ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

Volume 3, Special Issue 3, March 2014

2014 International Conference on Innovations in Engineering and Technology (ICIET'14)
On 21st & 22nd March Organized by

K.L.N. College of Engineering, Madurai, Tamil Nadu, India

Implementation of Image Processing Algorithm Using Partial Dynamic Reconfiguration in FPGA

C.Kalyana Sundaram1 , M.Elango2, P.Marichamy3

Asst. Prof., Department of ECE, ,2Mepco Schlenk Engineering College, Sivakasi, Tamilnadu, India

M.E. Student, Department of ECE, ,2Mepco Schlenk Engineering College, Sivakasi, Tamilnadu, India

Prof., Department of CSE, ,3 P.S.R Engineering College, Sivakasi, Tamilnadu, India

Abstract— High Performance Reconfigurable Computing (HPRC) is parallel computing systems that contain multiple Microprocessors and multiple FPGAs. In current settings, the design uses FPGAs as coprocessors that are deployed to execute the small portion of the application that takes most of the time, under the 10-90 rules, the 10 percent of code that takes 90 percent of the execution time. FPGAs can certainly accomplish this when computations lend themselves to implementation in hardware, subject to the limitations of the current FPGA chip architectures and the overall system data transfer constraints. Hardware reconfigurable devices that change their configurations under the control of a program can replace the FPGAs to satisfy the same key concepts behind this class of architectures. Hence FPGAs are the currently available technology that provides the most desirable level of hardware reconfigurability. This paper focuses on Image enhancement and Denoising process, reference to point processing methods such as brightness manipulation, threshold operation and median filtering operation in FPGA with Onboard reconfiguration and partial dynamic reconfiguration. When using a partial dynamic reconfiguration performance of FPGA increases and the resource requirement decreases.

Keywords— Partial Dynamic Reconfiguration, On Board Reconfiguration, Image Enhancement and Denoising.

I. INTRODUCTION

The digital image processing is impacted today in some way with a very large area of technical endeavor. Digital image processing is used in very large and expanding areas covering applications in multimedia services, arts, medicine, space exploration, surveillance, authentication, automated industry inspection and many more areas.

These kinds of applications involve different processes, including image quality enhancement and object detection. It is important to note that this kind of processing ask for appropriate resources, as memory availability or specific attached peripheral devices out of processing power needs. All these are not available on all ordinary general purpose computers and many times the related procedures are not very time efficient due to other additional constraints. Using application specific hardware as ASICs or FPGAs a much greater efficiency can be obtained compared with a Software implementation. The VLSI (Very Large Scale Integrated) technology offers today complex alternative hardware implementation solutions.

The use of configurable hardware and system level programming languages allow direct implementation of image processing algorithms with improved performances and with a short time-to-market interval.

There are many technologies available for hardware implementation. Application Specific Integrated Circuits (ASIC) and Field Programmable Gate Arrays (FPGA's) are both reconfigurable devices able to support techniques such as parallelism and pipelining. The use of reconfigurable hardware to implement algorithms for image processing minimizes the time-to market cost while rapid prototyping with simplified debugging and verification stages is also possible. Therefore, the reconfigurable devices seem to be the ideal choice for implementation of image processing algorithms.

Field Programmable Gate Arrays have traditionally been configured by hardware designers using specific so called Hardware Design Languages (HDLs). There are already few such languages available offering different levels of abstraction but the most important ones are Verilog HDL (Verilog) and Very High Speed Integrated Circuits (VHSIC) HDL (VHDL).

II. BLOCK MEMORY GENERATOR

The Block Memory Generator LogiCORE™ IP core automates the creation of area and performance optimized block memories for Xilinx FPGAs. Available

through the ISE® Design Suite CORE Generator™ System, the core enables users to create block memory functions to suit a variety of requirements. Built in knowledge about Xilinx device architectures allow it to leverage specialized FPGA architectural features to create the most compact, high performance solution. The Block Memory Generator core uses embedded Block Memory primitives in Xilinx FPGAs to extend the functionality and capability of a single primitive to memories of arbitrary widths and depths. Sophisticated algorithms within the Block Memory Generator core produce optimized solutions to provide convenient access to memories for a wide range of configurations.

