Structural Level Designing of Processing Elements using VHDL

Manisha P. Khorgade, Shweta Hajare, P.K.Dakhole

***Abstract*—This paper involves structural design and development of processing elements using Hardware Description Language (HDL) using Altera or Xilinx softwares and implements them on Field Programmable Gate Arrays (FPGAs). In this paper, we will simulate and synthesize the various parameters of processing elements by using VHDL on Xilinx ISE 13.1 and target it for SPARTAN 6 FPGA board. The output is displayed by means of Liquid Crystal Display (LCD) interface. The state of each output bit is shown by using Light Emitting Diodes (LED). The processor can perform 2n number of operations where n is the control bit. More number of designs can be implemented on FPGA as per user's needs.**

***Index Terms*— FPGA, XILINX ISE 13.1, SPARTAN 6.**

I. INTRODUCTION

This paper deals with the construction of processing elements using VHDL and we will implement these elements on FPGA to analyse the various design parameters. A processor is a digital device that can perform computations involving multiple steps. The term processor indicates the synonym for the Central Processing Unit (CPU). The processor and co-processors can be designed on software and hardware for many applications like Digital Signal Processing (DSP), Digital Image Processing (DIP) and video applications, communication, networking, multi-media and many other also. The processing elements carries our arithmetic and logic operations and a sequencing and control unit that can change the order of operation. The design and implementation of FPGA based co-processors is of core significance in digital technologies as it is an integral part of central processing unit. This paper deals with the designing of 8 bit Arithmetic and Logic Unit (ALU), memory and shift register. All the modules described in the design are coded using VHDL with its degree of concurrency to cope with the parallel nature of digital hardware. The VHDL software interface used in this design reduces the complexity and also provides a graphic presentation of the system. The key advantage of VHDL when used for systems design is that it allows the behavior of the required system to be described (modeled) and verified (simulated) before synthesis tools translate the design into real hardware (gates and wires). This software not only compiles the given VHDL code but also produces waveform results.

II. METHODOLOGY

A processing element is a generic term used to refer a hardware element that executes a stream of instructions.

Manuscript Received on May, 2014.

Prof. Manisha Khorgade, Electronics & Telecommunication Engineering Dept, RGCER, Nagpur, India.

Prof Shweta Hajare, Electronics Engineering Dept, YCCE, Nagpur, India.

Dr.P.K.Dakhole, Electronics Engineering Dept, YCCE, Nagpur, India.

In HDL, a processor is described at the bit level with little formal distinction between the control logic and the data path. The desired instruction set must somehow be externally verified against this description of the processor. Each processing element contains a data register file and three computation units: an arithmetic/logic unit (ALU), a multiplier and a shifter. Computational instructions for these elements include both fixed-point and floating-point operations, and each computational instruction can execute in a single clock cycle.

The computational units in a processing element handle different types of operations. The ALU performs arithmetic and logic operations on fixed-point and floating-point data. The multiplier does floating-point and fixed-point multiplication and executes fixed-point multiply/add and multiply/subtract operations.

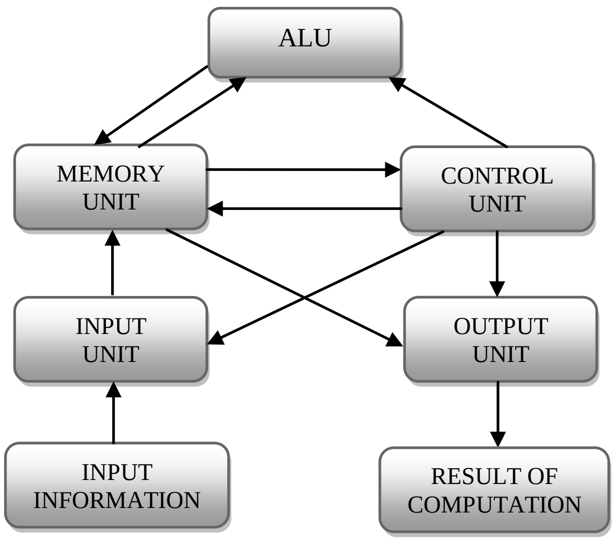

Fig 01: Processing unit

The shifter completes logical shifts, arithmetic shifts, bit manipulation, field deposit, and field extraction operations on 32-bit operands. Also, the Shifter can derive exponents. Data flow paths through the computational units are arranged in parallel. The output of any computation unit may serve as the input of any computation unit on the next instruction cycle.

