A Rapid Reconfigurable VLIW co-Processor for Ternary Emulation of Digital Designs

Flávio Miana[*1] miana@cpdee.ufmg.br

Wilton Padrão[*2] wpadrao@dcc.ufmg.br

Jones Oliveira[*2] joa@dcc.ufmg.br

Antônio O. Fernandes[*2] otavio@dcc.ufmg.br

Julio Cezar de Melo[*1] demelo@cpdee.ufmg.br

Claudionor N. Coelho Jr[*2] coelho@dcc.ufmg.br

Abstract

We present a cycle-based rapid reconfigurable VLIW (Very Long Instruction Word) bit-wide co-processor which allows ternary logic emulation. This approach allows the speed up of design process since debugging a hardware design may be costly if the designer have to iterate many times in the design and simulation steps. This VLIW processor efficiently emulates the hardware behavior verifying the system's functionality much faster than conventional simulation tools.

1. Introduction

The common steps during a hardware design are project specification, design, simulation, prototyping, and production[4]. In Figure 1, these steps are described according to a design time line. During the design of a digital circuit, we may have to iterate several times in the *Design*, *Simulation*, and *Prototyping* steps to debug a system. Therefore, it is imperative that these steps are executed quickly to speed up the design time.

Hardware emulation makes possible to verify the system's behavior by a prototype that is equivalent to the intended design. Recent advances in Field-Programmable Gate Array (FPGA)[1][2] made possible an increase in use of hardware emulation in order to reduce simulation time in up to six orders of magnitude than classical simulation[3].

One can see that FPGA may reduce the design time of processor designs[12]. The key for this new approach takes place because new FPGAs can be configured very quickly, allowing a whole processor to be mapped to a set of FPGAs emulating the

[*1] Electrical Engineering Department – Universidade Federal de Minas Gerais, Brazil
[*2] Computer Science Department – Universidade Federal de Minas Gerais, Brazil

processor's behavior. Thus, whenever it is necessary to change some definition in the design, the hardware can be reconfigured in milliseconds[2]. However, it may be slow to be efficiently used during the debugging cycle presented in Figure 1.

