COMMON OPERATORS DESIGN ON DYNAMICALLY RECONFIGURABLE HARDWARE FOR SDR SYSTEMS

Loïg GODARD, Hongzhi WANG, Christophe MOY, Pierre LERAY
IETR/Supelec-SCEE Laboratory, Campus of Rennes
{loig.godard, hongzhi.wang, christophe.moy, pierre.leray}@supelec.fr

ABSTRACT

This paper deals with *common operators* used to perform multi-standard, multi-function applications. In opposition with highly complex communication components, each exclusively dedicated to a given standard ("Velcro" approach), we argue that one of the key point in software radio is the (re)use of modular operators as proposed in [1]. Those operators can adapt their behavior according to the function to realize by parameters change. This indicates the need to dynamically adapt systems architecture at the hardware level. Taking into account the possibility of partial reconfiguration offered by the FPGA [2], we can reconfigure *common operators* while the rest of the design is still running. *Common operators* approach also offers possibility to use reconfiguration by difference that aims at reducing bitstream size and then decrease the cost of the reconfiguration in terms of memory and time allocation.

1. INTRODUCTION

Software radio basically refers to a collection of techniques that permit the reconfiguration of communication systems without changing any hardware part. We can see then that one of the challenges of future generation of communication systems is the reconfiguration of the device. This reconfiguration may be needed at highest level of the system as at lowest level. Highest has to be understood here like a complete reconfiguration of a system (for a standard change for example). Lowest level represents local reconfiguration (like bug fixing).
The topic of this paper is to give an overview of the *common operators* approach on dynamically reconfigurable platform and show examples of implementation of such operators. As we will see, we argue that such operators offer an answer to successful reconfiguration of SDR systems. Indeed, in opposition with static communication component, the (re)use of *common operators* which can adapt their behavior according to functions needs are key points in software radio architectures. Parameterization studies become a very important issue in this context, mainly because it decreases the size of the software to be downloaded, and also because it shortens the runtime of the software reconfiguration. Moreover this type of technique optimizes the sharing between the software and the hardware of the execution platform.
The following part shows how a *common operators* approach aims at improving multi-standard SDR systems operation in terms of reconfiguration time and performance. Then we introduce two commonly used operators in SDR systems which are FIR filter and CORDIC. From this we extract parameters that can be used to design those operators on *common operators*. The final part of this paper introduces an example of CORDIC implementation with a *common operators* approach in a MIMO context. Then we conclude on the result of this work.

2. COMMON OPERATORS APPROACH

Common operators approach aims at, independently of the standard, researching the minimum number of highest level operators, which are used in a maximum of functions of a maximum of standards implemented in a unique device. As standards and functions proposed for communication systems grew up drastically, it can be derived it may be usefull to have a maximum reuse of processing elements. *Common operators* approach gives an interesting opportunity to perform the maximum reuse of hardware part for multi-standard, multi-function systems. This approach aims at increasing the granularity of classical arithmetic operators in the context of multi-standard SDR systems design. Fig. 1 shows how those *common operators* can be identified. Reference [3] proposes a formal method to find the optimal *common operators* for a given multi-standard system by a graph optimization.

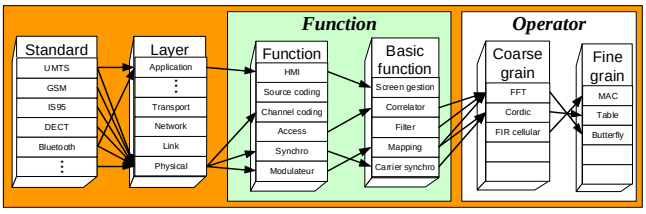

Fig.1 *Common operator*s definition

The main characteristic of a *common operator* is its ability to change its behavior by parameterization in order to realize different operations at different times. It is important to

notice that, as such operator has to be generic enough to be reused in different contexts, it will not have the same performance as a dedicated one.
Make an operator generic also offers the possibility to consider on-the-fly reconfiguration. With simple and quick parameterization indeed a *common operator* can change its behavior and give the possibility to perform a reconfiguration in a short time. Nevertheless as we will see in this document this approach offers a good trade-off in terms of complexity and cost (in area occupation and resource).

