4-Bit Processing Unit Design UsingVHDL Structural Modeling For Multiprocessor Architecture

Mohamed EL KHAILI
Laboratory of SSIDA, ENSET, Mohammedia
Hassan II University of Casablanca, Morocco
elkhailimed@gmail.com

***Abstract*—This paper presents design concept of 4-bit Processing Unit (PU) based on 4-bit Arithmetic and Logic Unit (ALU) for multiprocessor architecture on one FPGA chip. Design methodology has been changing from schematic design to HDL based design. We use a VHDL structural and dataflow level design. Each module of the Processing Unit is divided into smaller modules. All the modules in logical unit and Arithmetic and Logic Unit (ALU) design are realized using VHDL design. Functionalities are validated through synthesis and simulation process. Processing Unit (PU) using VHDL fulfills the needs for different high performance applications such as processor for mono architecture and parallel architecture.**

Keywords—component; Arithmetic and logic unit (ALU), FPGA Circuits, Very high scale integrated circuit Hardware Description Language.

I. INTRODUCTION

The Arithmetic and Logic Unit ALU is the part of the processor that is involved with executing operations of an arithmetic or logical nature. In ECL, TTL and CMOS family, there are available integrated packages which are referred to as Arithmetic and Logic Units ALU. All logic circuits in this units are entirely combinational (i.e. consists of gates with no feedback and no flipflops).The ALU is an extremely versatile and useful device since, it makes available, in single package, facility for performing many different logical and arithmetic operations. ALU is a critical component of a microprocessor and is the core component of central processing unit. ALU can perform all the 16 possible logic operations or 16 different arithmetic operations on active HIGH or active LOW operands[9][10].

In this paper, 4 Bit processing unit is presented and designed by a VHDL structural modeling. The processing unit uses also switches and decoder device. This unit will be used in an implementation of a multiprocessor system on one FPGA chip. Our project aimed to design and implement an FPGA-based multiprocessor architecture to speed up multiprocessor architecture research and ease parallel software simulation. This system had to be a flexible, inexpensive multiprocessor machine that would provide reliable, fast simulation results for parallel software and hardware development and testing.

II. CONTEXT OF THE PROJECT

In a multiprocessing system, all Processors Elements PEs may be equal, or some may be reserved for special purposes. A combination of hardware and operating system software design considerations determine the symmetry (or lack thereof) in a given system[11][13]. For example, hardware or software considerations may require that only one particular Processor Element PE respond to all hardware interrupts, whereas all other work in the system may be distributed equally among PEs; or execution of kernel-mode code may be restricted to only one particular PE, whereas user-mode code may be executed in any combination of processors. Multiprocessing systems are often easier to design if such restrictions are imposed, but they tend to be less efficient than systems in which all PEs are utilized. M. Youssfi et al. proposed an emulator for multiprocessor system [13]. Him study is focused on a fine grained parallel architecture that has been largely studied in the literature and for which several parallel algorithms for scientific calculus were developed. From the theoretical point of view, each computational model has its motivations and its exciting proofs. In the practice, some models were technologically realized and served as real computational supports, but some others remain in their theoretical proposition state. They are not realized because of their technological complexities and to their very high production cost.

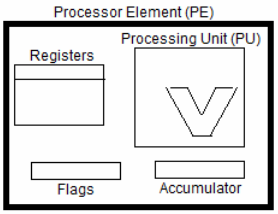

Fig. 1. Basic presentation of Processor Element (PE)

Our project provides a hard implementation of a multiprocessor system on one FPGA chip reducing production cost. The target is a reconfigurable system that we present in the following [12].

A. SISD multiprocessing

In a Single-Instruction stream, Single-Data stream computer, one PE sequentially processes instructions; each instruction processes one data item. One example is the "von Neumann" architecture with RISC.

B. SIMD multiprocessing

In a Single-Instruction stream, Multiple Data stream computer, one PE handles a stream of instructions, each one of

which can perform calculations in parallel on multiple data locations. SIMD multiprocessing is well suited to parallel or vector processing, in which a very large set of data can be divided into parts that are individually subjected to identical but independent operations. A single instruction stream directs the operation of multiple processing units to perform the same manipulations simultaneously on potentially large amounts of data.

