 7/2 | 7 × 7/3 | 7 × 7/4 | 7 × 7/5 | 7 × 7/6 | 7 × 7/7 |
|---|---|---|---|---|---|---|---|
| ▢ V4 FF | 2068 | 2879 | 3712 | 4156 | 5721 | 7193 | 5283 |
| ▨ V4 LUT | 1792 | 3164 | 4666 | 5398 | 7852 | 10654 | 7012 |
| ▨ V5 FF | 1736 | 2521 | 3399 | 3581 | 6009 | 7259 | 5262 |
| ▢ V5 LUT | 1446 | 2657 | 3448 | 4028 | 6119 | 8030 | 4953 |

Figure 8: Resource requirements.

Table 3: Operating frequency and resource requirements using Spartan-3 (9 and 25 taps).

| Configuration | 3×3/1 | 5×5/1 | 5×5/2 | 5×5/3 |
|---|---|---|---|---|
| FFs | 567 | 1234 | 1832 | 2044 |
| LUTs | 395 | 937 | 1420 | 1913 |
| BRAMs | 6 | 12 | 12 | 12 |
| $F_{\text{CLK max}}$ | 245 | 195 | 180 | 165 |

Table 4: Operating frequency and resource requirements using Spartan-3 (49 taps).

| Configuration | 7×7/1 | 7×7/2 | 7×7/3 | 7×7/4 |
|---|---|---|---|---|
| FFs | 2121 | 3030 | 3849 | 4242 |
| LUTs | 1862 | 3302 | 4866 | 5627 |
| BRAMs | 18 | 18 | 18 | 18 |
| $F_{\text{CLK max}}$ | 175 | 160 | 150 | 150 |

These configurations are inferior to the full parallel architecture in terms of throughput and silicon real estate. The presented architecture can take advantage of the 6-input LUTs of the Virtex-5 FPGA family, resulting in 20–30% reduction in the design size.

Tables 3, 4, 5, and 6 summarize the achievable operating frequencies and resource requirements of several filter configurations using Spartan-3, Virtex-4, and Virtex-5 devices, respectively. For every filter size and FPGA family, the configurations marked with light grey background can be used to filter HDTV (1920×1080 30 p—75 MHz pixel clock) input. The lower-performance configurations are still adequate for lower-resolution video inputs, like SDTV.

Although the longest register-to-register path does not depend on the filter configuration, as the complexity of the filter increases, the achievable operating frequency still decreases. This is common when using FPGAs and should be taken into consideration when selecting the filter configuration for given input format and filter size.

Table 5: Operating frequency and resource requirements using Virtex-4.

| Configuration | 3×3/1 | 5×5/1 | 7×7/1 | 7×7/2 |
|---|---|---|---|---|
| FFs | 594 | 1088 | 2068 | 2879 |
| LUTs | 406 | 950 | 1792 | 3164 |
| BRAMs | 6 | 12 | 18 | 18 |
| $F_{\mathrm{CLK\,max}}$ | 400 | 375 | 355 | 300 |

Table 6: Operating frequency and resource requirements using Virtex-5.

| Configuration | 3×3/1 | 5×5/1 | 7×7/1 | 7×7/2 |
|---|---|---|---|---|
| FFs | 580 | 1050 | 1736 | 2521 |
| LUTs | 290 | 750 | 1446 | 2657 |
| BRAMs | 3 | 6 | 9 | 9 |
| $F_{\mathrm{CLK\,max}}$ | 460 | 420 | 400 | 340 |

## 5. CONCLUSION

An efficient architecture for high-performance two-dimensional rank filters was presented. Rank-order filters, especially median filters, are used extensively for removing non-Gaussian (salt and pepper) noise from images and video streams. Targeting FPGA implementations for video applications, a parameterizable structure was proposed which delivers an efficient solution custom tailored to different pixel clock rates, available resources, and operating speeds. Compared to previous 2D architectures, the size and complexity of the filter structure were considerably reduced by balancing the number of new input samples entered into the core and the available operating frequency of the filter. The proposed solution is independent of input data type, as it offers great flexibility to generate magnitude information corresponding to RGB data, or it can take advantage of preexisting magnitude information if such data are already available. The solution presented can handle nonrectangular filter windows or weighted samples as well, which widens the domain of possible applications even further.

## REFERENCES

[1] R. Roncella, R. Saletti, and P. Terreni, “70-MHz 2-$\mu$m CMOS bit-level systolic array median filter,” *IEEE Journal of Solid State Circuits*, vol. 28, no. 5, pp. 530–536, 1993.

[2] B. K. Kar and D. K. Pradhan, “New algorithm for order statistic and sorting,” *IEEE Transactions on Signal Processing*, vol. 41, no. 8, pp. 2688–2694, 1993.

[3] C. Chakrabarti and L.-Y. Wang, “Novel sorting network-based architectures for rank order filters,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 502–507, 1994.

[4] P. Szántó, G. Szedő, and B. Fehér, “Implementing 2D median filter in FPGAs,” in *Proceedings of the 7th International Carpathian Control Conference (ICCC ’06)*, Roznov, Czech Republic, May 2006.

[5] C. Chakrabarti, “High sample rate array architectures for median filters,” *IEEE Transactions on Signal Processing*, vol. 42, no. 3, pp. 707–712, 1994.

Hindawi Publishing Corporation
VLSI Design
Volume 2008, Article ID 738983, 6 pages
doi:10.1155/2008/738983

*Research Article*

# Wave Pipelining Using Self Reset Logic

**Miguel E. Litvin and Samiha Mourad**

*Department of Electrical Engineering, School of Engineering, Santa Clara University, 500 El Camino Real, Santa Clara, CA 95053, USA*

Correspondence should be addressed to Miguel Litvin, melitvin@gmail.com

Received 1 May 2007; Accepted 9 December 2007

Recommended by Jean-Baptiste Begueret

This study presents a novel design approach combining wave pipelining and self reset logic, which provides an elegant solution at high-speed data throughput with significant savings in power and area as compared with other dynamic CMOS logic implementations. To overcome some limitations in SRL art, we employ a new SRL family, namely, dual-rail self reset logic with input disable (DRSRL-ID). These gates depict fairly constant timing parameters, specially the width of the output pulse, for varying fan-out and logic depth, helping accommodate process, supply voltage, and temperature variations (PVT). These properties simplify the implementation of wave pipelined circuits. General timing analysis is provided and compared with previous implementations. Results of circuit implementation are presented together with conclusions and future work.

## 1. INTRODUCTION

Wave pipelining (WP) is a suitable solution for fast arithmetic circuit implementation since it renders high throughput, while reducing the area and power overhead in a pipeline by removing intermediate registers. Such registers result in a latency penalty due to their setup and clock-to-output times and introduce delays for each stage. The area savings are realized due to (a) area devoted to the registers themselves, and (b) area needed for the clock distribution and buffering to control such registers.

In WP designs, each stage holds its output just enough time to guarantee that the next stage will be able to capture the data and start the computation of its own outputs to the following element in the pipe. So ideally, data have to progress *simultaneously* through the stages to achieve the maximum data throughput. Specific timing constraints apply to guarantee no data corruption. These designs are essentially asynchronous, but can be synchronized by the use of input and output registers (implemented with latches, or flip-flops) as long as timing conditions are met so that outputs are captured at an appropriate time. This circuit arrangement is shown in Figure 1.

Self reset logic (SRL) provides circuit implementations where "everything is quiet" when no new data are received. For single-rail implementations, power consumption is "data dependent." In a dual-rail implementation, there will be pulses propagating along the wave path (either at the direct or inverse outputs of each stage); every time new data is presented at primary inputs. We introduce a new family of dual-rail SRL with input disable (DRSRL-ID). A typical cell is shown in Figure 2. In Section 2, we discuss its operation. Section 3 describes the timing constraints that apply when designing wave-pipelining circuits with DRSRL-ID. Section 4 presents an application circuit, and Section 5 presents conclusions and future work.

## 2. SELF RESET LOGIC

A DRSRL-ID buffer-inverter cell is shown in Figure 2. The gate will generate an output pulse at the direct output (buf) or the inverse output (inv) only if the inputs validate the logic function $F$ or its inverse; otherwise, both outputs will remain at zero. Once inputs evaluation starts, the gate disconnects the inputs for the duration of the cycle time $\tau$ defined as follows:

$$\tau = t_{df} + w_p + t_{\text{rec}}, \qquad (1)$$

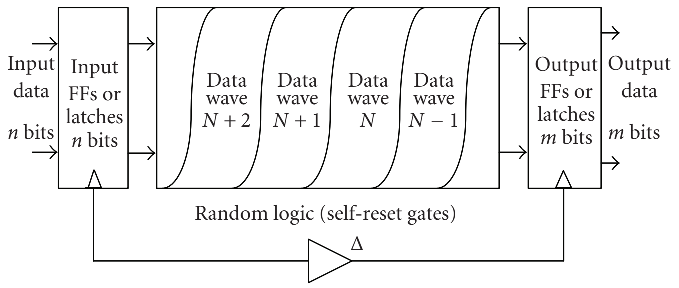


FIGURE 1: Basic wave pipelined circuit.

