INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

DIVISION ALGORITHM DESIGN USING FIELD PROGRAMMABLE GATE ARRAY

A. Jaafar[1], M. M. Lazim[2], N. M. Z. Hashim[3], A. Salleh[4], A. S. Jaafar[5]

[1, 2, 3, 4, 5]Centre for Telecommunication Research and Innovation (CeTRI)
Faculty of Electronic and Computer Engineering
Universiti Teknikal Malaysia Melaka (UTeM)
Hang Tuah Jaya 76100, Durian Tunggal, Melaka, Malaysia
[1]anuarjaafar@utem.edu.my, [2]muhammad_shukri89@yahoo.com, [3]nikzarifie@utem.edu.my,
[4]azahari@utem.edu.my, [5]shukur@utem.edu.my

Abstract*: This project is to design eight bit division algorithm program by using Xilinx ISE 10.1 software for simulation algorithm circuit partitioning through hardware Field Programmable Gate Array (FPGA). The algorithms are divide 8-bit dividend by 8-bit divisor for input and get the result 16-bit for the output. Circuit partitioning algorithms eight bits used to implement the distribution process for each program using the arithmetic and logic unit operations, called (ALU). All these operations using Verilog language in a program to be displayed on (LED) using the FPGA board. FPGA is a semiconductor device containing programmable logic components called "logic blocks", and programmable. Logic block can be programmed to perform the functions of basic logic gates such as AND, and XOR, or more complex combination of functions such as decoders or simple mathematical functions such as addition, subtraction, multiplication, and division (+, -, x, ÷). Finally, this project outlines the design and implementation of a new hardware divisor for performing 8-bit division. The error probability function of this division algorithm is fully characterized and contrasted against existing hardware division algorithms.*

Keywords: *Arithmetic Logic Unite (ALU), Division Algorithm, Full Adder, Hardware Description Language (HDL), Xilinx*

1. INTRODUCTION

This project is to design a program using the software Xilinx ISE 10.1 for simulator Eight Bit Division algorithm circuit using hardware Field Programmable Gate Array (FPGA). Designing the eight bits division algorithms circuit for each bit by performs arithmetic operation and logic unit operations, called the Arithmetic Logic Unit (ALU). All these operation using Verilog language in a program to be displayed on seven segment using the FPGA board [1]. FPGA is a semiconductor device containing programmable logic components called logic blocks, and programmable. Logic block can be programmed to perform the functions of basic logic gates such as AND, and XOR, or more complex combination of functions such as decoders or simple mathematical functions such as addition, subtraction, multiplication, and division (+, - , x, ÷). In most FPGA, the logic blocks also include memory elements such as flip- flops simple. The combination of FPGA and ALU will generate eight bit division circuit design [1]. The purpose of the design of eight bits division is to perform the required operations.

Nowadays, some of the devices such as microprocessor, microcontroller and microchip are complex digital devices to have been design in digital system. Digital system has been used in industrial field compared with Analog system because it has some benefits. Some of the benefits are digital technique is very easy to learned and digital circuits have quality of memory which makes these circuits highly favorable for computers, calculators, watches and telephone. Division is the most important parts in devices to make high speed performance in all the devices.

So, the division operation is used in all devices. The basic division operation is easy to do and learned but the complex division operation in binary number is hard to do in Verilog language.

This project is being done to help design or create a prototype of digital system design that can operate as division operation that would be implemented into digital devices.

The first objectives to achieve the target in the completion of this project are to design a program for 8-bit division algorithm using Verilog language. The second objective is to design the two's complement division algorithm design. The third is to implement division algorithm technology by using FPGA as a device.

Verilog language is used for design Division algorithm system. The input will be in 8 bit divide by 8 bit which produced 16 bit of the output division. The system will only process and produce in fixed point value and also used two's complement number or call as negative value in sign number. Xilinx ISE

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

10.1 software is used to be running all the process after design the top module and test bench.

2. LITERATURE REVIEW

2.1. Division Algorithm

The four basic arithmetic operations; addition, subtraction, multiplication, and division is the easy to implement in hardware compared with complex arithmetic operation. One of the main reasons is that while addition, subtraction and multiplication are well defined and give exact answers. The result of a division between integers (or even between floating point numbers with finite precision) will in general be a rational number. Many cases cannot be represented exactly in binary with a fixed number of bits. This leads either to an approximate answer, or to a second definition of division [2].

Integer division considers both the dividend and divisor to be integers, and expresses the result uniquely as a quotient and remainder. The equation for division satisfies as Dividend = Q × Div + Rem where the quotient is also an integer and the remainder satisfies as 0≤Rem<Div.

Division is somewhat more complex but it based on the same general principles than multiplication. As before, the operation involved for shifting and addition or subtraction like the basis of the algorithm for the manual calculated approached.

Figure 1 shows an example of the division of unsigned binary integers [8]. It used 8-bit dividend divide by 4-bit divisor input which produced 8-bit quotient and some of remainder. The process instructive to be described in detail by follow the step of division like a manual division. First, the bits of the dividend are examined represents a number greater than or equal to the divisor by referred to as the divisor being able to divide the number. Until this event occurs, 0s are placed in the quotient from left to right. When the event occurs, a 1 placed in the quotient and the divisor is subtracted from the partial dividend. The result is referred to as a partial remainder [8].

