ALGORITHMIC AND ARCHITECTURAL OPTIMIZATION OF A 3D RECONSTRUCTION MEDICAL IMAGES ALGORITHM FOR A HARDWARE IMPLEMENTATION

MILI Manel[1], MAHMOUD Bouraoui[2], BOUBAKER Mohamed[1], BEN ABDALLAH Asma[1], BEDOUI Mohamed Hédi[1], AKIL Mohamed[3]

[1]Research Team of Medical Imaging Technology (TIM), Faculty of Medicine at Monastir University of Monastir, Monastir, Tunisia
Mili.manel@gmail.com , medhedi.Bedoui@fmm.rnu.tn
[2]National Engineering School of Sousse, University of Sousse, Sousse, Tunisia
Bouraoui.Mahmoud@eniso.rnu.tn
[3]Laboratoiry of informatic Gaspard-Monge, Team A3SI, ESIEE Paris, University of Paris EST, Bld Blaise Pascal, BP 99, Noisy-Le-Grand, 93162, France
akilm@esiee.fr

ABSTRACT

This paper presents an optimization of an FPGA circuit implementation of 3D reconstruction algorithm of medicals images. It is based on an algorithmic specification in the shape of a Factorized and Conditioned Data Dependences Graph (GFCDD). An automatic and optimized implementation of the algorithm of « Marching Cubes » has been carried out. The repetitive property of the algorithm has been exploited, as much as possible, by means of the methodology "Adequacy Algorithm Structures".

KEYWORDS

3D reconstruction; marching cubes; optimization implementation; AAA; FPGA.

1. INTRODUCTION

The 3D reconstruction consists in providing a volumetric representation of an object from a set of minimum information to ensure the description of the 3D volume. Its involvements are increasingly important in various fields especially that in medicine and biology.

In this paper, we focus on the optimized implementation of latency and resource material for the reconstruction of volume flux on programmable media to exploit the parallelism of implantation and reduce the latency and area.

This article is organized as follow: Section 2 describes the Marching cubes algorithm, Section 3 presents the AAA methodology and SynDEx-IC tool. Section 4 describe Marching cubes development by using SynDEx-IC. In Section 5 we propose the FPGA based system for the Marching Cubes algorithm. Finally, the conclusion is given in Section 6.

DOI : 10.5121/ijcseit.2012.2203

2. MARCHING CUBES

Since the Marching Cubes algorithm is not only fast but also easy to implement, it has become a reference in the 3D reconstruction algorithm for the medical image data.

This algorithm was designed by William E. Lorensen and Harvey E. Cline in 1987 [1], to generate a 3D model of an interesting anatomical structure, characterized by a given threshold. It uses a set of 2D images previously segmented. The generated surface is described by a 3D triangular mesh.

The basic principle of the Marching Cubes algorithm [1] [2] is to subdivide the space into a set of cubes called voxels. Each voxel has 8 vertices and 12 edges [3]. Since each vertex is classified as above or below a given isovalue (threshold), there are $256(2^8)$ possible topologies inside a voxel. Thanks to rotation symmetry and inversion of inner and outer points, these 265 initial configurations can be reduced to 15 basic configurations.

The algorithm consists of drawing a polygon which is in fact, the intersection of a cube and the plan which locally modelizes the surface. For example, if the polygon formed by the intersection cuts only three edges, there will be a triangle (figure 5).

All the possible intersection configurations (256) are classified by the algorithm's designers in two tables [1]. The first one, named Edge-table, identifies the edges cut by the surface. The second table, called Tri-Table, determines the direction to follow to connect the points which form the corresponding polygon.

The organization chart of the Marching Cubes algorithm is described by figure 1.

