Introduction to FPGA System Design Coursework 2

***Abstract*—This research shows how field programmable gate arrays designs meet with the required specifications of the circuits and the stages and process its gone through to achieve this. This research paper contains the microarchitecture of the various circuits, Register transfer level Validation and The FPGA implementation used. Various pictures in the research paper show the post placement/Routing validation and timing simulation. Several findings and results are seen throughout the research on each circuit such as the Mac unit composition.**

***Index Terms*—Arrays, programmable, specifications, Microarchitecture, implementation, Validation, Simulation , inferring, Dense**

I. INTRODUCTION

Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs have two huge things going for them. First, they enable you to build exactly the hardware you need, instead of having to use the same application-specific standard product (ASSP) all your competitors are using, or having to undertake the time, cost, and risk of an application-specific integrated circuit (ASIC) design. But just as important, that ability to customize the FPGA means that often, in an FPGA, you can do operations in a simpler, faster, more energy-efficient way than they could be done in the microprocessor cores of an ASSP. g. An FPGA enables you to program product features and functions, adapt to new standards, and reconfigure hardware for specific applications even after the product has been installed in the field — hence the term field programmable. And gate arrays are two-dimensional arrays of logic gates. If you get enough of these things put together, you can make those simple calculations add up to do something meaningful. an FPGA allows you flexibility in your designs and is a way to change how parts of a system work without introducing a large amount of cost and risk of delays into the design schedule. Our design flow includes the use of **Linux** (UNIX OS), this is highly used cause its free, its portable on any hard ware, stable, secure, it keeps on running. Also the **BASH shell** is used, Bash is a command processor that typically runs in a text window. Bash can also read commands from a file, called a script. Like all Unix shells, it supports filename globbing (wildcard matching), piping, here documents, command substitution, variables and control structures for condition-testing and iteration.

II. MICROARCHITECTURE

Microarchitecture is the way a given instruction set architecture (ISA) is implemented in a particular processor. It implements the architecture and defines specific mechanisms and structures for achieving that implementation of the circuit. The mac unit which has been designed consists of four registers comprising of 4 D type flip flops, having two signals a clk and a reset and an input ‘D’ and one output ‘Q’. It also has the multiply which could be either an array multiplier Wallace tree multiplier and DADDA , which is an internal signal and the adder named ‘I_ADD’ also an internal signal . The multiply accumulate has two input signals ‘a’ and ‘b’ which are driven, six internal vector signals, and one output ‘acc_r’. Multiply and accumulate is used to implement most digital

signal processing applications because they require the computation of the sum of the products. A MAC consists of a multiplier and a special register called Accumulator. This circuits function is designed firstly by programming in vhdl the multiplier for called a parametric multiplier then the adder for it called a parametric adder.

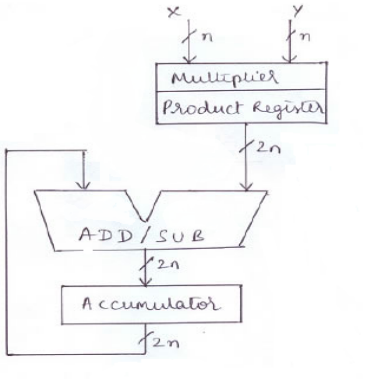


Fig 1.1 A mac unit architecture model

It consists of a multiplier and a special register called Accumulator. Addition and multiplication are two different operations, although they can be
performed in parallel. By the time the multiplier is computing the product, accumulator accumulates the product of the previous multiplications. Hence if N products are to be

accumulated, N-1 multiplications can overlap with N-1 additions. During the very first
multiplication, accumulator will be idle and during the last accumulation, multiplier will
be idle. Thus, N+1 clock cycles are required to compute the sum of N products. The functionality of the circuit is provided in three areas being signed and unsigned multipliers, multiply- accumulate supporting 16 x16 unsigned mac in 1 cycle or 32 x32 unsigned in 3 clk cycles. Also fixed point fractional operands. Lastly Register-based arithmetic operations.

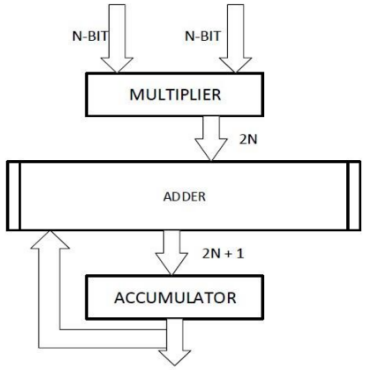


Fig 1.2 Block diagram

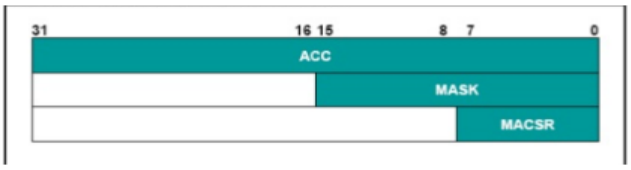


Fig 1.3 register based arithmetic operations.

The Truth table for the D type flip flop gates which are implemented in the mac unit

Inputs		Outputs	
Clk	D	Q	Reset
0	x	No change	No change
1	0	0	1
1	1	1	0

Fig 1.4 (Truth Table D type flip flop)

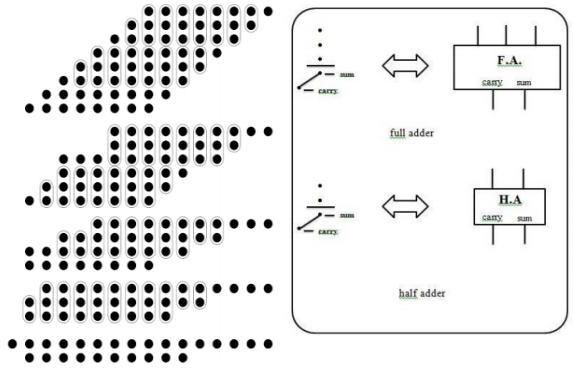


Fig 1.5 Multiplier (Example of an 8 bit Wallace tree multiplier).

