32- BIT MIPS SINGLE CYCLE PROCESSOR USING FPGA SPARTAN-6

(HIGH LEVEL SYNTHESIS- ESD 602)

BY

V.V.KUSUMITHA(MT2017527)

PARITOSH SAHAY(MT2017510)

A) Design Modules :

- **Single_cycle (mips.v)**

Top Module of the design connecting each of the functionality of the processor

- **Adder_PC (adder_PC.v)**

For every instruction, the PC should increment by 4 as MIPS is a byte addressable processor.

- **Adder_Branch(adder_branch.v)**

For Implementing Branch instruction ,the Immediate address is added to the PC ,resulting in the branch target address.

- **ALU (alu.v)**

ALU takes a set of data inputs each of 32 bits width and an ALU control signal of 4 bits, resulting in the implementation of instructions addition ,subtraction, Multiplication, Division, logical OR, logical NOR, Set less than (slt), Load,store , increment, decrement.

- **ALU Control Unit (alu_ctrl.v)**

For types of instruction formats i.e R-type, I-type & J-type , ALUop control signal results in the control signal of ALU .

- **Jump (Calc_Jump.v)**

If the instruction type is J i.e. Jump, the corresponding PC will be taken in PC resulting in the Jump target location. Jump has 26 bit immediate address which will be extended to 32 bits.

- **Control Unit (cu.v)**

For the datapath modules to function according to the instruction, corresponding control signals should be asserted, which will be controlled by the **control unit.** It includes control signals :

a)RegDst

b) Jump

c) Branch

d) MemRead

e) MemtoReg

f) MemWrite

g) ALUSrc

h) RegWrite

i)ALUControl[3:0]

- **Data Memory (data_mem.v)**

Data Memory is accessed for load & store instructions to take the data from the particular instruction memory address. Memory is initialized with some set of values.

LOAD : This instruction takes the data from memory and writes to the register, with *MemRead* control signal enabled.

STORE : Instruction writes the data to the memory with *MemWrite* control signal enabled.

- **Instruction Memory (inst_mem.v)**

At every positive edge of clock signal, PC is incremented and corresponding instructions are being read from the Instruction Memory address.

Depending on the instruction type, 32 bits are assigned to the output.

- **Register File(reg_file.v)**

It is a module which has 2 Read ports and 1 write port, with control signal , *RegWrite* which lets you read or write to the register file.

- There are 32 32-bit registers in a single cycle MIPS.

- **Sign Extension (sign_ext.v)**

For Load or store instruction , the immediate address is 16-bit which should be sign extended to 32-bit. This functionality takes the MSB bit and copies it to 16 bits.

- **Multiplexers (_5b_MUX.v , _32b_MUX.v)**
  (Mux1,Mux2,Mux3,Mux4 & Mux5)

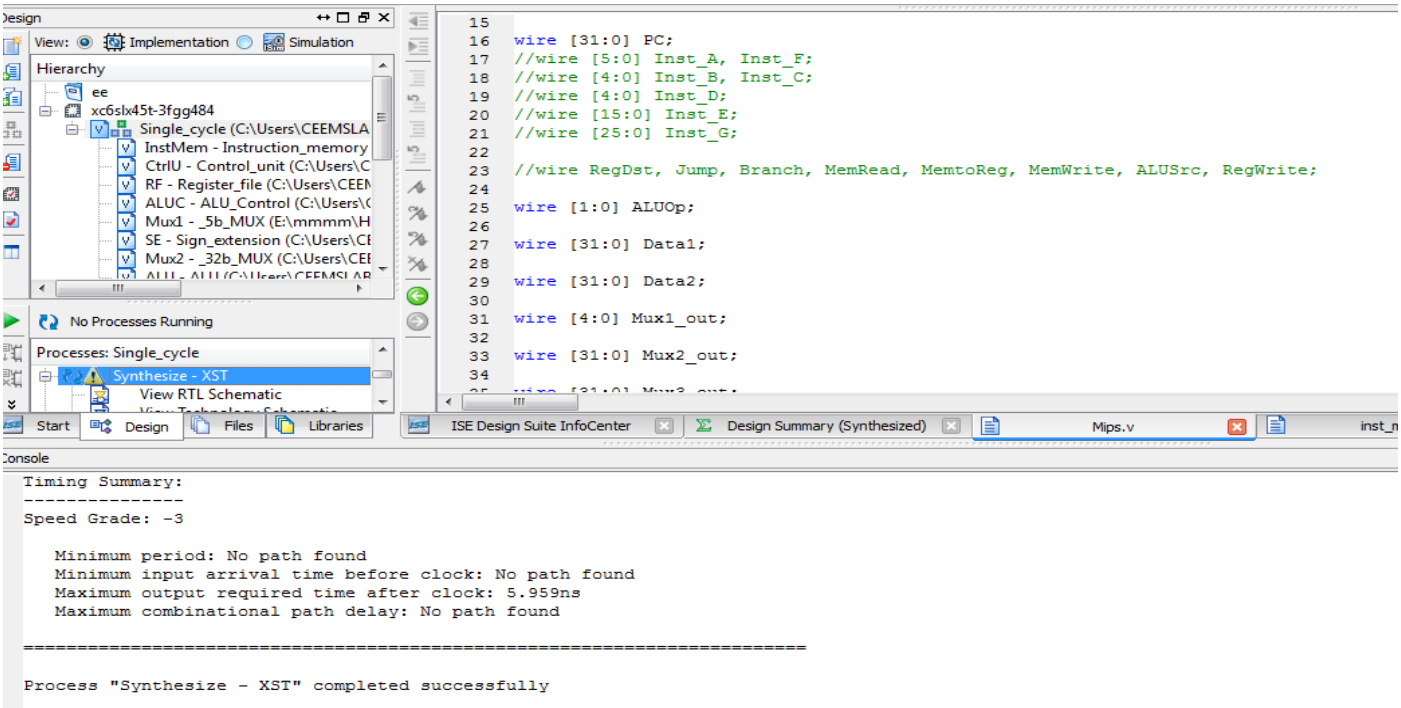
There are 5-bit & 32-bit multiplexers required at the design modules for implementing different types of instructions.

B) The following are the steps for synthesizing our MIPS design :

*Step 1:*Make the design module hierarchy with sub modules under the top module.