A. Single Port Block Memory

The Single-Port Block Memory module (Fig.1) is generated based on the user-specified width and depth. This module for Spartan-II and Virtex is composed of single or multiple 4 Kb blocks called Select RAM+. The Virtex-II, Virtex-II Pro, Virtex-5, and Spartan-3 Single-Port Block Memory modules, on the other hand, are composed of single or multiple 18 Kb blocks called Select RAM- II. Since Spartan-II and Virtex both use the 4 Kb Select RAM+ blocks, any particular reference to a Virtex implementation also applies to a Spartan-II, Virtex-V, Virtex- II Pro, or Spartan-IIE implementation. When Block Memory is enabled, all memory operations occur on the active edge of the clock input (CLK). The Block Memory can be configured to be active on the rising edge and the falling edge. When the block memory is disabled (enable inactive), the memory configuration and output value remain unaltered.

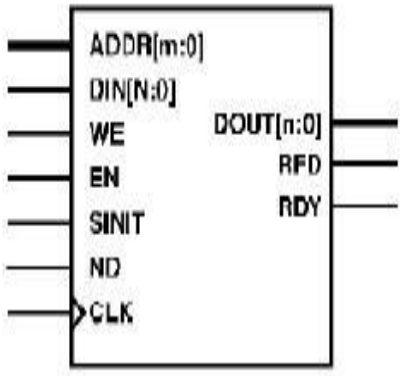

Fig. 1 Core Schematic Symbol

During Write Operation (WE asserted), the data presented at the port's data input is stored in memory at the location Selected by the port's address input.

B. Dual Port Block Memory

The Dual Port Block RAM (Fig.2) has two independent access ports that permit shared access to a central pool of memory. The data width and memory depth of each access port can be independently configured providing straightforward dual-port memory functionality or optional data formatting capability. Both ports are Partial reconfiguration (PR) is the ability to write access. Simultaneous reads from the same memory location may occur, but all other simultaneous, same location operations should be avoided. Simultaneously reading from and writing-to the same location results in the correct data being written into the memory, but invalid data being presented at the reading port.

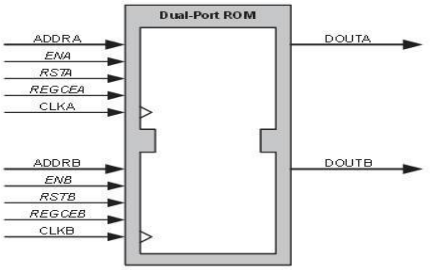

Fig. 2 Core schematic symbol

III. IMAGE STORAGE

The core generation process produces the logic for the core, partitions it into configurable logic blocks (CLBs), and then places the CLBs relative to each other. This logic design coupled with a CLB-floor-plan or physical design is what makes our cores predictable. The relative locations are maintained as the core is integrated into the overall design and placed anywhere in a large FPGA.

"Block Memory Generator" is one of the IP cores that are provided by the Xilinx IP Core Generator which is present under the "Memory and Storage Element". Block Memory Generator wizard is launched showing different memory types and IP Symbol."Single Port ROM" is selected under memory types and the values for read width and read depth are specified based on horizontal width and vertical length of the Image that is being processed. After defining all of your parameters, simply click on the Generate button. The output is an optimized CORE for the targeted FPGA device.

The size of the image that is available is 512x512 and this data is stored as .coe file in single port Block ROM using Xilinx Core Generator. Mat lab function IMG2coe8 (imgfile, outfile) is used to convert bmp, jpeg, png, gif and tif image file formats to .coe file format.

IV. DYNAMIC PARTIAL RECONFIGURATION

Dynamic Partial Reconfigurable (DPR) FPGAs offer new design space with a variety of benefits: reduce the configuration time and save memory as the partial reconfiguration files (bit streams) are smaller than full ones.

reconfigure select areas of an FPGA any time after its initial configuration. Dynamic partial reconfiguration is carried out to allow the FPGA to adapt to changing hardware algorithms, improve fault tolerance and resource utilization, to enhance performance or to reduce power consumption. DPR is especially valuable where devices operate in a mission critical environment that cannot be disrupted while some subsystems are being redefined.

In DPR two bit streams are generated, they are

partial and full bit stream. Full bit stream contains whole main program, partial bit stream contains the program which going to be replaced while the full stream is processed.