The Inferring single-port block RAMs. The BRAM Block is a configurable memory module that attaches to a variety of BRAM Interface Controllers, they are used for dense storage in applications. The BRAM is a structural design that instantiates a number of BRAM primitives, depending on specific factors.There are synchronous dual port and single port Brams.

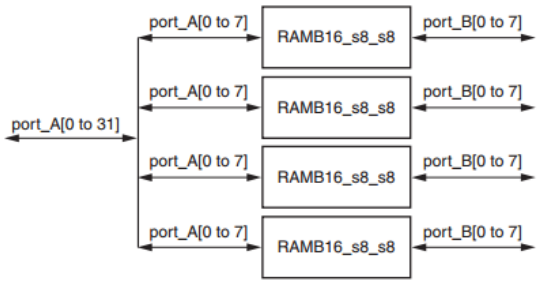


Fig 2.1 Block Ram block implementation

When Block Memory is enabled, all memory operations occur on the active edge of the clock input (CLK). The Block Memory can be configured to be active on the rising edge and the falling edge. When the block memory is disabled (enable inactive), the memory configuration and output value remain unaltered.

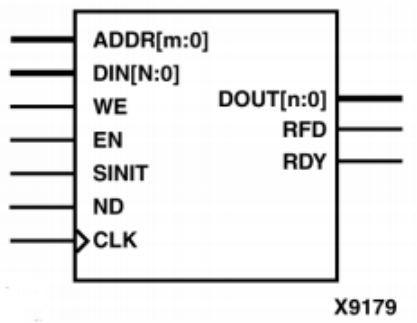


Fig 2.2 single port block memory schematic

The Single-Port Block Memory module is generated based on the user-specified width and depth. This module for Spartan-II and Virtex is composed of single or multiple 4 Kb blocks called SelectRAM+. The Virtex-II, Virtex-II Pro, Virtex-4, and Spartan-3 Single-Port Block Memory modules, on the other hand, are composed of single or multiple 18 Kb blocks called Select RAM-II. Since Spartan-II and Virtex both use the 4 Kb Select RAM+ blocks, any particular reference to a Virtex implementation also applies to a Spartan-II, Virtex-E, Virtex-II Pro, or Spartan-IIE implementation. The enable pin in the bram affects the read, write, and SINIT functionality of the port. When the Block Memory has an inactive enable pin, the output pins are held in the previous state and writing to the memory is disabled. By default the enable pin is active high. Users, however, have the option to configure the enable pin active high or active low. Configuring the enable pin active low will not use extra resources.

The write enable pin enables writing to the memory locations. When active, the contents of the DIN bus is written to memory at the address pointed to by the ADDR bus. The output latches are loaded or not loaded according to the write configuration (Write First, Read First, No Change). When WE is inactive, a read operation occurs, and the contents of the memory addressed by the ADDR bus are driven on the DOUT bus. In the Read-Only port configuration (ROM configuration), the WE pin is not available. By default, the write enable pin is active high. Users, however, have the option to configure the write enable pin active high or active low. Configuring the write enable pin active low will not use extra resources.

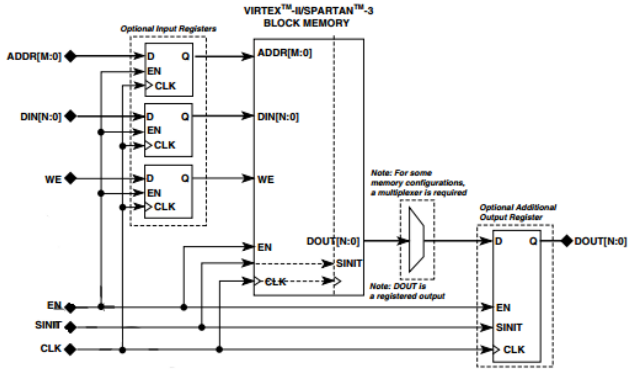


Fig 2.3 single port memory block diagram

Moore finite state machine. The general architecture of an FSM consists of a combinational block of next state logic, state registers, and combinational output logic the Moore machine's output depends on state variables only not on inputs. Moore machine's output depends on state variables only not on inputs. An FSM is specified by five entries, symbolic states, input signals, output signals, ext-state function and output function. FSM transits from one state to another, the new state is then determined by the next state function, which is a function of the current state and input signal.

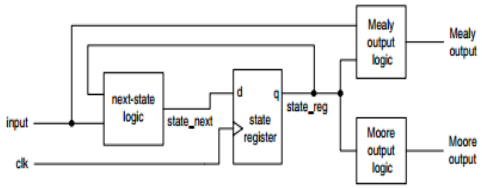


Fig 3.1 Block diagram of an fsm.

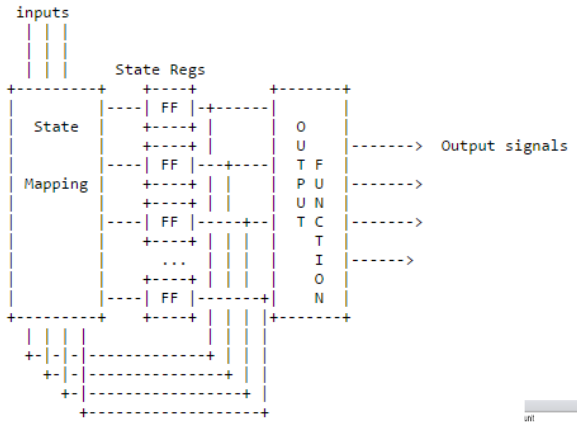


Fig 3.2 Moore finite state machine schematic.