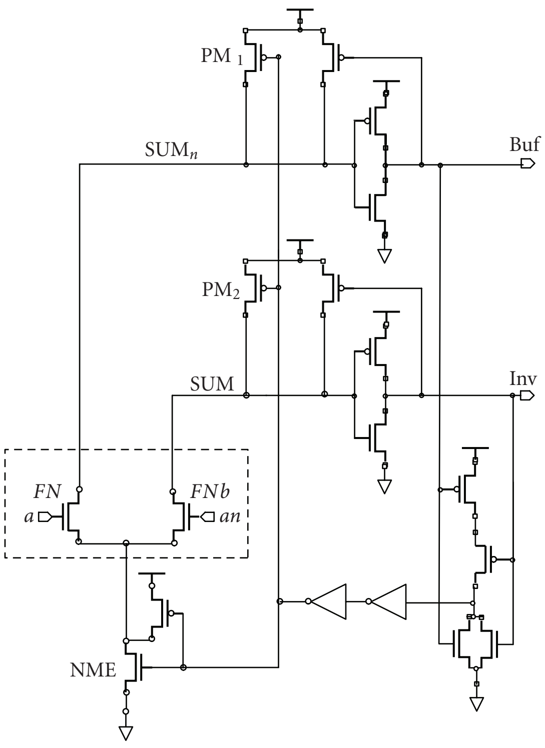


FIGURE 2: A DRSRL-ID buffer-inverter gate.

where the following definitions hold:

$t_{df}$ : *denotes data delay forward* (the time from the leading edge of the input data transition that validates *F* or *FN* to the leading edge of the pulse at the output);

$w_p$ : *denotes width* of the *output pulse*;

$t_{\text{rec}}$ : *denotes recovery time* (the time elapsed from the trailing edge of the output pulse to the trailing edge of the reset pulse).

For inputs to be evaluated, they have to be active for a minimum overlapping time, $t_{\text{ovlmin}}$, that must be longer than the capture time $t_c$.

$t_c$ *denotes* capture time, the time from the leading edge of the input data transition that validates *F* or *FN* to the falling edge of the pulse at the internal summing node (*SUM or SUM$_n$*).

We prefer to refer to the output pulse (at either output) so the condition is written as

$$t_{\text{ovlmin}} \geq t_{df}. \tag{2}$$

The width of the output pulse $w_p$ depends strongly on the characteristics of the output stage of the gate, but is independent of the loading while *fan-out* is equal to or less than 8 (for the gate families we have worked with). When a set of inputs validates the logic function *F* (or *Fn*), the corresponding output pulse starts only after the delay $t_{df}$, but its width $w_p$ depends on the delay through the feedback loop, which postcharges the summing nodes. Also, since we disable inputs, once an output pulse starts, $w_p$ is also *independent of the width of input pulses,* while they satisfy condition (2). Recovery time $t_{\text{rec}}$ and delay forward $t_{df}$ can also be made equal for a family of gates. Then, the *cycle time* $\tau$ will be a constant for the circuit implemented with these gates. It defines the minimum clock period at which new data can be pushed into the combinational circuit when received from an input register.

Figure 4 shows these timing parameters depicted for idealized waveforms corresponding to the behavior of a typical gate of this type. The outputs $Y$ and $Y_n$ correspond to outputs buf and inv of the buffer-inverter cell shown in Figure 2. For a complete description and characterization of these gates, we refer the reader to [1, 2].

An XOR/XNOR gate is shown in Figure 3. In this case, the use of shared elements between the FN and FNb blocks minimizes the number of devices needed. This approach is especially useful when implementing more complex gates. Additionally, in this case, we show the implementation of the self reset without using the extra inverters in the feedback loop. In the present case, as we actually use the internal reset *pulse_rst* signal to disable input readout as well as to control the postcharge of the summing nodes, we can safely play with the width of the resetting pulse, without being affected by the switching activity at the gate inputs.

## 3. WAVE PIPELINING WITH DRSRL-ID

The wave-pipelining circuit is an asynchronous structure, which can be made to work in a synchronous structure by adding an input and an output register, controlled by clock, as shown in Figure 1. This requires careful selection of the

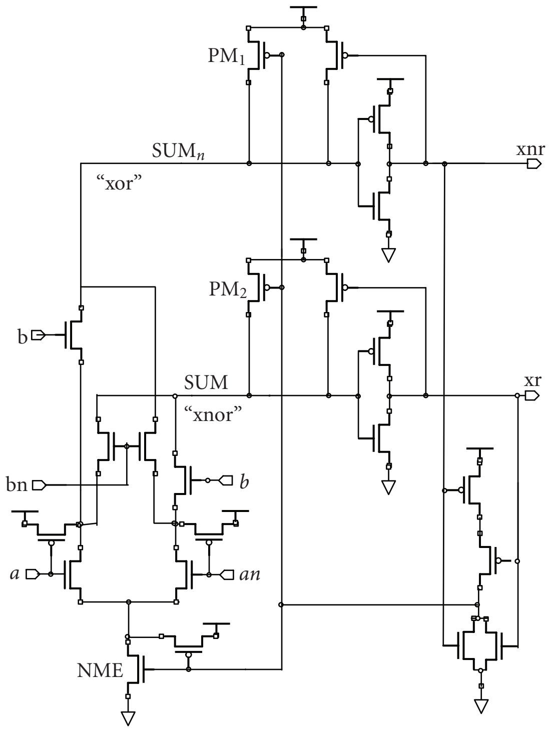


Figure 3: Dual-rail SRL XOR/XNOR gate with input disable.

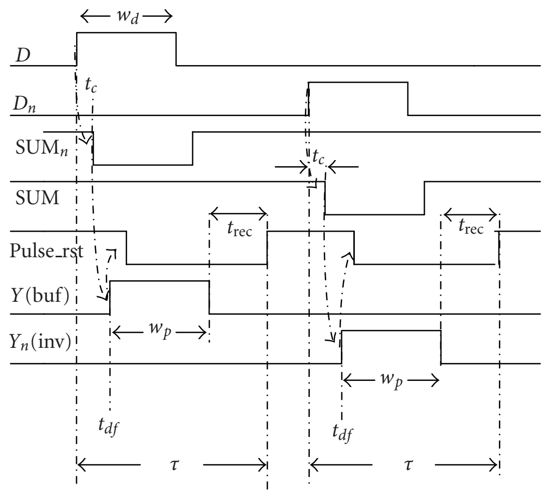


Figure 4: Timing parameters.

timing parameters. In the rest of this section, we explain the relationship between these different timing parameters using the following symbols:

| | |
|---|---|
| $k$: | number of data waves in the pipeline; |
| $ck$: | global clock; |
| $T_{ck}$: | period of the global clock; |
| $ck_{ir}$: | clock at input register; |
| $ck_{\text{out}}$: | clock at output register; |
| $T_L$: | total latency (time elapsed from launching a data wave from the input register until the corresponding result arrives at the output register); |
| $T_P$: | maximum delay through the combinational logic; |
| $\Delta T_P$: | maximum path delay difference through the combinational logic; |
| $\Delta o$: | phase shift between $ck$ and $ck_{\text{out}}$; |
| $\Delta i$: | phase shift between $ck$ and $ck_{ir}$; |
| $\Delta$: | $T_L \bmod T_{ck} : \Delta o - \Delta i$ : constructive skew (phase shift between the clocks that control the launching and receiving registers); |
| $t_d$: | register clock-to-$Q$ delay; |
| $t_s$: | register setup time; |
| $t_h$: | register hold time; |
| $t_{sk}$: | uncontrollable clock skew. |