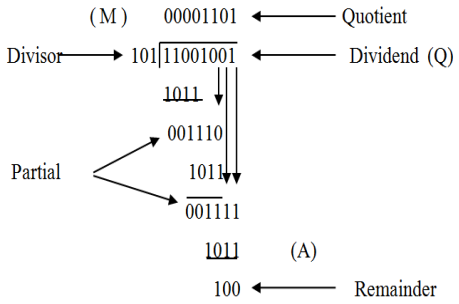

Figure 1: Example of Unsigned Binary Integers division

2.1.1. Non-Restoring Unsigned Division

The restoring division designs, the remainder given R (R<0) after subtraction. By added divisor D back, it have (R+D). While shifting the result, it have2×(R+D) =2×R+2×D. If subtract the divisor in the next step, it have 2×(R+2) × (D-D) =2×(R+D). This is equivalent to left-shifting R by 1 bit and then adding D [3].

Let do again the operation for non-restoring unsigned division. Used 0111/0010 (7/2) for this operation based on Table 1 below. For the first step, start with iteration 0, set dividend =0111 and divisor =0010 as the 4-bit input which produced remainder as the 8-bit output. The operation starts to shift left by 1 from remainder =00000111 and get 0000 1110 [4].

Next iteration 1, the remainder = remainder – divisor result get remainder=11101110. Then, shift left by 1 the remainder. After that, the operation in iteration 2, rem=rem + divisor to get 11111100 while the first loop iteration 1 has 0. For the next iteration get the remainder 00011000 while remainder=remainder + divisor. If remainder>0, shift left remainder and put r0=1. The next iteration 4 calculated remainder = remainder - divisorand get the input after shift lest is 00100011 while r0 = 0. Lastly, the operation had been done when shift left half of remainder right by 1. The answers for remainder get 00010011 for the output [4].

Table 1: Non-restoring Unsigned Division

Iteration	Division	Hardware design 3, non-restoring	
		Step	Remainder
0	0010	initial value	0000 0111
		shift remainder left by 1	0000 1110
1	0010	remainder = remainder - divisor	**1110** 1110
		(remainder<0)=>shift left; r0=0	1101 1100
2	0010	remainder = remainder + divisor	**1111** 1100
		(remainder<0)=>shift left; r0=0	1111 1000
3	0010	remainder = remainder + divisor	**0001** 1000
		(remainder>0)=>shift left; r0=1	0011 0001
4	0010	remainder = remainder - divisor	**0011** 0001
		(remainder>0)=>shift left; r0=1	0010 0011
Done	0010	Shift "left half of remainder" right by 1	**0001 0011**

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

2.1.2. Research of VHDL Implementation of Non Restoring Division Algorithm Using High Speed Adder/Subtracted

The binary division digit of non-restoring division algorithm is designed using high speed subtracted and adder. The procedure is to determine how many times the divisor M divides the dividend Q thus resulting in the quotient. At each step in the process, the divisor M divides Q into a group of bits. The value of divisor have less than or equal to the value of those bits after the divisor has dividend a group of bits. Therefore, the value for quotient has 1 or 0 in that operation. Addition or subtraction used based on the signs of the divisor and the partial remainder in the division algorithm. There are some of binary division algorithm such as Digit Recurrence Algorithm restoring, non-restoring and SRT Division (Sweeney, Robertson, and Torcher). High speed adder and subtracted are used to speed up the operation of division [5].

2.2.1. Restoring Division

Quotient is the method of restoring division represented using a non-redundant number system. The full width comparisons required to reduce the new quotient digit as the main characteristic of restoring division [5]. The operation at that bit position is unsuccessful if subtraction of the divisor produces a negative result at any bit position relative to the dividend. For 0 placed in the corresponding location of the quotient. The result of the division operation is added back (restored) the divisor, and then the next highest bit of the dividend is shifted into the left bit position of the result. The each bit of dividend is shifted from right to left and the quotient is built up from left to right. The division operation is completed after n-bit shifts, where n represents the number of bits in the dividend.

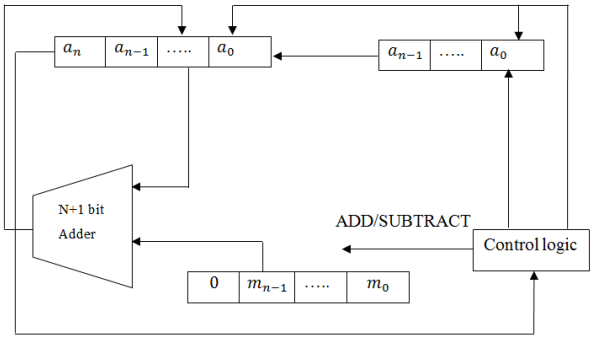

Figure 2: Hardware Design of Restoring Division Algorithm

Figure 2 has shown the complete hardware for restoring division [5]. In this figure an n-bit positive divisor is loaded into register M and n-bit dividend is loaded into register Q at the start of the operation. The n-bit quotient is in register Q and the remainder is in register a while completed the division. The result for last restoring operation is remainder [5].

The manually performing for long division is very similar in restoring division algorithm. Restoring division algorithm is mentioned below along with flow chart shown in Figure 3 [5]. First, set count to 0 and put 0 in register A to start loop for n times. After that, the operation is shift A & Q left one binary position. Then, the next step of the operation is subtract M from a and placing the answer back in A. If the sign of A<0, set Q0 to 0 and add M back to A (restore A), otherwise set Q0 to 1. The last step is check the count, when count=n-1 . Then stop the loop in the operation.