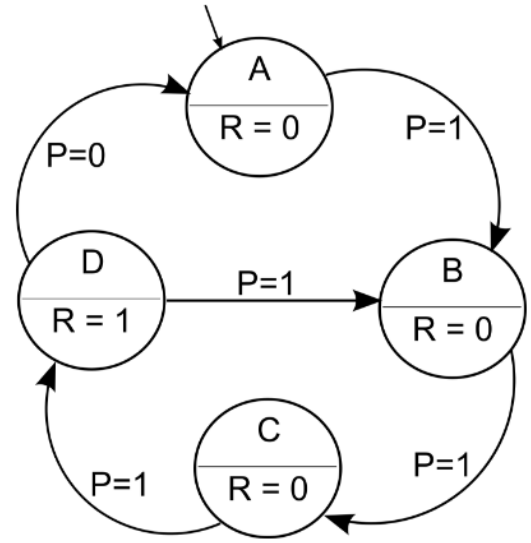


Fig 3.3. A finite state machine.

III. RTL VALIDATION.

Register transfer level is used to create high level representations of the circuits, from which lower level representations and ultimately actual wiring can be derived. Design at the RTL level is typically practiced in modern digital design. The mac unit which has been designed has a test scenario which is called a test bench for it to be able to test the design fully. The implemented test scenario has an architecture which describes the functionality of the design. The architecture contains declared components, Internal signals, definitions of clock and reset processes, stimulus process. It also verifies the output acc_r and drives the inputs A and B.

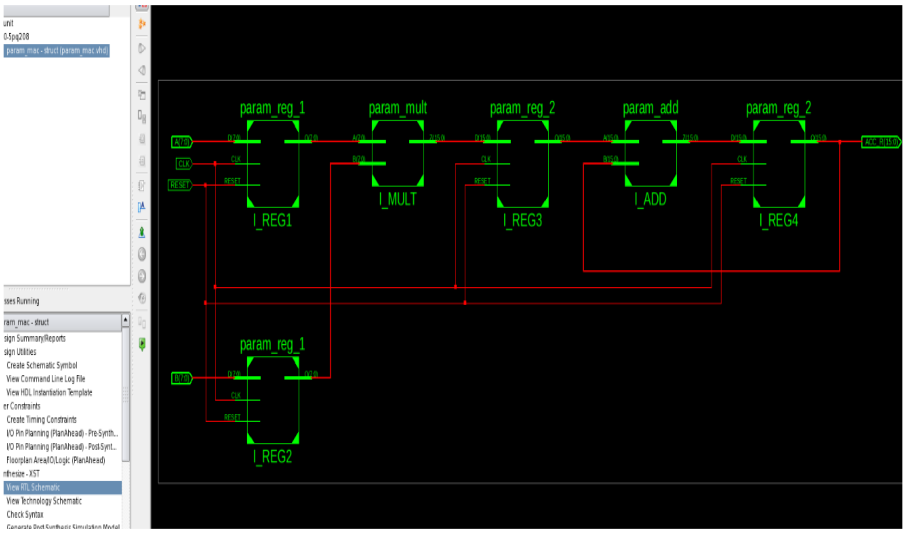


Fig 4.1 RTL schematic of mac unit.

It takes 10 nanoseconds for the clock to go from a ‘1’ to a ‘0’ and as the clock goes from 1 to 0 the reset remains at 0 with it being defined in the test bench for it to go to a ‘1’ in 10ns once when the clock goes high, low. The inputs a and b start with a ‘0’at first then go up in steps going at 1 first at 150ns then input b follows a ‘1’ at 170ns, then 3 and like that b follows 20ns after input a.

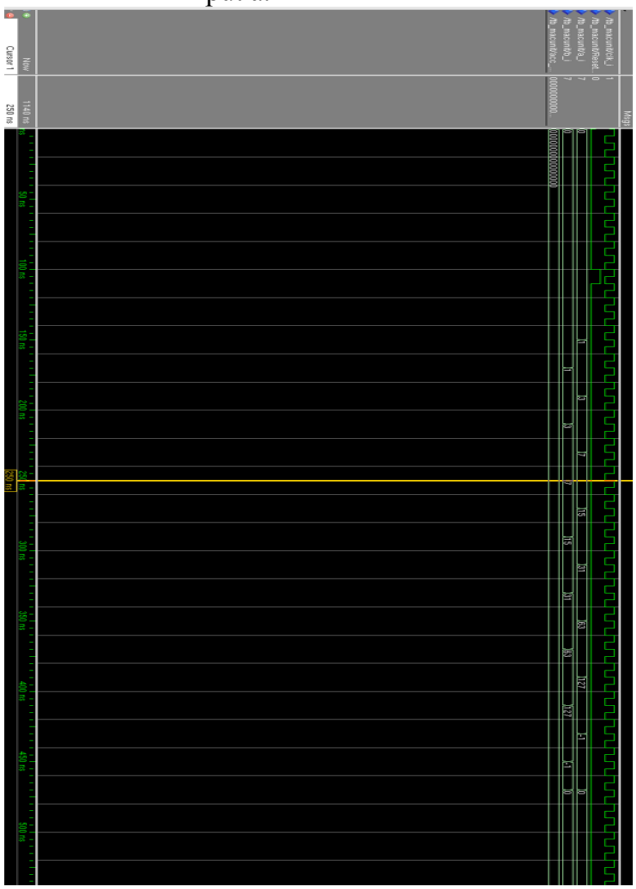


Fig 4.2 RTL Validation diagram for mac unit

Validation of the Bram. The block ram test scenario involves linking two d type flip flop with an enable to the parametric bram. The test scenario also contained the architecture which is the functionality of the design where the internal signals where declared and various processes written .The data goes through only when the enable is on and then it is stored, then the process repeats it self. The test scenario defines and shows where it first writes then reads.