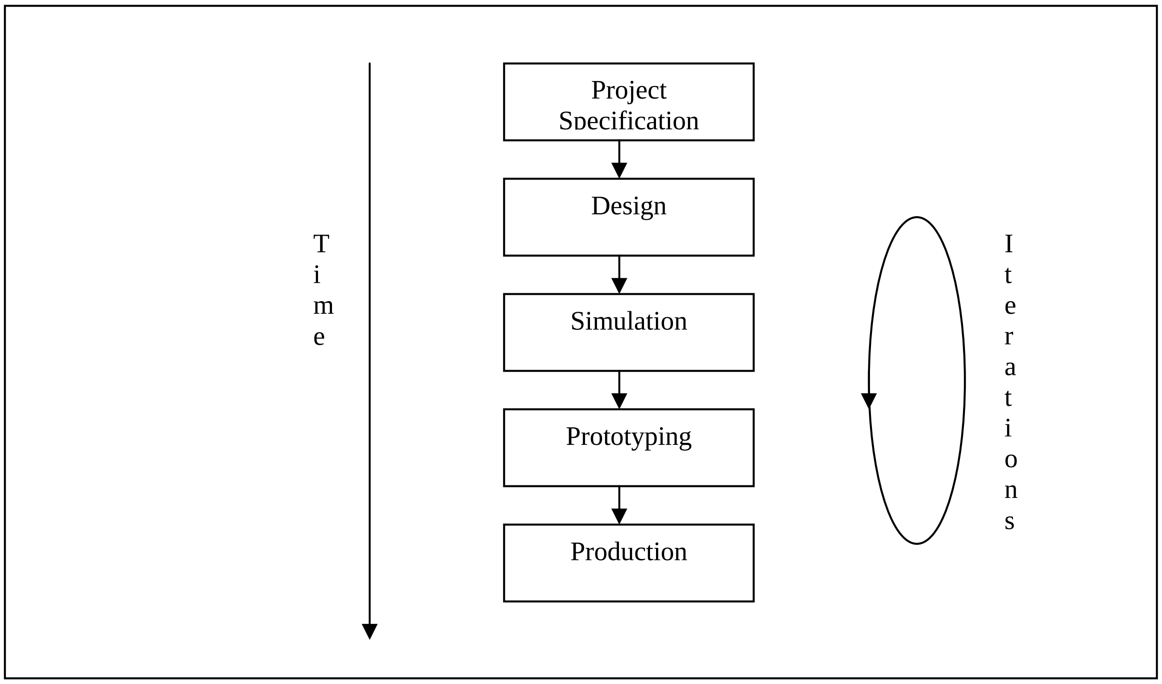


Figure 1. Steps of a Hardware Design

There are different levels of reconfiguration, such as coarse or fine-grained. At coarse-level (also named as process-level), whenever the application context has to be changed, the system is reconfigured, taking an order of milliseconds. On the other hand, each phase of a computation may require different hardware, requiring reconfiguration times on the order of microseconds. This kind of reconfiguration is called in the literature block-level or fine-grained reconfiguration.

We propose here a new type of reconfiguration whose reconfiguration time is higher than process-level reconfiguration, but with a number of configurations so high that we cannot store them or synthesize them each time a new program/process is invoked. In order to speed up simulation time, this technique is used in the following way. We have to separate the reconfiguration into two phases. The first phase comprises the generation of the hardware that can be used for all different configurations. In the second phase, the specific data configuration is generated quickly.

For the first phase, we synthesize a Rapid Reconfigurable Very Long Instruction Word (VLIW) bit-wide co-processor that allows ternary logic emulation. The VLIW co-processor emulates a digital design at a cycle level. This VLIW processor is comprised of multiple ALUs performing any two-input/one-output ternary functions. Ternary logic means a third *unknown* or *indeterminate logic value* (usually named X) is added to the binary logic set {0,1}. The third value X can be used to reduce the number of cases of the system to be tested[5] by encapsulating values that are *unknown* or *indeterminate* to the co-processor. For the second phase, on each new iteration, we generate code for this VLIW processor that emulates the circuit behavior. Our approach for design development is represented in Figure 2. We eliminate the hardware synthesis step from the *Iterations* of Figure 1 because it would tremendously increase the iteration time. In this way, we can quickly reconfigure a new design version into the emulation system without synthesizing a new chip.

