Optimized Placement Approach on Reconfigurable FPGA

Mohammad Naouss and Marwa Hannachi

***Abstract*—Adaptive systems based on Field-Programmable Gate Array (FPGA) architectures can benefit from the high degree of flexibility offered by Dynamic Partial Reconfiguration (DPR). In DPR, hardware modules composing an application can be allocated on demand or depending on a dynamically changing system. However, founders DPR tools are limited in functionality because it does not support an automatic place- ment, and require a manual inputs from the design. Manual placement not allows an efficient placement. The placement step represents a critical step in DPR flow on FPGA. It is highly impact routability, timing and density, hence performance of the system. In this paper, we present a novel placement algorithm to address these constraints by offering minimal fragmentation that minimizes resource utilization and reduces the total wirelength. The selection of the Partial Reconfigurable Region (PRR) is based on the shapes, location and communication with others. The proposed approach has been experimentally evaluated with a case study. Experimental results show the effectiveness of the proposed algorithm in the terms of exploration area and communication cost.**

***Index Terms*—Placement, FPGA, dynamic partial reconfigura- tion, wirelength**

I. Introduction

Processors are very powerful at the level of data management and control, but they are less efficient at the level of calculation and processing because of its sequential architecture [1]. On the other hand, hardware solutions such as FPGAs have a high computing power because of its parallel architecture in hardware; they are low-power calculators, but not performing at the level of management and control of data. To meet the needs of the market for a powerful system at both levels and low consumption, the founders decided to implement a hybrid architecture hardware (HW) / software (SW) and in particular FPGA / CPU (Central Processins Unit) [2]. To run an application on a hybrid system, it is necessary to split the application into several modules. Each module is a treatment unit. A set of modules are implemented on the software part (CPU) and others on the hardware part (FPGA) [3]. Furthermore, to improve flexibility and performance of system, DPR technique is used to implement the hardware modules. DPR allows the modification of certain blocks on the FPGA at run time [4], [5]. It can improve the efficiency of FPGAs by enabling different processing functions to share the same hardware resources and therefore, reduce the use overall resources. However, this technique leads to high configuration overhead, defragmentation, and complex allocation situations of hardware tasks [6]. In addition, founders tools have evolved to provide successful dynamic partial reconfiguration flows, but they are limited in functionality. They need the manual involvement of designer to specify the shapes and locations of each reconfigurable region. Consequently, the quality of place- ment affects the system performance. A bad placement will increase the amount of unused hardware resources and will increase the latency of communication between the different shapes, therefore decreased the performance of the system. The placement problem is NP-complete [7] because of the number of parameters and constraints to consider. So, there are exists a serious need to define an efficient method for an efficient placement in FPGA design flow.

The main contribution of this paper is the description of new placement algorithm that to deal with the HW/SW partitioning problem with DPR. We propose a new placement algorithm for efficient hardware space exploration and wirelength utilization. The defined algorithm proposes to split the FPGA into several regions and then to allocate a module in a dedicated region that has the shape closest to a square. The goal is to reduce latency metric inside the region and reduce the area fragmentation. Then, an extended version of the algorithm is proposed to improve the wire length of critical path for delay minimization. In the new algorithm, each module is mapped within its region in order to minimize wirelength by taking into account the distance and number of connection wires between modules. On a case study, we show that the proposed placement algorithm allows to significantly reduce the static region fragmentation and wirelength cost.

The remainder of this paper is as follows. Section II presents the existing works. Section III formulates the proposed place- ment algorithm. Section IV describes how we minimize the wirelength cost. Section V presents the results obtained on realistic applications. Finally, this paper is concluded by the Section VI.