The width of the output pulse of the input register $w_{pIR}$ must satisfy the following:

$$t_{df} \le w_{pIR} < T_{ck}. \tag{3}$$

The timing conditions are

$$w_p > \Delta T_{pi} + t_{df} + t_{sk}, \tag{4}$$

where $\Delta T_{pi}$ is the worst timing difference expected at any given stage. Then,

$$T_{ck} \ge w_p + t_{\text{rec}}, \tag{5}$$

considering also the delay forward $t_{df}$,

$$T_{ck} \ge w_p + t_{\text{rec}} + t_{df}, \tag{6}$$

that is,

$$T_{ck} \ge \tau. \tag{7}$$

As can be observed in Figure 5, the total latency is

$$T_L = kT_{ck} + \Delta o. \tag{8}$$

Analyzing the situation corresponding to a "late arriving" pulse versus an "early arriving" one, as shown in Figure 6, one can demonstrate that for wave pipelining with DRSRL-ID,

$$T_{ck} \ge \frac{t_d + T_p + t_s + t_{sk} - \Delta}{k}, \tag{9}$$

$$w_p > \Delta T_p + t_s + t_h + 2t_{sk.} \tag{10}$$

Comparing with regular WP CMOS implementation, as shown by [3–5], in that case the conditions for safe pipelining include the following (11) and (12):

$$T_{ck} > \Delta T_p + t_s + t_h + 2t_{sk}. \tag{11}$$

Condition (12) is a two-sided constraint on $k$, $T_{ck}$, and $\boldsymbol{\Delta}$, showing the behavior as we sweep frequencies:

$$\frac{T_p + t_d + t_s + t_{sk} - \Delta}{k} \le T_{ck} < \frac{T_p - \Delta T_p + t_d - t_h - t_{sk} - \Delta}{k-1}. \tag{12}$$

Condition (12) applies to the regular WP circuits and defines a set of "valid intervals" of frequencies at which the circuit

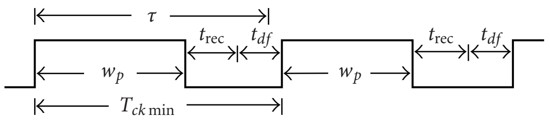


FIGURE 5: Timing diagram for successive pulses.

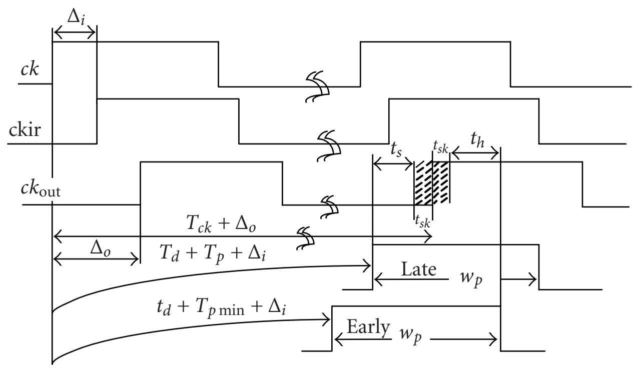


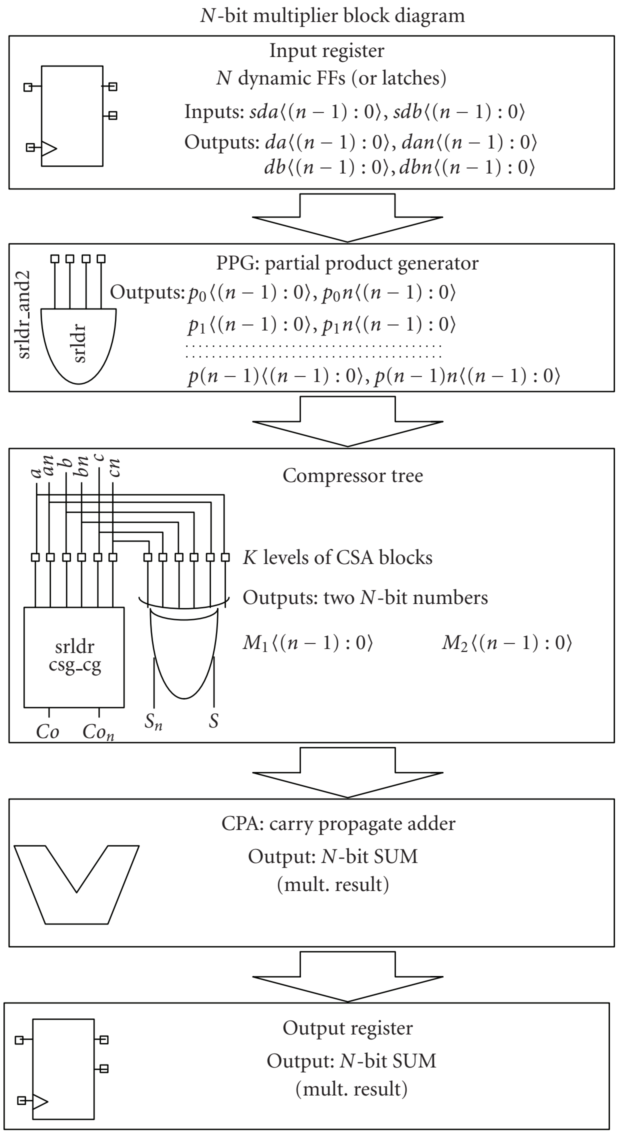
FIGURE 6: Pulses of the same data wave, with phase shift between input and output register clocks.

will behave in a wave-pipelining mode. The higher the frequency, the narrower the valid clocking interval [4, 5]. So it becomes extremely important to control delays carefully, and there is a strong dependency on the process, voltage, and temperature (PVT) conditions.

In contrast, in DRSRL-ID there is a maximum frequency at which the circuit can operate in WP mode, as stated by conditions (6) and (7). As long as the clock period $T_{ck}$ is greater than the cycle time $\tau$ of the gates, early arrival data from data wave $(N + 1)$ will not interfere with late arrivals from the previous data wave. Trying to operate at higher frequencies will generate a situation where, at a given stage, bits would arrive "too early" and will be ignored by that stage, since these gates will still have their inputs disabled. For DRSRL-ID, at frequencies below what condition (7) states, the combinational logic will still function properly (with different $k$ values). The only difficulty resides in capturing the computation result at the output register. Such behavior can be obtained by adding a latch at the end of the combinational logic, which will update when and if new data arrives, that is, converting the last stage into a static one. These characteristics render the technique presented in this paper very desirable.

The conditions on $w_p$ in DRSRL-ID are similar to the conditions on $T_{ck}$ for CMOS WP, rendering a theoretic lower data rate. In other words, we could design for a suboptimal frequency, but building headroom for process, voltage, and temperature (PVT), that is, we accept a maximum operating frequency, and design with a built-in margin.

Observe that condition (4) implies that we need to minimize the timing difference among signals arriving at any given stage $\Delta T_{pi}$, since this directly impacts the maximum achievable operating frequency. One still needs to do "rough tuning" to equalize timing paths at each stage: add gates to shorter paths, and maintain a solid layout engineering that



$N$-bit multiplier block diagram
Input register
$N$ dynamic FFs (or latches)
Inputs: $sda\langle(n-1):0\rangle$, $sdb\langle(n-1):0\rangle$
Outputs: $da\langle(n-1):0\rangle$, $dan\langle(n-1):0\rangle$
$db\langle(n-1):0\rangle$, $dbn\langle(n-1):0\rangle$
PPG: partial product generator
srldr_and2
srldr
Outputs: $p_0\langle(n-1):0\rangle$, $p_0n\langle(n-1):0\rangle$
$p_1\langle(n-1):0\rangle$, $p_1n\langle(n-1):0\rangle$
$p(n-1)\langle(n-1):0\rangle$, $p(n-1)n\langle(n-1):0\rangle$
Compressor tree
$a$ $an$ $b$ $bn$ $c$ $cn$
$K$ levels of CSA blocks
Outputs: two $N$-bit numbers
$M_1\langle(n-1):0\rangle$
$M_2\langle(n-1):0\rangle$
srldr csg_cg
$Co$ $Co_n$
$S_n$ $S$
CPA: carry propagate adder
Output: $N$-bit SUM
(mult. result)
Output register
Output: $N$-bit SUM
(mult. result)

FIGURE 7: Multiplier block diagram.

looks into equalizing wire loads. The "fine-tuning" proposed in other implementations [4] may not add much in this case because of the "built-in" headroom by the gates. (Fine-tuning refers to the careful resizing of gates at transistor level, according to the needs of each signal path.)

The method described in this paper renders a stable circuit that may meet all specifications on the first approach, at the price of having added this extra margin in the gates themselves.