3. FINITE IMPULSE RESPONSE (FIR) OPERATOR

This part presents a FIR filter *common operator* architecture for software radio applications. This filter is designed for a FPGA implementation in order to take advantage of reconfiguration possibilities of this technology. Indeed in a software radio context, systems may have to dynamically adapt architecture at the hardware level. Furthermore, we will see that this approach, combined with partial reconfiguration, aims at coping with on-the-fly reconfiguration.

3.1 Classical Approach

A FIR is a well known operator for the implementation of filtering functions. Fig.2 shows a representation of FIR function.

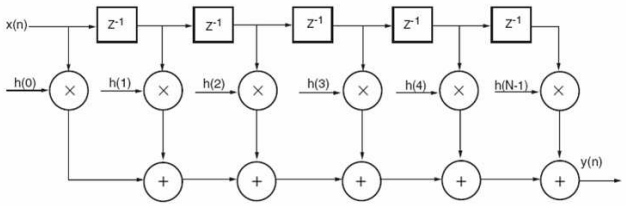

Fig.2 Finite Impulse Response (FIR) filter

Mathematical expression is given by the following equation:

$$y(n) = \sum_{k=0}^{k=N-1} h(k)x(n-k)$$

There are many FIR's architecture in the literature, each of them specialized in a certain context. We can find optimization in speed computation, in area occupation or whatever. The coefficients choice shapes the filter frequency response. In a classical approach those coefficients are set once at design time and can not be changed. Then if the system needs to reconfigure itself, it has to erase previous filter and implement a new one. As we can imagine, on the one hand this can not cope with on-the-fly reconfiguration as this would require too much time for such an amount of data. On the other hand memory resources needed to store FIR IP with different coefficients and architectures to face to standard change is relatively important. From this point, we can imagine a FIR that can reconfigure its coefficients and behavior (tap number, filter size or over sampling factor). It is important to notice that this kind of architecture is still generic. Indeed our goal is to realize an operator with re use facility.

3.2 Design Methodology

In a software radio context, architecture has to be reconfigurable, but the ways to implement it may vary. This reconfiguration can be made at system level (for standard change for example) or at local level (behavior change of an operator). We investigate the second proposition here.
A FIR operator has been designed with a static and a reconfigurable part. Its static part is a computational part that does not need to be reconfigured, and control is located in the reconfigurable part. FIR control is managed by a finite state machine (FSM) that can handle numerous configurations of filters. Keep also in mind that FIR IP can be duplicated and pipelined in order to realize more powerful and complex FIR functions.

3.3 FIR Design

FSM has two main entries: number of taps (modulo 4) and filter size (number of coefficients, modulo 8). Coefficients are stored into Block SelectRAM configured as ROM. This solution decreases risks of erroneous FIR response inherent to addressing problems after a reconfiguration step. Moreover we take advantage here of reconfiguration capacity of FPGA by decreasing logic control. We choose a dynamic of 8 bits for data and coefficients for this IP, which can cope with many situations. Fig.3 shows a representation of the IP architecture.
A key point for reconfiguration is that as filter coefficients are stored in Block SelectRAM, a partial reconfiguration with a reconfiguration by difference can be done. The use of such reconfiguration aims on one hand at decreasing bitstream size for reconfiguration, and on the other hand at authorizing on-the-fly reconfiguration.

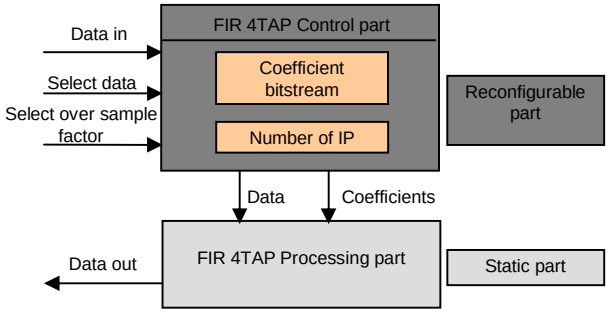

Fig.3 FIR architecture

In a running system indeed, as only a sub-part of the FPGA (Block SelectRAM for coefficients) is reconfigured, the rest of the FPGA is still in operation. Reconfiguration by difference methodology aims at generating a bitstream underlying only the changes done between the previous and the current configuration. In this case, for coefficients change, only some memory slices are changed, so that the bitstream generated by this method is drastically smaller in size than original one.