In this paper Image enhancement and denoising process done by using dynamic partial reconfiguration. Here the Image enhancement and denoising programs are partial bit files. When running the main bit file we can switch over from one process to another process by changing the download process of partial bit files. The initial steps in the PR design flow are similar to the initial steps in the standard modular design flow.

A. Hardware Description Language (HDL) Design and Synthesis

1) Top-Level design: The top-level module that does not contain any logic only contains I/O instantiations, clock primitives, static module instantiations, partial reconfiguration module instantiations, and signal declarations. In addition, the top-level module must define bus macros. The based design and each Partial Reconfiguration Module (PRM) must be connected through the bus macro.

2) Base design: The static modules contain logic that will remain constant during reconfiguration. This step is same with traditional HDL design method, but the static modules cannot contain any clock or reset-related primitives. In the proposed design, the control manager is implemented by the based design.

3) PRMs design: Similar to the static modules, the partial reconfiguration modules must also not contain global clock signals, but may use those from the top-level module. When designing multiple reconfigurable modules to utilize the same reconfigurable area, the component name and port reconfiguration of each module must match the reconfigurable module instantiation located in the top-level module. The proposed system has two PRMs which are Image Enhancement and Denoising process.

B. Set Design Constraints

After the HDL design description and synthesis, the next step is to set design constraints. Design constraints include the area group, reconfiguration mode, timing constraint and location constraints. The area group constraint specifies which modules in the top-level module are static and which are reconfigurable. Each module instantiated by the top-level module is assigned to a group. The reconfiguration mode constraint is also only applied to the reconfigurable group, which specifies that the group is reconfigurable. Location constraints must be set for every pin, clocking primitive, and bus macro in the top-level design.

C. Implement Base Design

Before the static modules are implemented, the top-level is translated to ensure that the constraints file has been properly created. The information generated by implementing the base design is used for PRM implementation phase. The base design implementation follows three steps: i.e., translate, map and Place & Route (PAR).

D. Implement PRMs

After the base design is implemented, each PRM must be implemented separately and follows base design implementation steps: translate, map, and PAR.

E. Merge

The final implementation phase is the merge phase. During the merge phase, a complete design is built from the base design and each PRM. In this step, many partial bit streams for each PRM and initial full bit streams is created to configure the FPGA.

V. IMAGE ENHANCEMENT METHODS

One of the most spectacular and interesting image processing approaches is the image enhancement. The interest for this domain stems from two principal application directions:

1. Improve the human interpretation and enhance the pictorial visual information;
2. Modify the data structure of image representation in order to optimize it for data storage, transmission or other representation for autonomous machine perception.

The main goal of any enhancement method is too obtain a more suitable result compared with the original as is from the point of view of a specific application. Any image enhancement procedures can be categorized into two approaches methods: spatial domain methods and frequency domain methods. The *spatial domain* refers to the pixels structure of the image plane itself and this kind of enhancement is based on direct manipulation of those pixels of an image. *Frequency domain* processing techniques are using mathematical transforms to induce different enhancements.

The Fourier transform of an image is well known for these purposes. Some of the simplest, yet useful, image processing operations in the spatial domain involves the adjustment of brightness, contrast or color in an image. A reason for manipulating these attributes is the need to compensate for difficulties in image acquisition and with image processing we can increase the overall brightness of the object of interest and magnify the tiny residual variations in contrast across it. This image processing operations can reveal enough detail to allow proper interpretation. Point operations perform a modification of the pixel values without changing the size, geometry, or local structure of the image. The pixel value is given by $a = I(u, v)$ which depends exclusively on the previous value $a = I(u, v)$ at the same position. To map the original pixel values to the new values a function $f(a)$ is used,

$$a' \leftarrow f(a) \qquad I'(u,v) \leftarrow f(I(u,v)) \tag{1}$$

For each image point with (u, v) coordinates. When the Function $f()$ is independent of the image coordinates, the operation has the name *global* or *homogeneous*. The hardware description language Verilog HDL was developed to carry out readings and writings of files with ASCII characters and it does not allow to process bitmap or jpeg files. For that reason, it is necessary to represent

binary information with ASCII characters in the hex format.

A. *Brightness Manipulation*

A dark region in an image may become brighter after the point operation and the common used point operation are increasing and decreasing of brightness. If an operator takes each pixel value and adds a constant number to it, then this point operation increases the brightness of the image and similar subtraction operator reduces the brightness. Brightness improvement operation improves the human perception of an image so that the viewer can view the information which present in the dark region. This shows in Fig.6.

