Systolic FIR filter Based FPGA

A. GOUGAM and D. BENAZZOUZ
University M'Hamed Bougara - Boumerdes, Algeria.
Solids mechanical and systems laboratory (LMSS)
Signals and systems laboratory (LSS).
ahgougam@yahoo.fr, dbenazzouz@yahoo.fr

Abstract: **In this paper, we first review in detail the basic building blocks of reconfigurable devices, essentially, the field-programmable gate arrays (FPGAs), then we describes a high-speed, reconfigurable, Systolic FIR filter design implemented in the Virtex-II series of FPGAs. The VHDL description of this filter is used for simulation and EDIF for implementation using Xilinx's place and route tools. The VHDL simulation shows that the filter behaves as expected**

I. INTRODUCTION

As field programmable gate array (FPGA) technology has steadily improved, FPGAs have become alternatives to other technology implementations for high-speed classes of digital signal processing (DSP) applications. In this paper, we first review in detail the basic building blocks of reconfigurable devices, essentially, the field-programmable gate arrays (FPGAs), then we describes a high-speed, reconfigurable, Systolic FIR filter design implemented in the Virtex-II series of FPGAs.

II REPROGRAMMABLE COMPUTING AND THE FPGA ARCHITECTURE

Reconfigurable computing (RC) is computation using hardware that can adapt at the logic level to solve specific problems. Figure 1a shows the implementation spectrum in reconfigurable computing [1]. The spectrum is bounded by three axes symbolising performances, flexibility and cost. The figure clearly shows that ASIC gives high performance at cost of inflexibility, processor is very flexible but not tuned to the application and that RC hardware (FPGA) is a nice compromise.

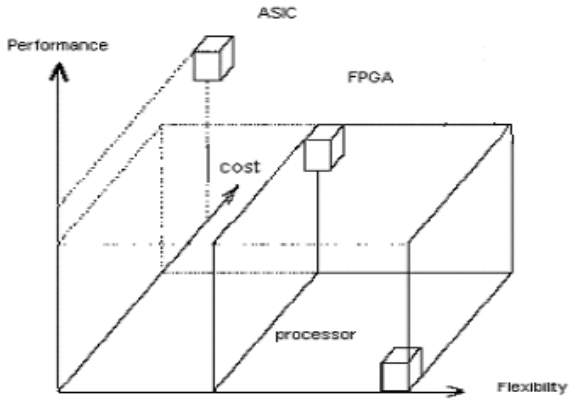

Figure 1a: Implementation Spectrum

Reconfigurable hardware can be classified according to their granularity level, which are: the system level, the functional level and the logic level.

At system level the reconfiguration correspond to the programming of the computing resources such the different processors and memory space.

At functional level, the reconfiguration focuses on the interconnections between resources such as the different arithmetic modules.

Finally, at logic level the reconfiguration deals with the different L.U.Ts and the network of interconnects at bit level.

In reconfigurable computing we distinguish two types of reconfiguration as shown in figure 1b:

- Static reconfiguration : the application is not running
- Dynamic reconfiguration – application is modified in response to changing environmental issues.

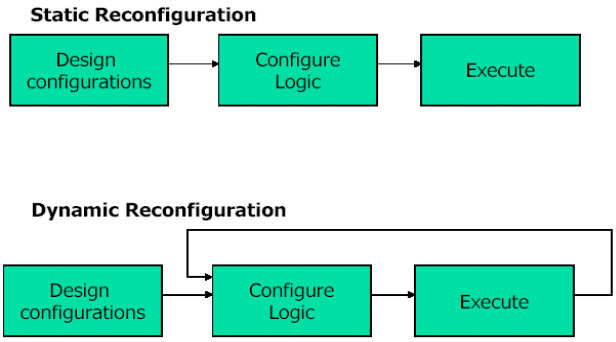

Figure 1b: Reconfiguration types

The basic structure of an FPGAs is *array-based*, meaning that each chip comprises a two dimensional array of logic blocks that can be interconnected via horizontal and vertical routing channels. An illustration of this type of architecture is shown in Figure 2.

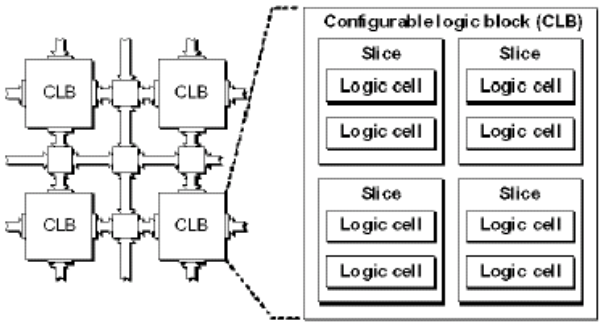

Figure 2: FPGA: Basic Structure

The features of a logic block (called a *Configurable Logic Block* (CLB) by Xilinx) shown in figure 3 is based on look-up tables (LUTs) . A LUT is a small one bit wide memory array, where the address lines for the memory are inputs of the logic block and the one bit output from the memory is the LUT output.