This architecture has been implemented on a *VIRTEX II*. Configuration was one IP FIR_4TAP, in order to perform RRC filter for UMTS, with a set of coefficients of 32 and an over-sampling factor of 2 and 4 which are classical values to reach the performance for UMTS. For this configuration, partial reconfiguration bitstream size obtained by difference is 2 KB, whereas a usual partial bitstream is near from 850 KB. Repercussion of such a size reduction is a diminution of resource occupancy and also a reduction of time reconfiguration (depending on the communication bus used for the design). It is not in the scope of this paper to expose the complete methodology to design such an operator but it's an important part of the work needed to perform this realization [4].

4. CORDIC OPERATOR

This part presents CORDIC (COordinate Rotation Digital Computing) concept that allows the computing of elementary operations such as products, divisions and trigonometric functions. It performs rotations without using multiplication operations.

4.1 Principle of the CORDIC algorithm

In the CORDIC concept [5], a rotation of 2-D vector is performed with a required angle φ decomposing this into a sum of micro-rotations of elementary angles φ_i expressed as values depending on the *i*-th power of 2 that can be performed by hardware through simple shift-add operation. The result is more and more accurate as the number of iterations n increases since the vector orientation is successively closer to its target.

The CORDIC algorithm for trigonometric computing is defined by the equations:

$$x_{i+1} = x_i - d_i \,.\, dy_i \qquad \text{where } dy_i = y_i \times 2^{-i}$$
$$y_{i+1} = yi + d_i \,.\, dx_i \qquad dx_i = xi \times 2^{-i}$$
$$a_{i+1} = a_i - d_i \,.\, da_i \qquad da_i = tan^{-1}(2^{-i})$$

Where *x* and *y* are the coordinates of the vector as shown in Fig. 4, *a* is the angle accumulator that stores the effective rotation, and *d* is the sign of rotation. The general principle of the CORDIC algorithm consists of making the rotation vector turn in the appropriate direction by an increasingly small angle until the angle *a* or the values *x* and *y* are approximately equal to 0. Furthermore, a CORDIC micro-rotation is a rotation with an intrinsic increase gain of the magnitude *r* of the vector quantified by the factor A:

$$A_n = \prod_n (1 + 2^{-2i})^{1/2}$$

The CORDIC method can be employed in two different modes, known as the "rotation" mode and the "vector" mode. In the rotation mode, the co-ordinate components of a vector and an angle of rotation are given and the co-ordinate components of the original vector, after rotation through a given angle, are computed. In the vector mode, the co-ordinate components of a vector are given and the magnitude and angular argument of the original vector are computed.

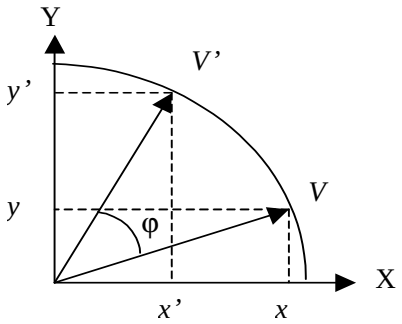

Fig. 4 Rotation of vector V in the Cartesian plane

4.2 CORDIC operator architecture

An architecture of CORDIC operator is proposed by Rader [6]. It's a simple and effective method for calculating a range of complex functions, which relies on a technique of additions and shifters. The CORDIC operator calculates most trigonometric-based functions by approximation. The iterative structure of the CORDIC algorithm makes possible an implementation using the pipeline structure of Fig. 5 thus limiting critical path length in such a way as to speed up operation.