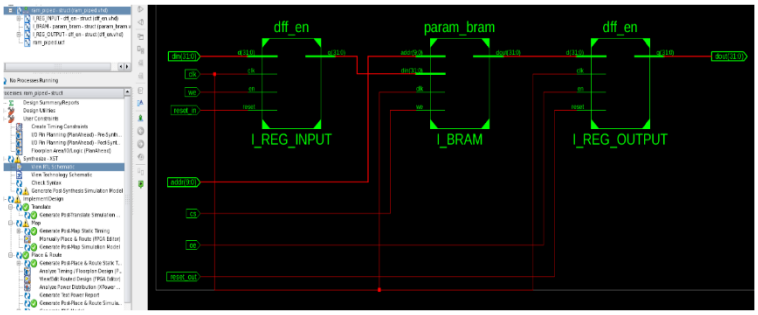


Fig 4.3 RTL schematic of the BRAM

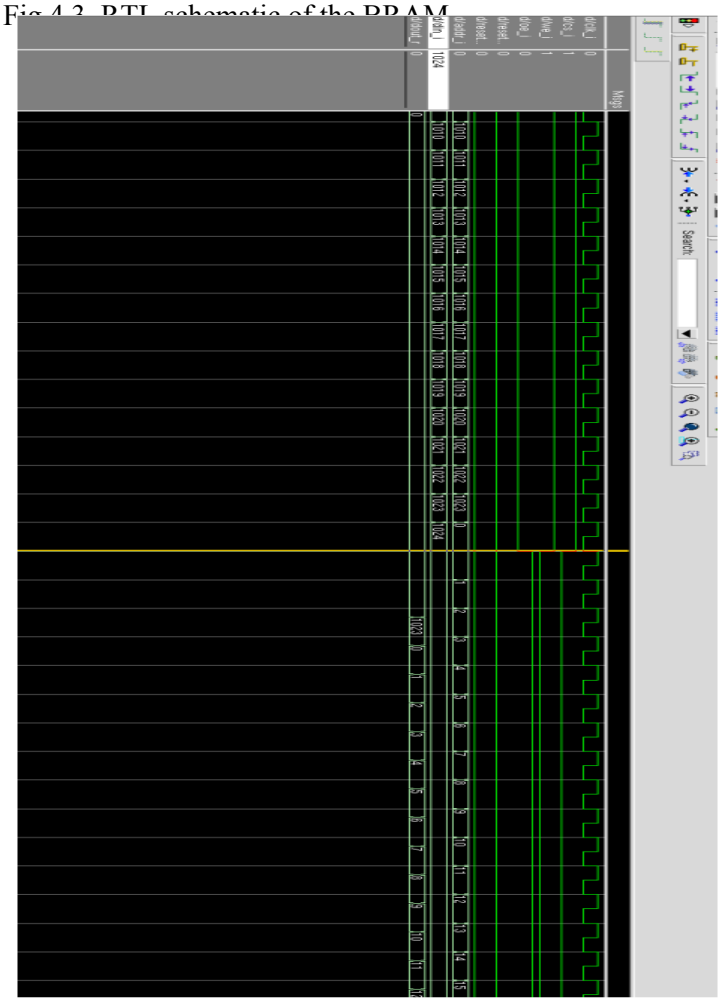


Fig 4.4. RTL validation of the Bram, (also showing the read section first then the write).

IV. FPGA IMPLEMENTATION.

The fpga flow which has been followed comprises design entry, design synthesis, design implementation, and Xilinx device programming. Design. Design verification which

includes both functional verification and timing verification, takes places at different point during the design flow. Design entry is the process of creating a project, created files are added to the project including the UCF file which contains the timing constraints used.

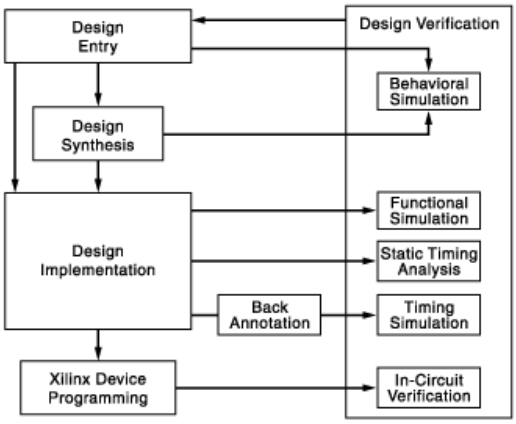


Fig 5.1 Fpga Implementation diagram.

The verification of the function for the design was done after functional simulation has been ran and translated (gate level simulation) using the simprim library. Design synthesis checked the code syntax and analyzed the hierarchy of the design ensuring its optimized for the design architecture selected.

Timing constraints describes the timing performance requirements of the design it allows comparing constraints to the performance of the resulting design. The timing constraints used for the circuits involves clock domain, inputs and outputs. The clock domain have a duty cycle of 50%, 20ns period , the EDGE set to High.

The input and output set to half of the clk period, 10ns, and the clock edge to rising with a valid duration of half the clock period(10ns).