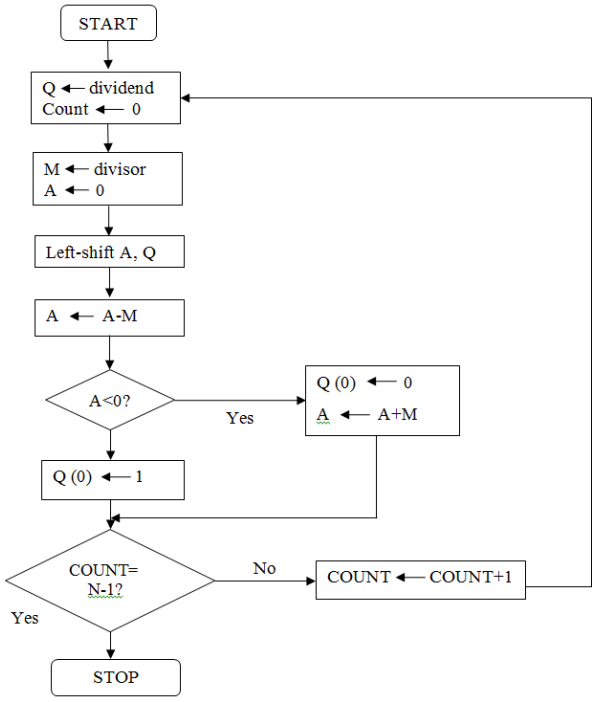

Figure 3: Flow chart of Restoring Division Algorithm

2.2.2. Non-restoring Division

The non-restoring division is based on the restoring division. The operation of each step depends on the result of the previous step. Quotient value in non-restoring division has been set of (1,-1) based of the conventional binary digit. The -1 of the quotient bit can be simply set to 0 and the quotient is the actual quotient that to find in non-restoring division. First, set Q as dividend a remainder and M as a divisor. Set the initial value of remainder and divisor. After that, the process has been shift left A, Q. the value for difference get from A-M. While the process has been shift left, Q0 value can be checked either 0 or 1. If the value A<0, the shift left start shit A, Q until get

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

the last result. The divisor subtract from the most significant bit (MSB) of the dividend. Set the next MSB of the quotient to 1 if the result positive. Subtract the divisor from the result to produce a new result. Set the next MSB of the quotient to 0 if the result is negative. The process will repeat until all the bits of the quotient are determined [5].

2.2.3. High Speed Adder/subtracted

The high speed of adder and subtracted used the binary numbers in parallel operation which is all the bits of adder and subtracted are available for computed at the same time. These design used the combination of gate logic such as XOR, AND, OR to implement the full-adder design. These are implemented by using carry adder selected to reduce the delay. The carry output of each stage is connected to the carry input of the next higher stage (ripple carry) in a parallel operation. Therefore, carry outputs of any stage and sum can't produce until the input carry occurs. Carry propagation delay means the time delay in the addition process while the sum output of each bit depends on the value of carry input. The value of "Si" in any given stage in the adder will be in its steady state final value only after the input carry to that stage has been propagated [5].

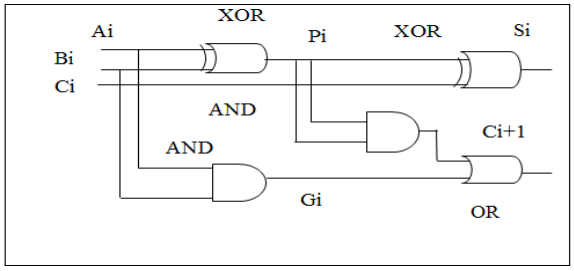

Figure 4: Logic diagram full adder

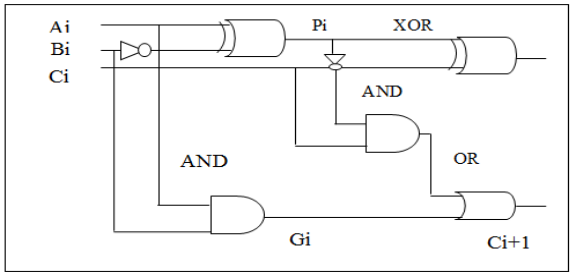

Figure 5: Logic diagram full subtraction

2.2.4. The result and simulation

The result for top RTL view of the division circuits shows in Figure 6. In that figure showed that the hardware of division which has two inputs and two outputs. The input had been declared in 8 bit divide by 8 bit which produced 16 bit of the output division. Quotient in used as the dividend of the operation in the design.

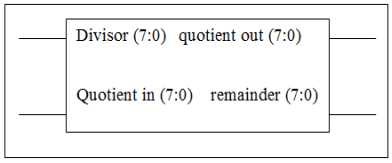

Figure 6: Top RTL view

Figure 7 shows the simulation result of 0111/0011(7/3). The 7 is stored in quotient in which is dividend and 3 are stored in divisor. Both values are in binary. After simulation of 7/3, the output of the division stored in quotient out and remainder.

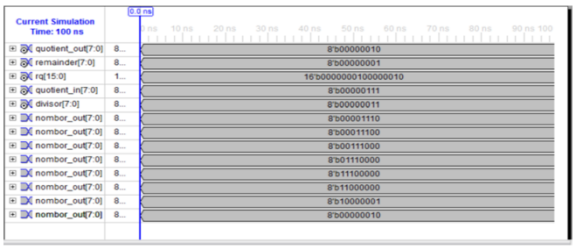

Figure 7: Output simulations

2.3. Field Programmable Gate Array

Random Access Memory (RAM) and Field Programmable Gate Array (FPGA) is a logic gate that performed the digital calculations and can be programmed. There are several types that are used as Xilinx FPGA, Altera, Atmel and Lattice Semiconductor. FPGA configuration is generally specified using a hardware description language (HDL) as that used for the specific use of an integrated circuit (ASIC). FPGA is also a semiconductor device containing programmable logic components called "logic blocks", and programmable [8]. Logic block can be programmed to perform the functions of basic logic gates such as AND,XOR or a combination of more complex functions such as decoders or simple mathematical functions such as addition, subtraction, multiplication, and division (+, -, x, ÷) [4].