II. Related Work

In the early research work, the authors treat in their algorithms the aspect of static placement in the FPGA without any regard for its reconfiguration ability [8], [9]. These algorithms are based on the simulated annealing approach (SA) to find optimal placement solutions. Their formulations can be applied to heterogeneous FPGAs, but the resulting floorplan may contain irregular shapes, which does not conform to the requirements of the reconfigurable regions. In dynamic reconfigurable system, Tomono et al. [10] used area matrix to represent 2D Configurable Logic Block (CLB), and managed by 2D-based array space to

Manuscript received February 24, 2019; revised April 12, 2019.

M. Naouss and M. Hannachi are with the Department of research, Altran Technologies, Toulouse, France (e-mail: mohammad.naouss@altran.com, marwa.hannachi@altran.com).

DOI: 10.7763/IJMO.2019.V9.689

search the free space in FPGA 2 resource. Recently, Vipin and Fahmy [11] have introduced a floor planner named: Columnar kernel tessellation to define the location and the size of each reconfigurable region. Their approach characterizes the FPGA in terms of tile: Each tile contains a number and a type of logical resources. Therefore, the requirements of the reconfigurable regions are translated in terms of tile requirements. Reconfigurable regions are sorted in order of priority based on the type and the number of tiles required. Floor planning is done by merging adjacent tiles on the same line to form areas called kernel. A column verification procedure (coulnmar direction) is performed at the end of each iteration by moving the region vertically in order to improve the total wire length (communication cost. And the best result obtained with respect to an objective function is then considered as a solution. However, the final result of the placement results in a large defragmentation of the remaining part. Rabozzi and al. [12] have shown that the quality of the solutions obtained can be further improved by analytic methods based on mixed integer linear programming technique (MILP. The analytic model is formulated to represent the characteristics of all reconfigurable regions in terms of their resource requirements and the connectivity between them. Since the MILP-based algorithm has shown that he is able to consider the global space in his research for finding optimal solution. Nevertheless the algorithm takes a long time to solve more complex problems that require multiple reconfigurable regions. Therefore the search time exponentially increases with the number of regions to find.

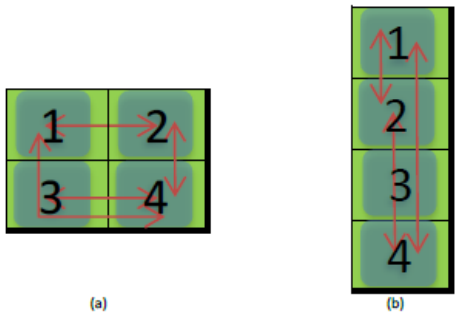

Fig. 1. 4 regions (a) Proposed conditions (b) Without conditions

III. Proposed Placement Algorithm

In DPR technique, the size of each module must be calculated as the number of reconfigurable unit (*Ru*) used for the implementation (using the clock region technique). *Ru* is a LABs (Logic Array Blocks) for Altera and Frame for Xilinx. The goal is to avoid sharing the same *Ru* between two modules. The FPGA takes a rectangular shape of *X Ru* on the x axis and *Y Ru* on the y axis, so an FPGA contains *X _ Y* Ru. The problem of placement can be announced as follows.

_ Let $M = fM_1; M_2; :::; M_{mg}$ is a set of modules to be placed

_ Let Si is the number of *Ru* required for each module M_i

_ Let *X _Y* is the number of *Ru* on the *x* axis and the *y* axis of FPGA

The placement algorithm consists of finding a set of *n* PRR

$R = fR_1; R_2; :::; R_{ng}$. Each PRR is defined as a rectangular shape that is characterized by the following data: X_i _ Y_i

Which is the size of the defined region. A. Resource Requested

A reconfigurable region can accommodate different recon- figural modules at different times; hence their sizes should be large enough to allocate the biggest task. This area (R_i) is defined taking into consideration the maximum resources needed between the different tasks allocated.

$$R_i = Max(S_j)$$

B. Placement Process

This section focuses on the process steps of our placement strategy. Our algorithm will start with a search of three different shapes for each region and after, it will select the most appropriate one.