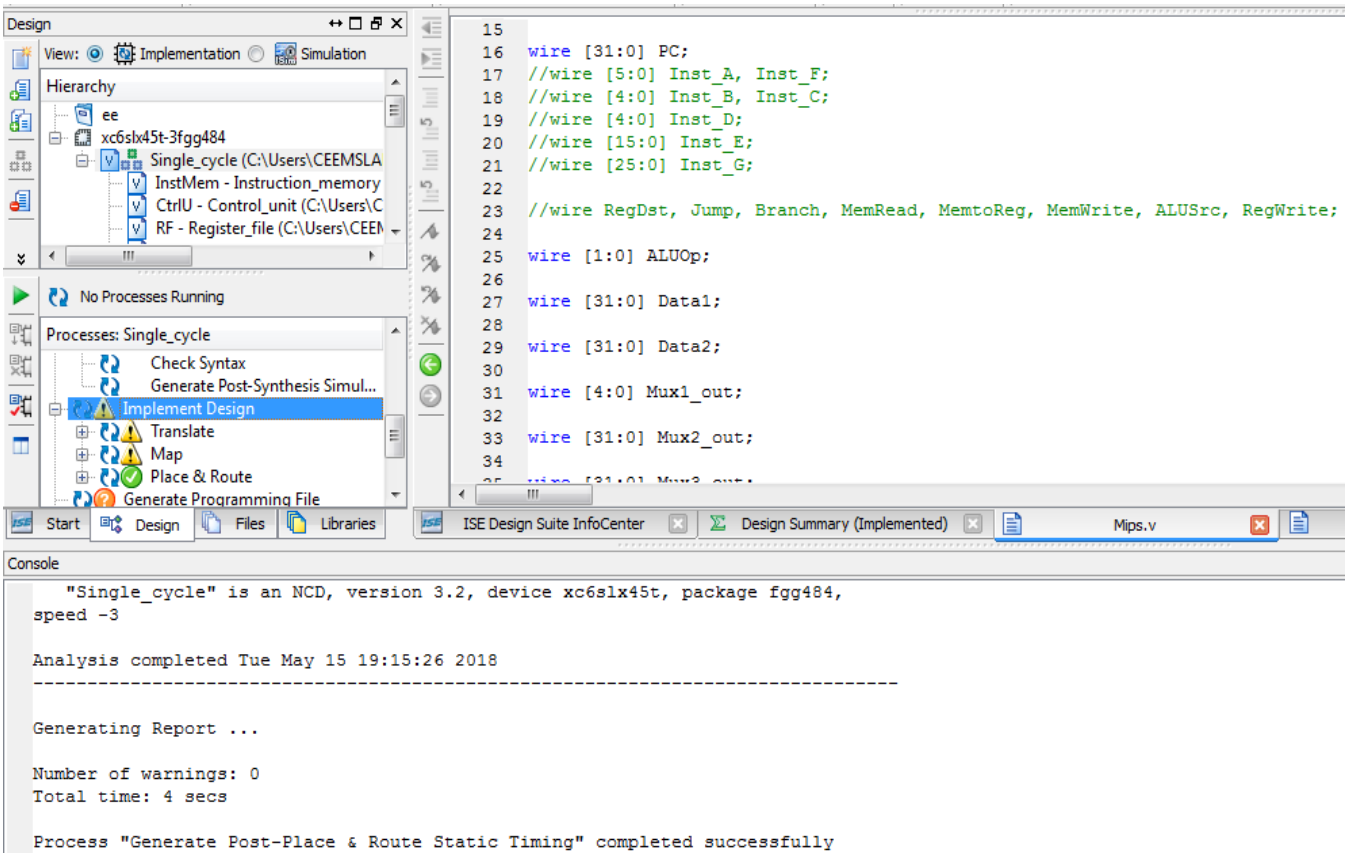
*Step 2:*Synthesize

Includes RTL Schematic



*Step 3:*Implement the design

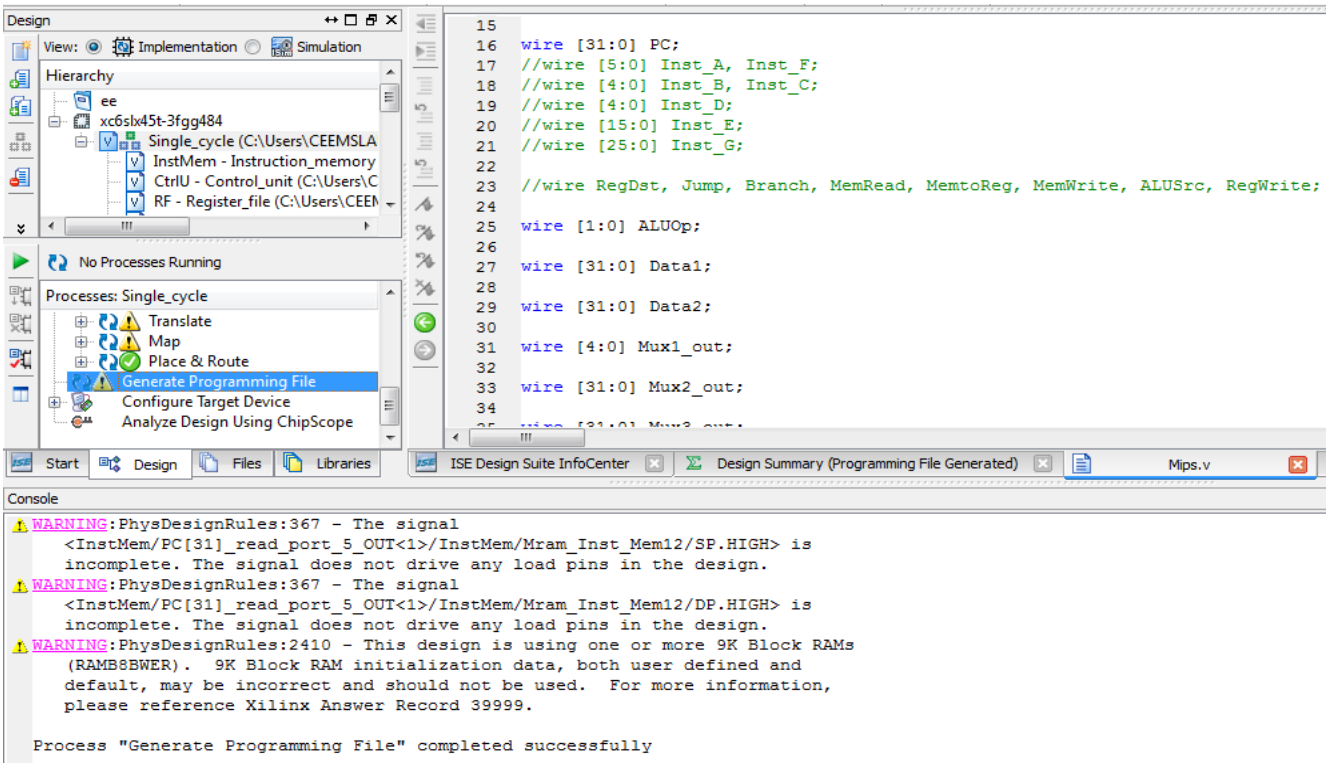
Includes Translate , Map & Place and Route.