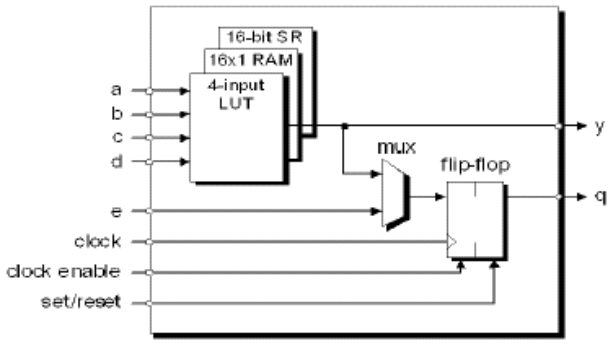

Figure 3: Configurable Logic Block (CLB)

Modern VLSI FPGAs architecture shown in figure 4 are characterized by the integration of different building blocks [2] such as:

- Logic cell (Combinational and Sequential) .
- Dedicated Arithmetic Logic, processors,
- Input/Ouput, JTAG, Gigabits transceiver blocks ,

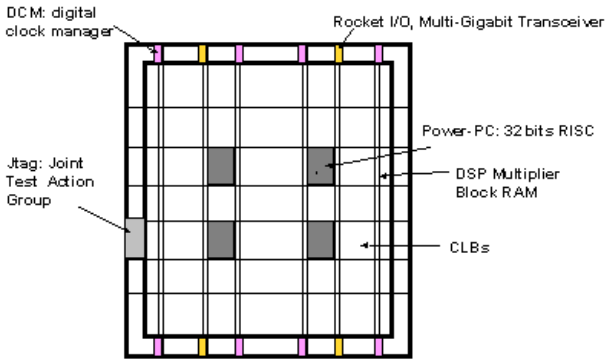

Figure 4: FPGA with embedded system functionality

III SYSTOLIC FIR APPLICATIONS

Systolic system consists of an array of processing elements (typically multiplier-accumulator chips) in a pipeline structure that is used for applications such as image and signal processing. The "systolic," introduced by H. T. Kung of Carnegie-Mellon in 1978, refers to the rhythmic transfer of data through the pipeline, like blood flowing through the vascular system [3,4].

Systolic approach can speed up a compute-bound computation in a relatively simple and inexpensive manner. A systolic array in particular achieves higher computation throughput without increasing memory bandwidth as shown in figure 5.

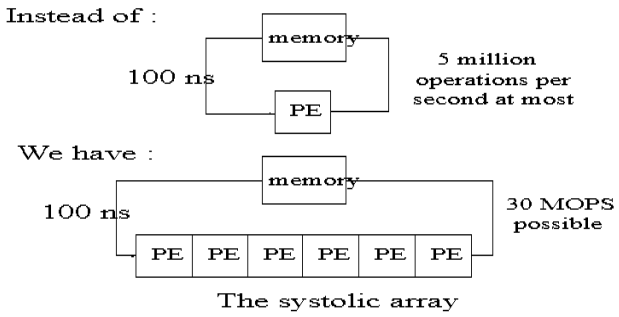

Figure 5: systolic array throughput

One such application is the well known finite impulse response (FIR) digital filter, also known as the transversal filters or as a tapped delay line. The behavior of the finite impulse response filter can be described by the equation:

$$y_t = \sum_{k=0}^{N-1} a_k x_{t-k} \qquad (1)$$

where y_t denotes the output at time t and x_t represents the input at time t and a_k are the filter coefficients.

The processing element of the 1D semi systolic FIR is shown in fig 5. The inner product processing element will take as inputs an accumulated sum from previous processing elements (y_{in}), a filter coefficient (a_i) and a sample value from the input stream (x_{in}) and return two values: the x_{in} is passed to x_{out} and the y_{out} is computed by performing the inner product calculation and adding it to the accumulated sum.

$$y_{out} = y_{in} + a_i * x_{in}. \qquad (2)$$

An implementation for an inner product processor is shown in figure 6. Both the x values and the accumulated results flow from left to right. Registers are added at the inputs and outputs for pipelining in a way that makes sure the accumulated sums and x values stay in synch.

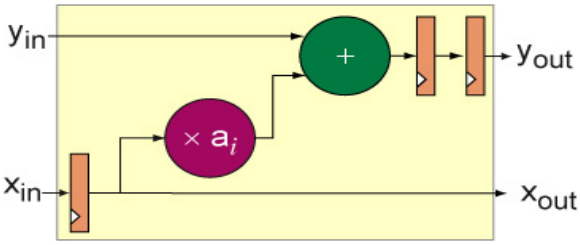

Figure 6: FIR processing element

An example of a four tap filter using this processing element is shown in fig 6. This is formed by simply replicating the processing element horizontally. The x input has to be delayed by one clock tick to synchronise with the y inputs. This filter has a much higher latency (8 ticks) than its direct implementation counterpart (semi systolic filter).