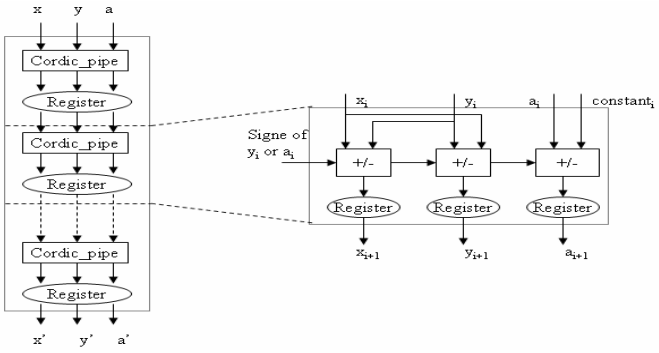

Fig. 5 CORDIC operator pipeline architecture

5. SCENARIO OF RECONFIGURATION OF CORDIC OPERATOR

This part presents a scenario of reconfiguration involving CORDIC operator. It highlights both application and implementation points of view in the context of a Multiple-Input Multiple-Output (MIMO) system. MIMO is one of the most promising technologies to enhance the wireless communications performances because of the increase in terms of bandwidth capacity it may provide [7]. In the various MIMO detection algorithms, V-BLAST square root decoder is an interesting trade-off to obtain a high performance with a reasonable complexity.

5.1. Description of the "V-BLAST Square Root" algorithm

The "V-BLAST Square Root" algorithm is proposed by B. Hassibi [8]; it avoids the repeated calculation of the pseudo-inverse of the channel matrix, as well as the matrix inversion by using unitary transformations. It makes possible the reduction of the calculation load from $O(M^4)$ to $O(M^3)$ without degrading the BER. To do this, B. Hassibi uses a recurrence relationship well known in adaptive *RLS* (Recursive Least Square) filtering. He demonstrates that if one applies a Givens rotation sequence to the recurrence relationship, one obtains $P^{1/2}$ after i iterations. But it still remains to calculate Q_α. The best solution will be to apply a relationship that provides $P^{1/2}$ and Q_α at the same time. This is why B. Hassibi puts forward a new recurrence relationship starting from the preceding matrix block, to which he adds a block vector.

The algorithm is summarized below:

Step 1: Calculation of $P^{1/2}$ and Q_α,

- Initialization

$$\begin{bmatrix} 1 & H_i^{1\times M} P_{i-1}^{1/2M\times M} \\ 0^{M\times 1} & P_0^{1/2M\times M} \\ -e_i^{N\times 1} & Q_0^{N\times M} \end{bmatrix} \text{ with } P_0^{1/2M\times M} = \beta\, I \ , \ Q_0 = 0^{N\times M} \quad (1)$$

- for $i = 1$ to N

$$\begin{bmatrix} 1 & H_i^{1\times M} P_{i-1}^{1/2M\times M} \\ 0^{M\times 1} & P_{i-1}^{1/2M\times M} \\ -e_i^{N\times 1} & Q_{i-1}^{N\times M} \end{bmatrix} \Theta_i = \begin{bmatrix} \times & 0^{1\times M} \\ \times & P_i^{1/2M\times M} \\ \times & Q_i^{N\times M} \end{bmatrix} \quad (2)$$

End

After N iterations one obtains $P^{1/2}{}_N = P^{1/2}$, $Q_N = Q_\alpha$,

Q_i representing the i^{th} iteration with Q_0 initialized to 0,

e_i indicates the i^{th} column of the identity matrix,

Θ_i corresponds to a unitary transformation, which transforms the matrix of equation (2) into a lower triangular matrix. The methods for finding this type of unitary transformation are well known [8].

Iteration: For $i=0$ to $M-1$

Step 2: Determine the minimum norm of the lines of $P^{1/2}$ and permutate this line so that it becomes the last one. This means to determine the optimal detecting order to obtain the strongest transmit signal. Likewise permutate the index of the received symbol correspondingly. Perform a unitary transformation that satisfies relationship (3).

$$P^{1/2}\Sigma = \begin{bmatrix} P^{1/2}{}_{i-1} & P_i^{i-1\times 1} \\ 0 & p_i \end{bmatrix} \ , \quad (3)$$

As for equation (2), Σ is a unitary transformation that transforms $P^{1/2}$ into an upper triangular matrix.