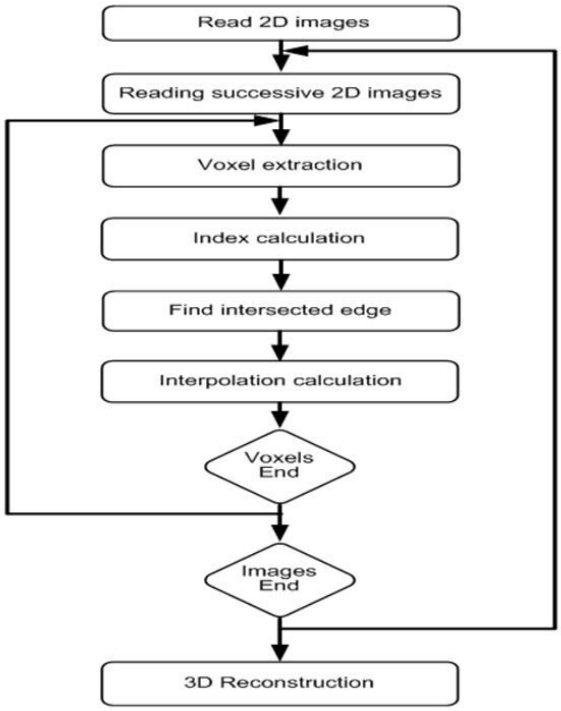


Figure 1: The Marching cubes algorithm chart

3. AAA/SYNDEX-IC METHODOLOGY

3.1 AAA methodology for specified integrated circuits

The AAA methodology aims rapid prototyping and optimized implementation of real time applications. It is founded on models of graphs forming the algorithm, the architecture and the adequation.

The data flow graph modeling the algorithm at the behavioral level is gradually transformed until it matches the hardware graph architecture, which includes the control path graph and data path graph [4] [5] [6]. The result of these transformations corresponds to optimized hardware implementation [4] [5] [6].

The adequation is to choose among all possible algorithm implementations on an architecture the one whose performance characteristics derived from the components of the architecture (the current version of SynDEx-IC architecture supports a single-component), respecting the real-time constraints, while minimizing the consumption of hardware resources (surface), thus an optimization problem.

Figure 2 describes the stages of the implementation process AAA applied to the circuits.

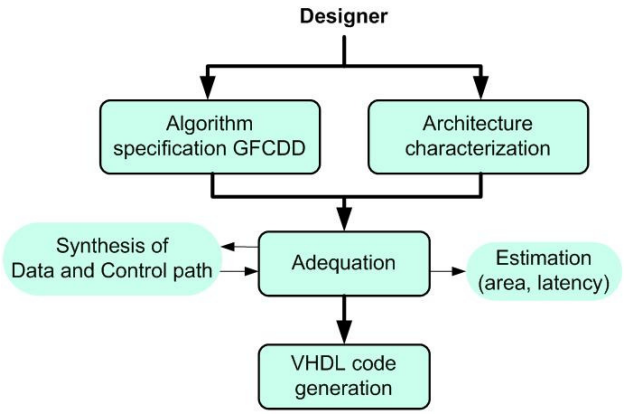


Figure 2: The AAA methodology for circuits

3.2 SynDEx-IC tool

SynDEx-IC (Synchronized Distibuted Executive-IC) is free software developed by the ESIEE team "design architecture" of A2SI Laboratory, focusing on the AAA methodology. This software tries to be, before the execution, an implementation of the algorithm that meets a given latency constraint while being implanted on the target circuit. For this, SynDEx-Ic works on factorization ie the more factorized loop, the more parallelism.

But, the area is necessary for its implementation increases. It allows therefore exploring the nested loops in order to find for each loop the rate defactorization, which can respect the execution data contraints.

Because it is impossible to explore all possibilities defactorization in reasonable times, SynDEx-IC use approximate methods based on heuristics. The result of the heuristic is then converted into

a synthesizable VHDL code to be executed directly on the target component or to be simulated. The specification of the algorithm is in the form a Data Dependence Graph (Figure 3) including the regular parts (regular repeating patterns) and not regular. This specification should be independent of any hardware implementation constraints and requires the implementation of a process of decomposition of the algorithm in implantable operations (addition, subtraction and multiplication) [6].