Fig.3 shows internal architecture of image enhancement technique shows the architecture for both brightness operation and threshold operation. Upper portion of the architecture shows the threshold operation such as average value calculation and comparison operations. Lower portion of the architecture shows the brightness operation such as values add function and comparison operation.

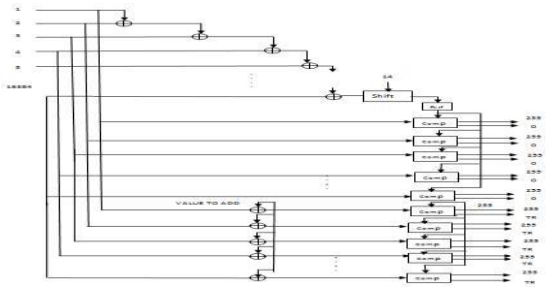

Fig. 3 Internal architecture of image Enhancement Block (IEB) without parallel process

When using an image enhancement technique without parallel process then the single port ROM used for read the pixels of an image. But when processing pixels serially that took larger time. This problem can be overcome by using a parallel process technique for that dual port ROM used for read the pixels. That serves the two pixels at the time so that time required for process reduced into half. When using two dual ports ROM that will serve the four pixels at the time and process time reduced into another half.

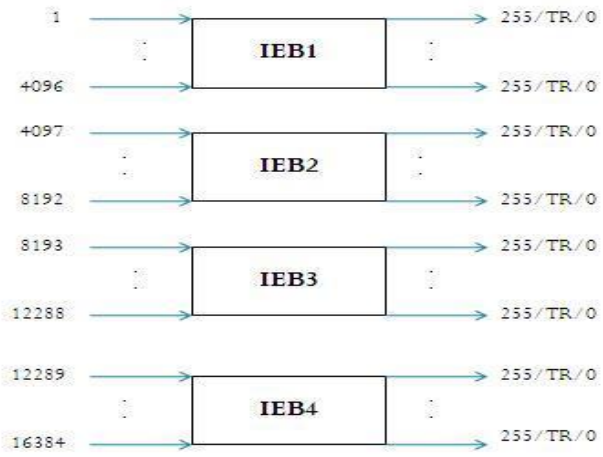

Fig. 4 Architecture for Image Enhancement with Parallel Process

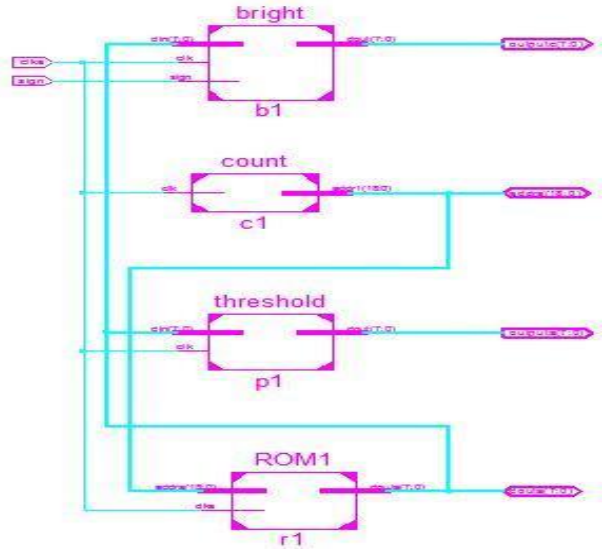

Fig. 5 RTL Schematic of Image Enhancement Technique without Parallel Process

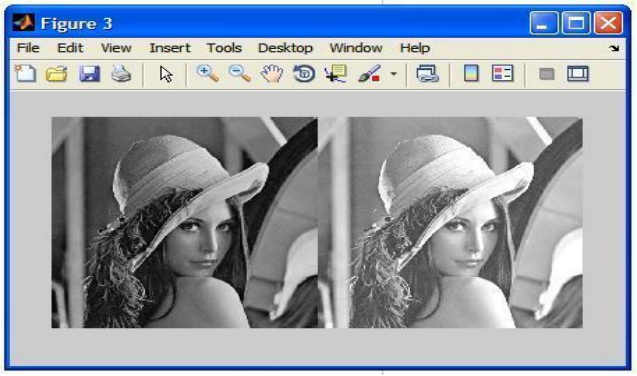

Fig. 6 Verilog result for Brightness Operation using *value =60(right image).*

B. *Threshold Operation*

Thresholding operations are particularly interesting for segmentation in the process of isolating an object of interest from its background.