## 4. AN ILLUSTRATIVE EXAMPLE

### 4.1. *Wave-pipelining parallel multiplier*

Amultiplier was used to illustrate the concepts. It was implemented in a 1.2 V–0.18 $\mu$m CMOS process using a library of DRSRL-ID cells. The multiplier consists of three major blocks: the partial product generator (PPG), the partial

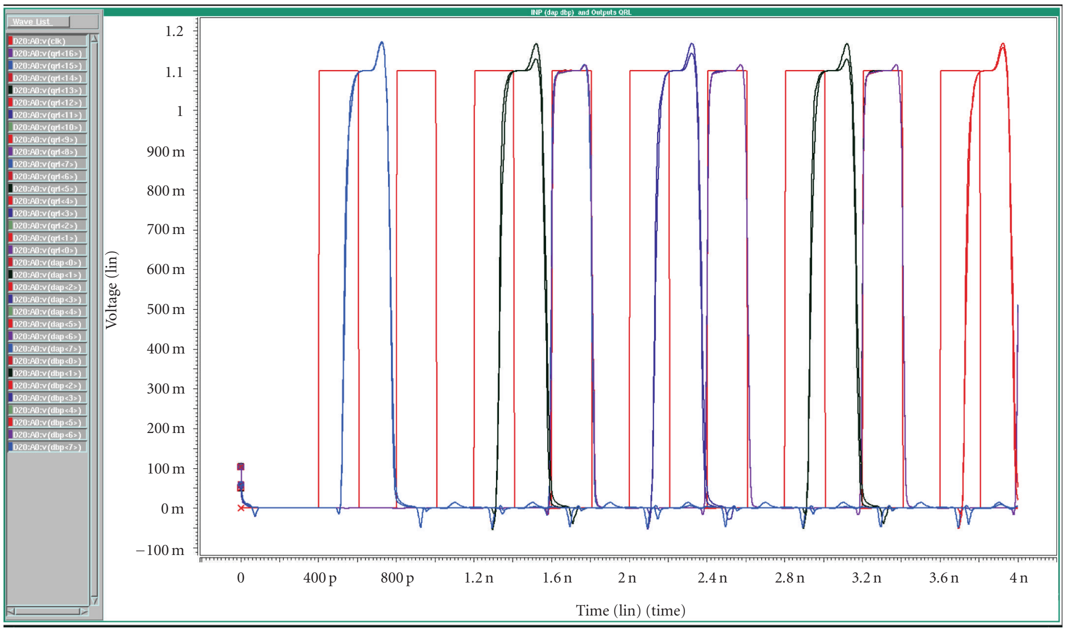


FIGURE 8: Advancing waves: *clk*, output: $qrl\langle 15:0\rangle$, and inputs $dap$ and $dbp$.

product reducer (PPR), and an adder (*carry propagate* adder). Figure 7 shows a block diagram.

In the first stage, the partial products (PPs) are generated. Each PP is the product of each bit of the multiplier by every bit of the multiplicand. Thus, for an $n \times n$ multiplication, $n$ PPs ($n$-bit wide) are generated. These PPs have to be added to obtain the final result.

The next stage is the partial product reducer (PPR), which reduces the $n$ PPs of an $n$-bit multiplier to two, hence the name of *reducer*. This is the main block of the multiplier, which we have implemented as a Wallace tree using *carry-save adders* (CSA).Timing of the CSA cell has been adjusted so that the delay forward of both outputs (S, Co) is approximately the same. The two final elements are added by means of an adder to generate the final result. We have used the *carry look-ahead* structure proposed in [5], with a slight modification to control the fan-out and the loading at critical points [2].

This block by itself is essentially asynchronous. We have added input and output registers for timing analysis when the multiplier is inserted in a synchronized pipeline. Theregisters were implemented by a set of edge-triggered flip-flops. The output register must sample the final stage of the adder while the result pulses are available. For this wave-pipelining application, all paths have been equalized using "rough padding," that is, adding buffers to the shorter paths to get the same number of stages in all cases.

To make the delays through the longest and shortest paths through the logic as close as possible, we tried to tightly control the difference in arrival times of all signals connected to a certain stage. This implies not only the delay equalization of the different elements of logic at a given stage, but also the delay equalization of the interconnects between successive stages and controlling the total loading of a given intermediate driver. A careful layout plan is important, but in the present design there is a certain tolerance level for differences in arrival times. This is true as long as one can guarantee that all valid input pulses at a given stage will overlap long enough to generate the output pulse within the time frame of valid inputs for the next stage.

### 4.2. *Simulation results and analysis*

Results of spice simulation of 8-bit multiplier, implemented in a 0.18 $\mu$m CMOS process, running at 2.5 GHz data rate, are shown in Figure 8. It can be observed that as the pulse waves advance through the stages of the multiplier, the timing difference among signals at a given stage is minimal, so they conform to a coherent data wave. Here, the following signals are depicted: the global ideal clock *clk*, the output $qrl\langle 15:0\rangle$, together with inputs $dap\langle 7:0\rangle$ and $dbp\langle 7:0\rangle$. Since the inputs shown leave a clock cycle in between, where all input bits are made zero, for clarity, it is easy to observe two nonzero input patterns, before the first output is shown. The pattern shown corresponds to decimal products: $(255\times255)$, $(3\times3)$, $(15 \times 15)$, $(3 \times 3)$, and $(63 \times 63)$, alternated with $(0 \times 0)$ for power analysis.

In Figure 8, the global clock signal is shown as reference; it is the almost perfect rectangular wave. At the end of its first pulse, we see the overlapping input signals, entering the multiplier. As mentioned above, in this case, inputs remain at zero, during the next clock cycle, and so we do not observe any other activity. At the next clock cycle, the 2nd set of nonzero inputs is applied, but only within the following pulse of the clock we see the first set of outputs (very close together, and almost completely enclosed within that clock pulse).

A new set of nonzero inputs will enter the pipelined circuit, before the multiplier outputs the result corresponding to the second nonzero set of inputs. Actually, the multiplier is able to calculate a new result at each clock tick; we have just interleaved a set of zero-valued inputs in between for clarity.

The maximum timing difference among output bits occurs in this design between bits $rl\langle 0\rangle$ (early arrival) and $rl\langle 9\rangle$ (late arrival) and is approximately 74 picoseconds = $\Delta T_P$. The maximum delay through the combinational logic $T_P$ is 978 picoseconds. The delay through the FF (timing difference between $dap\langle 0\rangle$ and *ckira* is 52 picoseconds = $t_d$.

Here, $T_{ck}$ = 400 picoseconds, that is, $F_{ck}$ = 2.5 GHz. Setup time $t_s$ = 20 picoseconds, and hold time $t_h$ = 50 picoseconds.

Looking at signals between different stages in the compressor, we have measured the following:

(i) $w_p$ = 210 picoseconds, $t_{\text{rec}}$ = 89 picoseconds,
(ii) $t_{df}$; = 101 picoseconds, at the slowest path, and
(iii) $\tau = w_p + t_{\text{rec}} + t_{df} := (210 + 89 + 101)$ picoseconds = 400 picoseconds.

Since the input register is always sending pulses, by means of the direct outputs Q or the inverse ones Qn, then the power consumption is average no matter what pattern is presented at the inputs. Whenever single rail implementations are possible, there will be power savings, since pulses will be generated at gate outputs only if input signals validate the gate logic function, but gate outputs will remain at zero otherwise.

It is worth noting that as the width of the multiplier grows, the total latency increases, but the data throughput remains unchanged, as far as we can control the wire loading, since the maximum operating frequency depends on the cycle time of the gates.

## 5. CONCLUSIONS AND PROPOSED FUTURE WORK

Wave pipelining is especially suitable for designs that show a high degree of parallelism and regularity. If that were not the case, the circuit has to be first transformed to achieve such parallelism.The design shown provides a practical proof of the feasibility of using the proposed technique in many applications, where pipelining is suitable. Wave pipelining provides savings in area and timing, since all intermediate storage elements are removed from the circuit, saving also from the point of view of timing overhead. The use of self reset logic provides savings in power and area with respect to a comparable CMOS dynamic implementation, since clock distribution for dynamic gates is avoided, as shown in [2], where a comparison was made between two implementations of an adder: domino logic versus DRSRL-ID. The use of dual-rail self reset logic with input disable functionality (DRSRL-ID) has additional advantages, providing a fairly constant pulse width, and in so doing avoiding "pulse-width adjusting structures" [6]. It provides an additional tolerance in the design for differences in arrival times of signals at any stage. While such tolerance is built-in in the structure of the gate family, it comes at the price of adding to the total cycle time and affects the minimum clock period $T_{ck_{\min}}$ used to pump in new data into the circuit. The reduction in area and power savings, plus the simplified equalization mechanism due to the built-in tolerance, makes this approach suitable for many fast processing designs.

Additionally, if we use as the last stage an SR-latch, which will only be updated each time new data has arrived, then, we are making the last stage "static," and in so doing, we can reduce the operating frequency as we need to interface with the next stage (moving the design from a $k$-wave mode to a single wave, if so needed). At the same time, we must maintain constraint (3) on the width of the input pulse to the first stage implemented with DRSRL-ID. The recommended approach would be to use a pulse generator, which will generate one pulse at the valid input clock transition.