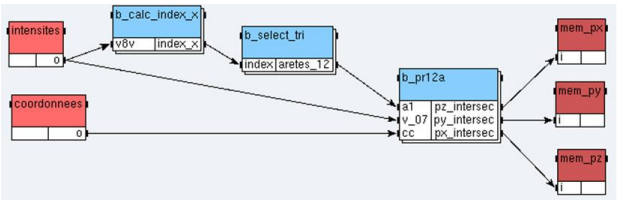


Figure 3.: Example of the SynDEx-IC main algorithm graph

To reduce the specification size and adopt these regular parts, we use a factorization process, which shows four specific types of nodes (factorization frontiers nodes): F (Fork node), J (Join node), D (Diffusion node) and I (Iterate node). This factorization process specifies different ways to factor the data and operation across a factoring frontier.

The designer models the target architecture (the type of FPGA) through the graphic interface of SynDEx-IC (Figure 4). Once the architecture has been specified, it is necessary to characterize the set operations of the algorithm graph in terms of area by number of slices, and in terms of latency, to ensure its hardware implementation on the target component. These values can come from libraries or measurements.

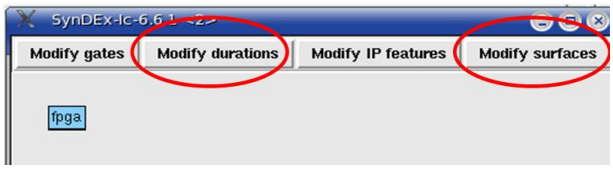


Figure 4: Example of the SynDEx-IC architecture graph.

4. MARCHING CUBES DEVELOPMENT BY USING SYNDEX-IC

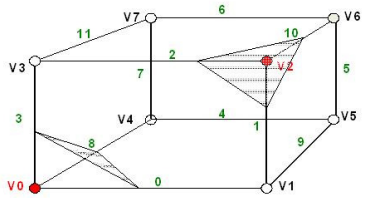
4.1 Algorithmic specification

The index calculation presents the second stage of the Marching cubes operation. This index is calculated by comparing each vertex of the cube with the threshold. This latter, was chosen by the user to characterize the interesting surface (in our case we chose a threshold = 80).

The index is a binary sequence of 8 bits size which the value varies from 0 to 255. It provides information on the positions of the inside and outside for the interested surface.

Each bit of the index is associated with one vertex; the bit is "1" if the point is internal, "0" if not. The surface then cuts the edges of the cube when the two vertices forming the edge are signs opposites (one is 0 the other is "1").

Figure 5 shows an example of index calculation where we have a cube whose vertices 0 and 2 are internal points on the surface; thus producing an index equal to 5.



V7	V6	V5	V4	V3	V2	V1	V0
0	0	0	0	0	1	0	1

Index = 1×1 + 0×2 + 1×4 + 0×8 + 0×16 + 0×32 + 0×64 + 0×128 = 5

Figure 5: Example of Index calculation

Thus, by factoring the index calculation block, we have FF2 that represents the factorization frontier of repetitive patterns of the index calculations block (the repeat unit contains a comparator, multiplier and adder).these operators are specified in loop "For s = 0 ... 7 do." (Figure 6).