A. ALU

Since ALU is the main part of CPU, this arithmetic and logic unit design is very important to know. ALU chip design involves changing the logic functions into equivalent circuits and creating fast switching networks.

There are two kinds of operations that an ALU can perform:

- First part deals with arithmetic computations and is referred to as Arithmetic unit.

*Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication*

It is capable of addition, subtraction, multiplication, division, increment and decrement.

- Second part deals with the gated circuits in the shape of AND, OR, XOR, inverter, rotate, left shift and right shift which is referred to as Logic unit. Logic unit of ALU does not need as many gates as required in Arithmetic Unit and if done separately, the LOGIC unit can be implemented using Complex Programmable Logic Devices (CPLD) or other Programmable Logic Device (PLD) technologies instead of using FPGA.

In general, the ALU includes storage places for input operands, operands that are being added, the accumulated result (stored in accumulator), and shifted results. The flow of bits and operations performed in the subunits of the ALU is controlled by gated circuits.

In this paper, we will design 8 bit ALU with 3 function select inputs: Mode (M), Select inputs (S1 and S0). The mode input (M) selects between a logic (M=0) and arithmetic (M=1) operations.

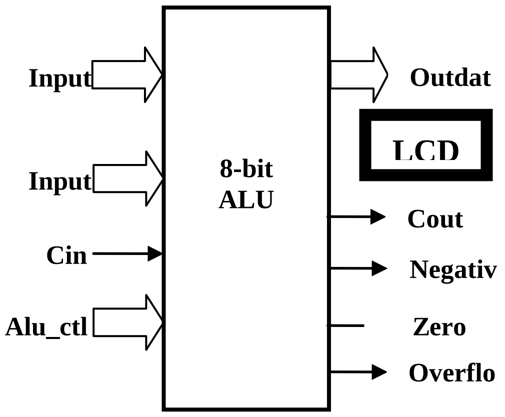

Fig 02: Hardware representation of 8 bit ALU

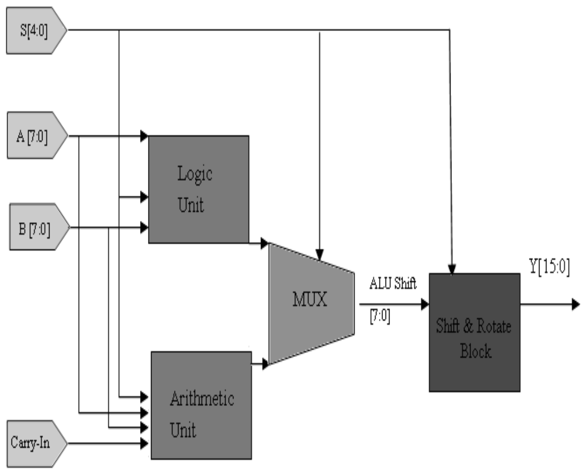

Fig 03: Block diagram of 8 bit ALU

An ALU is chopped into several segments each incorporating its specific operations. The ALU has been modelled with a separate arithmetic unit, logic unit and shifter as indicated by the circuit structure above. This modularization is closer to reality, makes it easier to follow the processes and produces better pre-optimized timing.

Table 01

Select lines	Operation	Function	Implemen-tation
000	Y<=A+B	Addition	Arithmetic unit
001	Y<=A-B	Subtrac-tion	Arithmetic unit
010	Y<=/A	NOT	Logic unit
011	Y<=A nand B	NAND	Logic unit
100	Y<=A nor B	NOR	Logic unit
101	Y<=A and B	AND	Logic unit
110	Y<=A or B	OR	Logic unit
111	Y<=A xor B	XOR	Logic unit

The above table shows the status of select lines and operations performed by ALU.

More out of ALU:

Carry-Out and Overflow

It is advantageous to act as if your ALU would only deal with unsigned numbers as that way you can also deal with two's complement numbers. However, there are now two conditions that can mess us up in the systems, carry-out and overflow.