For certain types of computing applications, this type of architecture can produce enormous increases in performance, in terms of the elapsed time required to complete a given task. However, a drawback to this architecture is that a large part of the system falls idle when programs or system tasks are executed that cannot be divided into units that can be processed in parallel.

Additionally, programs must be carefully and specially written to take maximum advantage of the architecture, and often special optimizing compilers designed to produce code specifically for this environment must be used. Some compilers in this category provide special constructs or extensions to allow programmers to directly specify operations to be performed in parallel (e.g., DO FOR ALL statements in the version of FORTRAN used on the ILLIAC IV, which was a SIMD multiprocessing supercomputer).

SIMD multiprocessing finds wide use in certain domains such as computer simulation, but is of little use in general-purpose desktop and business computing environments.

C. MISD multiprocessing

MISD multiprocessing offers mainly the advantage of redundancy, since multiple PEs perform the same tasks on the same data, reducing the chances of incorrect results if one of the units fails. MISD architectures may involve comparisons between PEs to detect failures. Apart from the redundant and fail-safe character of this type of multiprocessing, it has few advantages, and it is very expensive. It does not improve performance. It can be implemented in a way that is transparent to software. It is used in array processors and is implemented in fault tolerant machines.

Another example of MISD is pipelined image processing where every image pixel is piped through several hardware units performing several steps of image transformation.

D. MIMD multiprocessing

MIMD multiprocessing architecture is suitable for a wide variety of tasks in which completely independent and parallel execution of instructions touching different sets of data can be put to productive use. For this reason, and because it is easy to implement, MIMD predominates in multiprocessing.

Processing is divided into multiple threads, each with its own hardware PE state, within a single software-defined process or within multiple processes. Insofar as a system has multiple threads awaiting dispatch (either system or user threads), this architecture makes good use of hardware resources.

MIMD does raise issues of deadlock and resource contention, however, since threads may collide in their access to resources in an unpredictable way that is difficult to manage efficiently. MIMD requires special coding in the operating system of a computer but does not require application changes unless the programs themselves use multiple threads (MIMD is transparent to single-threaded programs under most operating systems, if the programs do not voluntarily relinquish control to the OS). Both system and user software may need to use software constructs such as semaphores (also called locks or gates) to prevent one thread from interfering with another if they should happen to cross paths in referencing the same data. This gating or locking process increases code complexity, lowers performance, and greatly increases the amount of testing required, although not usually enough to negate the advantages of multiprocessing.

Similar conflicts can arise at the hardware level between PEs (cache contention and corruption, for example), and must usually be resolved in hardware, or with a combination of software and hardware (e.g., cache-clear instructions).

E. Interconnection Topology

A network allows communication between PEs, Output/Input devices and memory. Topology has a significant influence on the properties of the system. Almost all topologies were used, one can nevertheless illustrate some:

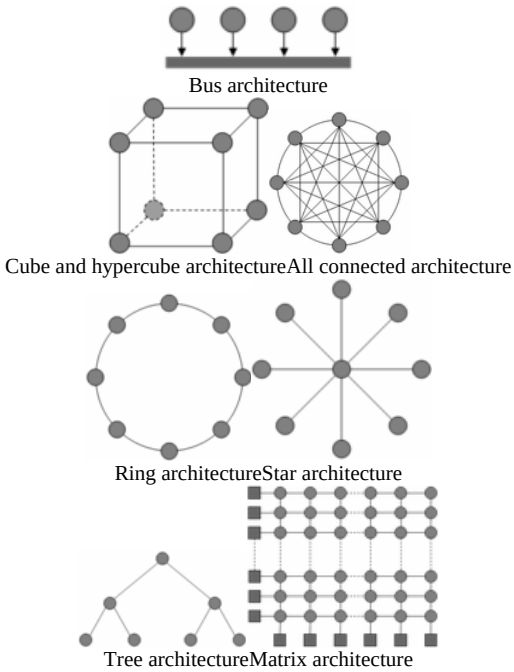

Fig. 2. Multiprocessor ArchitectureTypologies

III. Overview on Processor Element (PE)

In each PE, the ALU works in conjunction with the register array for many of these, in particular, the accumulator and flag registers. The accumulator holds the results of operations, while the flag register contains a number of individual bits that are used to store information about the last operation carried out by the ALU. More on these registers can be found in the register array section. In this paper, we present only the Processing Unit.