Step 4: Add the User Constraints File(ee_cs.ucf) which includes the location of the 4 LEDs in the Spartan-6 FPGA Board and the clock signal(**100MHz**) location in the FPGA.

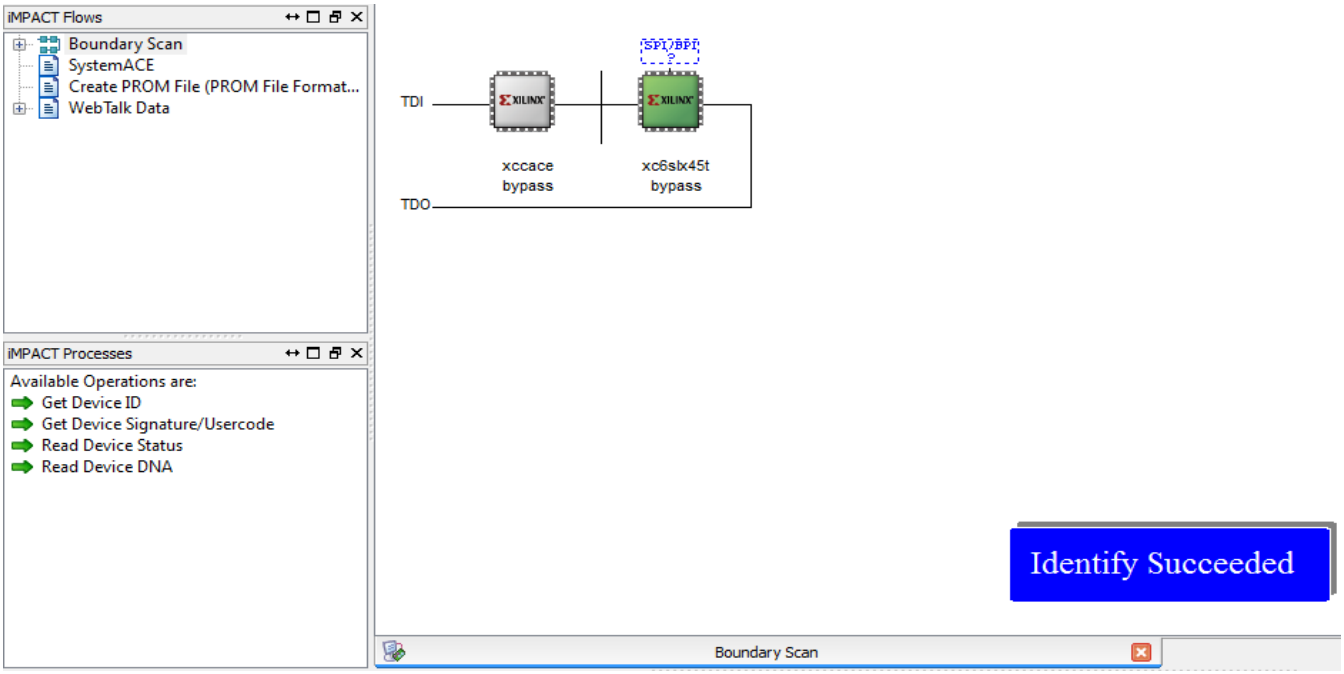
Step 5: Generate Programming File

(Make sure that FPGA has power cable and JTAG cable connected).