1. Carry-Out

The highest number we can display on the 7-Segment display is 'F' which symbolizes 15. We get this number for example just by setting all A-switches high when all selection switches and the carry-in switch are low. Up to now the output register was: '1111'. If we set the carry-in high too, then adding another one now will set it to '0000' and we would have a 1 in the fifth position. As this position however does not exist, this number – our carry-out – is lost.

So in this case, we can increase the register. However, then we cannot anymore display all the numbers the register can contain on the 7-segment display. To take care of that, we can use the decimal point of the 7-Segment display to show if we have a carry-out or not and thus we can now display numbers from 0 to 31 instead of only from 0 to 15.

2. Overflow

Carry-out does of course only make sense when we are dealing with unsigned numbers, when we deal with two's complement numbers we might get overflow which basically means that our result will have the wrong sign. To exemplify this, assume that you are adding the numbers 1001 and 1011 in 4-bit two's complement. This means of course, that both numbers are negative then the result will be 0100, a positive number, which is a completely wrong result.

There are now several ways to test for overflow however, the easiest one is to follow the rule: 'When the two numbers coming in, have the same sign-bit, and the number going out has a different one, then overflow has occurred'. We can implement this, using the other decimal point on the system however, always pay attention now to the interpretation of your output.

Carry-out is meaningless for signed numbers and overflow for unsigned numbers.

B. MEMORY

Memory is the process in which information is stored, encoded and retrieved.

Memory structures are crucial in digital design system. All memory structures have an address bus and a data bus. The processing elements of the DSP applications must be mapped with ALU, memory and shift register. As we are using 8 bit ALU, hence input to the ALU will be of 8 bit. Accordingly, it is able to perform 8 different tasks as per the requirement of users. This would be achieved if we have a minimum memory of 8 bit. Then the shift register should be also of 8 bit, so that the shifting is done sequentially and the complete memory is utilised.

The fastest possible memory option is to put everything in local memory. Xilinx local memory is made up of large FPGA memory blocks called Block RAM (BRAM). Embedded processor accesses to BRAM happen in a single bus cycle. Xilinx FPGA BRAM quantities differ by device. For example, the 1.5 million gate Spartan-3 device (XC3S1500) has a total capacity of 64KB, whereas the 400,000 gate Spartan-3 device (XC3S400) has half as much at 32KB. An embedded designer using FPGAs should refer to the device family datasheet to review a specific chip's BRAM capacity. If the designer's program fits entirely within local memory, then the designer achieves optimal memory performance [7].

Using various configuration options, Select RAM blocks create RAM, ROM, FIFOs, large look-up tables, data width converters, circular buffers, and shift registers, each supporting various data widths and depths. Each block RAM contains 18,432 bits of fast static RAM, 16K bits of which is allocated to data storage and, in some memory configurations, an additional 2K bits allocated to parity or additional "plus" data bits.

Physically, the block RAM memory has two completely independent access ports, labelled Port A and Port B. The structure is fully symmetrical, and both ports are interchangeable and both ports support data read and write operations. Each memory port is synchronous, with its own clock, clock enable, and write enable.

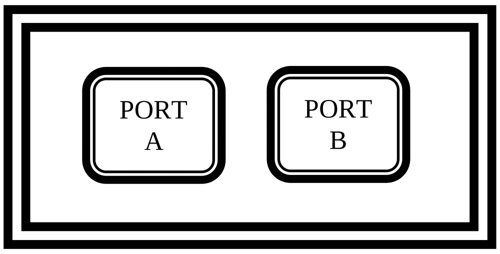

Fig 04: Block Random Access Memory (BRAM)

The two ports, Port A and Port B perform the following functions:

1. Port A behaves as an independent single-port RAM supporting simultaneous read and write operations using a single set of address lines.
2. Port B behaves as an independent single-port RAM supporting simultaneous read and write operations using a single set of address lines.
3. Port A is the write port with a separate write address and Port B is the read port with a separate read address. The data widths for Port A and Port B can be different also.
4. Port B is the write port with a separate write address and Port A is the read port with a separate read address. The data widths for Port B and Port A can be different also.