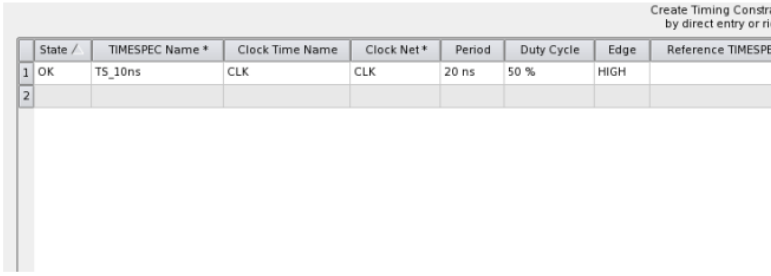


Fig 5.2. Time constraints for both mac unit and Bram(Clock domain).

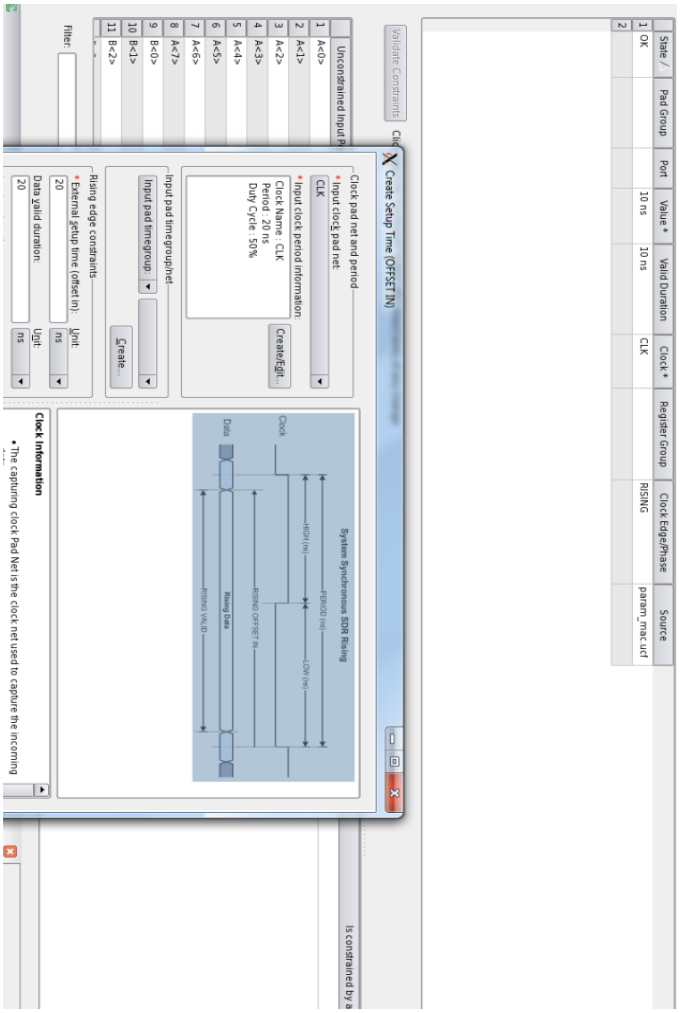


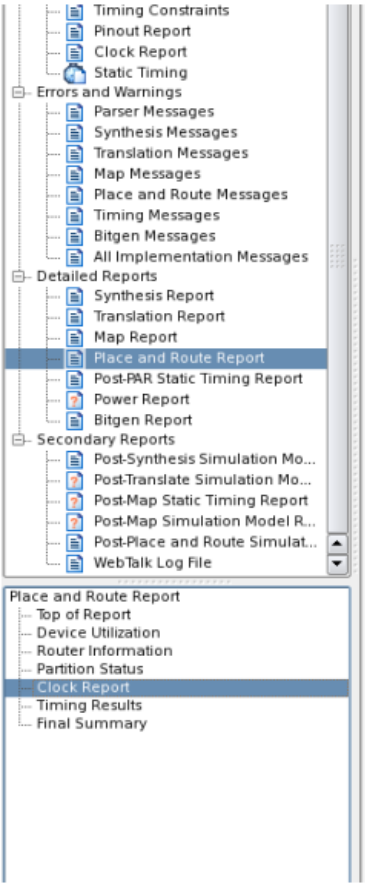
Fig 5.3..(Inputs and output defined constraints)

The place and route places and routes the design to the timing constraints. The Map report or the

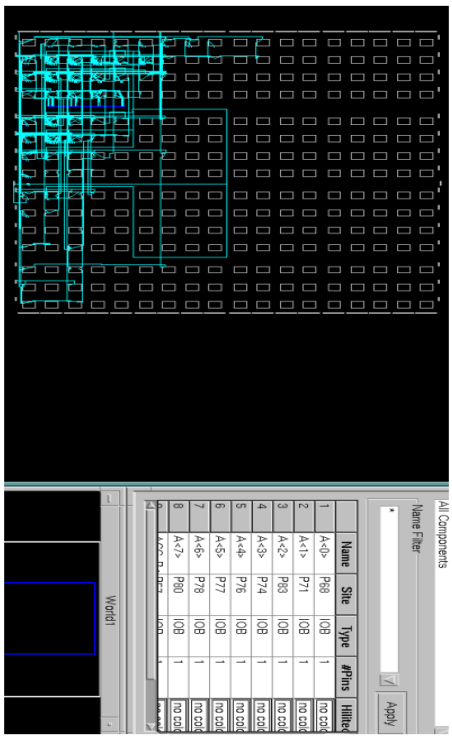
Place and route report shows the circuit timing of the circuits.

Fig 5.4. The place and route timing results report for the circuits.