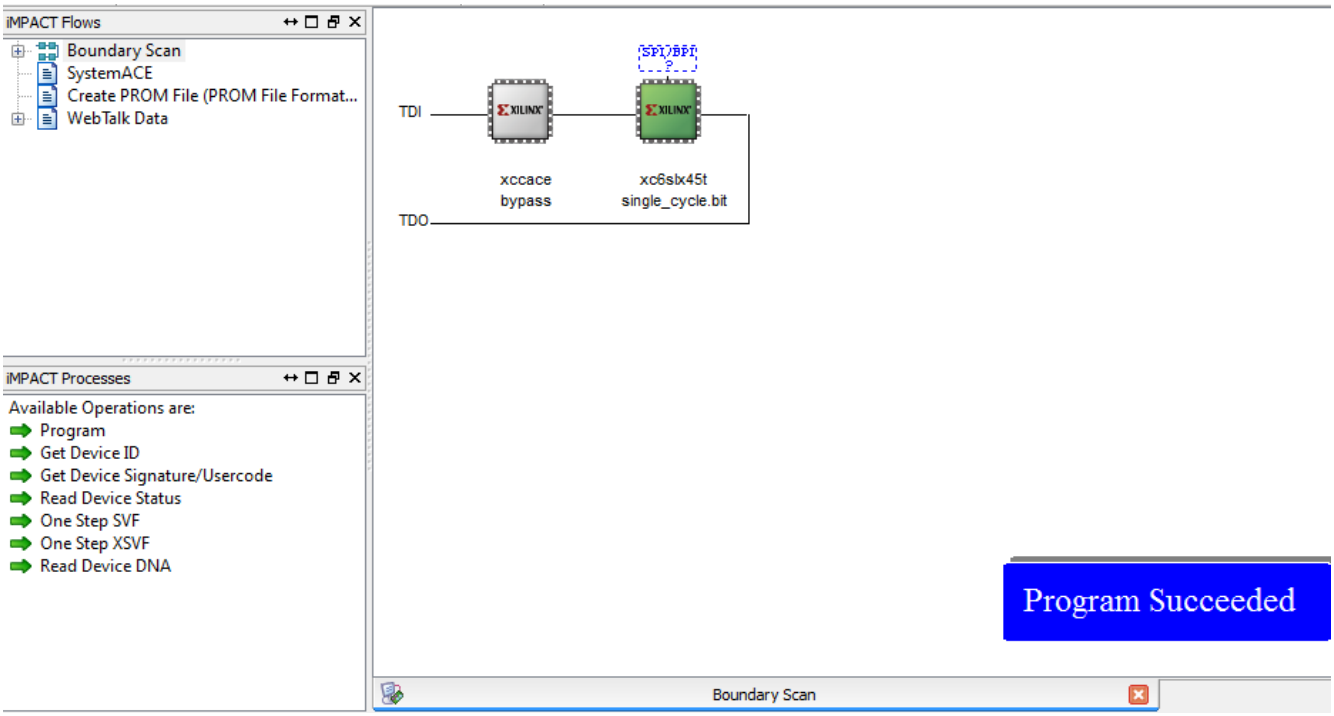
*Step 6:*Configure Target Device

IMPACT window , lets you select the processor among the 2 processors available in it.