Finding different shapes: Filling a part of an FPGA with modules requires defining the rectangular shapes taken by each module. Each module M_i is described by its size S_i, assuming that each *Ru* consists of a number of LUTs and registers, depending on the FPGA used. For each form, we note UN_i the number of unused *Ru* by M_i and we thus have: $UN_i = (X_i\ Y_i)\ S_i$, where $(X_i\ Y_i)$ is the size of the selected module. We arbitrarily calculate three possible rectangular forms X_i, Y_i of minimum values UN_i and whose sizes are equal or higher than S_i. In the calculation of shapes, the algorithm forbidden to have UN_i greater than Xi or $Y\,i$. These calculated shapes are therefore as close as possible to squares to minimize latency of module. We therefore forbid X_i and Y_i to be equal to 1 if $S_i > 3$. Table I shows the computed shapes for each modules of a case study as well as the value of UN_i.

Selection of shapes: In order to map application on the FPGA, the most appropriate form for each module is selected. However, this selection depends on the possible forms of other modules to avoid generating a remaining region unable to map another module. For the latter, $min\ xy = min(min(Xi), min(Y\,i))$ is calculated for all the modules. min_XY is the minimum size of the row or column of a module. The following constraints on $min\ XY$ therefore select the form that prevents a module from being fragmented over several regions [13].

$$[(L - X\text{ ``}\ min\_xy) \vee (L - X = 0)] \wedge$$

Fig. 2 illustrates the different steps of module mapping for this case study. The first corresponding module is M6 (SM6=35 Ru) which has the highest Si value. To select the shape of M6, we calculate minXY for all the remaining modules (except M6) which is equal to two in this case. Since the first form 6 6 for M6 keeps free more than 2 columns and 2 lines, then this form is selected. We start by filling the FPGA from its lower left corner and therefore we map M6 to this corner, as is shown in Fig. 2.a. We then follow a direction in the counterclockwise rotation as shown by the other figures. The next region considered is 6 6. M8 is selected to fill this region because its SM8 value is equal to the size of this region. We continue on the same path until we fill all the existing modules. It should be noted that most of the regions selected for each module belong to the first column and a little less to the second. This means that

our approach has selected regions that have shapes very close to a square or a square.

$$[(W - Y \text{“ } min\_xy) \vee (W - Y = 0)]$$

IV. Minimizing Communication Cost

In this section, an improved version of placement algorithm is proposed. The goal of this second version is to reduce the communication cost by applying specific rules for mapping tasks and around the paths routing.

TABLE I: DIFFERENT SHAPES FOR EACH MODULE

Module	S_i	Region1		Region2		Region3	
		$X1 \times Y1$	UN_1	$X2 \times Y2$	UN_2	$X3 \times Y3$	UN_3
M1	17	4 × 5	3	4 × 6	1	2 ×9	1
M2	26	5 × 6	4	4 × 7	2	3 ×9	1
M3	7	3 × 3	2	2 × 4	1	NC	NC
M4	16	4 ×4	0	3 ×6	2	2 ×8	0
M5	13	4 ×4	3	3 ×5	2	2 ×7	1
M6	35	6 ×6	1	5 ×7	0	4 ×9	1
M7	22	5 × 5	3	4 × 6	2	3 ×8	2
M8	34	6 × 6	2	5 × 7	1	4 ×9	1
M9	15	4 ×4	1	3 ×5		2 ×8	1

TABLE II: NUMBER OF RESOURCES FOR EACH MODULE

Module	LUT	Frame
M1	2737	7
M2	5907	15
M3	2583	7
M4	1620	5
M5	196	5
M6	38	1
M7	5594	15
M8	1619	5
M9	448	2