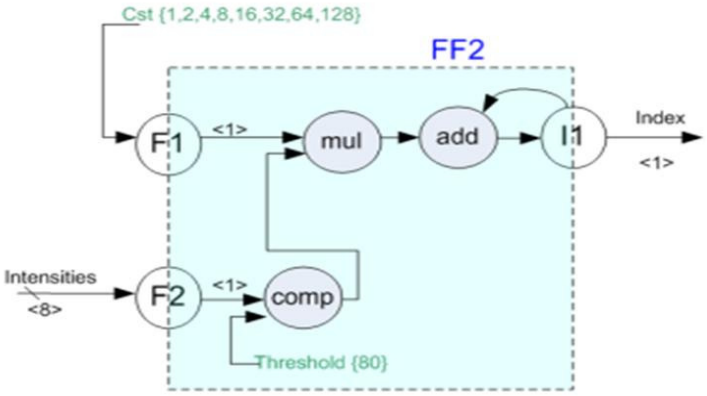


Figure 6: Factorization of index calculation block

FF2 is applied eight times and is delimited by factorization node (F1, F2, I1). The two nodes F1 and F2 are separate its input array (8 intensities and 8 constants) in its elements, while the node (I1) makes the factorization of inter-pattern data.

Added to the factorization frontier FF2, the GFCDD of the Marching Cubes algorithm contains two nested frontiers FF3 and FF4 their factorization factor respectively 15 and 3. The first (marked by the degree of repetition) is the factorization of unit calculations contained in the loop "For a=1...15 do." The second is the factorization contained in the loop "For c = x, y, z do" (Figure 7).

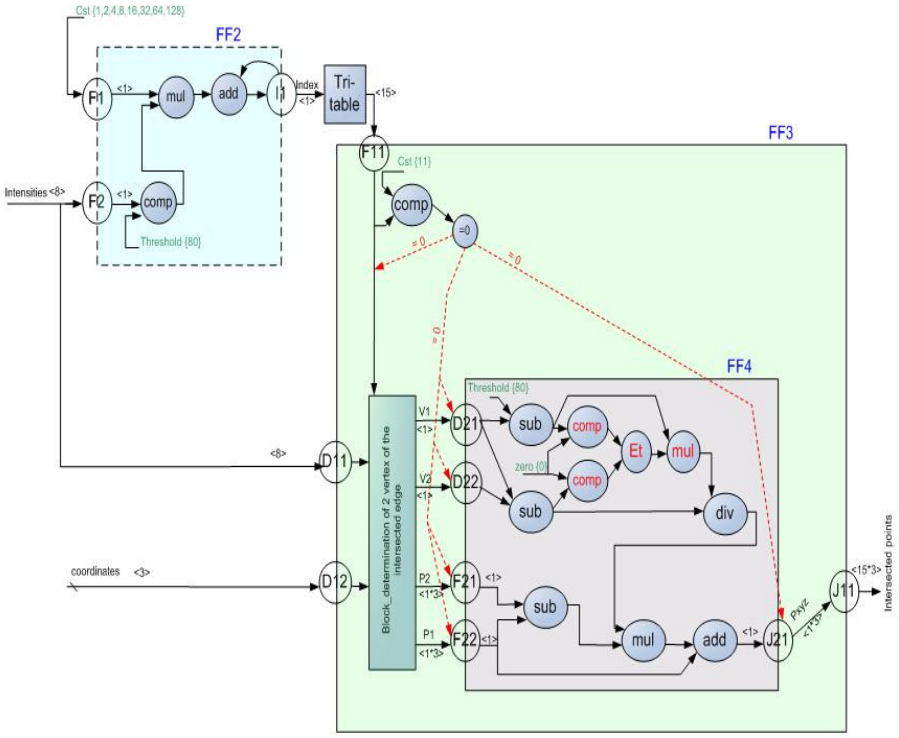


Figure 7: GFCDD of Marching Cubes algorithm for a voxel processing

4.2 Architectural exploration of the Marching cubes implementation

The architectural exploration is based on a heuristic, which builds the solution iteratively. The heuristic determines, at each iteration, a defactorization frontier and its defactorization degree. The factorization frontier is the one that generates the weakest possible area and respects the imposed latency. The evaluation of each solution is based on the predictive model performance (latency and area). This prediction requires the characterization of every provider according to the target technology of the used circuit.

Figure 8 presents the solutions obtained by heuristic optimization under different constraints (µs). For each case presented, we give the estimated latency and area (number of slices) and the defactorization degree (DF) of two frontiers FF3 and FF4 (TF means totally factorized).