In most FPGA, the logic blocks also include memory elements such as simple flip-flops. A hierarchy of programmable interconnects allows logic blocks to each other as required by the system designer. Logic blocks and interconnects can be programmed by the customer or designer, after the FPGA generated, to perform any logic function hence the name "field - programmable". FPGAs are usually slower than an application specific integrated circuit (ASIC) counterpart because they cannot handle complex designs.

2.3.1. Application of FPGA

Most of the complex designs needed to implement into FPGAs for all the features. RAMs, ROMs,

FIFOs, or CAMs are memory blocks that configured as a basic single port. The clock management is facilitated by on-chip PLL (phase-locked loop) or DLL (delay-locked loop) circuitry. Data processing is varies widely in the logic fabric devices. The FPGA with backplanes, high-speed buses, and memories are afforded the ability to link by supporting the various single added and differential I/O standards. Nowadays, FPGA can found through the system building resources such as high-speed serial I/O, arithmetic modules, embedded processors, and large amounts of memory. FPGAs have spread into a host of applications that initially seen as a vehicle for rapid prototyping and emulation systems. They were once too simple, and too costly, for anything but small-volume production. FPGAs are finding their way off the prototyping bench and into production through that the much of the larger devices and declined cost per part [1].

2.3.2. FPGA Design

FPGA can be broken down while implemented the design process. The design also make loosely definable as design entry or capture, synthesis, and place and route. Along the process of simulation, the design at the various levels abstraction as in ASIC design. FPGA design available of sophisticated and coherent tool suites to makes all the design more attractive [6].

Design entry was performed in the form of schematic capture and technology at one time. Most of the designers design the entry of the schematic used hardware description languages (HDLs). Some of the designer will preferred to mixture of the two techniques while do the design. Meanwhile, language based design entry is faster, but often at the expense of performance or density. The schematic based on design capture tools gave the designer a good performance to control the physical placement and partitioning of logic on the device [1].

Most of designers choose their own choice to used schematic or based on HDL design entry to complete the concept of their design. HDL is the better choice software or algorithmic well suited for highly complex design too easy for designer handle on how the logic must be structured. Designer work very useful to actual hardware implemented for designing the small functions when having the time. On the other hand, from the details of the hardware implementation HDLs can represent a level of abstraction that can isolate designers. Schematic-based entry gives designers much more visibility into the hardware. It's a better method for those who are hardware-oriented [7].

The design more difficult to modified or port to another FPGA while used the downside of schematic based entry [1]

2.3.3. FPGA Architecture

Generally, the typical basic of FPGA architecture that consists of an array for configuration of logic block (CLBs) and routing channel. In the architecture of FPGA have multiple I/O pads may fit into the height of one row and the width of one column at the array pads. All the routing channels have the same width (number of wires) in that architecture. An application circuit must be mapped into an FPGA with adequate resources [8].

The logic block architecture of classic FPGA as shown below consists of a 4-input lookup table (LUT), and flip-flop. Manufacturers of designing the architecture started moving configured of their logic block architecture consists of 6-input LUTs in their high performance parts, claiming increased performance in a recent years. Typical logic block there is only one output, which can be either the registered or the unregistered LUT output [8].

Lookup table (LUT) has four inputs and a clock input in the logic block architecture. Routed via special purpose dedicated routing networks in FPGA commercial as normally by since clock signals (and often other high-fan out signals) and other signal are separately managed at the clock signal. Each input of the logic block pin locations architecture is accessible from one side of the logic block, while the output pin are connected to route wires in both the channel below the logic block and the right channel of the logic block. The output pin of each logic block can connected in the channels adjacent to any of the wiring segments [8].

In generally, UN segmented is the FPGA routing show in Figure 8. That is, only one logic block of each wiring segment span before it terminated in a switch box. By turning on some of the programmable switches within a switch box, longer paths can be constructed. Some FPGA architectures used longer routing lines that span multiple logic blocks in higher speed interconnected. Modern FPGA families expand upon the above capabilities to include higher level functionality fixed into the silicon. The common of functions embedded having reduces the area required into the silicon and the speed increased for those functions compared to building from primitives. FPGAs are also widely used for systems validation including pre-silicon validation, post-silicon validation, and firmware development. The time has been reduced to the market that allows the chip companies to validate the design before the chip is produced in the factory [8].

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

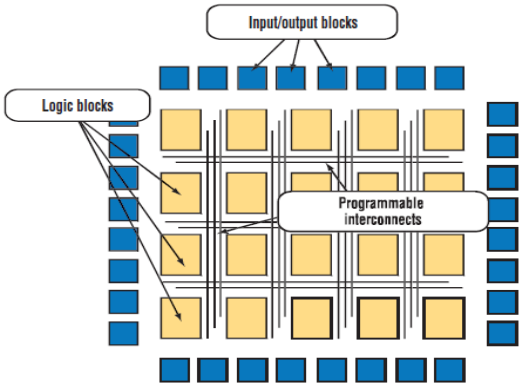

Figure 8: Architecture of FPGA

2.4. *Verilog Hardware Description Language (VHDL)*

Hardware description language (HDL) or first provide a description of the desired functional modules in the form of either schematic by users are configured to FPGA hardware. Then, the description language may synthesized to produce a binary file (usually using software provide by the FPGA manufacturer) used to configure the FPGA device. Hardware description language has some advantage is that it allows the user to both verify and a system before implemented on hardware functionality also verify. HDLs also allow for the succinct description of concurrent systems, with multiple subcomponents all operating at the same time. CPU is designed to be executed sequentially that is contrast to standard programming language. HDL programming also allowed for a more flexible and expression of system behavior is powerful between simply connecting components together using a schematic [9].