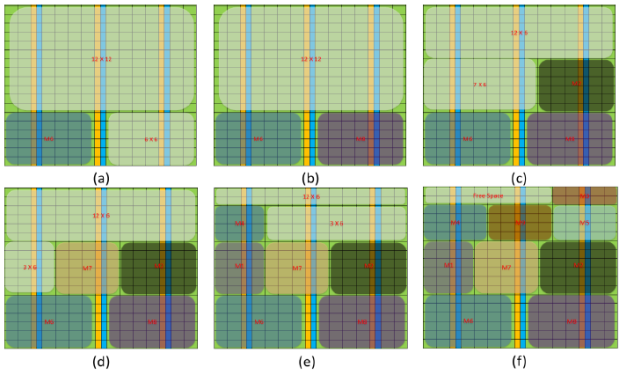

Fig. 2. Steps of mapping the 9 modules using the proposed approach.

A. Problematic

The reduction of communication costs between reconfig- urable regions is also an objective to be achieved by the placement algorithms. The routing resources available in a reconfigurable architecture are configured to link the reconfig- urable tasks to each other. During the design phase, these rout- ing resources are often of fixed density and quantity. Indeed, more the distance between the two regions is important, more the routing resources are important to link the modules that will be allocated in these regions. In addition, the wirelength inter-module not only affects the cost of routing paths, but it also increases data communication latency between tasks and affects system performance. With regard to all these considerations, it is necessary to minimize the Wirelength between tasks. In this context, we propose an improvement on the proposed placement algorithm. The goal is to find the best location for each module by minimizing the cost of communication and respecting the constraint shape and placement of the initial algorithm.

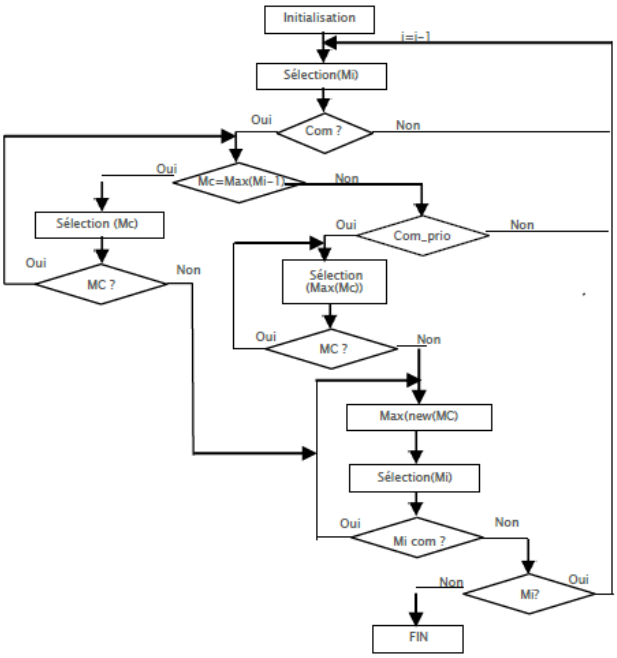

Fig. 3. Placement algorithm process.

B. Formulations

The total wirelength is measured by applying the most commonly used method, Half Perimeter Wire Length (HPWL). HPWL assumes that each pin is in the center of the area. The wirelength between two regions (WL) is calculated as the product of the Manhattan distance between them and the number of wires connecting them.

We start by calculating the centroid of each region.

$$cx_i = \bar{x}_i + w_i/2$$
$$cy_i = y_i + h_i/2$$

where (cx_i, cy_i) represent the (x, y) coordinates of region i centroid, w_i and h_i represent respectively the width and high of region i and x_i and y_i represent respectively the leftmost and the lowest positions occupied by region i. Then, we calculate the Manhattan distance between the centroids of two regions.

$$dcx_{(i1,i2)} = |cx_{i1} - cx_{i2}|$$
$$dcy_{(i1,n2)} = |cy_{i1} - cy_{i2}|$$

where ($dcx(i1, i2)$, dcy ($i1$, $i2$)) represent the distances between the centroids of the regions $i1$ and $i2$ on the x and y axes.