The results presented in terms of surface and latency, prove that most defactorization solutions reduce latency, but increase resource consumption. Thus, within the imposed constraints, the variation of latency and area by the defactorization degree change in level Figure 8.

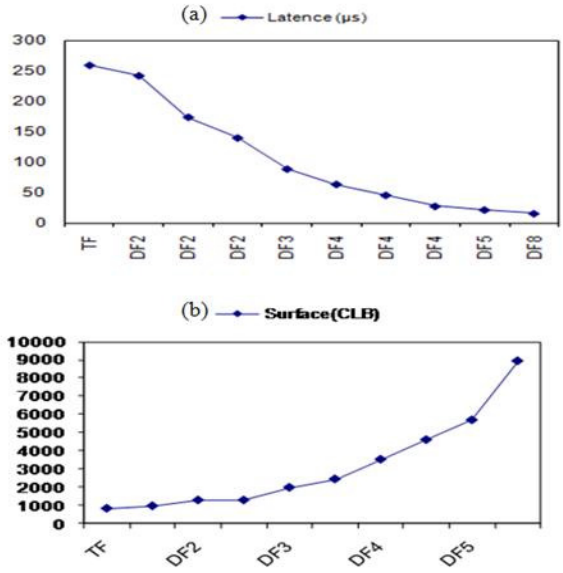


Figure 8: Results of the heuristic according to constraints variation

It should be noted that, in terms of surface, these results correspond only to the area occupied by the arithmetic and logical operator's application. The area occupied by the control units of different factorization frontier nodes are not counted by the SynDEx-IC. The comparator, the divisor, the logical AND and the memories are defined as empty blocks (with the corresponding inputs and outputs) in the data flow graph. We defined and allocated to each operator the occupied area memory blocks (number of slices) and the latency in the architectural specifications. The parallelism depends on the degree of each defactorization frontier following the constraint. For a constant surface, the variation of latency depending on defactorization degree is due to the sequential execution or partly parallel of the frontier.

4.3 Implementation optimization

4.3.1 Data graph

The graph in Figure 9 shows the data path optimized implementation of the Marching Cubes algorithm for a voxel processing. These blocks include: the calculation operators, the memory and the registers.