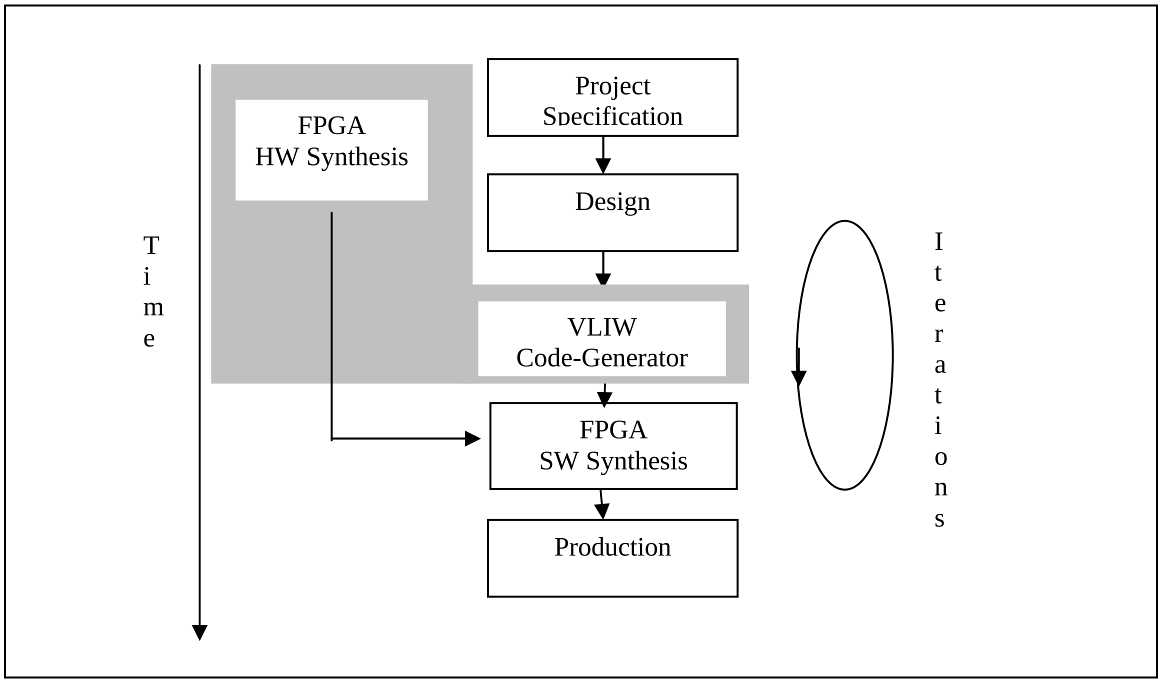


Figure 2. Emulation Cycle Using Rapid Reconfigurable VLIW co-Processor

This paper is organized as follows. In Section 2, we present a briefly description of the Development System XC6200 board. The architecture of the co-processor is presented in Section 3. In Section 4 and 5 we present the software support and the system integration, respectively. Finally, Section 6 concludes the paper and suggests some future work.

2. Xilinx Development System XC6200

This section briefly presents the XC6200 Development System that is used as the emulation engine proposed in this paper. For further information, referred to [6].

The XC6200 Development System (XC6200DS) offers some features such as:

- PCI Based Development System;
- High speed Data Transfers to board memory;
- 16K User Programmable Gates with the XC6216;
- Up to 2 Mb fast SRAM;
- Flexible clock generation for Xilinx 6200;
- Debugger software supports validation of XC6200 based design and reduces time to market;

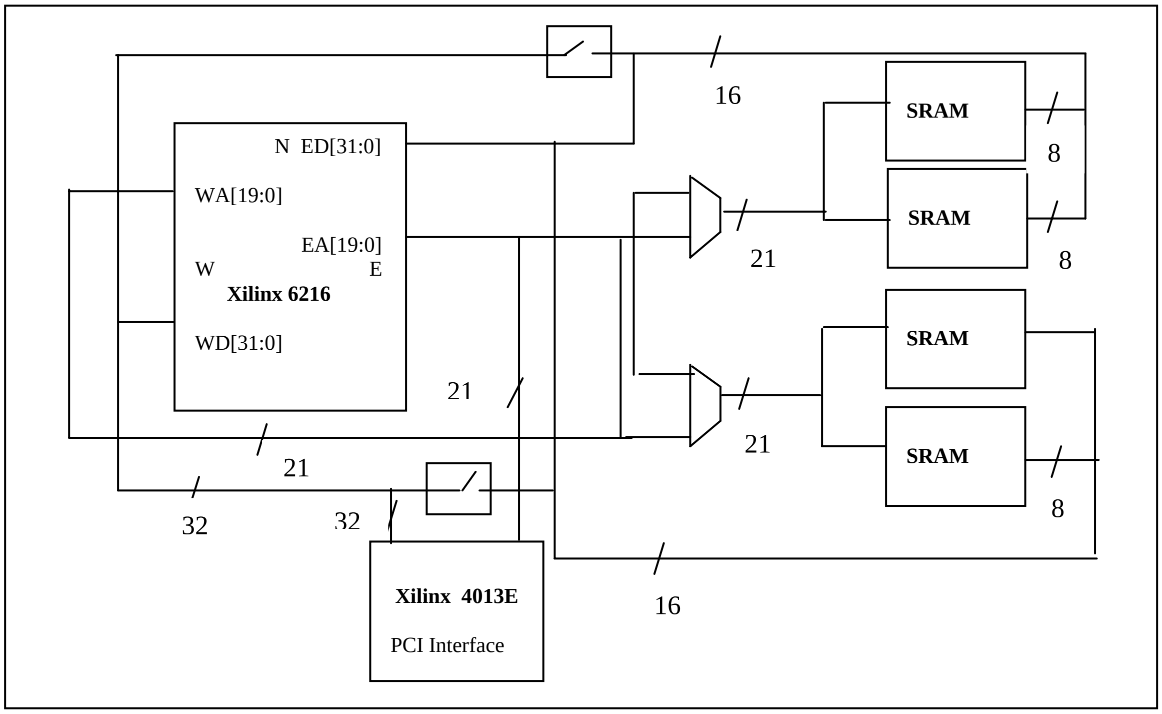
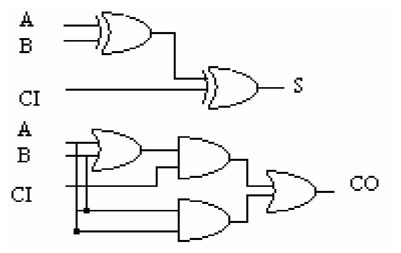