The objective of the algorithm is to minimize the total communication cost WLT which is equal to the sum of the interconnection cost between the different zones multiplied by the numbers of wires connecting them.

$$WLT = \sum_{i=0} b_{i,i+1} \times (dcx_{i,i+1} + dcy_{i,i+1}) = WL$$

With $b_{i,i+1}$ is the number of wires connecting the regions i and $i + 1$.

C. Algorithm Process

In order to add the communication constraints in the algorithm, we need to define some additional parameters:

Mn = (*M* 1, *M* 2, ...*Mn*) is the set of modules to be placed, with n the number of modules.

Mc is the set of modules that communicate with the modules already placed.

The first rule is to allocate the first module (*Mi*) which is the largest one of *Mn*, in the lower left position of the reconfigurable part by choosing the most appropriate shape.

After placing the first module (*Mi*), the algorithm will test its dependence with the other modules and build the first set *Mc*. If *Mi* wouldn't communicate with others, the algorithm will move to place the largest module between the remaining modules. But, if *Mi* communicates with other modules, it will compare the set of modules *Mc* with the remaining modules *Mn*. If there is a module in *Mc* that is larger than the remaining modules, it will look for a shape and a nearest location. If not, it will test the priority between communication and hardware resources. If the location has the highest priority, the algorithm will look for the shape and location of the largest module between the remaining modules. If communication has the highest priority, it will look for largest module in *Mc* a suitable form and at closest position.

In the next step, the list *Mc* is updated by adding the communicating modules with the new module placed and removing the placed module. Then, the algorithm determines the largest module from *Mc* and looks for its shape and location. These steps are repeated several times until no placed module communicates with the remaining modules. In this case, if there are still modules not placed the search process will start again from the first step until all the modules will be placed. The proposed algorithm process is illustrated in Fig. 3.

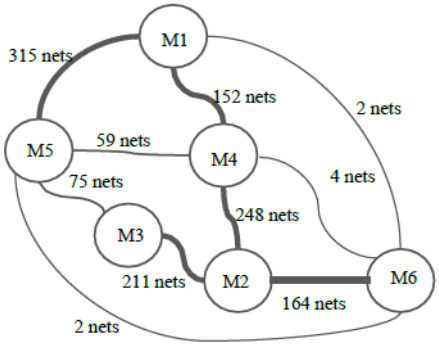

Fig. 4. Task graph illustrating H264 Codec case study.

V. Experimental Results

This section shows an experimental evaluation of the place- ment algorithm in the two cases: without and with communica- tion. The design was implemented on the Xilinx series 7 FPGA *xc7a*35tifgg484-1L. This algorithm has been implemented using C++ in order to make several tests and to validate its operation.

The considered application is based on a codec H264 and DCT-2*d* application. The hardware architecture of the code developed within the ALTRAN team is mainly composed of six modules (*M*1, *M*2, *M*3, *M*4, *M*5 and *M*6) connected between themselves (Fig. 4). The DCT-2*d* is also composed of three modules (*M*7, *M*8 and *M*9) connected between themselves. All this modules will be implemented to evaluate the impact of their locations on the wire length cost.

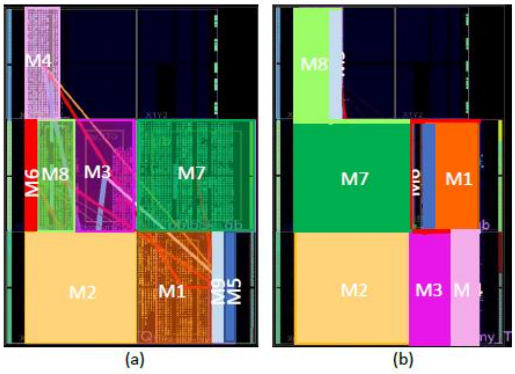

Fig. 5. Placement results in cases (a) without communication (b) with communication.