Thresholding an image means transforming all pixels in two values only. This is a special type of quantization comparing the pixel values with a given threshold value *a*th that is usually constant. That allows the separate the pixel values in two classes. The described threshold function *f*threshold (*a*) has to map all pixels to one of two fixed intensity values *a*0 or *a*1.

A black and white image (binary) image can be obtained from a grayscale image through a thresholding operation. In our implementation we use the thresholding with a fixed threshold value of a grayscale image. The thresholding operation will be performed by scanning the values of each pixel from the input image and replacing the corresponding pixel in the destination image using a0 = 0 and a1 = 255. The value of the threshold can be established inline Verilog testing code.

C. *Median Filter*

A median filter is a non-linear digital filter which

is able to preserve sharp signal changes and is very effective in removing impulse noise (or salt and pepper noise). An impulse noise has a gray level with higher or lower value that is different from the neighborhood point. Linear filters have no ability to remove this type of noise without affecting the distinguishing characteristics of the signal. Median filters have remarkable advantages over linear filters for this particular type of noise. Therefore median filter is very widely used in digital signal and image/video processing applications. A standard median operation is implemented by sliding a window of odd size (e.g. 3x3 windows) over an image. At each window position the sampled values of signal or image are sorted, and the median value of the samples replaces the sample in the center of the window.

Let W be a window with an odd number of points. Then the median filter is given by

$$ys = median\{xr + s : r \hat{I} W\} \qquad (2)$$

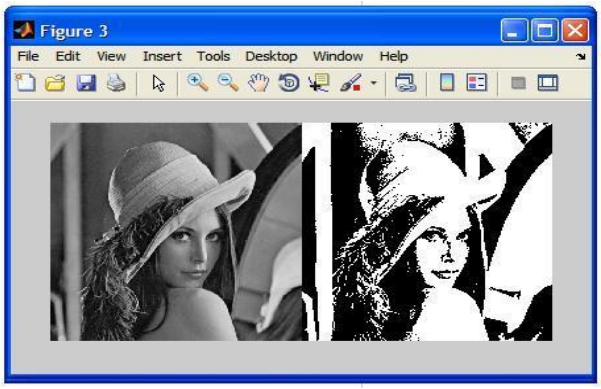

Fig. 8 Verilog result for Denoising using Median Filter operation (right image

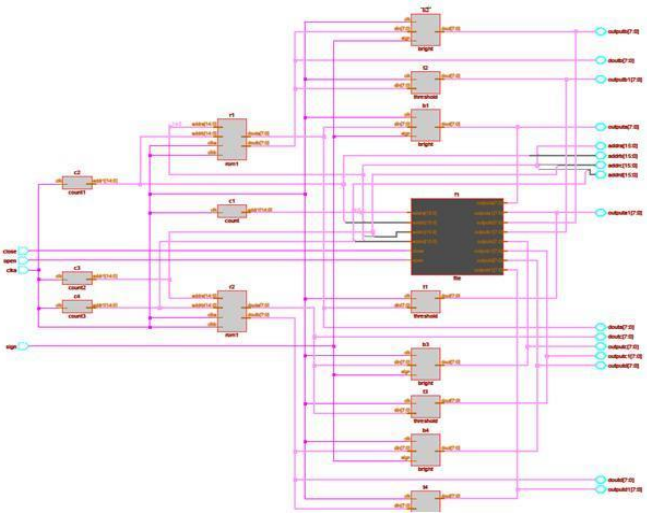

Fig. 7 Verilog result for Threshold Operation using *threshold =90(right image).*