Step 3: Update Q_α on the basis of $Q_\alpha\Sigma$

Step 4: Calculate the MMSE *nulling vectors*

$$w_i = p_i\, q^*_{\alpha,i} \quad (4)$$

Step 5: Calculate the strongest transmit signal $\hat{y}_i$

$$\hat{y}_i = w_i\, r_i \quad (5)$$

Step 6: Slice $\hat{y}_i$ to the nearest value in the signal constellation

$$\hat{s}_i = decision(\hat{y}_i) \quad (6)$$

Step 7: Cancel the interference of the sliced strongest transmit signal from the vector of received signals and return to step 2

$$r_{i-1} = r_i - h_i\, \hat{s}_i \quad (7)$$

End

5.2. Functional description of the "V-BLAST Square Root" algorithm

The architecture of the "V-BLAST square-root" MIMO decoder is illustrated in Fig. 6. It consists of 6 processing modules. The inputs consist of the received messages r and the values of the channel matrix H. The first three modules (M_1, M_2, M_3) perform the decomposition of the matrix H using unitary transformations. These modules calculate the dimensions $P^{1/2}$, Q_a (Step *1*), p_i (Step *2*) and $q^*_{a,i}$ (Step *3*). The next module M_4 determines the optimal decoding order and calculates the vector w_i (*nulling vector*) (Step *4*). Module M_5 decides the transmitted symbol vector (Step *6*) and the last module M_6 performs interference cancellation between the symbols (Step *7*).

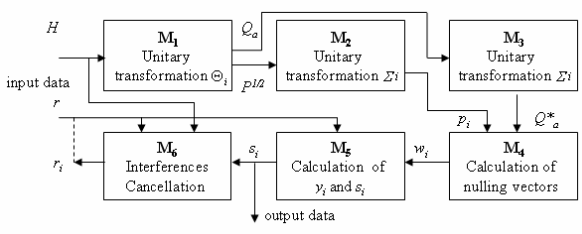

Fig. 6 Functional architecture of the "V-Blast square-root" MIMO decoder

The three modules (M_1, M_2, M_3) exhibit similar architectures. These modules are designed using CORDIC

operators (see example below for the calculation of $P^{1/2}$ and Q_α in module M_1). Instead of performing QR decomposition by a triangular network, we use a CORDIC-based Givens rotation sequence.

5.3. Square Root Decoder based on CORDIC operator

A total parallel architecture of the calculation of $P^{1/2}$ and Q_α, in module $M1$ is shown in Fig. 7. This calculation requires 29 CORDIC operators in rotation mode. They use different angles ($\theta 1$, $\theta 2$, $\varphi 1$, $\varphi 2$, $\theta 3$, $\theta 4$, $\varphi 3$, $\varphi 4$) that are pre-calculated by a CORDIC operator in vectoring mode (not shown in Fig. 7).

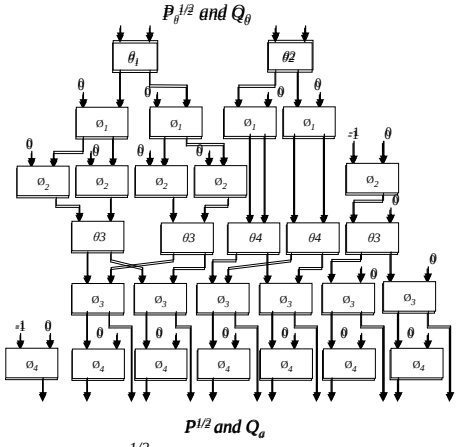

Fig. 7 Calculation of $P^{1/2}$ and Q_α in the module $M1$ in total parallel structure

This total parallel structure may lead to a waste of computational capabilities, since the *common operators* are implemented repetitively and they do not work at the same time. Therefore the iterative use of several CORDIC operators can optimize the resources. So an iterative structure of decoder can be implemented using for example three parallel CORDIC operators instead of 29 CORDIC operators in the total parallel structure. The number of CORDIC operators can be changed to adapt different requirements of wireless communication (different number of antennas and different throughputs) [9].