Common HDLs used in FPGA design are VHDL (VHSIC (Very High Speed Integrated Circuit) Hardware Description Language) and Verilog [8]. ADA programming language develops VHDL and has relatively the syntax. In addition, VHDL is most strongly typed and case insensitive [10]. Verilog devolved out the C programming language, and such as is a much more terse language than VHDL. Verilog is case sensitive, and also more weakly typed than VHDL [8]. The highly functionality, and widely supported by software synthesis tools similar by the two languages [10]. The VHDL has chosen used for describing and synthesizing the FPGA modules. VHDL is the stronger typing means certain errors will be caught during synthesis which might otherwise be missed in Verilog.

2.4.1. *Using Verilog in Design*

HDL has been used for some decades used to model the hardware designs as an IEEE standard. Designers are capable of partitioning their design into components that work concurrently and are able to communicate with each other using digital simulators HDL. The design of HDL simulators can simulate the design in the presence of the real hardware delays. HDL can imitate concurrency by switching between design parts in small time slots called “delta” delays. The basic features of Verilog HDL for simulation and synthesis are described in the following subsection [14].

2.4.2. *Using Verilog for Simulation*

Module called as the basic structure of Verilog in which all hardware components and test bench are described. Language constructs, in accordance to Verilog syntax and semantics form the inside of a module. These constructs are designed to facilitate the description of hardware components. The hardware component constructs for simulation, synthesis, and specification of test bench to specify test data and monitor circuit responses. Encloses a design’s of a module has been described to test the module under design in which case it is regarded as the test bench of the design [11]. Figure 9 show the model of simulation of a design that consists with a Verilog test bench [11]. Verilog model of the Verilog constructs being tested are responsible for the description of its hardware. While the language constructs the model used in a test bench are in charge of providing appropriate input data. Verilog model also applied the data stored in a text file to the module being tested, and analysis or display of its outputs. The output of simulation is generated in the form of a waveform for visual inspection. Data files for record or for machine readability generated the output simulation in the form [11].

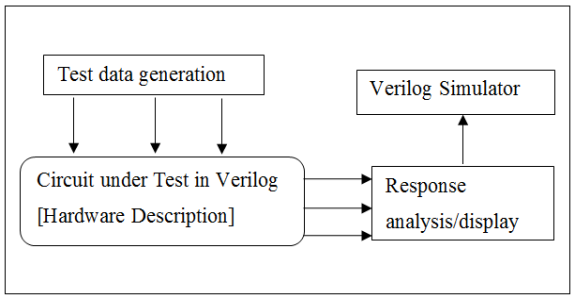

Figure 9: Simulations in Verilog

2.4.3. *Using Verilog for Synthesis*

The design synthesized into a net list of components in a target library after the design passes the basic functional validation. The target library is the specification of the hardware that the design is being synthesized. Design for its verification Verilog description or those for timing checks and timing

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

specifications are not synthesizable using the Verilog constructs. A Verilog design that is to be synthesized must use language constructs that have a complete hardware correspondence.

A block diagram specifying the synthesis process shows in Figure 10 [11]. The target library of specification is the input of a synthesis tool. The outputs of synthesis are a net list of components of the target library, and timing specification and other physical details of the synthesized design. The synthesis tools have an option to generate this net list in Verilog [11].

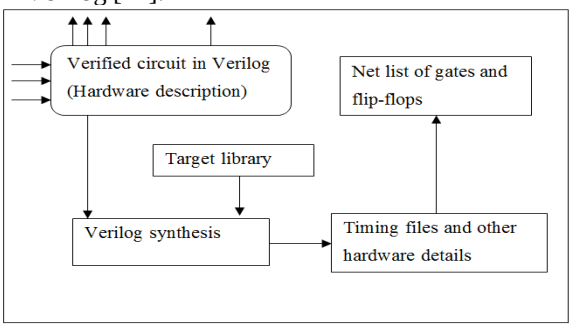

Figure 10: Synthesis of Verilog Design

2.5. *Xilinx Spartan-II FPGA*

FPGA is the Spartan-II family which has a regular flexible, programmable architecture of Configurable Logic Blocks (CLBs), surrounded by a perimeter of programmable Input/output Blocks (IOBs). Spartan-II has four Delay-Locked Loops (DLLs), one in each corner of the die. The block RAM lie has two columns on opposite side of the die, between the CLBs and the IOB columns. These functional elements are interconnected by a powerful hierarchy of versatile routing channels. Customize the Spartan-II FPGA by loading the configuration data into internal static memory cells [1].

Figure 11 shows an external serial PROM (master serial mode) can be read the data configuration. It also can be written into the FPGA in slave serial, slave parallel, or Boundary Scan modes. High-volume application typically use in Spartan-II FPGA where the versatility of a fast programmable solution adds benefits. The cost is effective solution for high volume production for Spartan-II FPGA. It is ideal for shortening product development cycles. Spartan-II FPGAs also achieve high-performance, low-cost operation through advanced architecture and semiconductor technology [12]. The system clock rates up to 200MHz provide for their devices. On chip synchronous single port and dual port RAM are the conventional benefits for Spartan II FPGA. Other benefits of high volume programmable logic solutions are DLL clock drivers, reset on all flip flop and programmable set [13].

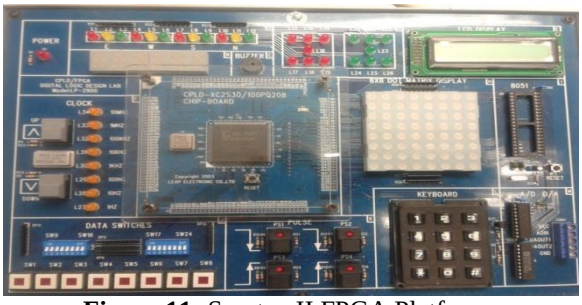

Figure 11: Spartan II FPGA Platforms

3. METHODOLOGY

Design flow and methodology is defines as the tools and methods used by designer to build the design. Although design flows vary, it is difficult for a single design that the flow is defined and well supported.