The basic DRSRL-ID is suitable for structures with feedback, and this is an area we will investigate further.There is also special interest in asynchronous circuit applications. The DRSRL-ID application shown here uses the simplest protocol: "just sending data" and sacrifices elasticity for higher throughput. Many variations are possible according to circuit needs.

## ACKNOWLEDGMENT

M. E. Litvin is an IEEE regular member and S. Mourad is an IEEE Fellow. They thank Dr. Fabian Klass for his invaluable comments and advise during the research period.

## REFERENCES

[1] M. E. Litvin, "Wave pipelining with self reset logic," Doctoral dissertation, Santa Clara University, Santa Clara, Calif, USA, 2005.

[2] M. E. Litvin and S. Mourad, "Self-reset logic for fast arithmetic applications," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 13, no. 4, pp. 462–475, 2005.

[3] M. E. Litvin and S. Mourad, "Wave pipelining with self reset logic," in *Proceedings of IEEE International Conference on Electronic Circuits & Systems (ICECS '06)*, Nice, France, December 2006.

[4] E. F. Klass, "Wave pipelining theoretical & practical issues in CMOS," Doctoral dissertation, Stanford Univrsity, Stanford, Calif, USA, 1994.

[5] W. K. C. Lam, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Valid clocking in wavepipelined circuits," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD '92)*, pp. 518–525, Santa Clara, Calif, USA, November 1992.

[6] L. Wentai, C. T. Gray, D. Fan, W. J. Farlow, T. A. Hughes, and R. K. Cavin, "250-MHz wave pipelined adder in 2-$\mu$m CMOS," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 9, pp. 1117–1128, 1994.

Hindawi Publishing Corporation
VLSI Design
Volume 2008, Article ID 674259, 7 pages
doi:10.1155/2008/674259

*Research Article*

# Power Considerations in Banked CAMs: A Leakage Reduction Approach

**Pedro Echeverría, José L. Ayala, and Marisa López-Vallejo**

*Departamento de Ingeniería Electrónica, Universidad Politécnica de Madrid, 28040 Madrid, Spain*

Correspondence should be addressed to Pedro Echeverría, petxebe@die.upm.es

Received 26 April 2007; Accepted 9 December 2007

Recommended by Jean-Baptiste Begueret

The content-based access of CAMs makes them of great interest in lookup-based operations. However, the large amounts of parallel comparisons required cause an expensive cost in power dissipation. In this work, we present a novel banked precomputation-based architecture for low-power and storage-demanding applications where the reduction of both dynamic and leakage power consumption is addressed. Experimental results show that the proposed banked architecture reduces up to an 89% of dynamic power consumption during the search process while the leakage power consumption is also minimized up to a 91%.

## 1. INTRODUCTION

A broad range of modern applications demand large storage devices with fast search capabilities. Content-addressable memories (CAMs) have emerged as one of the favorite devices for such applications [1–3]. In CAMs data are accessed based on their content, rather than their physical address. This functionality has shown to be specially efficient in lookup-based applications like TLBs [1], associative computing and data compression [2]. High-speed networks such as gigabit Ethernet and ATM switches [3] also benefit from this particular structure [4].

However, CAMs pay a high hardware cost for this content-based access because the memory cell must include comparison circuitry, negatively impacting the size/speed tradeoff and complexity of the implementation. Usually, a 9-transistor cell is required instead of the 6-transistor cell used in SRAM. Moreover, the large amounts of parallel comparisons performed in conventional CAMs make the device consume too much power, preventing the implementation of large-scale CAMs in a single chip as the leading-edge applications demand.

In this paper, we present the implementation and characterization of a CAM with low-power constraints. The proposed architecture is highly scalable and provides high-performance functioning at large sizes. Moreover, it achieves great savings for both dynamic (89%) and leakage (91%) power consumption. Therefore, this architecture overcomes previous limitations of the CAM implementation and makes it suitable to all the applications where a high-performance low-power data search functioning is needed.

Previous work on CAM design has focused only on reducing the dynamic power consumption of the match line [5] and enhancing the search speed [6]. Although many approaches addressing dynamic power dissipation have been reported [7–9], the resulting circuit techniques have either substantial area overhead, deficiencies in noise immunity, or cannot be easily scaled without a negative impact on performance.

Our work overcomes these limitations by a novel and effective design of the CAM architecture and also addresses leakage energy reduction in an efficient way. The architecture presented here improves the energy savings obtained by Lin et al. in [10], and recently extended by Noda et al. [11] and Choi et al. [9]. These recent works provide a low power implementation of the CAM based on the precomputation of an index parameter. Nevertheless, they are constrained to specific small sizes, lack of scalability and present an increased search delay.

Moreover, the leakage energy consumption (which is one of the main issues regarding current electronic design) has not been addressed by previous approaches. Static power dissipation is becoming as important as dynamic dissipation as transistor gate sizes are being reduced [12]. Our work improves the dynamic power savings of the referred approaches and also reduces the leakage energy consumption of the memory to a minimum.

As stated before, this work is based on a parameter precomputation-based architecture [10] (PB-CAM from now on); however, we are able to reduce the parameter word's size with respect to [10], decreasing in this way the logic complexity, area, and power consumption related to this parameter. Moreover, the energy savings obtained with the proposed banked architecture (up to 89% of the dynamic power consumption and a 91% of the leakage power consumption) improve the previous implementations of similar technologies and also improve the scalability capabilities of architectures like [9, 11].

Our research work on this field has already shown good results in terms of area and dynamic power consumption [13, 14]. This paper expands [14] where an improved architecture was presented (with a novel hardware mechanism to reduce the static power consumption and increase the dynamic energy savings) with new experimental results and a deeper analysis of the consequences of applying leakage reduction techniques over CAM memories.

The paper is composed as follows. Section 2 presents the first designed approach where the minimization of the dynamic power consumption is addressed, while the leakage power reduction technique is shown in Section 3. The experimental results are introduced in Section 4. Finally, some conclusions are drawn.

## 2. DEVISED ARCHITECTURE: BANKED APPROACH

The first architecture presented in this section describes a banked implementation of the mentioned PB-CAM [10] in order to reduce the dynamic power consumption during the search operation. The main idea of the PB-CAM is to store a parameter word (obtained by a formula, e.g., a one's count) to perform the comparison process in a reduced number of memory positions, saving dynamic power consumption. However, the total power consumption of the logic can still be too high for low-power applications. Additionally, the PB-CAM architecture is also based on a novel seven-transistor data memory cell (that can only be used on conventional architectures) instead of the common one of nine or ten transistors, that can only be used on conventional architectures.

Our architecture employs the precomputed parameter to perform a power-aware ordering of the data. The extracted parameter allows us the classification of the memory contents based on the one's count. This classification can be used to store the memory contents in such an efficient way that the search operation is restricted to a smaller memory size.

The order of the memory data attending to the one's count parameter makes possible to split the memory architecture into independent banks where every data in the bank has the same value for one subset of the parameter (e.g., *N-least significant bits*). Moreover, the logic needed for this ordering is very simple and does not present a serious overhead in terms of delay and energy consumption [13].

In Figure 1 our architecture is depicted (RAM area corresponds to the memory where the output address associated to each tag will be found to complete a CAM-search engine). It can be observed that each memory word is composed of a validation bit, the data word and part of its parameter. The parameter extractor computes the parameter of the input tag and a bank decoder selects the proper bank with a subset of the bits of the calculated parameter. Then, the rest of the parameter bits and the input data are searched in the decoded bank.

Due to the banked implementation of the memory, the operation of the architecture is restricted to just one bank every cycle. One of the advantages of this banked structure is the reduction of the dynamic power consumption as the charge in the bit lines is limited to one bank (the driven line is simplified to the bit line of the accessed bank of the memory). This behavior is also shown by the parameter lines and also has a positive influence in the memory speed. The complexity of the logic shared for the banks (buffers, priority encoders, and address decoders) is reduced when the bank approach is applied. This simplification saves area, power consumption and improves the delay of these devices.

## 3. DEVISED ARCHITECTURE: DROWSY APPROACH

The banked PB-CAM presented in Section 2 has been improved to reduce the static power consumption of the architecture. In this case, the unused memory banks are put into a low-power state and a pipeline carefully manages the operations to avoid any performance penalty. This approach is based on the design of a low-power cell and a further pipelined subbanked implementation.

### 3.1. Low-power cells (data and parameter)

As has been previously described, the memory area has been split into several banks, which can be independently accessed. Then, a dynamic voltage scaling (DVS) technique is applied to turn the unused banks into a low-power state and thus save as much energy as possible in the system. However, when the objective is a memory device, the cost of recovering the lost information could hide any power saving or, at least, represent a very significant time penalty. Moreover, something has to be done in the powered-down banks to prevent the information from being lost.