Processing Unit consists of three blocks for different operations:

- 4-bit Arithmetic and Logic Unit ALU;
- Switching systemAIGi;
- 1/4 Decoder DEC.

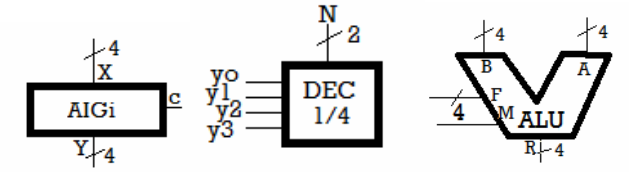

Fig. 3. Bloc systems used in Processing Unit architecture

A. Arithmetic and Logic Unit (ALU) Description

An ALU which was widely used when only TTL integrated circuits were available: 74181, a 4-bit ALU. This circuit is intended to be connected in cascade to treat other integers greater than 4 bits size. It is the basis of UAL Alto and many models PDP-11 (16 bits)[1][7][9][10].

TABLE I. THE TRUTH TABLE OF THE TTL 74181 CIRCUIT ACCORDING TO THE INPUT M AND F VALUES

Function	**M = 1**	**M = 0**
F	**Logic Operation**	**Arithmetic Operation**
0000	R = not A	R = A
0001	R = not (A or B)	R = A or B
0010	R = (not A) and B	R = A or (not B)
0011	R = 0	R = - 1
0100	R = not (A and B)	R = A + (A and (not B))
0101	R = not B	R = (A or B) + (A and (not B))
0110	R = A xor B	R = A - B - 1
0111	R = A and (not B)	R = (A and (not B)) - 1
1000	R = (not A) or B	R = A + (A and B)
1001	R = not (A xor B)	R = A + B
1010	R = B	R = (A or (not B)) + (A and B)
1011	R = A and B	R = (A et B) - 1
1100	R = 1	R = A + (A << 1)
1101	R = A or (not B)	R = (A or B) + A
1110	R = A or B	R = (A or (not B)) + A
1111	R = A	R = A - 1

1) Arithmetic operations

These operations are performed by constructs of logic gates, such as half adders and full adders. While they may be termed 'adders', with the aid of they can also perform subtraction via use of inverters and 'two's complement' arithmetic. A binary adder-subtractor is a combinational circuit that performs the arithmetic operations of addition and subtraction with binary numbers. Connecting n full adders in cascade produces a binary adder for two n-bit numbers.

2) Logical operations

Further logic gates are used within the ALU to perform a number of different logical tests, including seeing if an operation produces a result of zero. Most of these logical tests are used to then change the values stored in the flag register, so that they may be checked later by separate operations or instructions. Others produce a result which is then stored, and used later in further processing.

B. Switching system AIGi

AIGiisatri-state device. Ifc ='0', the output Ycoded on4 bits,is setto high impedance(HZ). Ifc ='1', the output Y assumesthevalue of the inputX, ascoded on4 bits.

C. Decoder DEC

Dec 1/4 is a decoder1from 4,receives anumber N coded on 2 bitsand providesan active output at‘1’ level among thefour outputs.

D. VHDL descriptions of Bloc systems used

The VHDL descriptions of Aigi, Dec 1/4 and ALU are as follows [2][3][5][6] :

```
Entity aigi is
Port (x : in std_logic_vector(3 downto 0); c : in std_logic;
        Y : out std_logic_vector(3 downto 0));
End aigi ;

Architecture str_aigi of aigi is
Begin
        Y <= x when c='1'  else "ZZZZ";
End str_aigi;

Entity dec_1_4  is
Port (n : in std_logic_vector(1 downto 0);
        Y : out std_logic_vector(3 downto 0));
End dec_1_4  ;

Architecture str_dec of dec_1_4 is
Begin
        Y <= "0001" when n=0 else
        <="0010" when n=1 else
        <= "0100" when n=2 else
        <= "1000" when n=3;
End str_dec;

Entity alu  is
Port (a,b,f : in std_logic_vector(3 downto 0); m : in std_logic;
        r : out std_logic_vector(3 downto 0));
End alu  ;

Archticturestr_alu of alu is
Begin
--The same implementation described in [9]
End str_alu;
```

IV. PROCESSING UNIT (PU) DESCRIPTION

The processing unitreceives the numberofRi input toswitchtothe operandBof theALUandthe instruction code to execute. TheALUprovides aresult that will beswitched to anaccumulator ACCor a register REG. The dataOPis recoveredfrom theaccumulator ACC. The signalsNumIN, RoutandRoutare providedfrom a logicblock thatis not partof our study.