All iterations of CORDIC algorithm are performed in parallel, using a 20 steps pipelined structure. The input data of the CORDIC periodically changes and static implementation of the interconnections frameworks uses a great number of multiplexers to switch from one interconnection context to the next one. They take a lot of surface of FPGA and lead to waste of power consumption. Nevertheless, these multiplexers remain in the same state during 20 steps of CORDIC operations. The only difference between every 20 steps is the interconnections. This fact lets inspire the implementation on dynamically reconfigurable hardware to improve the configuration time, area efficiency and flexibility.

Our approach splits the processing into a static hardware skeleton which is composed of decoding processing elements and a reconfigurable part where the interconnections are mapped and can evolved at run-time depending on the step of processing to perform. In this approach, dynamic reconfiguration is used to change the state of interconnections instead of multiplexers. Every cycle shown in Fig.8 represents one state of multiplexers.

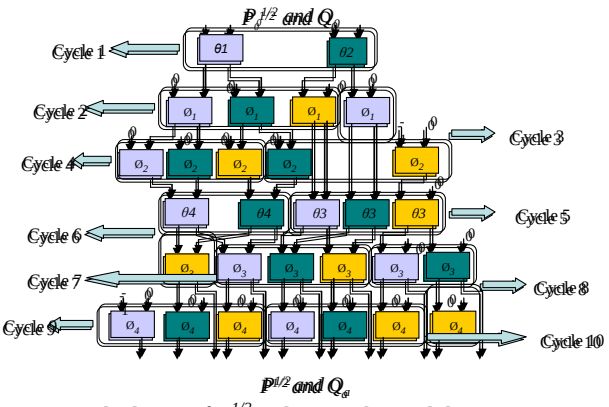

Fig. 8 Calculation of $P^{1/2}$ and Q_α in the module $M1$ in iterative structure

The main computation is preformed in the CORDIC operator and only the interconnections between them are changed at certain regularly moment. Fig. 9 illustrates the entire decoder architecture which contains two parts. The first one is a fixed part that contains 4 CORDIC units (3 CRUs: CORDIC Rotation Unit and 1 CVU: CORDIC Vectoring Unit), a multiplier and an adder or substractor. The second one is the reconfigurable one, allowing the implementation of interconnections between fixed part modules. In the reconfigurable part, only routing and registers are implemented. The same register is placed and routed in the same area of FPGA in order to retain the information stored in the register after a reconfiguration. It reduces power consumption because only wire resources are used in the reconfigurable module.

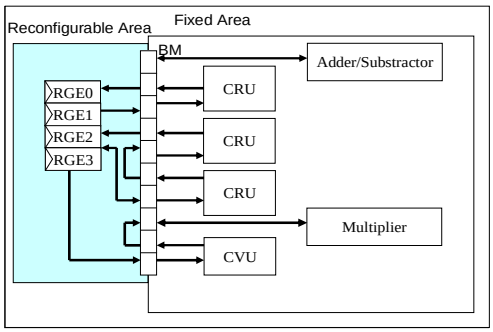

Fig. 9 Position of fixed modules and reconfigurable module

The reconfigurable part is connected to the modules of fixed part through LUT-based Bus Macro provided by Xilinx to ensure the right place and routing crossing over partial reconfigurable area. This MIMO decoder for 2 antennas system with QPSK signal constellation is implemented on a Virtex-II xc2v-2000 form Xilinx. Table 1 shows some synthesis results of fixed part and reconfigurable part of decoder.