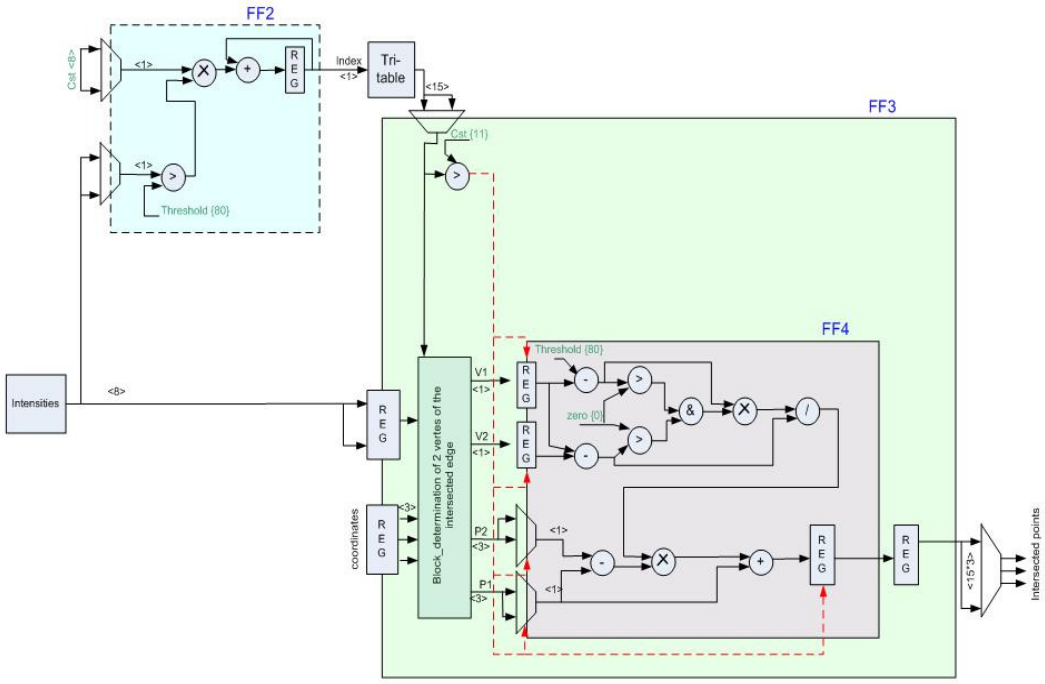


Figure 9: Data path of the optimized implementation

4.3.2 Control graph

Figure 10 presents only the control path and shows the action of each control unit UC1, UC2 and UC3 on different nodes of the factorization frontiers. Each UC contains a counter (corresponding to the factorized repetitions), a requests (r) and acknowledge (a) signal implanting the consumption / production of data between the different frontiers.

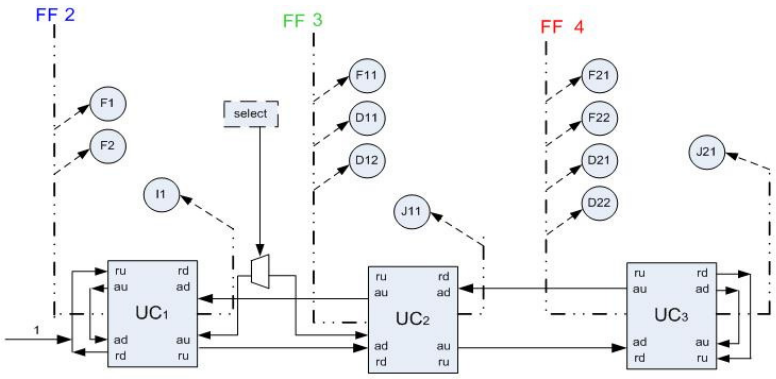


Figure 10: control path of the optimized implementation

Based on the neighborhood relations between the various factorizations frontiers of the algorithms

graph, the AAA methodology generates a neighborhood graph Gv as described in figure 11.

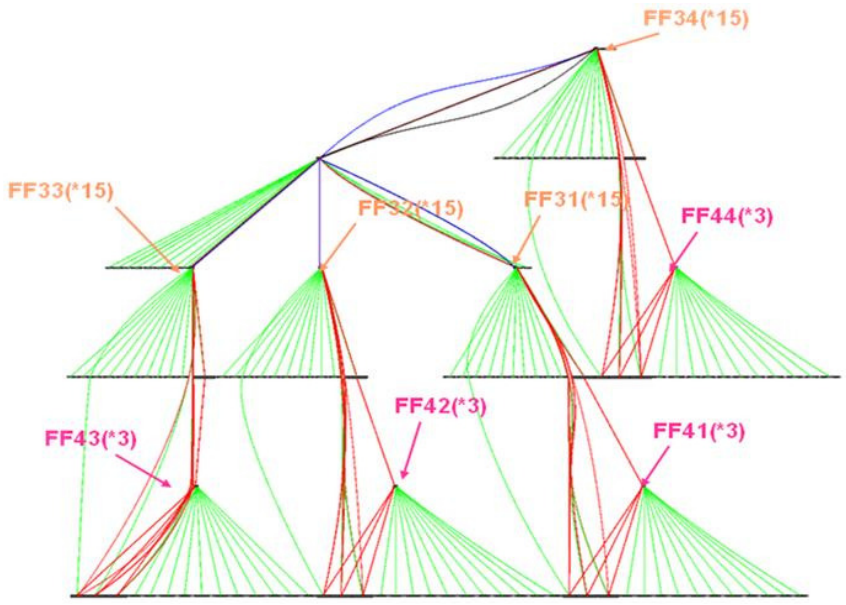


Figure 11: the neighborhood graph corresponding to a defactorization by four

5. FPGA BASED SYSTEM FOR THE MARCHING CUBES

Since the processing core is concentrated in a voxel, we propose to keep the processing block of an optimal cube made by SynDEx-IC like IP and integrated into a platform created for the 3D reconstruction (Marching Cubes), as described below by Figure 12.