Figure 3. Board Architecture

The XC620DS is a high performance PCI based computing and input/output coprocessor for Win95 based PC's and consists of a Xilinx 4013E[6] and a compute element. The compute element is a Xilinx 6200 FPGA[7], four 8-bit wide SRAM's and six bus controller chips to control data flow. Figure 3 shows the primary components of the architecture. A Xilinx 4013E FPGA is used as the PCI bus interface. Approximately 50% of the chip are used for this function and the remaining area is used for card control logic. The 4013E FPGA is electrically and functionally 100% PCI compliant. For details of the PCI LogiCore see product description which is available separately from Xilinx[8]. The primary component of the compute element is the Xilinx 6216. The board architecture allows the XC6216 to be reconfigured through the PCI interface during run-time. The PCI interface provides direct access from the host PC to logic cells within the user circuit. The output of any cell's function unit can be read and the flip-flop within any cell can be written through the PCI interface. The compute element memory is organized into two banks. Each bank consists of a maximum of two 512K x 8 SRAM's (128K x 8 SRAM's are normally supplied). A bank of RAM can be accessed from either the signals to control the RAMs individually. The development system provides a flexible architecture in order to implement a wide variety of algorithms.

3. Bit-Level Description of the VLIW Reconfigurable co-Processor

The VLIW bit-wide co-processor allowing ternary logic emulation is a hardware cycle-based emulator that evaluates one basic block per cycle. We can describe a basic block as any sequence of instructions containing only assignments and, without any branching conditions. Each cycle of simulation corresponds to executing all assignments of the basic block. In Figure 4, we exemplify a basic block that simulates the logic behavior of a full adder circuit, where *A*, B and *CI (carry-in)* are the inputs of the circuit; *S (Sum)* and *CO (carry-out)* are the output and aux1,...aux5 are the internal wires of the circuit.



```
aux1 = A∧B;
S = aux1 ∧ CI;

aux2 = A │ B;
aux3 = aux2 & CI;
aux5 =  A & B;
CO = aux5 │ aux2;
```

Basic Block

Figure 4. Example of Basic Block

3.1 VLIW co-processor

The datapath of the co-processor is represented as in Figure 5. The block labeled ALU shows the four independent ALUs with two registers of two bits each (regALU). Although we represented the *Instruction Memory* and *Data Memory* in Figure 5, this is only one memory unit in the VLIW co-processor.

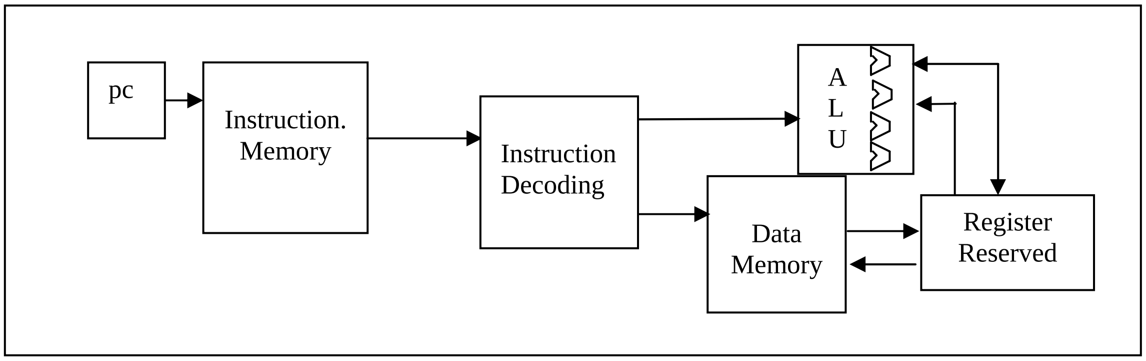


Figure 5 VLIW co-Processor datapath

The VLIW co-processor has a 32-bit bus. There is a 32-bit register, named *Reserved* (regR), which exchanges data with *Data Memory*. The block *Instruction Decoding* decodes the fetched instruction checking if it is Memory-Register, Register-Register or an ALU instruction. The instruction set of the co-processor is presented in Table 1.