In our design, we have nine modules and we have defined for each one a reconfigurable region. To define the location and shape of each module, we first need to determine the number of resources of each reconfigurable region according to the number of frames. The number of logical resources per frame depends on the FPGA family. In a series 7 FPGA, a CLB frame is equivalent to 400 LUTs. Table II illustrates the ressources needed for each module.

For the improved version of the placement algorithm, we also need to know the number of wires that connect modules between them. This number will influence the cost of routing because it will be multiplied by the distance of routing path. The table III shows the different connections between modules and the number of wires between them.

Applying the first version of the placement algorithm, we found the placement of the modules in the following order: *M* 2, *M* 1, *M* 9, *M* 5, *M* 7, *M* 3, *M* 8, *M* 6, *M* 4. In the case of a placement with communication, the modules are placed in the following order: *M* 2, *M* 3, *M* 4, *M* 1, *M* 5, *M* 6, *M* 7, *M* 8, *M* 9. Fig. 5.a and 5.b respectively illustrate the placement results in both cases without and with communication.

Comparing the two figures (Fig. 5.a and 5.b), we note that the two algorithms give better results in the selection of shapes and location. These results minimize latency and fragmenta- tion of the remaining area of the static part. Nevertheless, the interconnection results are not the same. In the case where the algorithm does not take into account the communication between the different modules, we notice that the routing paths are long and not well distributed. On the other hand, the algo- rithm takes into account the communication, we notice that the modules which communicate between them are close and the routing paths

are short and more optimized. Table III illustrates the overall communication cost results in both cases. As shown in the Table III, the communication cost between modules M4 and M6 is higher in the case with communication than in the case without communication, because the objective of our algorithm is to minimize the total cost of communication in the application. In this case, the modules which have larger size and more wire number are more critical in their location, so they have the priority to choose their locations firstly. The results show that, taking into account the communication, the routing paths are optimized on the order of 48%.

TABLE III: INTER-MODULE COMMUNICATION COST

Communication	wires	without communication			with communication		
		dcy	dcx	WL	dcy	dcx	WL
(M1,M4)	152	100	16	17632	0	6.5	988
(M1,M5)	315	0	6.5	2047.5	50	1.5	16222.5
(M1,M6)	2	50	17	134	50	1.5	103
(M2,M3)	211	50	2	10972	0	7.5	1582.5
(M2,M4)	248	100	5	26040	50	10	14880
(M2,M6)	164	50	6.5	9266	0	20.5	3362
(M3,M5)	75	50	13.5	4762.5	0	3	225
(M4,M5)	59	100	23.5	7286.5	50	5	3245
(M4,M6)	4	50	1.5	206	50	6.5	226
(M5,M6)	2	50	25	150	0	3	6
(M7,M8)	2	0	15	30	50	4	108
(M7,M9)	128	50	4.5	6976	50	1.5	6592
(M8,M9)	128	50	19.5	7616	0	4	512
WLT				93118.5			48052

VI. CONCLUSION AND FUTURE WORK

This paper was dedicated to the issues encountered during the placement step in DPR flow. Among these problems: shape, location and communication. A new strategy for place- ment of reconfigurable hardware tasks is proposed. The main goal of this algorithm is to find optimal shapes and location for each module to reduce internal latency.

In the remainder of this paper, we have proposed an improvement of the placement algorithm taking into account the inter-communication between the different modules. Indeed, the goal of this part was to optimize the routing resources while seeking an ideal shape and location for each module. A real application developed within the Altran team is imple- mented to evaluate our placement algorithm.

With the increasing design complexity, modern FPGAs have a heterogeneous architecture with distributed I/O, DSPs (Digital Signal Processor) and BlockRAMs (Block Memory). In the future, we will be looking at more emerging challenges for FPGA placement on heterogeneous architectures.