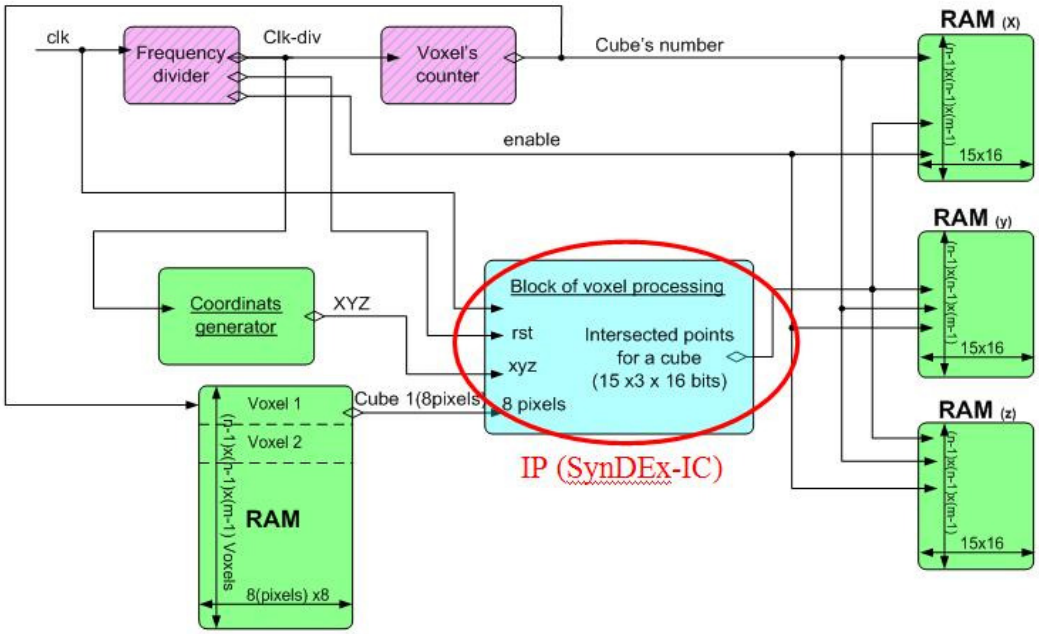


Figure 12: Block schema of the Marching Cubes algorithm

Table 1 : the implementation resource of blok processing of on cube on Virtex 4

	Sequential algorithm		Optimization algorithm (DF4)	
	Used	Rate recovery	Used	Rate recovery
Area(Slice)	1675	27%	3713	60%
Flip/Flop	1704	13%	3024	24%
DSP	2	6%	8	25%
Latency	23 cycles		9 cycles	

We subdivise the Marching Cubes algorithm into two modules named respectively: Managing Memory and Calculation of triangles. The memory management system can store the image slices into a RAM. Every eight intensity pixel values forming a voxel will be loaded into a line. This ensures the processing of voxels, one by one. The intensities of the pixels and their coordinates (x, y, z) are coded by 8 bits. Z is the number of slices which are been processing.

The architecture of the memory management module, allows reading intensities of 8 pixels (4 pixels of each section). The eight intensity values are then put into a comparator to be compared to a threshold called iso-value (this is the segmentation image by thresholding in real time). An index will be generated by the comparator noted cube_index coded by 8 bits. The coordinates (x, y, z) of each pixel will be used in the second part of the algorithm for calculating the coordinates of the intersection points and the triangles creation of the interested surface.