Table 02

Input data	Function
00	Port A is an independent single-port RAM.
01	Port B is an independent single-port RAM.
10	Port A is the write port. Port B is the read port.
11	Port B is the write port. Port A is the read port.

C. SHIFT REGISTER

Shift register is a register that is designed to allow the bits of its contents to be moved to left or right. Shift registers can have both parallel and serial inputs and outputs. These are often configured as serial-in, parallel out (SIPO) or as parallel-in, serial-out (PISO). There are also types that have serial and parallel input and types with serial and parallel output. There are also bi-directional shift registers which allow shifting in both directions: Left to Right or Right to Left.

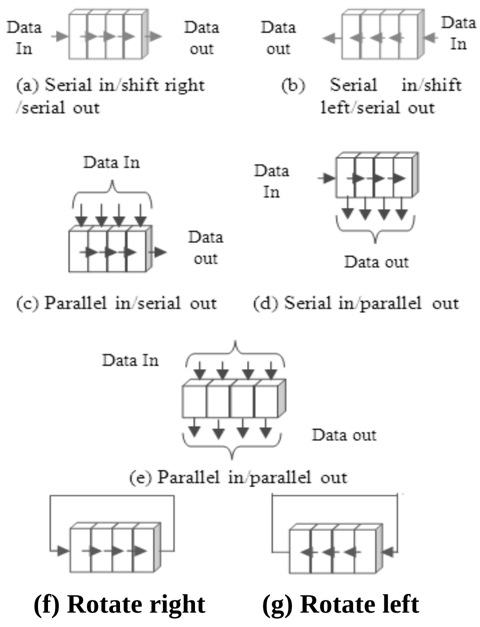
Data In
Data out
Data out
Data In
(a) Serial in/shift right /serial out
(b) Serial in/shift left/serial out
Data In
Data out
Data In
Data out
(c) Parallel in/serial out
(d) Serial in/parallel out
Data In
Data out
(e) Parallel in/parallel out
(f) Rotate right
(g) Rotate left

Fig 05: Basic data movement in shift register

Shift registers delay data by one clock time for each stage. They will store a bit of data for each register. The data string is presented at Data in and is shifted right one stage each time Data advance is brought high. At each advance, the bit on the far left (i.e., Data in) is shifted into the output. The bit on the far right (i.e., Data out) is shifted out and lost.

III. IMPLEMENTATION

The standard FPGA design flow starts with the design entries using hardware description language (HDL), such as Verilog HDL or VHDL. The flow then goes through programming, compilation, simulation and verification in FPGA hardware [10]. There are two types of simulation:

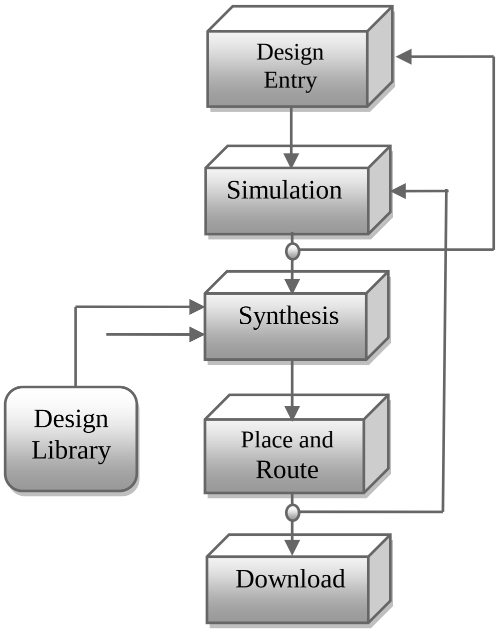

Fig 06: FPGA Process

1. RTL (or functional) simulation: It allows us to verify that the code is manipulating the input and output respectively
2. Timing (or post place and route) simulation: It verifies that the design meets timings and functions appropriately.

The VHDL code which implies the hardware part of ALU is downloaded on FPGA processor using JTAG cable interfacing PC and the hardware element. When a VHDL model is translated into the "gates and wires" that are mapped onto a programmable logic device i.e., FPGA, then it is the actual hardware being configured, rather than the VHDL code being "executed" as if on some form of a processor chip.