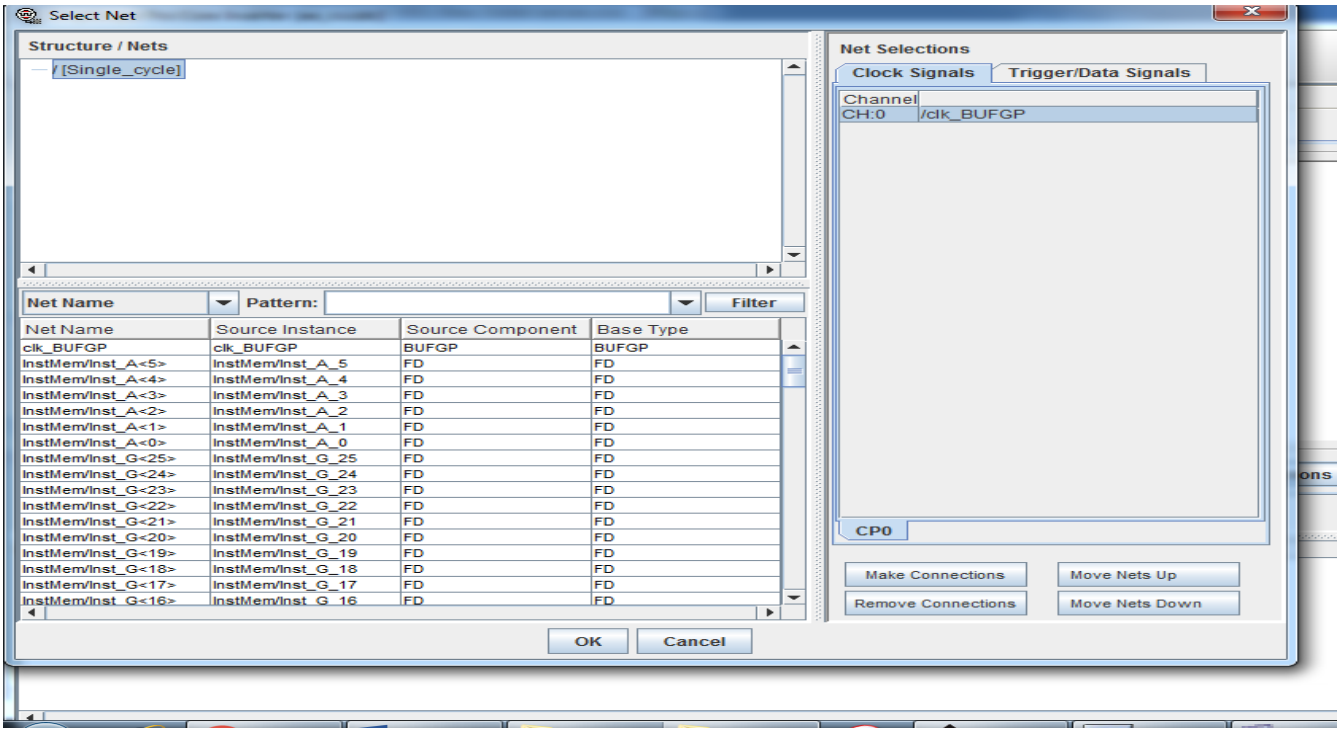


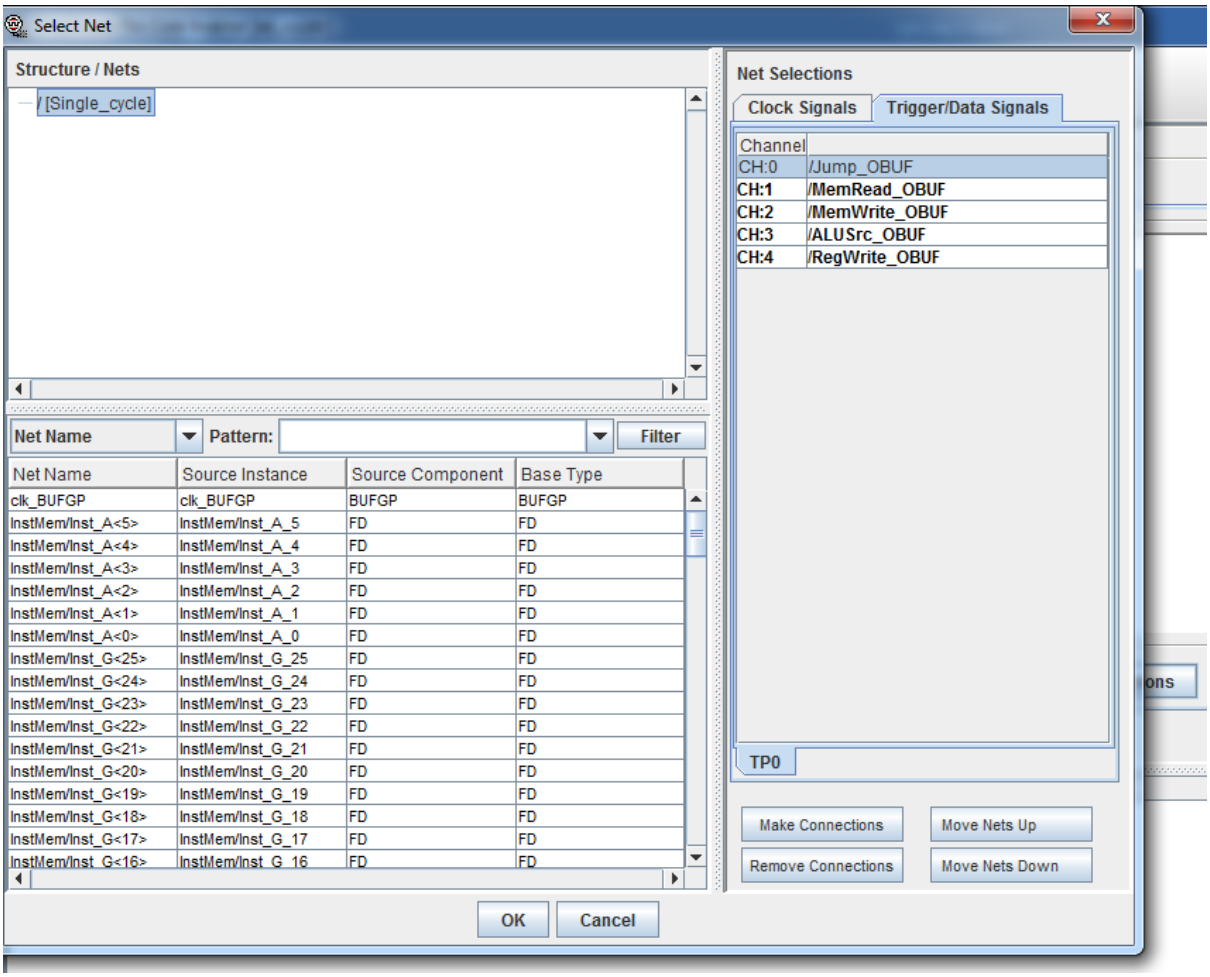
a) Assign New Configuration file- .bit file is added

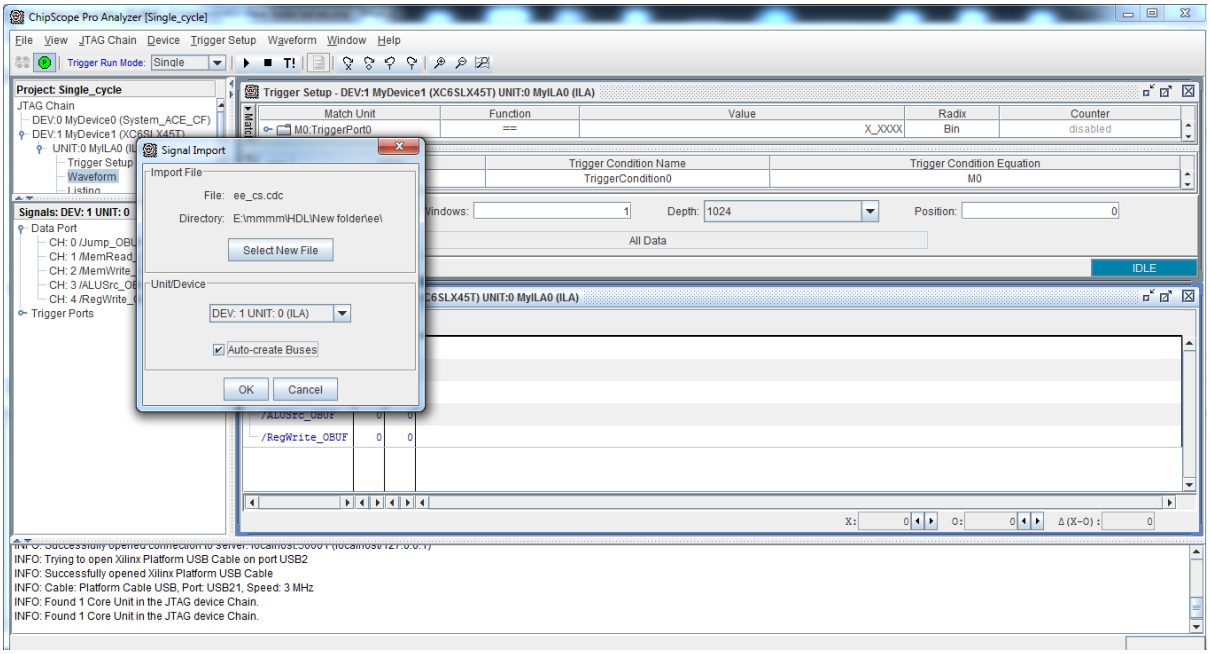
b) Program – Output will be displayed on FPGA.



Step 7: Add .cdc file (Chipscope and Definition file), inserting the clock & trigger ports which we wanted to display in the chipscope.


Select Net
Structure / Nets
/ [Single_cycle]
Net Selections
Clock Signals
Trigger/Data Signals
Channel
CH:0
/clk_BUFGP
CP0
Net Name
Pattern:
Filter
Net Name
Source Instance
Source Component
Base Type
clk_BUFGP
clk_BUFGP
BUFGP
BUFGP
InstMem/Inst_A<5>
InstMem/Inst_A_5
FD
FD
InstMem/Inst_A<4>
InstMem/Inst_A_4
FD
FD
InstMem/Inst_A<3>
InstMem/Inst_A_3
FD
FD
InstMem/Inst_A<2>
InstMem/Inst_A_2
FD
FD
InstMem/Inst_A<1>
InstMem/Inst_A_1
FD
FD
InstMem/Inst_A<0>
InstMem/Inst_A_0
FD
FD
InstMem/Inst_G<25>
InstMem/Inst_G_25
FD
FD
InstMem/Inst_G<24>
InstMem/Inst_G_24
FD
FD
InstMem/Inst_G<23>
InstMem/Inst_G_23
FD
FD
InstMem/Inst_G<22>
InstMem/Inst_G_22
FD
FD
InstMem/Inst_G<21>
InstMem/Inst_G_21
FD
FD
InstMem/Inst_G<20>
InstMem/Inst_G_20
FD
FD
InstMem/Inst_G<19>
InstMem/Inst_G_19
FD
FD
InstMem/Inst_G<18>
InstMem/Inst_G_18
FD
FD
InstMem/Inst_G<17>
InstMem/Inst_G_17
FD
FD
InstMem/Inst_G<16>
InstMem/Inst_G_16
FD
FD
Make Connections
Move Nets Up
Remove Connections
Move Nets Down
OK
Cancel