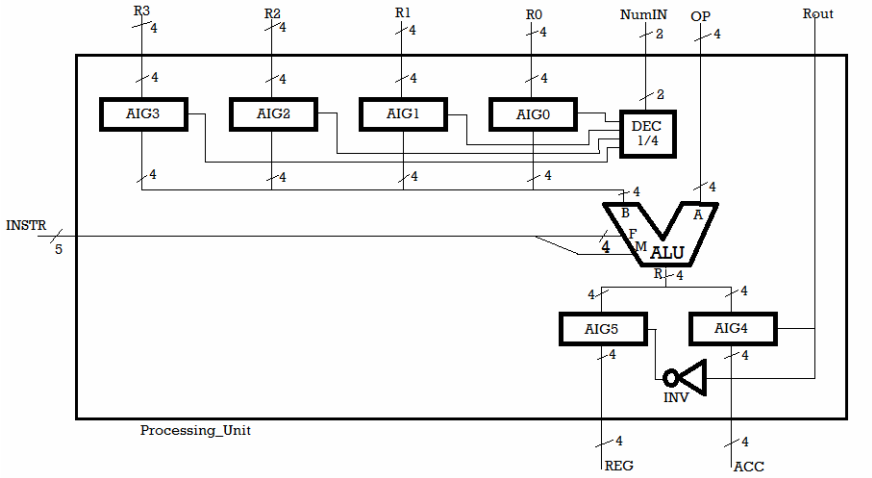

Fig. 4. Processing Unit Architecture

VHDL Code

```
Entity Processing_unit  is
Port (instr : in std_logic_vector(4 downto 0);
        R3,R2,R1,R0,OP : in std_logic_vector(3 downto 0);
        numIn : in std_logic_vector(1 downto 0);
        Rout : in std_logic;
        Reg, Acc : out std_logic_vector(3 downto 0);)
End Processing_unit  ;

Architecture str_ Processing_unit  is

Component aigi
Port (x : in std_logic_vector(3 downto 0); c : in std_logic;
        Y : out std_logic_vector(3 downto 0))
End Component  ;

Component  dec_1_4
Port (n : in std_logic_vector(1 downto 0);
        Y : out std_logic_vector(3 downto 0))
End Component  ;

Component  alu
Port (a,b,f : in std_logic_vector(3 downto 0); m : in std_logic;
        r : out std_logic_vector(3 downto 0))
End Component  ;

  for Aig0, Aig1,  Aig2,  Aig3,  Aig4,  Aig5   : aigi use entity
                work.aigi(str_aigi);
 fordec   : dec_1_4  use entity work. dec_1_4  (str_ dec_1_4  );
          for alu_4   : alu use entity work.alu(str_alu);

signalR_alu, B_alu, co : std_logic_vector(3 downto 0) ;

Begin
Aig0 :aigi port map (R0,co(0),B_alu);
Aig1 :aigi port map (R1,co(1), B_alu);
Aig2 :aigi port map (R2,co(2), B_alu);
Aig3 : aigi port map (R3,co(3), B_alu);
Aig4 : aigi port map (R_alu,Rout,Acc);
Aig5 : aigi port map (R_alu,not(Rout),Reg);
dec : dec_1_4  port map (NumIn,co);
alu_4   :   alu   port   map   (op,B_alu,Instr(3   downto
0),Instr(4),R_alu);
End str_ Processing_unit  ;
```

V. SIMULATION RESULTS

Design is verified through simulation, which is done in a bottom-up fashion. Each small moduleis simulated in separate test benches before the all are integrated and tested as a whole[4][8]. All operations available in the design are tested with the various inputs. The sequence of operations done in the simulation is addition. The results of operation on the test vectors are manually computed and are referred to as expected result.

VI. FUTURE WORK

The current implementation is a prototype that represents aproof-of-concept. Future research will be devoted to thedevelopment of a complete multiprocessor platform like a reconfigurable machine. The remainder of thiswork sketches the most important hardware and softwareissues.On the hardware side, the current VHDL code is to be enhanced by all therequired interfaces such as a LAN-connection fortransferring data, interfacing the peripherals, and connectinga CAN bus.