We will create a design that will cause the LED to blink at a speed that can be controlled by an input button. We can also run other designs on the development board as well. For the LED design, we will write VHDL code for 8 bit ALU, memory and shift register. The device can accept two numbers of 8 bit binary data and can perform arithmetic/logical operations depending on the 3 bit operation code given. The design of the elements is done through Xilinx which will give simulation waveforms and synthesis report for the design. When the design is running on FPGA board, we can press an input switch to change the bits and can see the respective change in output through LED's.

The function of FPGA is embedded on the kit along with PROM, LCD, LEDs and DIP switches. A Joint Test Action Group (JTAG) interface connects the FPGA chip with PROM and leads to PC through a serial interface. Since FPGA is a user programmable, therefore JTAG is of core significance. After the process of compilation and simulation of the VHDL design, the hardware realization is carried out and tested. Here the 3-bit inputs will be given by means of two sets of DIP switches and the output can be displayed on a LCD panel and the result can be verified with the simulated output. The status of the flag registers is indicated by a series of 8 bit LED's. The provision of a select switch used in this hardware enables the user to perform the required operation on the FPGA processor.

IV. SIMULATION WAVEFORMS

Various designs and gates are implemented and processing speed is analyzed by using PE. ALU, memory designs are also simulated and analyzed speed of PE We have tried to implement and then dumped it on Spartan 6 Kit. It give us more accurate results using for analysis of PE performance. This is purely a static implementation of PE on structural basis using ALU,Memory and IO component. Few results are as shown in following figures.

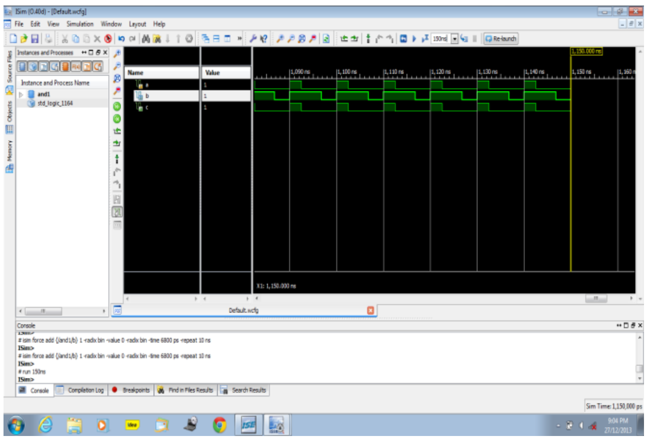
```
# isim force add {/and1/b} 1 -radix bin -value 0 -radix bin -time 6800 ps -repeat 10 ns
ISim>
# isim force add {/and1/b} 1 -radix bin -value 0 -radix bin -time 6800 ps -repeat 10 ns
ISim>
# run 150ns
ISim>
```

Fig 07: And Gate

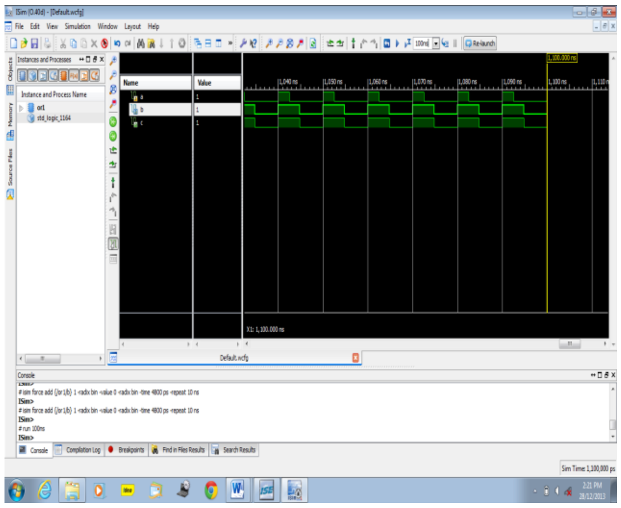
```
# isim force add {/or1/b} 1 -radix bin -value 0 -radix bin -time 4800 ps -repeat 10 ns
ISim>
# isim force add {/or1/b} 1 -radix bin -value 0 -radix bin -time 4800 ps -repeat 10 ns
ISim>
# run 100ns
ISim>
```