Type	Instruction
Memory-Register	load [Mem] ∏ regR
	store regR ∏ [Mem]
Register-Register	mov regR ∏ regALU
	mov regALU ∏ regR
	zeros regR
	zeros regALU
	ones regR
	ones regALU
ALU	and R_j,R_k : $R_j = R_j \wedge R_k$
	or R_j,R_k : $R_j = R_j \vee R_k$
	xor R_j,R_k : $R_j = R_{j\otimes}R_k$
	$R_j \rightarrow R_k$: $R_j = !R_j \vee R_j$

Table 1 VLIW co-Processor Instruction Set

The four basic ALU instructions combined with their complementations and symmetries can represent all sixteen two-input logic functions. In Figure 6, we present the architecture of the ALU block. The interconnections and multiplexers/demultiplexers allow an easy way to send a data already computed at one ALU to any other ALU with its correspondent register.

In Figure 6, the gray block represents the demultiplexers and dark gray, multiplexers. After the instruction is decoded at *Instruction Decode*, the demultiplexers send the data to the correct two-bit registers R0-R15. The multiplexers select the input of the ALUs and, ALUs place the result of its operation in the demultiplexers.

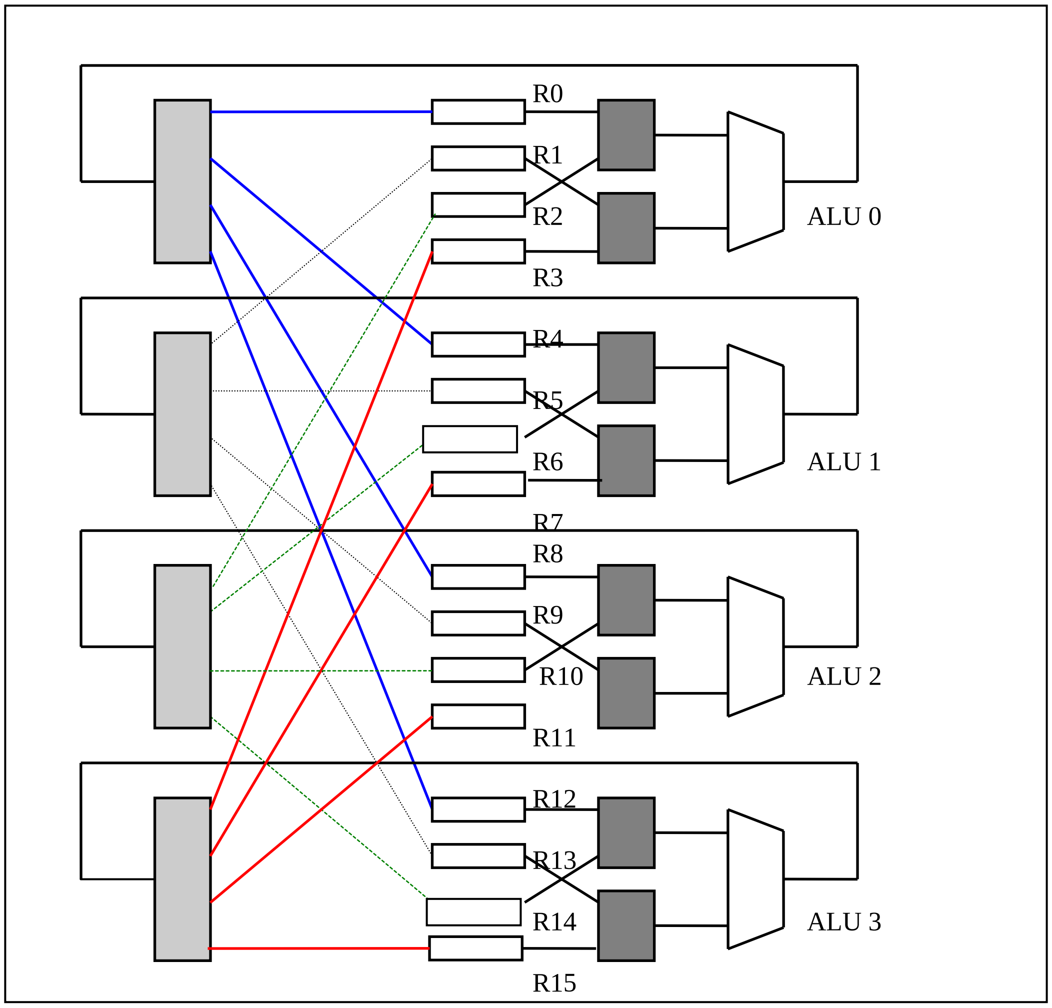


Figure 6. ALU Block

Since we are working with ternary logic emulation, each number that is processed by the ALU can assume the values in the set {0,1,X}. In order to encode these three values, we use the two bit encoding of Table 2.

Ternary Number	Representation
0	00
1	11
X	01

Table 2. Ternary Number Representation

The complementation of X is X in ternary logic, with all other Boolean operations being performed bit-wise. Therefore, we define a special NOT operation that can be described in VHDL according to Figure 7.

```
.....
architecture BEHAVIORAL of alu_emul is
begin