3.1. *Design flow*

First step is to design the division algorithm by using Xilinx ISE 10.1. After that, the program has been synthesized. The second step is compiling the program by check the syntax of behavioral. Third step is simulated the test bench program and check the output by using Xilinx ISE simulator. Fourth step is creating a pin input output on a floor plan area and generate the map. Lastly, download the program to the FPGA trainer and verify the input and output [14].

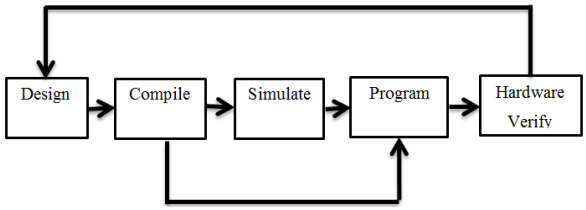

Figure 12: Design flow [15-20]

3.2. *Project Methodology*

The project methodology is illustrated in Figure 13. There are few steps that have been involved in order to achieve the objectives of the project. Each project needs suitable strategy on planning in order to run the project smoothly before start the project[21-31].

As soon as the purpose and function of the project are clear, the method of the project is divided into several parts. In this system show the process of developing the program of eight bit division algorithm circuit using the Verilog language. This coding is generating using the block diagram that was designed which includes logic gates.

Furthermore, after designing the diagram of program, the algorithm developed using the Xilinx ISE software to generate the circuit for eight bits division. If the program has error, the process goes to the previous process. If the program no error, the next

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

process is burn program into FPGA memory. Once program of eight bit division is fully finished, the coding has been testing on board FPGA. If successful, the next process has been done.

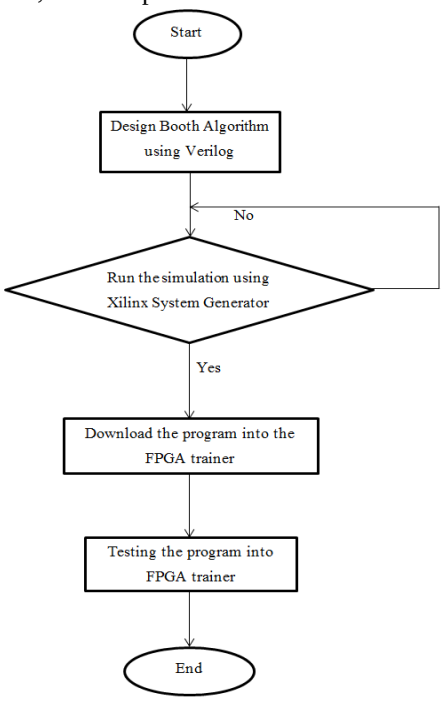

Figure 13: Flowchart of Division Algorithm

4. RESULT AND DISCUSSION

Implementation of 8 bit division using Verilog High Description Language and its testing on the Spartan 2 FPGA. The algorithm will implement from the simple division design until the complex design by follow the flowchart and table.

4.1. Design division algorithm

Eight bit Division will work when as the operation of division started. From the Figure 14 show that the top module has an input output of the diagram. The input eight bit declared as divisor and quotient in. Meanwhile at the output also shows the eight bit for quotient out and remainder. From the top module of eight bit division, assume that divisor (A3), quotient in (dividend), quotient out (A1) and remainder (A0). LED_COM at the block diagram show while the program has been connected at the FPGA.

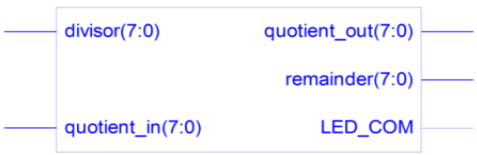

Figure 14: Block diagram

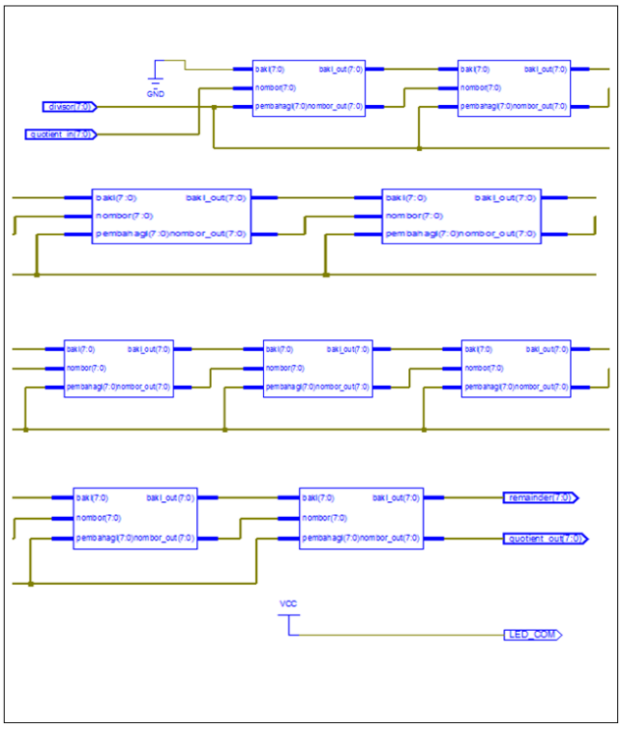

Figure 15: Full RTL schematic

Figure 15 shows that the top design of full operation schematic for connection of eight bit division. At the top design of division program have state1, state2, state3, state4, state5, state6, state7, and state8 will be declared for the wire operation in the top module.

Figure 16 shows the combinational logic gate gets from full RTL schematic. In combinational logic gate have some of logic gate used for algorithm. The logic gate has been used such as ALU logic, AND gate, and NOR gate. The entire gates connect to the input and output algorithm. The block ALU logic connect to NOR gate and the output of NOR gate connect to AND gate.