Fig 5.5. Placed and routed design of the mac unit.




```
+---------------------+--------------+------+------+------------+-------------+
|        Clock Net    |   Resource   |Locked|Fanout|Net Skew(ns)|Max Delay(ns)|
+---------------------+--------------+------+------+------------+-------------+
|           CLK_BUFGP |      BUFGMUX0| No   |   40 |  0.109     |  0.676      |
+---------------------+--------------+------+------+------------+-------------+

* Net Skew is the difference between the minimum and maximum routing
only delays for the net. Note this is different from Clock Skew which
is reported in TRCE timing report. Clock Skew is the difference between
the minimum and maximum path delays which includes logic delays.

* The fanout is the number of component pins not the individual BEL loads,
for example SLICE loads not FF loads.

Timing Score: 0 (Setup: 0, Hold: 0, Component Switching Limit: 0)

Asterisk (*) preceding a constraint indicates it was not met.
   This may be due to a setup or hold violation.

------------------------------------------------------------------------------------------------------
  Constraint                                |    Check    | Worst Case |  Best Case | Timing |   Timing
                                            |             |    Slack   | Achievable | Errors |    Score
------------------------------------------------------------------------------------------------------
  OFFSET = OUT 10 ns AFTER COMP "CLK" "RISI | MAXDELAY    |     1.971ns|     8.029ns|       0|           0
  NG"                                       |             |            |            |        |
------------------------------------------------------------------------------------------------------
  OFFSET = IN 10 ns VALID 10 ns BEFORE COMP | SETUP       |     3.598ns|     6.402ns|       0|           0
   "CLK" "RISING"                           | HOLD        |     0.541ns|            |       0|           0
------------------------------------------------------------------------------------------------------
  TS_10ns = PERIOD TIMEGRP "CLK" 20 ns HIGH | SETUP       |    12.424ns|     7.576ns|       0|           0
   50%                                      | HOLD        |     1.160ns|            |       0|           0
------------------------------------------------------------------------------------------------------


All constraints were met.


Generating Pad Report.

All signals are completely routed.

Total REAL time to PAR completion: 3 secs
Total CPU time to PAR completion: 3 secs

Peak Memory Usage:  590 MB

Placement: Completed - No errors found.
Routing: Completed - No errors found.
Timing: Completed - No errors found.

Number of error messages: 0
Number of warning messages: 0
```



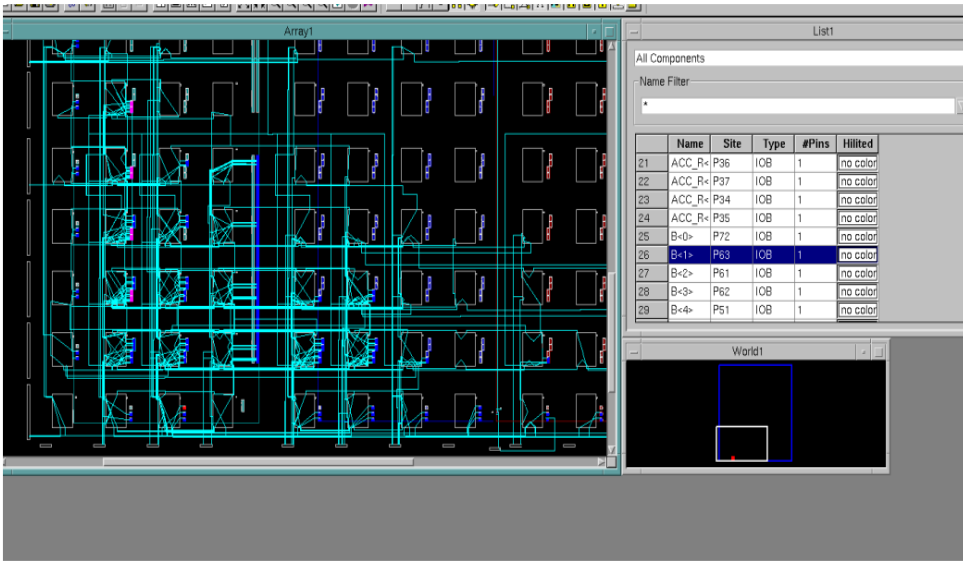


Fig 5.7. Zoom showing more placed and routed design of mac unit.

The Bram placed and routed design.

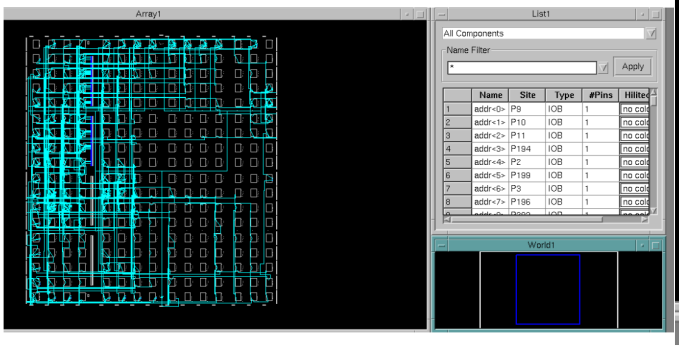


Fig 5.8

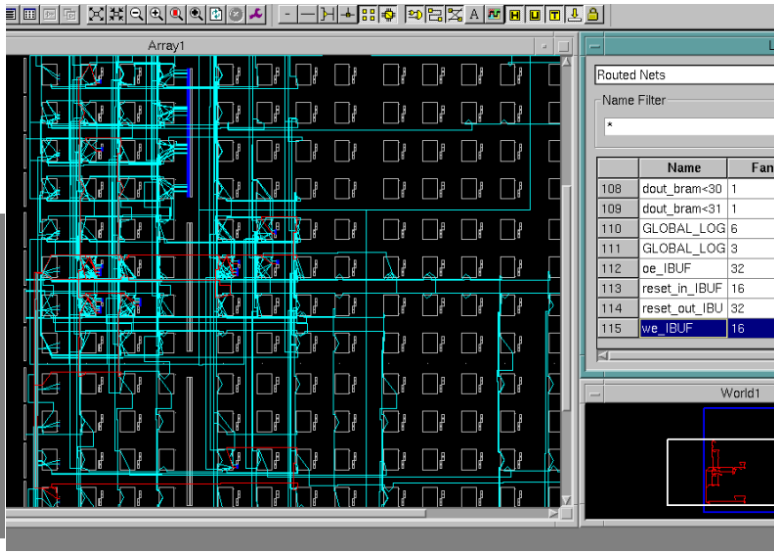


Fig 6.1. The write enable design highlighted in red.