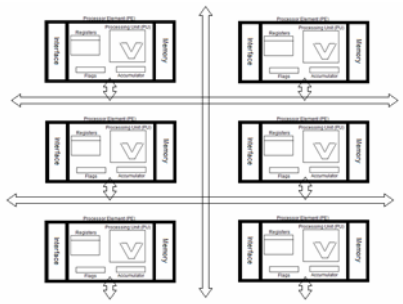

Fig. 5. A reconfigurable machine using a Multi Processing Unit

On the software side, the next steps are as follows: First ofall, the existing software functionalities have to be ported

tothe new system. Then, the distribution of the data andcommunication objects, i.e., the memory map, should beoptimized with respect to the system's runtime behavior.

After the completion of the steps mentioned above, a finalfield study has to prove both sufficient processing speed andfulfillment of the required timing behavior.

VII. CONCLUSIONS

In this paper, we have proposed efficient VHDL behavioral coding verification method. We have also proposed a VHDL program using different design levels. Our proposals have been implemented in Verilog and verified using Xilinx ISE 14 analyzer. We have reduced the number of bus lines and all the designs have been implemented and tested. This Processing Unit design using VHDL is successfully designed, implemented, and tested.

In the future work, we are conducting further researches that integrate a reconfigurable multiprocessor system on one FPGA chip.Several areasof application will betreatedfor validationprocessing algorithms.

REFERENCE

[1] Brown S., Vranesic Z "Fundamental of Digital Logic Design with VHDL" McGraw Hill, 2nd Edition.

[2] Bhasker J, "A *VHDL Primer"*, P T R Prentice Hall, Pages 1-2, 4-13, 28-30.

[3] Jiang Hao, Li Zheying, "FPGA design flow based on a variety of EDA tools" in *Micro-computer information*, 2007(23)11-2:201-203.

[4] Xilinx, Inc. Xilinx Libraries Guide, 2011.

[5] *A Comparative Study"*, In Proce. of the *IEEE Student Conference on Electrical, Electronics and Computer Sciences 2012*, 1-2 Mar 2012, NIT Bhopal, India.

[6] Douglas L. Perry, *VHDL*, McGraw-Hill, 1995.

[7] "Digital Integrated Circuits" by Jan M. Rabaey , Anantha Chandrakasan and Borivoje Nikolic

[8] Jiang Hao, Li Zheying, "FPGA design flow based on a variety of EDA tools" in *Micro-computer information*, 2007(23)11-2:201-203.

[9] Rupali Jarwal, Ulka Khire "4-Bit Arithmetic And Logic Unit Design Using Modelling In VHDL" International Journal of Engineering Research & Technology (IJERT) Vol. 2 Issue 3, March - 2013

[10] Navdeep Kaur, Neeru Malhotra, « FPGA Implementation Of ALU Using BIST » International Journal of Engineering Research & Technology (IJERT) Vol. 2 Issue 3, March - 2013

[11] Hauck, Scott and Dehon, André. 2008. Reconfigurable computing. Burlington, Massachusetts, USA : Morgan Kauffman, 2008.

[12] Sumedh.S.Jadhav, C.N.Bhoyar, « FPGA Based Embedded Multiprocessor Architecture » International Journal of Electrical and Electronics Engineering (IJEEE), Vol-1, Iss-3, 2012

[13] Mohamed YOUSSFI, Omar BOUATTANE, Mohamed O. BENSALAH , "A Massively Parallel Re-Configurable Mesh Computer Emulator: Design, Modeling and Realization » J. Software Engineering & Applications, January 2010, 3: 11-26

AUTHORS PROFILE

Mohamed EL KHAILI received Bachelor's Degree of Electronics Engineeringfrom ENSET (Ecole Normale de l'Enseignement Technique) of Mohammedia,Morocco in 1992 and received a Master's Degree of science in Information Processing (1994) and Doctorat in Image Processing (1998). He is a Assistant Professor at ENSET (Ecole Normale de l'Enseignement Technique) of Mohammedia, Moroccowhere his field of specialization is Electronics Engineering, Information Processing and Embedded Systems.