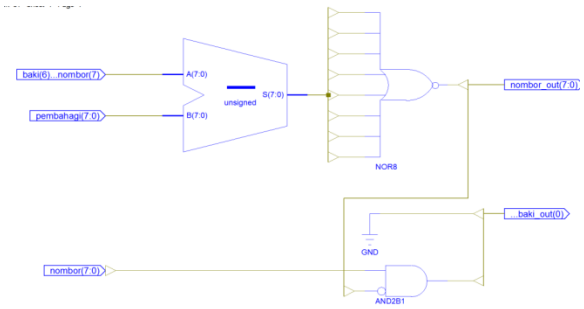

Figure 16: Combinational logic gates

Figure 17 shows that operation of division. The result will shift left for 8 bit division. Shift left at the result start with divisor divide by quotient in. The shift left from number out state1 until state8. At state7, the

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

shift stop while the remainder=different value. Set 1 at quotient value and add the value after last shift operation. After that, the shifts continue from the new value and get the last value from the division operation in 8 bit at state8. The full result to display at FPGA is RQ value in 16 bit of the division algorithm.

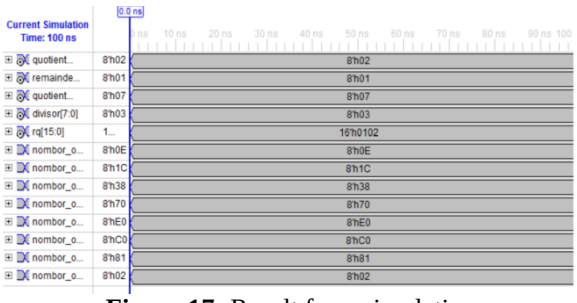

Figure 17: Result from simulation

After simulated the program at the software, FPGA used to burn the division algorithm. The pin has been set to connect the input and output program at FPGA. The quotient_in and divisor is used DIP switch. The output has been display at LED. Start from L1 until L8 for the quotient_out output. Results for remainder start from L8 until L12. The left hand side is MSB and right hand side is LSB.

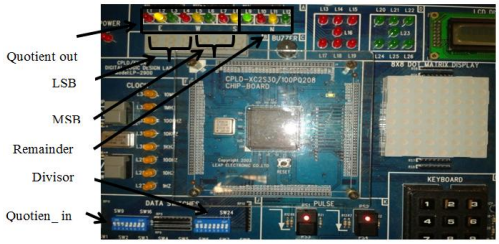

Figure 18: Results at FPGA trainer

4.2. Discussion

This project is to design 8-bit division algorithm using Verilog language. The programs are dividing by 2 top modules. First top module for called the function of division program and second is top module for division algorithm. The project design 2 input present by quotient in and divisor, 2 outputs present by quotient out and remainder. Quotient in and divisor will present 8-bit input and the output will represent 16-bit.

The algorithm implemented using Xilinx ISE 10.1 software to generate the circuit for eight bits division. If the program has some error, the process is not done. Fine the error until the program successful. Once the program of eight bit division is finish, the program can be testing on FPGA trainer board.

5. CONCLUSIONS AND FUTURE WORK

In conclusion, eight bit division circuit has been designed and developed. Furthermore, eight bit division circuit is involved in performing the division for each bit by performs arithmetic operation such as addition, subtraction and multiplier. This project has some benefit offered by proposed system in FPGA that simplify the complex design. In addition, the design proposed up to more benefit to get better profit. In future, the division algorithm is recommended to be implemented in other devices. Division algorithm also can be designed in a large bit such as 32 bits that will increase performance of the devices. Another recommendation is that change the display from LED to display on 7 segments or LCD display at FPGA trainer board.

Acknowledgments

We are grateful to Centre for Telecommunication Research and Innovation (CeTRI) and Universiti Teknikal Malaysia Melaka (UTeM) through PJP/2013/FKEKK (47B)/S01274 for their kind and help for supporting financially and supplying the electronic components and giving their laboratory facility to complete this study.

References

[1] **A. Jaafar, N. Arasid, N. M. Z. Hashim, A. A. Latiff, and H. Rafis, "INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY WINGS TO YOUR THOUGHTS ….. Three Bit Subtraction Circuit via Field Programmable Gate Array," vol. 1, no. Viii, pp. 1–13, 2013.**

[2] **D. G. Bailey, "Space Efficient Division on FPGAs," pp. 206–211, 2006.**

[3] **A. Language, "Binary division Hardware design 1 Hardware design 2 Hardware design 3," pp. 1–4.**

[4] **A. Language, "Binary division CS / COE0447 : Computer Organization and Assembly Language Hardware design 1 Hardware design 2 Hardware design 3 Exercise sheet Restoring division."**

[5] **S. Kaur, M. S. Manna, and R. Agarwal, "VHDL Implementation of Non Restoring Division Algorithm Using High Speed Adder / Subtractor," vol. 2, no. 7, pp. 3317–3324, 2013.**

[6] **L. Chin, "FPGA Based Embedded Vision Systems Final Year Project," no. October, 2006.**

[7] **A. J. Hodnett, "Addition , Subtraction , and Multiplication of Unsigned Binary Numbers Using FPGA," pp. 1–20, 2009.**

[8] **T. Abound, "in FPGA Design," 2003.**

[9] **I. Chiuchisan, A. D. Potorac, and A. Graur, "Finite State Machine Design and VHDL Coding Techniques," 2010.**