V. POST PLACEMENT/ROUTING

Full timing simulation of the Mac unit. (Fig 6.2

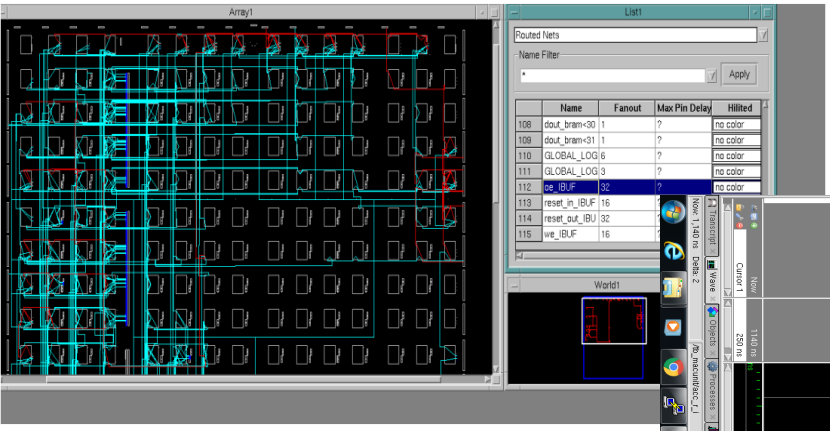


Fig 5.9. (Zoom in showing in oe of the I

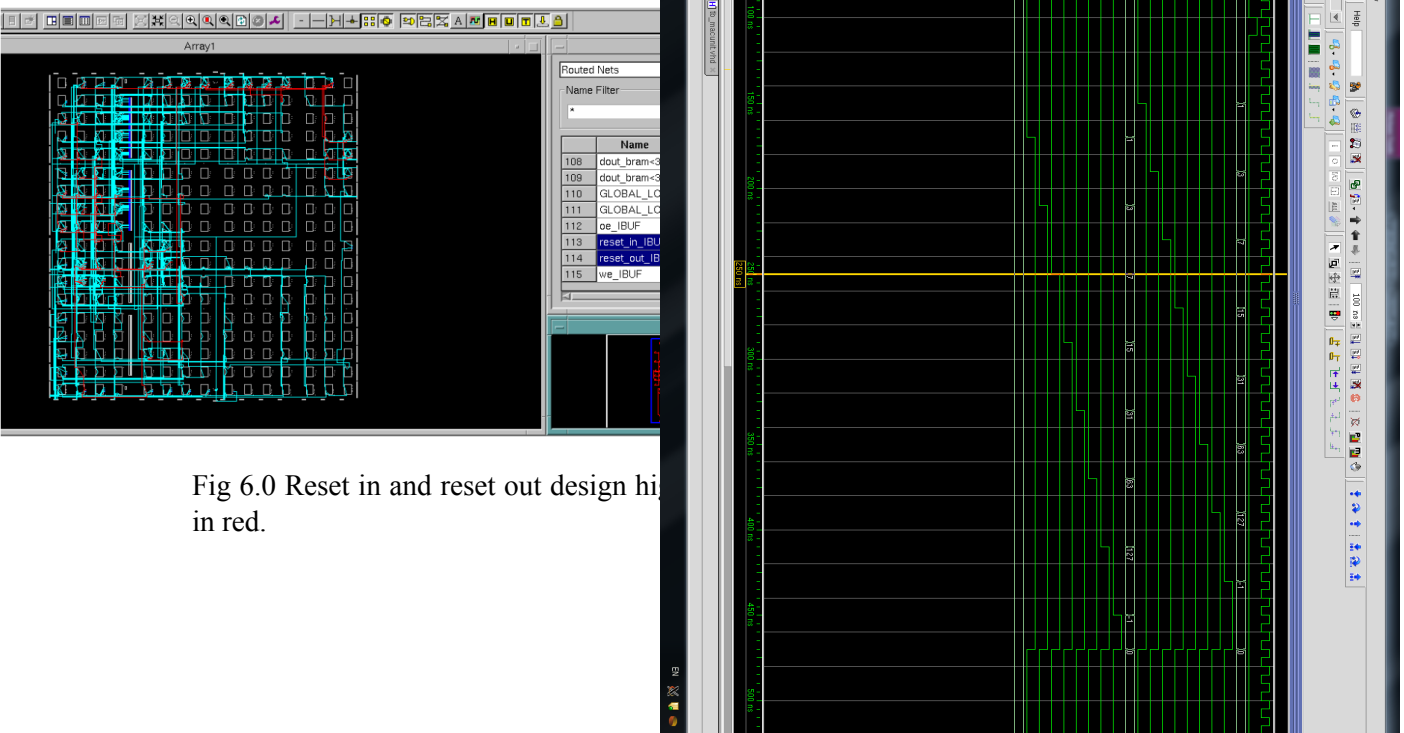


Fig 6.0 Reset in and reset out design hi in red.