An efficient approach to achieve the low-power (drowsy) state is proposed by [15], where a DVS technique is exploited to reduce static power consumption. As is well known, both dynamic and static power consumption are proportional to the supply voltage. DVS technique benefits of this fact by turning down the power supply to reduce power consumption. However, reducing the supply voltage also has a negative impact in other parameters such as speed, hence DVS techniques look for combining different voltages while not affecting other parameters.

The method proposed by Flautner et al. implements DVS to reduce the leakage power of cache cells by scaling the voltage of the cell to a lower voltage that ensures the preservation of the state of the memory cells. This voltage can be conservatively approximated to 1.5 times $V$th [15], but further

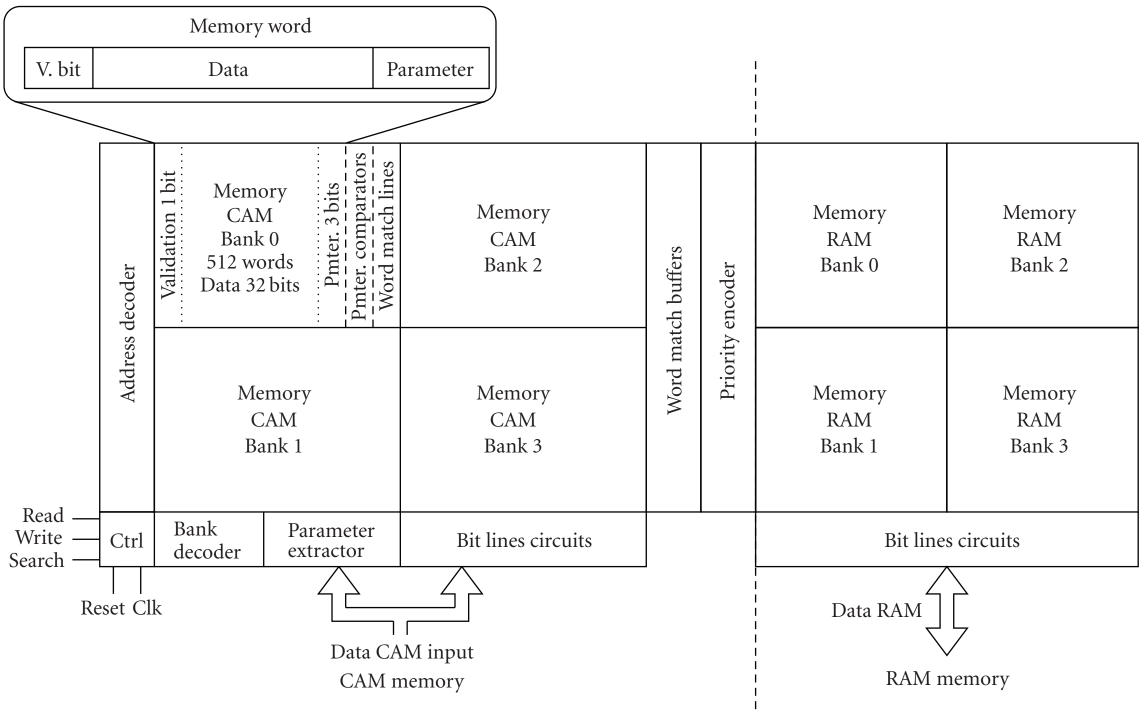


Figure 1: Banked architecture (4-bank implementation).

reductions of the scaled voltage would increase static power savings [16].

Figures 2 and 3 show the modified memory cells (data and parameter) used to support the drowsy state with a dual power supply (not part of the cell, it is shared by all the cells in a same bank). As can be observed, the dual power supply is switched to low $V_{DD}$ when the cell is in drowsy state. It is necessary to use *high-V*th devices as pass transistors because the voltage on bit lines could destroy the cell contents. Before a memory position can be accessed, the power supply has to be switched to high $V_{DD}$ (wake up) to restore the contents and allow the access. The careful management of these operations along the pipeline presented in the next section takes care of this extra clock cycle to avoid any performance penalty.

As mentioned before, each bank of the CAM architecture counts with the additional logic required to implement the DVS mechanism. Since the low-power consumption state is selected for the whole bank instead of a specific memory position, the overhead of the control logic is greatly minimized.

### 3.2. Pipeline

A clear way to improve the access time of a CAM is the use of a pipeline structure, which additionally provides greater scalability in the performance and density of the applications that make use of CAMs. The aforementioned DVS technique can take advantage of this pipeline to awake the drowsy cells one clock cycle before the access. Only one of the banks needs to be on while the rest can remain in the drowsy state saving leakage energy.

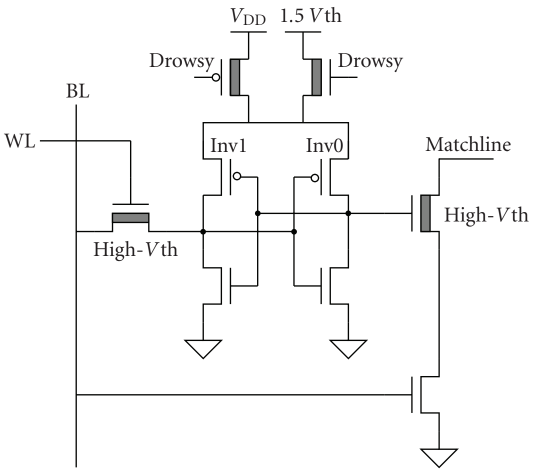


Figure 2: 7-transistor CAM cell with drowsy support.

In our approach, the devised pipeline configuration includes the three operations needed in a CAM-search engine: READ, OVERWRITE, and WRITE. READ is the read operation in the associated RAM memory after the tag is found in the CAM. In OVERWRITE, after the tag is found in the CAM, a write operation is done in the RAM memory, and WRITE is the operation to write a tag and its data in both CAM and RAM memories.

The pipeline stages defined within those operations are EXT (parameter extraction of the input tag and selection of the working bank), SEARCH (tag comparison in the CAM), DEC (decodification of internal address, common for both RAM and CAM), READ (only in RAM memory), and WRITE (in both memories or only in RAM):

(i) *READ operation:* EXT–SEARCH–READ_R,
(ii) *OVERWRITE operation:* EXT–SEARCH–WRITE_R,
(iii) *WRITE operation:* EXT–DEC–WRITE_CR.

However, this three-stage pipeline shows a structural and data hazard, as shown in Figure 4(a). The stages and the resources used in each stage (parameter extractor, address decoder, CAM, and RAM memories) are shown in the plot. This hazard is produced in the CAM structure between the READ (or OVERWRITE) operation and the WRITE operation because the CAM area is simultaneously accessed by the second and third stages, respectively. This problem can be solved by including a fourth pipeline stage splitting the WRITE operation into WRITE_C and WRITE_R and introducing a "no operation" stage, NOP (see Figure 4(b)). All the CAM accesses are in the third stage and the RAM accesses in the fourth:

(i) *READ:* EXT–NOP–SEARCH–READ_R,
(ii) *OVERWRITE:* EXT–NOP–SEARCH–WRITE_R,
(iii) *WRITE:* EXT–DEC–WRITE_C–WRITE_R.

The second stage would mean a cycle delay in the READ and OVERWRITE operations, but we take advantage of this cycle to wake up the memory cells of the CAM memory from the drowsy state. Therefore, the pipeline is not stalled and the performance is not compromised.

The throughput of the pipeline is set by the slowest stage, the third one in a READ or OVERWRITE operation with the SEARCH stage, where the parallel access to all the words in a bank of the CAM memory is carried out.

### 3.3. Banks subdivision

Once the leakage current control mechanism has been exposed, the natural goal consists in increasing the expected energy savings. The simplest idea is to divide the memory in as many banks as possible, using more parameter bits to decode the active bank. This technique presents the same advantages as those mentioned in Section 2.

However, dividing the memory in so many banks has two very important drawbacks. Firstly, the unbalanced use of the banks which means an increase in the failed search rate of the memory (number of times that a searched data is not in the memory and has to be written in the memory before a new search). The 4-bank implementation in Section 2 presents a homogeneous use of the banks (each bank with an almost exact 25% distribution of the input tags) but if a third bit is introduced to split the CAM into 8 banks, the distribution of inputs varies from 14.48% to 10.52% for the one's count parameter (that is a difference of 27.4% between the most and the least used bank). The second drawback is the complex layout that will require the memory, due to the common elements of the banks.