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

[10] N. M. Z. Hashim, M. H. A. Halim, H. Bakri, S. H. Husin, and M. M. Said, "Vehicle Security System Using Zigbee," vol. 3, no. 9, pp. 2–7, 2013.
[11] Z. Navabi, Digital System Test and Testable Design. Boston, MA: Springer US, 2011.
[12] D.- June and B. Ram, "Spartan-II FPGA Family Module 1 : Module 3 : DC and Switching Characteristics Module 2 : Module 4 : Spartan-II FPGA Family : Introduction and Ordering," pp. 1–99, 2008.
[13] T. Spartan and C. Solutions, "Spartan-II 2 . 5 V Family Field Programmable Gate Arrays Spartan-II 2 . 5 V Family Field Programmable Gate Arrays XC2S15 CLBs BLOCK RAM CLBs DLL I / O CELLs," vol. 001, pp. 1–49, 1999.
[14] Assoc.P.Dr.A.S.Jamal,"Digital Integrated Circuit Design", January 2011.
[15] N. M. Z. Hashim, N. A. Ali, A. Salleh, A. S. Ja'afar, and N. A. Z. Abidin, "Development of Optimal Photosensors Based Heart Pulse Detector," Int. J. Eng. Technol., vol. 5, no. 4, pp. 3601–3607, 2013.
[16] N. M. Z. Hashim and N. B. Hamdan, "Flood Detector Emergency Warning System," Int. J. Eng. Comput. Sci., vol. 2, no. 8, pp. 2332–2336, 2013.
[17] N. M. Z. Hashim, N. M. T. N. Ibrahim, Z. Zakaria, F. Syahrial, and H. Bakri, "Development New Press Machine using Programmable Logic Controller," Int. J. Eng. Comput. Sci., vol. 2, no. 8, pp. 2310–2314, 2013.
[18] N. M. Z. Hashim, S. H. Husin, A. S. Ja'afar, and N. A. A. Hamid, "Smart Wiper Control System," Int. J. Appl. or Innov. Eng. Manag., vol. 2, no. 7, pp. 409–415, 2013.
[19] N. M. Z. Hashim, A. F. Jaafar, Z. Zakaria, A. Salleh, and R. A. Hamzah, "Smart Casing for Desktop Personal Computer," Int. J. Eng. Comput. Sci., vol. 2, no. 8, pp. 2337–2342, 2013.
[20] A. Salleh, N. R. Mohanad, N. M. Z. Hashim, M. Z. A. A. Aziz, and M. H. Misran, "Design of Wideband Microstrip Bandpass Filter for S-Band Application," Aust. J. Basic Appl. Sci., vol. 8, no. 4, pp. 843–848, 2014.
[21] S. H. Husin, A. A. Ngahdiman, N. M. Z. Hashim, Y. Yusop, and A. S. Ja'afar, "Home Electrical Appliances Smart System," Int. J. Comput. Sci. Mob. Comput., vol. 2, no. 9, pp. 85–91, 2013.
[22] A. S. Ja'afar, N. M. Z. Hashim, A. A. M. Isa, N. A. Ali, and A. M. Darsono, "Analysis of Indoor Location and Positioning via Wi-Fi Signals at FKEKK , UTeM," Int. J. Eng. Technol., vol. 5, no. 4, pp. 3570–3579, 2013.
[23] N. R. Mohamad, A. S. A. M. Soh, A. Salleh, N. M. Z. Hashim, M. Z. A. A. Aziz, N. Sarimin, A. Othman, and Z. A. Ghani, "Development of Aquaponic System using Solar Powered Control Pump," IOSR J. Electr. Electron. Eng., vol. 8, no. 6, pp. 1–6, 2013.
[24] N. M. Z. Hashim and N. Arifin, "Laboratory Inventory System," Int. J. Sci. Res. (IJSR …, vol. 2, no. 8, pp. 261–264, 2013.
[25] N. M. Z. Hashim, N. A. Ibrahim, N. M. Saad, F. Sakaguchi, and Z. Zakaria, "Barcode Recognition System," Int. J. Emerg. Trends Technol. Comput. Sci., vol. 2, no. 4, pp. 278–283, 2013.
[26] N. M. Z. Hashim and S. N. K. S. Mohamed, "Development of Student Information System," Int. J. Sci. Res., vol. 2, no. 8, pp. 256–260, 2013.
[27] A. Salleh, N. R. Mohamad, M. Z. A. A. Aziz, M. N. Z. Hashim, M. H. Misran, and M. A. Othman, "Design the High Gain and Low Power Amplifier for Radio over Fiber Technology at 2 . 4 GHz," no. 09, pp. 163–170, 2013.
[28] N. M. Z. Hashim, N. H. Mohamad, Z. Zakaria, H. Bakri, and F. Sakaguchi, "Development of Tomato Inspection and Grading System using Image Processing," International Journal Of Engineering And Computer Science (IJECS), vol. 2 no. 8, pp. 2319-2326, 2013.
[29] A. Salleh, N. R. Mohamad, A. M. Aziz, M. H. Misran, M. A. Othman, and N. M. Z. Hashim, "Simulation of WiMAX System Based on OFDM Model with Difference Adaptive Modulation Techniques," International Journal of Computer Science and Mobile Computing, vol. 2 no. 9, pp. 178-183, 2013.
[30] S. H. Husin, M. Y. N. Hassan, N. M. Z. Hashim, Y. Yusop, A. Salleh, "Remote Temperature Monitoring and Controlling," International Journal for Advance Research in Engineering and Technology (IJARET), vol. 1. no. 8, pp. 40-47, 2013.
[31] A. Salleh, M. Z. A. Abd Aziz, N. R. Mohamad, M. H. Misran, M. A. Othman, N. M. Z. Hashim,"Simulation of 2.4 GHz Low Power RF Front End Design for Radio over Fiber Technology," International Journal of Emerging Trends in Engineering and Development, vol. 5, no. 3, pp. 345–354, 2013.