   ....
   constant O : BIT_VECTOR ( 1 downto 0) := B"11";
   constant Z : BIT_VECTOR ( 1 downto 0) := B"00";
   constant X : BIT_VECTOR ( 1 downto 0) := B"01";
   constant NOT : array (BIT_VECTOR ( 1 downto 0))
          of BIT_VECTOR ( 1 downto 0) :=  ( O => Z; Z => O; others => X);
....
```

Figure 7. Ternary Numbers in VHDL Representation

In the next sections, we are going to show the system's software support and how the co-processor operates and its interaction with an external environment.

4. Software Support

Our VLIW co-processor is a cycle-based emulator that manipulates a single basic-block per cycle. As mentioned before, the VLIW co-processor manipulates any logic function of two-input/one-output. Therefore, we start with the circuit we want to emulate described in some hardware description language, such as Verilog HDL[10] or VHDL[9]. A synthesis process generates a logic description of the circuit that is mapped into gates specified in a technology library[11] compatible with the VLIW instructions. Finally, the VLIW Compiler translates and schedules this circuit description to the co-processor Language, and the user can begin the testing phase. These steps are shown in Figure 8.

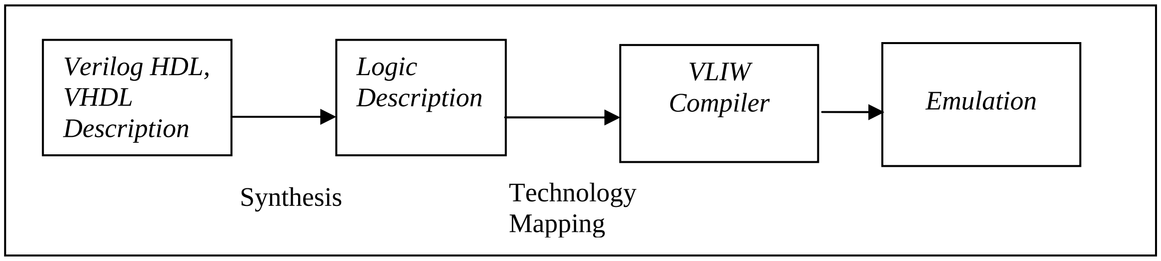


Figure 8. Diagram of pre-Processing a Circuit Description

5. The System : Interaction between Hardware Emulator and a Host Computer

An IBM-PC is used as a host computer that controls our cycle-based VLIW co-processor through sending data and control signals using PCI facilities as described in Section 2. The data signals are pre-processed basic blocks while control signals initialize and reset the system. In Figure 9, we detail the system.