REFERENCES

[1] S. Asano, T. Maruyama, and Y. Yamaguchi, "Perfor- mance comparison of fpga, gpu and cpu in image processing," in *Proc. International Conference on Field Programmable Logic and Applications*, 2009.

[2] Y. Ma, J. L. Liu, C. Zhang, and W. Luk, "Hw/sw partitioning for region-based dynamic partial reconfigurable fpgas," in *Proc. 2014 32nd IEEE International Conference on Computer Design*, 2014.

[3] R. Cordone, F. Redaelli, M. A. Redaelli, M. D. Santambrogio, and D. Sciuto, "Partitioning and scheduling of task graphs on partially dynamically reconfigurable fpgas," *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, vol. 28, no. 5, pp. 662–675, 2009.

[4] W. Lie and F. Y. Wu, "Dynamic partial reconfiguration in fpgas," in *Proc. IITA 2009. Third International Symposium on Intelligent Information Technology Application*, vol. 2, pp. 445–448, 2009.

[5] R. H Patel and K. A. Norman, "Partially reconfigurable pro- grammable logic device," 2000.

[6] A. A. E. Farag, H. M. E. Boghdadi, and S. I. Shaheen, "Improving utilization of reconfigurable resources using two-dimensional compaction," *The Journal of Supercomputing*, vol. 42, no. 2, pp. 235–250, 2007.

[7] H. R. Lewis, "Computers and intractability a guide to the theory of NP-completeness," 1983.

[8] L. Cheng and M. D. F. Wong, "Floorplan design for multimillion gate fpgas," *IEEE Transactions on Computer-Aided Design of* Integrated *Circuits and Systems*, vol. 25, no. 12, pp. 2795–2805, 2006.

[9] Y. Feng and D. P. Mehta, "Heterogeneous floorplanning for fpgas," in *Proc. 5th International Conference on Embedded Systems and Design*, 2006.

[10] M. Tomono, M. Nakanishi, S. Yamashita, K. Naka-jima, and K. Watanabe, "A new approach to online fpga placement," in *Proc. 2006 40th Annual Conference on Information Sciences and Systems*, 2006.

[11] K. Vipin and S. A. Fahmy, "Architecture-aware reconfiguration-centric floorplanning for partial reconfiguration," in *Proc. International Symposium on Applied Reconfigurable Computing*, pp. 13–25, 2012.

[12] M. Rabozzi, J. Lillis, and M. D. Santambrogio, "Floorplan- ning for partially-reconfigurable fpga systems via mixed-integer linear programming," in *Proc. 2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*.

[13] L. Abdallah, M. Jan, J. Ermont, and C. Fraboul, "Reducing the contention experienced by real-time core-to-i/o flows over a tilera-like network on chip," in *Proc. 2016 28th Euromicro Conference on Real-Time Systems*, pp. 86–96, 2016.

Mohammad Naouss was born in Lebanon on September 1, 1989. He received the B.S. degree in industrial engineer and maintenance from Lebanese university, Saida, Lebanon, in 2010, and the B.E. degree in electrical and electronics engineering from ISEN Graduate School of Engineering, Brest, France, in 2013. Then, He had a Ph.D on embedded electronics at IMS laboratory, Bordeaux, France, in 2016. Naouss's fields of interest include the modeling and the simulation of the aging mechanisms of electronic Devices especially FPGA and their effects on circuits reliability. The architecture of a System On a Programmable Chip which combines between the processors and the FPGA system is the one of his interesting fields on this day.

Marwa Hannachi received her degree in electrical en- gingering and her masters in electrical engineering, electron- ics from the National School of Engineering of Monastir (ENIM), Tunisia, respectively, in 2012 and 2013. Then, she had a Ph.D on embedded electronics at IJIL laboratory, Nancy, France, in 2017.

Her current researches interests include reconfigurable architectures and the architecture of a System On a Programmable Chip which combines between the processors and the FPGA system.