Therefore, another technique has been devised to preserve the homogeneous use of the banks and a realistic layout: the subdivision of each bank into a set of subbanks. The main idea is to combine the parameter decoding with a new

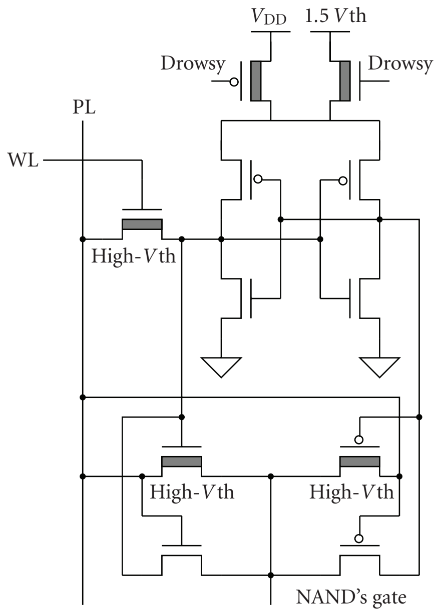


Figure 3: Parameter CAM cell with drowsy support.

ordering of the input tags, using in this case the value of some bits of the input tags. In this way, the tags found in the same bank are ordered in local subbanks attending to some bits (the tags belonging to the same subbank share the same value of some bits). This mechanism obtains a very homogeneous use of the subbanks without impacting the layout.

For example, in the previous 4-bank implementation, using any two bits of the tag to enable the bank subdivision, there will be 4 subbanks per bank (those banks correspond to the values 00, 01, 10, and 11 of any two tag bits), as depicted in Figure 5. Unlike the 8-bank implementation, this 4-bank configuration with subbanking presents a very homogeneous use of the subbanks, with only 0.013% of maximal difference between subbanks. Moreover, the obtained layout remains without appreciable changes when the subbanking approach is applied.

One of the key advantages of this subbanking technique is that any memory operation will be done only in the proper subbank of the decoded bank, while the other subbanks of that bank as well as the other banks will remain at the drowsy mode. Moreover, there are also dynamic power and area advantages of this technique very similar to the ones presented for the bank implementation. For example, the tag bits used for the subbanking do not need to be stored. Also, the complexity of the common logic (address decoder and priority encoders) can be simplified by designing a single element for the subbank and sharing this design for every subbank in the architecture. And finally, the power consumption of the comparison operation is restricted to the working subbank, which increases the savings in this factor.

## 4. EXPERIMENTAL RESULTS

Our experiments have been carried out with Spice simulations in the Cadence environment. The technology used

| Instruction | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Read | EXT | SEARCH | READ_R | | | |
| | Pmt. Extr. | C. Mem | R. Mem | | | |
| Write | | EXT | DEC | WRITE_CR | | |
| | | Pmt. Extr. | Addr. Dec | C. Mem R. Mem | | |
| Overwrite | | | EXT | SEARCH | WRITE_R | |
| | | | Pmt. Extr. | C. Mem | R. Mem | |

(a) 3-stage pipeline

| Instruction | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Read | EXT | NOP | SEARCH | READ_R | | |
| | Pmt. Extr. | | C. Mem | R. Mem | | |
| Write | | EXT | DEC | WRITE_C | WRITE_R | |
| | | Pmt. Extr. | Addr. Dec | C. Mem | R. Mem | |
| Overwrite | | | EXT | NOP | SEARCH | WRITE_R |
| | | | Pmt. Extr. | | C. Mem | R. Mem |

(b) 4-stage pipeline

Figure 4: Structure of the proposed pipeline.

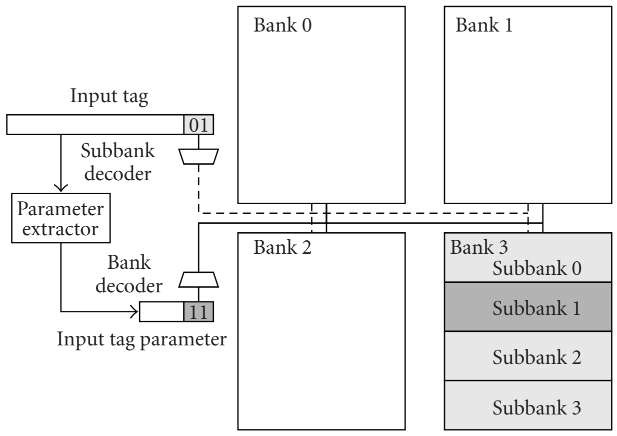


Figure 5: Subbanking scheme.

Table 1: Drowsy PB-CAM with two different scaled voltages.

| Voltage | 1.5 $V$th | 1.25 $V$th |
|---|---|---|
| Drowsy state; leakage reduction | 92% | 98% |
| Leakage savings | 86% | 91% |
| Dynamic wake-up penalty | 2.77 f J/bit | 3.06 f J/bit |

to implement the designed architecture has been 0.35 $\mu m$ from Austria MicroSystems (as the base architecture is also a 0.35 $\mu m$ design), while the estimation of the leakage energy savings has been carried out using the 70 nm BPTM [17] models, as the leakage power cannot be studied with the technology selected for the implementation.

### 4.1. Banked implementation

The banked architecture has been firstly evaluated in terms of the energy savings obtained after reducing dynamic power. The simulated memory is the architecture described in Figure 1, implemented as a memory of 2048 positions and 32 bits per word, and split into 4 independent banks. Our approach decreases the dynamic power consumption by a 78% (18.86 f J/bit in the banked architecture with respect to 86 f J/bit in [10]).

The area improvement achieved with the proposed architecture has also been evaluated in terms of number of transistors. When the memory size is fixed and the number of bits per word is varied, there is a reduced area improvement of the banked implementation with respect to the original PB-CAM due to the savings in the parameter word length and comparison logic. Compared to a traditional implementation, area savings are quite representative for architectures with more than 16 bits (up to 17.5% in the range considered: 8 to 128 bits per word and a 2048 words memory size) due to the use of 7-transistor memory data cells instead of the 9-transistor one of a traditional architecture.

For the implemented architecture, the area savings obtained are 13.5% with respect to a traditional architecture and 10.7% respect to the base PB-CAM.

Finally, the performance of the design has been analyzed to assure the required fast response. The results show a 7.5 ns delay for the search operation, which also includes the data write into a RAM memory. The comparison of these performance results with the ones described by Lin in [10] shows how the operation time in the banked architecture is a 25% faster than in the original PB-CAM (10 nanoseconds).

### 4.2. Drowsy implementation

Regarding the energy savings obtained, this approach presents further improvements both in dynamic and leakage energy due to the subbanking and drowsy techniques. The simulated architecture is a CAM with 8192 positions (notice the larger implementation with respect to the baseline

TABLE 2: Comparison of the three approaches (simple, banked, and drowsy PB-CAM [10]).

| | PB-CAM [10] | B. PB-CAM | D. PB-CAM |
|---|---|---|---|
| Technology | 0.35 $\mu m$ | 0.35 $\mu m$ | 0.35 $\mu m$ |
| CAM configuration | $128 \times 30$ | $2048 \times 32$ | $8192 \times 32$ |
| Dynamic power | 86 fJ/bit | 18.86 fJ/bit | 9.06 fJ/bit |
| Operation delay | 10 ns | 7.5 ns | 12 ns |
| Throughput | 100 MIPS | 133 MIPS | 333 MIPS |
| Static power reduction | — | — | 86%–91% |

architecture) and 32 bits per word, implemented with 4 independent banks, 4 subbanks per bank, and the described 4-stage pipeline.

With this design, the dynamic consumption is reduced to 9.06 fJ/bit (a decrease of an 89% and 52% with respect to the baseline architecture and our first approach, resp.) when the subbanking technique is used without the drowsy technique.

To apply the drowsy technique for leakage reduction, we have considered two scaled voltages, the one referred as the conservative approach [15] (scaled voltage 1.5 $V$th) and a smaller one [16] (1.25 $V$th). The BPTM simulations estimate a 92% of leakage energy savings for the memory cells at a low power state for the conservative voltage, that is increased up to a 98% for the second voltage. Given that the working subbank has to remain at full voltage, the total power consumption, for the simulated architecture, is reduced an 86% and a 91%, respectively.

However when the drowsy technique is applied, the waking up of the accessed subbank is a new source of dynamic power consumption. The total value of this dynamic consumption depends on the scaled voltage used for the low-power state, but in the opposite way for the leakage consumption. When the scaled voltage is reduced, the leakage power consumption is decreased while the waking up dynamic power consumption is increased (the swing voltage between low-power state and working state is increased).

If we extrapolate BPTM simulations to the 0.35 $\mu m$ simulated architecture and the two proposed voltages, the waking up of a subbank means an additional dynamic power consumption of 2.77 fJ/bit for the 1.5 $V$th and 3.06 fJ/bit for the 1.25 $V$th. In Table 1 the experimental results for the two voltages are summarized.