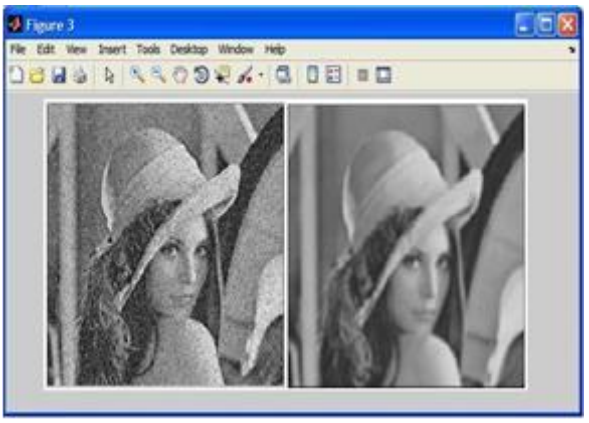

Fig.9 RTL Schematic of Denoising Process with Parallel Process

The main problem of the median filter is its high computational cost (for sorting *n* pixels, the time complexity is *O (n log n)*, even with the most efficient sorting algorithms). When the median filter is carried out in real time, the software implementation in general-purpose processors does not usually give good results. The execution times are reduced by implementing median filters on FPGAs.

VI. CONCLUSION

Firstly, the full bit stream (main program) is downloaded to initial the device, then, the partial bit streams (Image Enhancement or Denoising) are downloaded when the FPGA is running. Plan Ahead design tools as a platform for partial reconfiguration applications can greatly simplify the complexities of the dynamic operating environment, allowing a single device to operate in applications that previously required multiple FPGAs. Image enhancements methods offer a wide variety of approaches for modifying image to achieve visually improve images. The image enhancement operations were here described using Verilog hardware description language and the Verilog models were simulated using ISIM Simulator from Xilinx ISE Design Suite. Verilog can't handle the standard image formats so the images were converted to a Coe-file; The Coe-file was applied as vector to the Verilog block models. The output file was similarly converted and viewed using the same Mat lab tool, to show the original image and the results of the enhancement methods. The image enhancement methods considered include brightness manipulation, threshold operation and denoising process. When doing the image enhancement process without parallel process then the speed of operation is slower, because here the pixels are processed serially. When including the parallel process in the image enhancement technique then the speed of operation is high but the device utilization also increases. Here the future work is to show a resultant image with the help of VGA (Video Graphics Array) connection.

REFERENCES

[1] Filippo Borlenghi, Dominik Auras, Ernst Martin Witte, Torsten Kempf, "*An FPGA-Accelerated Testbed for Hardware Component Development in MIMO Wireless Communication Systems*", IEEE TRANSACTIONS ON COMPUTERS, 2012.

[2] Raman Maini, H. Aggarwal - "*A Comprehensive Review of Image enhancement Techniques*", Journal of Computing, vol. 2, issue 3, ISSN 2151-9617, pp. 269-300, 2010.

[3] V. Baumgarte, G. Ehlers, F. May, A Nückel, M. Vorbach, M. Weinhardt, ―PACT XPP – "*A Self-Reconfigurable Data Processing Architecture*", ǁ in J. Supercomputing, pp. 167-184 26, 2012.

[4] K. Benkrid, S.Belkacemi and S. Sukhsawas, "*An Integrated Framework for the High Level Design of High Performance Signal Processing Circuits on FPGAs*", In Proceedings of SPIE Opto-Ireland, Vol. 5823, No. 29, pg. 12, 2005.

[5] B. Hutchings et al., "*A CAD suite for high-performance FPGA design*", in Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machinespp.12-20, FCCM'99, 1999.

[6] Khaled Benkrid,," *High Performance Reconfigurable Computing: From Applications to Hardware*", IAENG International Journal of Computer Science, 35:1, IJCS_35_1_04,2010.

[7] Rubén Salvador, Andre´s Otero, Javier Mora, Eduardo de la Torre," *Self-Reconfigurable Evolvable Hardware System for Adaptive Image Processing*". IEEE TRANSACTIONS ON COMPUTERS, VOL. 62, NO. 8, AUGUST 2013.

[8] Victor Silva, Jorge R.fernandes, Mario P.Vestias," *A High-Performance Reconfigurable Computing Architecture using a Magnetic Configuration Memory*", IEEE TRANSACTIONS ON COMPUTERS, 2012.

[9] William K. Pratt - "*Digital Image Processing (fourth edition)*", A John Wiley & Sons, Inc. Publication, pp. 247-307, 2007.

[10] http://www.xilinx.com/ise/products/coregen_overview .pdf

[11] http://www.xilinx.com/tools/coregen.html.