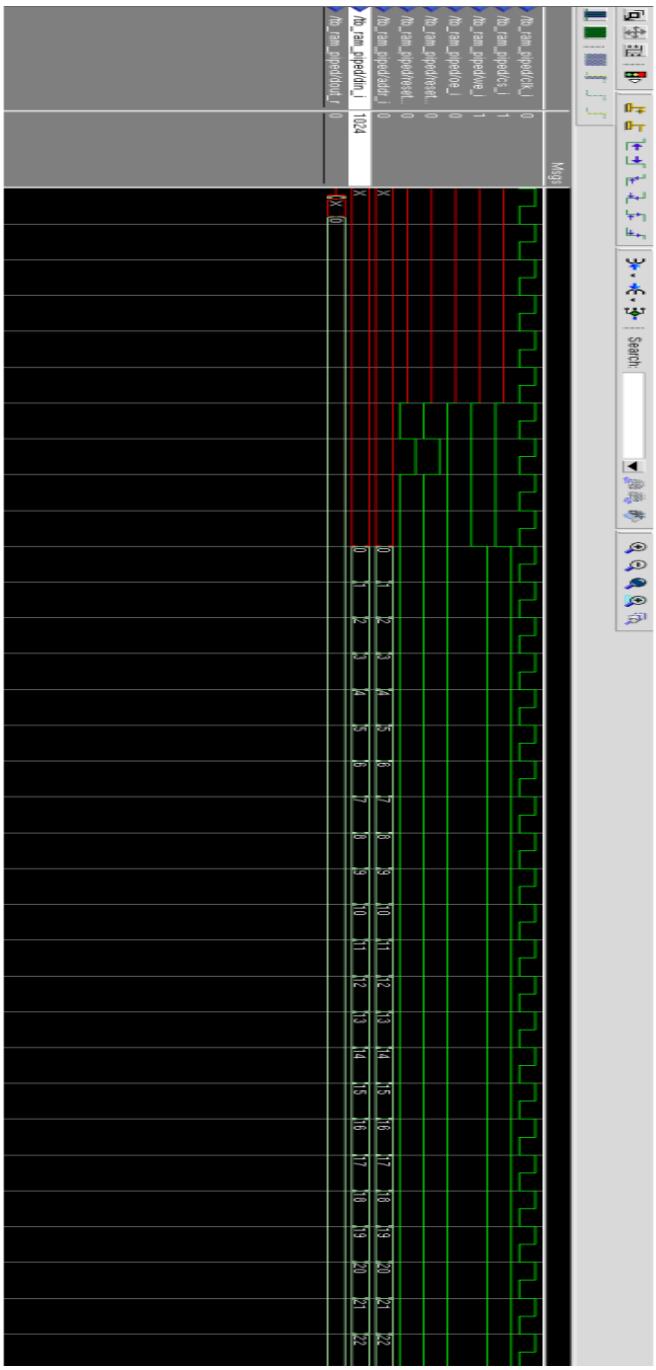


Fig 6.4.Full timing simulation of the Bram

This shows the Bram waits 129 nanoseconds before its signals starts giving an output while the clk starts immediately.

VI. CONCLUSION

The following circuit; Mac unit, Single port Block ram and Moore finite state machine have been designed simulated and validated at rtl level, they have produced significant information on themselves from taking them through the following stages and processes using Unix. The Multiply and accumulate unit is a circuit consisting of three registers(dff) one multiplier and one adder and it follows the implementation in strict accordance to the specification. The Block ram is a circuit comprising two registers with an enable and a parametric designed Bram.

The Moore finite state machine have one input and one outputs.The goal is to generate a set of outputs given a set of inputs.

Suggestions are in future works coming the rtl validation should have a more precise verification plan so less effort is required to verify a design, than writing code. Brian Kernighan, creator of the C language, stated that "Everyone knows debugging is twice as hard as writing a program in the first place." A lot of effort goes into specifying the requirements of the design. Given that verification is a larger task, more effort should go into specifying how to make sure the design is correct.

VII. REFERENCES

1. Allsyllabus.com. (2017). Multiply and Accumulate Unit. [online] Available at: http://www.allsyllabus.com/aj/note/ECE/DSP%20Algorithms%20and%20Architecture/unit2/Multiply%20and%20Accumulate%20Unit.php#.WR-_pdIrK00 .
2. Perry, D. (2017). [online] Available at: http://vhdl-manual.narod.ru/books/programming_by_example.pdf [Accessed 18 May 2017].
3. Anon, (2017). [online] Available at: https://www.ijircce.com/upload/2015/april/17_FPGA.pdf [Accessed 20 May 2017].

4. Anon, (2017). [online] Available at: https://www.xilinx.com/support/documentation/ip_documentation/sp_block_mem.pdf

5. Xilinx.com. (2017). Block Ram (BRAM) Block. [online] Available at: https://www.xilinx.com/products/intellectual-property/block_ram.html [Accessed 18 May 2017].

6. Generator (2017). [Full VHDL code] Matrix Multiplication Design using VHDL and Xilinx Core Generator. [online] Fpga4student.com. Available at: http://www.fpga4student.com/2016/11/matrix-multiplier-core-design.html

7. Anon, (2017). [online] Available at: https://learnabout-electronics.org/Digital/dig53.php

8. Williams, D., Teegarden, D., Munira, A. and Dicks, J. (2017). Implementing a Finite State Machine in VHDL. [online] Allaboutcircuits.com. Available at: https://www.allaboutcircuits.com/technical-articles/implementing-a-finite-state-machine-in-vhdl/ [Accessed 17 May 2017].

9. Anon, (2017). [online] Available at: http://academic.csuohio.edu/chu_p/rtl/chu_rtL_book/rtl_chap10_fsm.pdf