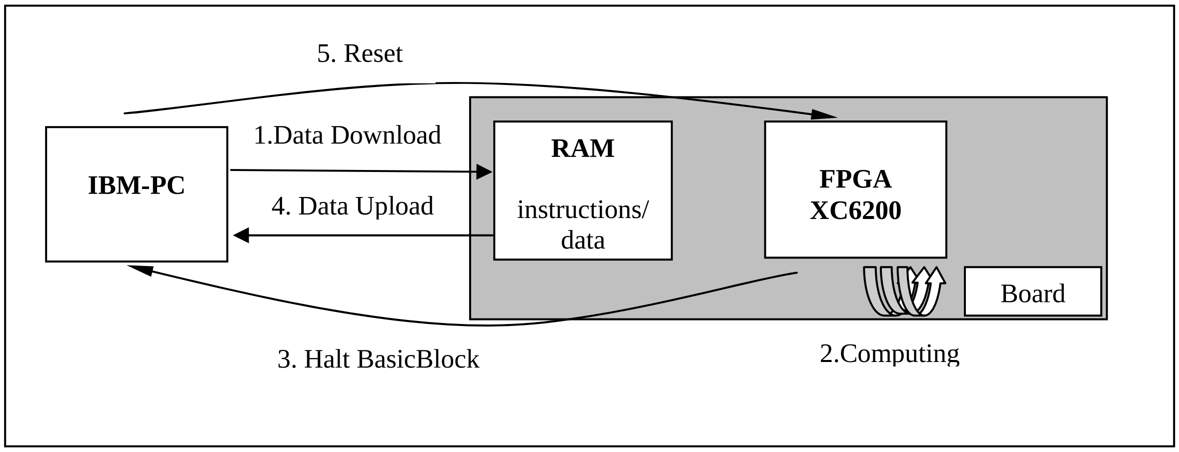


Figure 9. The System Integration

Once the pre-processed basic block is downloaded, VLIW co-processor in FPGA XC6200 starts its computation. When a *Halt Basic Block Instruction* is reached, the co-processor signals the host computer. The data available in RAM is uploaded, processed, and the computer resets the co-processor initiating a new basic block computing or the next emulation cycle.

As we only download instructions and data to the RAM instead of configuring the FPGA in debugging phase, we reach the system requirement of fast reconfiguration.

6. Conclusions

During a hardware design, there are some critical steps such as designing, simulation and prototyping that affects its design time since the designer may iterate many times in these steps. Due to recent advances in FPGAs, hardware emulation made possible to reduce this design time because FPGAs can be reconfigured very quickly, and simulating the system's behavior much faster than conventional event driven or compiled code simulators.

We proposed a cycle-based rapid reconfigurable VLIW co-processor that allows ternary logic emulation. This co-processor is being currently developed at LECOM (Computer Engineering Laboratory - Computer Science Department - UFMG) and we expect to complete its implementation by the end of 1997.

For future work we intend to integrate this ternary logic VLIW co-processor to a ternary logic simulator for a HW/SW co-design development. Also, we intend to introduce a remote interaction with this reconfigurable co-processor through the Web by a WebScope software[13].

7. References

[1] Brown, S., Rose, J., *FPGA and CPLD Architectures: A Tutorial*, IEEE Design & Test of Computers, Vol. 12, No. 2, pp. 42-57, Summer 1996.
[2] Villasenor, J., Mangione-Smith, W. H., *Configurable Computing*, Scientific American, pp 54-59, June 1997.
[3] Kumar, J., et al, *Emulation Verification of the Motorola 68060*, Proc. Int'l Conference Computer Design: VLSI in Computer and Processors, IEEE CS Press, 1995, pp. 150-158.
[4] Gajski, D., Vahid, F., Narayan, S., Gong, J., *Specification and Design of Embedded Systems*, Prentice-Hall, 1994.
[5] C. H. Seger and R. E. Bryant. *Formal Verification by Symbolic Evaluation of Partially-Ordered Trajectories*. Technical Report 93-08, Department of Computer Science, University of British Columbia, July 1993.
[6] Xilinx - *The Programmable Logic Data Book, 1996*.
[7] Xilinx – *XC6200 Field Programmable Gate Array DataSheet*
[8] Xilinx – *LogiCore PCI Master and Slave Interface User's Guide*
[9] R. Lipsett, C. Schaefer and C. Ussery. *VHDL : Hardware Description and Design*. Kluwer Academic Publishers, 1989.
[10] D. E. Thomas and P. R. Moorby. *The Verilog hardware description language*. Kluwer Academic Publishers, 1991.
[11] Sentovich, E. M., et al, *SIS: A System for Sequential circuit Synthesis*, Technical Report, Berkley University,1992
[12]Quickturn Design Systems, Inc. – HomePage : http://www.quickturn.com
[13]Xilinx – HomePage: http://www.xilinx.com