Fig 08. OR gate

Fig0 9.Spartan 6 Kit

V. APPLICATIONS

- Since ALU is the main part of the CPU, we can design the CPU for the system. General decoding problems will be solved for the system by using the ALU design. Since ALU has huge applications, we can develop micro-processors and micro-controllers.
- The Spartan-3E FPGA's abundant, fast and flexible block RAMs provide invaluable on-chip local storage for scratchpad memories, FIFOs, buffers, look-up tables, and much more. Using unique capabilities, block RAM implements such functions as shift registers, delay lines, counters, and wide, complex logic functions.
- The serial in -serial out shift register can be used as a time delay device.

The amount of delay can be controlled by:

1. The number of stages in the register
2. The clock frequency.

VI. CONCLUSION

The VHDL is a versatile language which has great flexibility of designing component and FPGA embedded processor has the power and ability to provide previously unachievable flexibility and performance. One can design hardware in a VHDL for FPGA implementation such as Xilinx ISE, Altera Quartus to produce the RTL schematic of the desired circuit. After that the generated schematic can be verified using simulation software which shows the waveforms of inputs and outputs of the circuit after generating the appropriate test bench. We learned how to produce different arithmetic operations and logical functions by using various select signals for a single circuit. Rapid implementation of parallel structures based on FPGAs using VHDL proves to be a very efficient, cost- effective and attractive methodology for design verification.

Verification of the designed static elements using Xilinx ISE tool is implemented using Very High Speed Hardware Descriptive Language and Xilinx Spartan 6 Field Programmable Gate Array. Further enhancements can be made on this system by adding more number of inputs with increased number of bit size. Digital Signal Processing (DSP) is being credited with lots of applications from VHDL designs for reconfigurable PE.

REFERENCES

1. V. Khorasani, B. V. Vahdat, and M. Mortazavi, "Design and implementation of floating point ALU on a FPGA processor", IEEE International Conference on Computing, Electronics and Electrical Technologies (ICCEET), pp. 772-776, 2012.
2. Suchita Kamble, Prof .N. N. Mhala, "VHDL Implementation of 8-Bit ALU", IOSR Journal of Electronics and Communication Engineering (IOSRJECE), ISSN : 2278-2834 Volume 1, Issue 1 (May-June 2012), PP 07-1.
3. Prof. S. Kaliamurthy & Ms. U. Sownmiya, "VHDL design of arithmetic processor", Global Journals Inc. (U.S.A), November 2011.
4. S.Kaliamurthy, R.Muralidharan, "VHDL Design of FPGA Arithmetic Processor" International Conference on Engineering and ICT, 2007.
5. Charles H. Roth, Jr., "Digital system design using VHDL", PWS publishing company, 2006.
6. B.Stephen Brown, V. Zvonko, "Fundamentals Of digital logic with VHDL Design", 2nd Edition, McGraw Hill International Edition, 2005.
7. Bryan H. Fletcher, "FPGA Embedded Processors", Embedded Systems Conference San Francisco 2005 ETP-367.
8. J. Bhaskar, "VHDL Primer", Pearson Education, 3rd edition, 2000. Douglas L.Perry, "VHDL", tata mc grawhill, international edition 1999.
9. Module 4: Design of Embedded Processor, Lesson 20: Field Programmable Gate Arrays and Applications, Version 2, EE IIT Kharagpur.

AUTHORS PROFILE

Prof. Manisha Khorgade, She is a Assistanr Professor in RGCER,Wanadongari,, persuing Ph D in embaded area consisting of 9 international publications in conference and journals.

Prof Shweta Hajare, She is a Assistant Professor in YCCE, Wanadongari,Nagpur ,perusing PhD in VLSI area and published almost 12 publications at international level.

Prof. P.K.Dakhole. He is Professor and Dean in YCCE, Wanadongari,Nagpur.He has doctorate in electronics engineering having research area in VLSI, Embedded system. He has almost 27 international publications.