These energy savings can be easily increased if an architecture with more banks or more subbanks per bank is selected. As stated before, and as can be seen in Figure 6, using an 8-banks implementation unbalances the usage of banks and increases the failed search rate, due to the different probabilities of each parameter. While for parameter zero or 32 there is only one possible input data, there are 601 hundred million different input data for parameter 16.

However, using more subbanks per bank has a very weak impact in that factor because all the input data bits have the same probability (and so the ones used to select the subbank), as can be seen in Figure 7. For a 4-bank implementation, the percentage of use between the most and the least used bank is 0.006%. When subbanks are introduced, it can be observed how that percentage duplicates when we quaduplicate the number of subbanks, and has a very small value for a very wide range of subbanks.

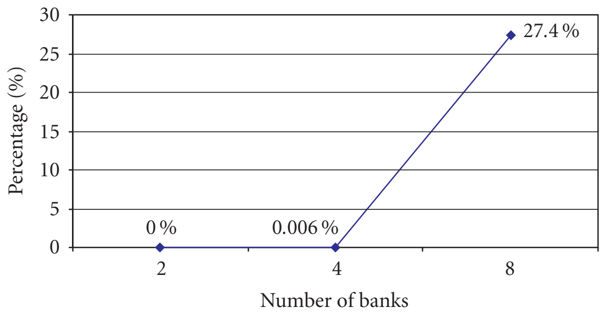


FIGURE 6: Unbalance between banks (no subbanking).

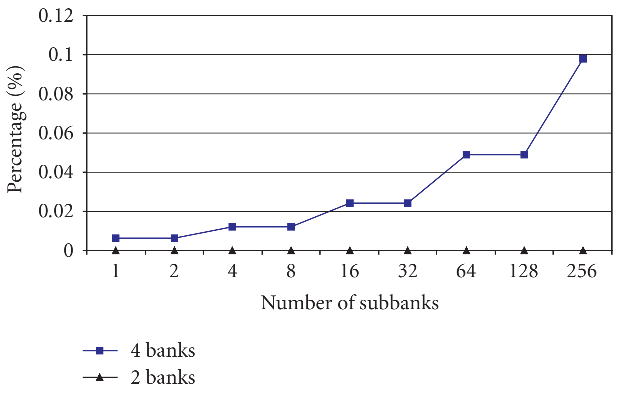


FIGURE 7: Unbalance between subbanks.

In this way, using more subbanks only increases slightly the unbalance between banks. However, the energy saved while using more subbanks is considerable, as the active part of the memory is smaller: for dynamic power consumption there is less comparison power consumed during a search and less wake up power consumption, and for leakage power more parts of the memory can stay at a low-power state.

The drowsy implementation with subbanking also presents area improvements because two less memory cells per data are stored, while the overhead of the DVS control has a very small impact given that it is shared by all the cells on the same subbank. In Figure 8 it can be observed the area savings when the word length is fixed (32b) and the memory size is ranged. For our simulated architecture we obtained an improvement of 16.1% with respect to the traditional

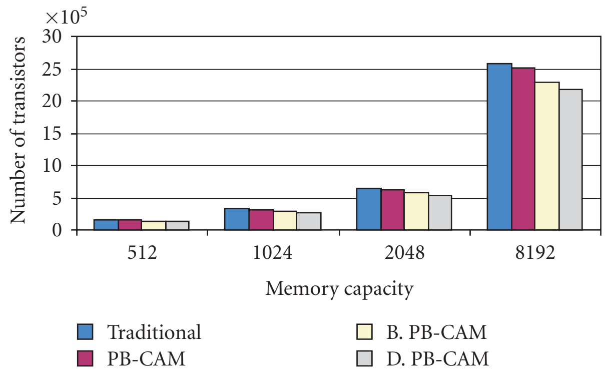


Figure 8: Number of transistors for a fixed word length.

implementation, 13.4% respect the base PB-CAM and a 5% compared with our banked implementation.

The throughput of the system is also improved (70% when compared with the baseline architecture, 60% when compared with the banked PB-CAM) due to the pipeline. A comparison of the two approaches presented in this paper, as well as the baseline architecture, can be found in Table 2.

## 5. CONCLUSIONS

Nowadays, the limiting factor in applications where the CAMs play a critical role is the power consumption of these devices. The integration levels achieved by current technology processes have turned the area and performance factors into secondary actors. Search-based applications with high-performance constrains demand efficient implementations of content-addressable memories to cover the astringent constraints. The work presented in this paper has shown efficient mechanisms to reduce the dynamic and static power consumption by means of hardware modifications. These approaches do not compromise the performance and area improvements achieved with the architecture.

## ACKNOWLEDGMENT

This work was supported by the Spanish Ministry of Science and Education under Contract TEC2006-00739.

## REFERENCES

[1] S. Swaminathan, S. B. Patel, J. Dieffenderfer, and J. Silberman, "Reducing power consumption during TLB lookups in a PowerPC/spl trade/ embedded processor," in *Proceedings of the 6th International Symposium on Quality of Electronic Design (ISQED '05)*, pp. 54–58, San Jose, Calif, USA, March 2005.

[2] K.-J. Lin and C.-W. Wu, "A low-power CAM design for LZ data compression," *IEEE Transactions on Computers*, vol. 49, no. 10, pp. 1139–1145, 2000.

[3] Y. Tang, Y. Jiang, and Y. Wang, "CAM-based label search engine for MPLS over ATM networks," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM '01)*, vol. 1, pp. 45–49, San Antonio, Tex, USA, November 2001.

[4] H. Liu, "Reducing routing table size using ternary-CAM," in *Proceedings of the 9th Symposium on High Performance Interconnects (HOTI '01)*, pp. 69–73, Stanford, Calif, USA, August 2001.

[5] I. Arsovski and A. Sheikholeslami, "A current-saving match-line sensing scheme for content-addressable memories," in *Proceedings of IEEE International Solid-State Circuits Conference (ISSCC '03)*, vol. 1, pp. 304–494, San Francisco, Calif, USA, February 2003.

[6] H. Miyatake, M. Tanaka, and Y. Mori, "A design for high-speed lowpower CMOS fully parallel content-addressable memory macros," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 6, pp. 956–968, 2001.

[7] I. Arsovski and A. Sheikholeslami, "A mismatch-dependent power allocation technique for match-line sensing in content-addressable memories," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 11, pp. 1958–1966, 2003.

[8] K. Pagiamtzis and A. Sheikholeslami, "Pipelined match-lines and hierarchical search-lines for low-power content-addressable memories," in *Proceedings of the IEEE Custom Integrated Circuits Conference (CICC '03)*, pp. 383–386, San Jose, Calif, USA, September 2003.

[9] S. Choi, K. Sohn, and H.-J. Yoo, "A 0.7-fJ/bit/search 2.2-ns search time hybrid-type TCAM architecture," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 1, pp. 254–260, 2005.

[10] C.-S. Lin, J.-C. Chang, and B.-D. Liu, "A low-power precomputation-based fully paralel content-addressable memory," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 4, pp. 654–662, 2003.

[11] H. Noda, K. Inoue, M. Kuroiwa, et al., "A cost-efficient high-performance dynamic TCAM with pipelined hierarchical searching and shift redundancy architecture," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 1, pp. 245–253, 2005.

[12] N. S. Kim, T. Austin, D. Blaauw, et al., "Leakage current: Moore's law meets static power," *Computer*, vol. 36, no. 12, pp. 68–75, 2003.

[13] P. Echeverría, J. L. Ayala, and M. López-Vallejo, "A banked precomputation-based CAM architecture for low-power storage-demanding Applications," in *Proceedings of the 13th IEEE Mediterranean Electrotechnical Conference (MELECON '06)*, pp. 57–60, Malaga, Spain, May 2006.

[14] P. Echeverría, J. L. Ayala, and M. López-Vallejo, "Leakage energy reduction in banked content addressable memories," in *Proceedings of the 13th IEEE International Conference on Electronics, Circuits and Systems (ICECS '06)*, pp. 1196–1199, Nice, France, December 2006.

[15] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. N. Mudge, "Drowsy caches: simple techniques for reducing leakage power," in *Proceedings of the 29th Annual International Symposium on Computer Architecture (ISCA '02)*, pp. 148–157, Anchorage, Alaska, USA, May 2002.

[16] N. S. Kim, K. Flautner, D. Blaauw, and T. N. Mudge, "Single-VDD and single-VT super-drowsy techniques for low-leakage high-performance instruction caches," in *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED '04)*, pp. 54–57, Newport, Calif, USA, August 2004.

[17] http://www.eas.asu.edu/ptm/.