To respect the latency constraint, the heuristic identified the optimal defactorization factors of each frontier component on the critical path. In our example, the heu heuristic has opted for a implementation defactorization both frontiers FF2 and FF3 by a factor of 4.

Table 1 presents the number of sections occupied by the processing block of a cube made by SynDEx-IC as IP for two different constraints on Virtex4 (XC4VLX15).

6. CONCLUSION

From an algorithmic specification of factorized and conditioned data dependence graph (GFCDD), we have adopted the AAA methodology and software SynDEx -IC to optimize the implementation of the 3D reconstruction algorithm "Marching Cubes" on programmable circuit.

We exploited the aspect of repetitive mode in this algorithm for a better factorization. This approach allowed the synthesis of the implementation of the Marching Cubes algorithm by imposing latency constraints.

We have showed the benefit of using the AAA methodology and the associated software tool to specify and explore the different possible architectural solutions and to select the optimal solution which corresponds to the target application. By using a Virtex 4 (XC4VLX15) device, the best match of the location of the Marching Cubes Algorithm for a cube processing is a fully factorized architecture whose execution time is equal to 43 81 776 ms at a frequency of 50 MHz. By

implementing defactorized one (defactorization degree equal to 4) the execution time is reduced to 16.43166 ms at a same frequency for 24 slices of size 64 x 64.

REFERENCES

[1] William E. Lorensen and Harvey E. Cline. (1987) "Marching cubes: A high resolution 3D surface construction algorithm," *Computer Graphics*, 21(4), 163–169.

[2] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald & W. Stuetzle (1992) "Surface Reconstruction from Unorganized Points". *Proceedings of Computer graphics*, pp 71 78.

[3] D. Marr et E. Hildreth (1980) "Theory of Edge Detection", *Proc. R. Soc. London*, B207, pp:1187-217.

[4] Boubaker Mohamed, Akil Mohamed, Ben Khalifa Khaled, Bedoui Mohamed Hedi, (2007) "Architecture dédiée pour le traitement temps réel des signaux physiologiques: spécification algorithmique et méthodologie d'implantation sur circuits reconfigurables" *Colloque GRETSI 2007, Troyes.*

[5] T. Grandpierre, Y. Sorel. (2003) "From Algorithm and Architecture Specifiations to Automatic Generation of Distributed Real-Time Executives: a Seamless Flow Graphs Transformations", *IN Procs. IEEE Formal Methods and Models for Codesign Conference (MEMCODE'2003)*, pp: 123-133, Mont Saint-Michel, France.

[6] L.Kaouane, M.Akil, Y.Sorel, T.grandpierre, (2004) "A methodology to implement real-time applications on reconfigurable circuits", *Special issue on Engineering of Configurable Systems of the Journal of Supercomputing, Kluwer Academic Publisher*, Vol.30, No.3.

Authors

Manel MILI received the Bachelor's and Master's Degree in the National Engineering School of Sfax, University of Sfax, in 2007 and 2008 respectively. Since 2009, she has been working as a Research Scientist at the Research team of Medical Imaging Technology (TIM), Faculty of Medicine at Monastir University of Monastir where she prepares his thesis. She has also four published paper in international conference in the same areas. His areas of interest include dedicated architecture of medical image processing.

Bouraoui MAHMOUD received the Bachelor's, Master's Degree and doctorate in the Faculty of Science at Monastir from Mastir University, in 1997, 1999 and 2006 respectively. Since 2008, He is currently Assistant Professor in the Department of Industrial Electronics in the National Engineering School of Sousse, University of Sousse. He has four published papers in international journals. His areas of interest include embedded processor, embedded system, medical imaging, codesign HW/SW.

Mohamed Hedi BEDOUI Since 2011, He is currently Professor in Biophysics in the Faculty of Medecine at Monastir from Monastir University. He has many published papers in international journals. His areas of interest include Biophysics, medical imaging processing, embedded system, codesign HW/SW.