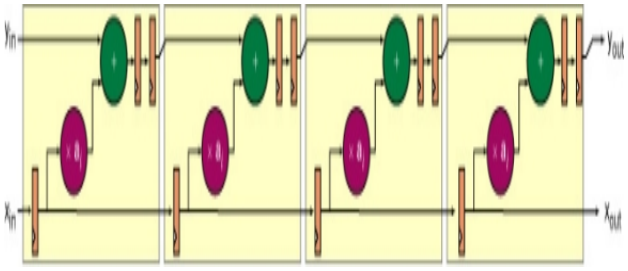

Figure 7: A 4 taps full systolic FIR filters

VHDL description of this filter is used for simulation and EDIF for implementation using Xilinx's place and route tools. The VHDL simulation shows that the filter behaves as expected as shown on figure 8, a serie of unsigned number b[15:0] produced a same serie of filtered unsigned numbers q[15:0] . The first filtered data output is produced after the eigth clock pulse then, an output is produced every clock pulse. The coefficients Wi, of the filter have been set to 1 for simplicity.

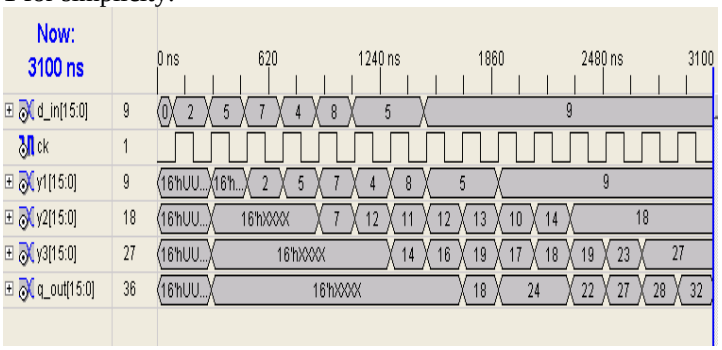

Figure 8 Simulation results using ISE 8i.

After configuration, ON CHIP debugging and verification is performed using ChipScope™ Pro tools which integrates logic analyzer hardware components with the target design inside Xilinx Virtex™. The ChipScope Pro tools communicate with these components and provides us with a complete logic analyzer. Figure 9 shows a block diagram of a ChipScope Pro system. We can place the ICON, ILA, cores (collectively called the ChipScope Pro cores) into the design by generating the cores with the ChipScope Pro Core Generator and instantiating them into the VHDL source code. The design is then placed and routed using the Xilinx ISE 8.1i implementation tools. Next, we download the bitstream into the device under test and analyzes

the design with the ChipScope Pro Analyzer software.

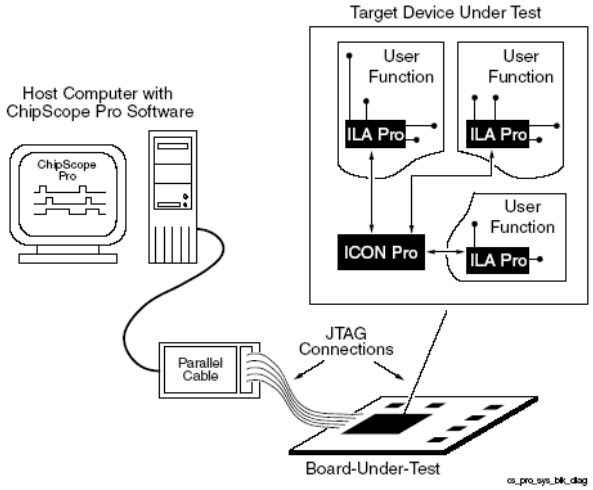

Figure 9 block diagram of a ChipScope Pro .

IV CONCLUSION

Our study shows the suitability in using FPGAs for spatially parallel applications such as systolic filters. By illustrating a design methodology for digital filters, the

advantages of using FPGAs for digital signal processing applications (DSP) are emphasized.

Finally, further works on this subject is being carried out including :Making the filter parameters scalable as the number of taps and the coefficients.Extending this systolic FIR filter to real data acquisition.

- Comparing performances against existing optimized filter implementation from Xilinx's Core Generator in order to propose a real IP (Intellectual Property) core for reuse .

V REFERENCES

[1] R. David , Architecture reconfigurable dynamiquement pour applications mobiles ‘, Thèse ,Université de Rennes, 2003.

[2] The Design Worrior’s Guide to FPGAs Devices, Tools, and Flows. ISBN 0750676043, Mentor Graphics Corp, 2004,

[3] H.T Kung, C.E Leiserson, “ Systolic Array for VLSI ”, Sparse Matrix Proc., 1979, pp. 256-282.

[4] A GOUGAM, A FARAH. “Systolic Arrays via dependancy graphs”. Journal of Technology (JOT), E.N.P El-Harrach, 1994