Target FPGA Xilinx Virtex-II	Number of slices	Flip Flops	Reconf time
Without Reconf	4505(40%)	5927(27%)	16ms
Fixed part	2857(26%)	4766(22%)	12ms
Reconf part 1	85	148	0.4ms
Reconf part 2	85	148	0.4ms

Table 1: Synthesis results of MIMO decoder

It can save about 36% slices comparing to the multiplexer based decoder. In this table, the transmission time from Host to FPGA is not counted and the reconfiguration times are calculated by the size of Bitstreams and ICAP frequency of Xilinx FPGA. In this application, several CORDIC operators are reused iteratively as a *common operator* in the MIMO decoding algorithm. The synthesis results show the configuration time, area efficiency and flexibility of MIMO decoder improvement, by using the dynamic partial reconfiguration in the CORDIC *common operator*. Furthermore, the CORDIC algorithm is widely used for VLSI implementation of digital signal processing applications. Reference [10] also shows the CORDIC algorithm is often used to implement different communication functions in Software Defined Radio (SDR) systems, like direct digital synthesizers, AM, FM and PM modulators and ASK, PSK and FSK modulators, etc.

6. CONCLUSION

We have presented a design approach for SDR systems based on *common operators*. This approach consists in reusing operators and adapting their behavior according to standards and functions need. A key point is to define *common operators* for multi-standards, multi-functions systems, and once those operators are identified, design them in the way that they can be reconfigurable by parameters change. This is classically done by splitting an operator in two parts. One is static and used for computation. Second one has to be dynamically reconfigurable and has the control of the static part. The "half-reconfigurable" *common operators* can be then implemented and cope with dynamic reconfiguration for SDR systems. We also introduce the FPGA facility in terms of reconfiguration. This, combined with partial reconfiguration and reconfiguration by difference, has an important impact on bitstream size. Use of such methodology decreases drastically the size of bitstreams and without doubt offers shorter reconfiguration time as well as releases memory resources.

7. ACKNOWLEDGMENT

This work was performed in project E2R II which has received research funding from the Community's Sixth Framework programme. This paper reflects only the authors' views and the Community is not liable for any use that may be made of the information contained therein. The contributions of colleagues from E2R II consortium are hereby acknowledged

REFERENCES

[1] V. Rodriguez, C. Moy, J. Palicot, "Install or invoke?: The optimal tradeoff between performance and cost in the design of multi-standard reconfigurable radios," *Wiley InterScience, Wireless Communications and Mobile Computing Journal*, to appear, 2007

[2] J.-P. Delahaye, P.Leray, C. Moy, J. Palicot, "Managing Dynamic Partial Reconfiguration on Heterogeneous Platform", *SDR Forum, Technical Conference 2005*, Anaheim, USA, November 2005

[3] C. Moy, J Palicot, V. Rodriguez, D. GIRI, "Optimal determination of common operators for multi-standard Softawre-Defined Radio", *4th Karlsruhe Workshop on Software Radios*, March 2006, Karlsruhe, Germany

[4] Jean-Philippe DELAHAYE, Jacques PALICOT, Christophe MOY, Pierre LERAY, "Partial Reconfiguration of FPGAs for Dynamical Reconfiguration of a Software Radio Platform", *IST Mobile and Wireless Communications Summit'07*, 1-5 July 2007, Budapest, Hungary

[5] J.E. Volder, "The CORDIC Trigonometric Computing Technique", *IRE Trans. Elect. Comput.*, EC(8):330-334, 1959.

[6] C. M. Rader, "VLSI Systolic Arrays for Adaptive Nulling", IEEE Sig. Proc. Mag, Vol. 13, No. 4, pp. 29–49

[7] G.J. FOSCHINI "Layered space time architecture for wireless communication in a fading environment when using multi-element antennas" Bell Labs Journal, pages 41-57, 1996

[8] B. Hassibi, "An efficient square-root algorithm for BLAST," http://mars.bell-labs.com/

[9] H. Wang, P. Leray, J. Palicot. "A reconfigurable architecture for MIMO square root decoder". Reconfigurable Computing: Architectures and Applications, p. 317–322, 2006

[10] J.Valls, et al. "The use of CORDIC in Software Defined Radios: A Tutorial", IEEE Communications Magazine, September, 2006