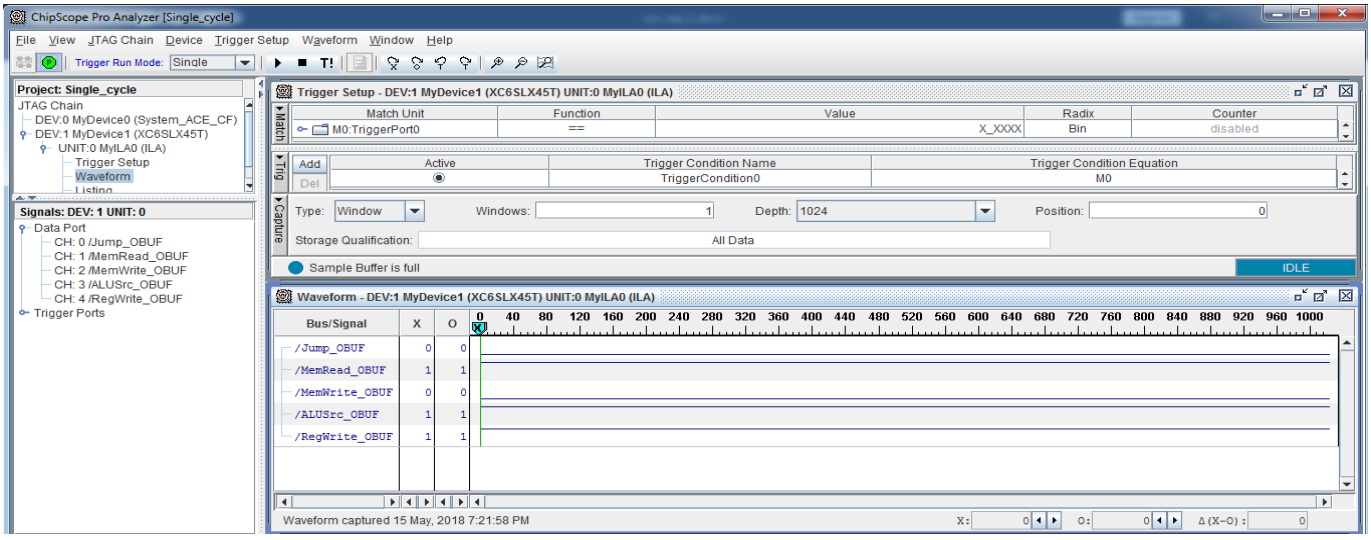

Select Net
Structure / Nets
/ [Single_cycle]
Net Selections
Clock Signals
Trigger/Data Signals
Channel
CH:0
/Jump_OBUF
CH:1
/MemRead_OBUF
CH:2
/MemWrite_OBUF
CH:3
/ALUSrc_OBUF
CH:4
/RegWrite_OBUF
TP0
Net Name
Pattern:
Filter
Net Name
Source Instance
Source Component
Base Type
clk_BUFGP
clk_BUFGP
BUFGP
BUFGP
InstMem/Inst_A<5>
InstMem/Inst_A_5
FD
FD
InstMem/Inst_A<4>
InstMem/Inst_A_4
FD
FD
InstMem/Inst_A<3>
InstMem/Inst_A_3
FD
FD
InstMem/Inst_A<2>
InstMem/Inst_A_2
FD
FD
InstMem/Inst_A<1>
InstMem/Inst_A_1
FD
FD
InstMem/Inst_A<0>
InstMem/Inst_A_0
FD
FD
InstMem/Inst_G<25>
InstMem/Inst_G_25
FD
FD
InstMem/Inst_G<24>
InstMem/Inst_G_24
FD
FD
InstMem/Inst_G<23>
InstMem/Inst_G_23
FD
FD
InstMem/Inst_G<22>
InstMem/Inst_G_22
FD
FD
InstMem/Inst_G<21>
InstMem/Inst_G_21
FD
FD
InstMem/Inst_G<20>
InstMem/Inst_G_20
FD
FD
InstMem/Inst_G<19>
InstMem/Inst_G_19
FD
FD
InstMem/Inst_G<18>
InstMem/Inst_G_18
FD
FD
InstMem/Inst_G<17>
InstMem/Inst_G_17
FD
FD
InstMem/Inst_G<16>
InstMem/Inst_G_16
FD
FD
Make Connections
Move Nets Up
Remove Connections
Move Nets Down
OK
Cancel



*Step 8:*Do the steps, Implement Design again & Analyze the Chipscope configuration device.

a) Import the .cdc file

b) Trigger Interrupt displays waveforms corresponding to your design.



C) In our Design, there is no requirement to changes in the design modules as we were able to get the expected RTL schematic with every module connected in proper. Design is Synthesizable.

D) SYNTHESIS REPORT OF OUR DESIGN

Macro Statistics

 # RAMs : 4

101x8-bit dual-port Read Only RAM : 2

32x32-bit dual-port RAM : 2

 # Multipliers : 1

32x32-bit multiplier : 1

 # Adders/Subtractors : 10

32-bit adder : 9

32-bit subtractor : 1

 # Registers : 208

16-bit register : 1

26-bit register : 1

32-bit register : 1

5-bit register : 3

6-bit register : 2

8-bit register : 200

 # Latches : 69

1-bit latch : 69

 # Comparators : 1

32-bit comparator greater : 1

 # Multiplexers : 25

1-bit 2-to-1 multiplexer : 13

32-bit 2-to-1 multiplexer : 4

4-bit 2-to-1 multiplexer : 3

5-bit 2-to-1 multiplexer : 1

8-bit 200-to-1 multiplexer : 4

 # Tristates : 800

8-bit tristate buffer : 800

E) As mentioned above, the design hierarchy is as shown in the figure:

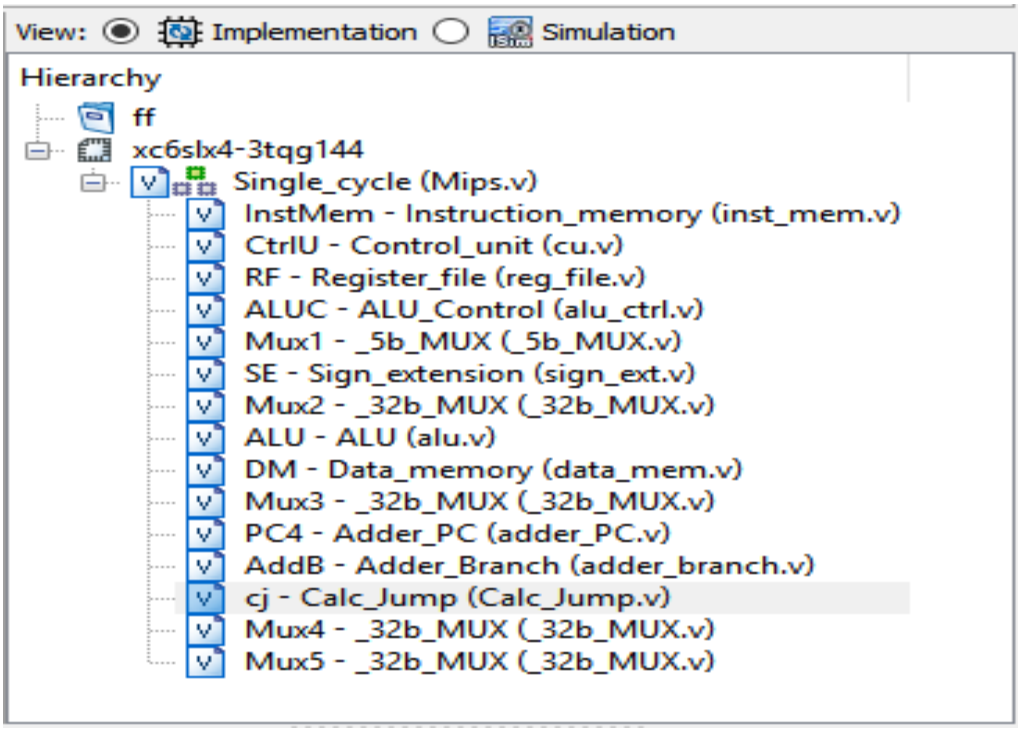
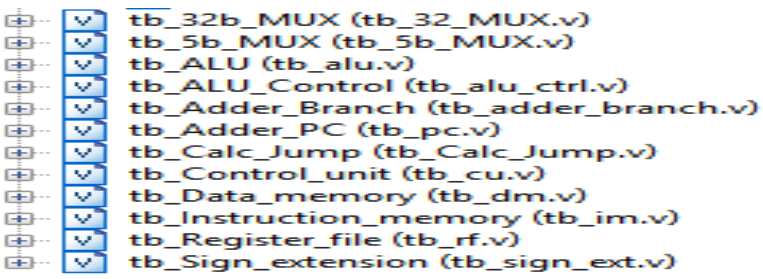


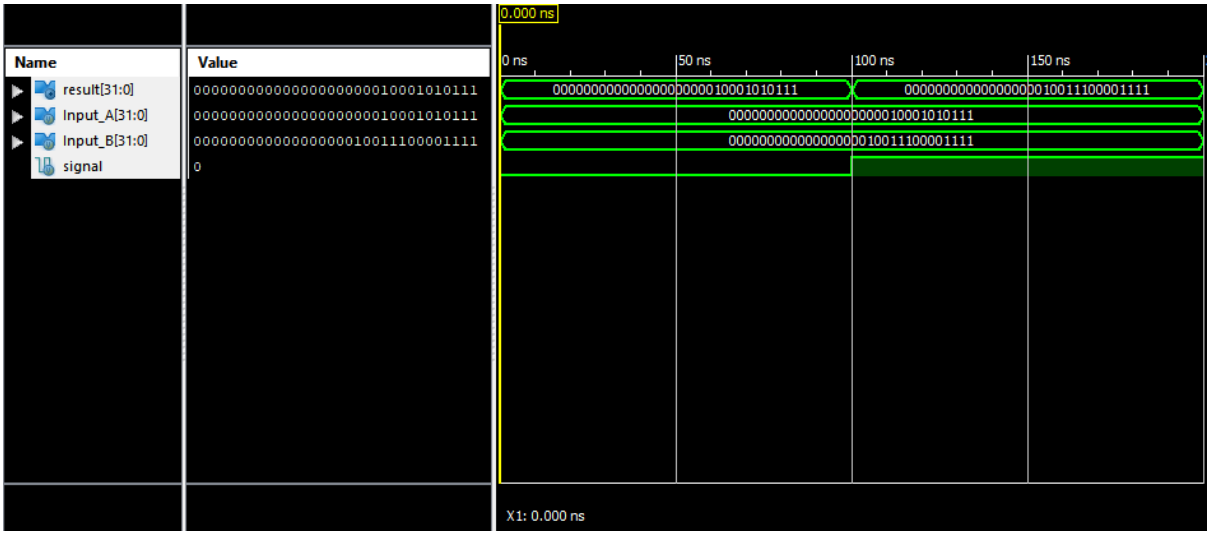
Fig: Design Hierarchy

Following are the testbenches for the corresponding design modules:

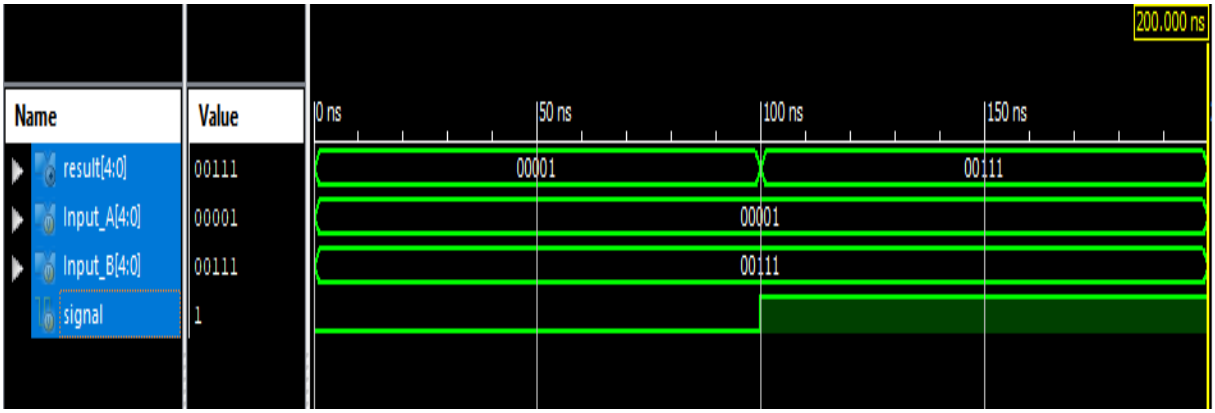


Simulation Testbench Results:

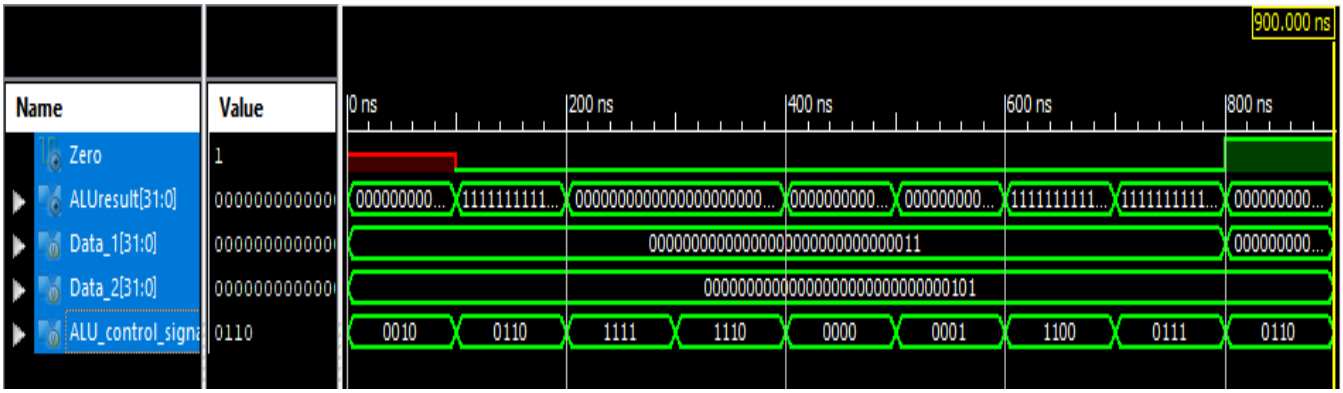
- **Multiplexer – 32 bit**



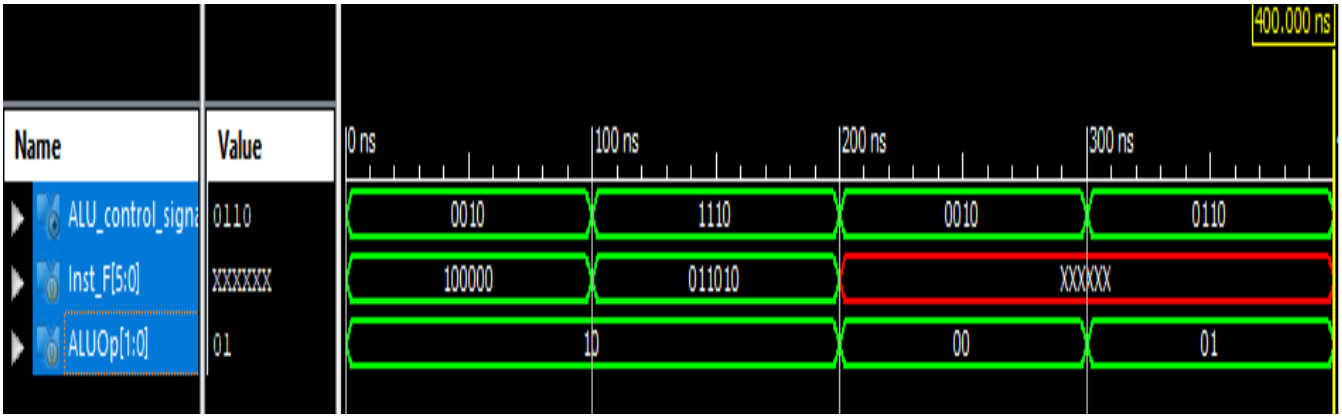
- **5bit Multiplexer**



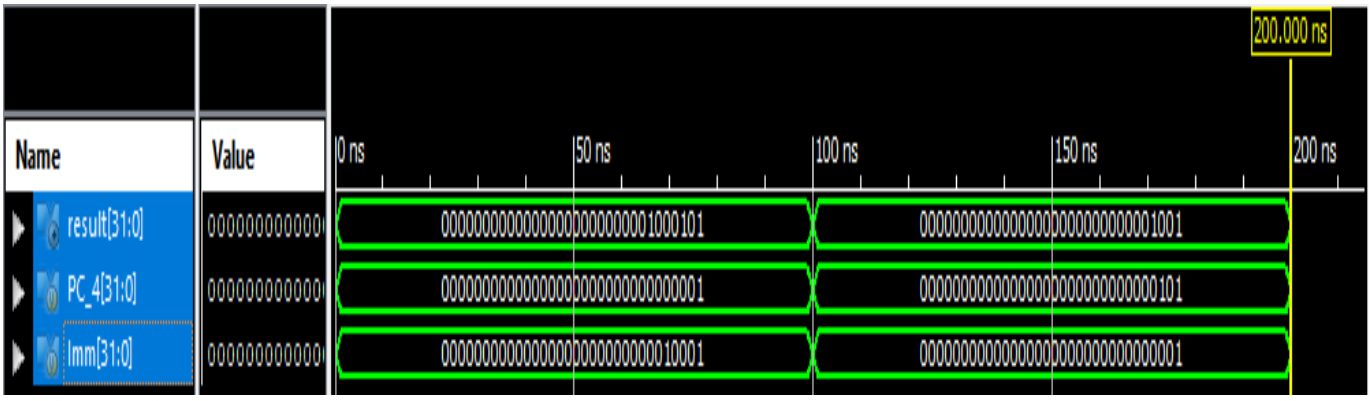
- **32-bit ALU**



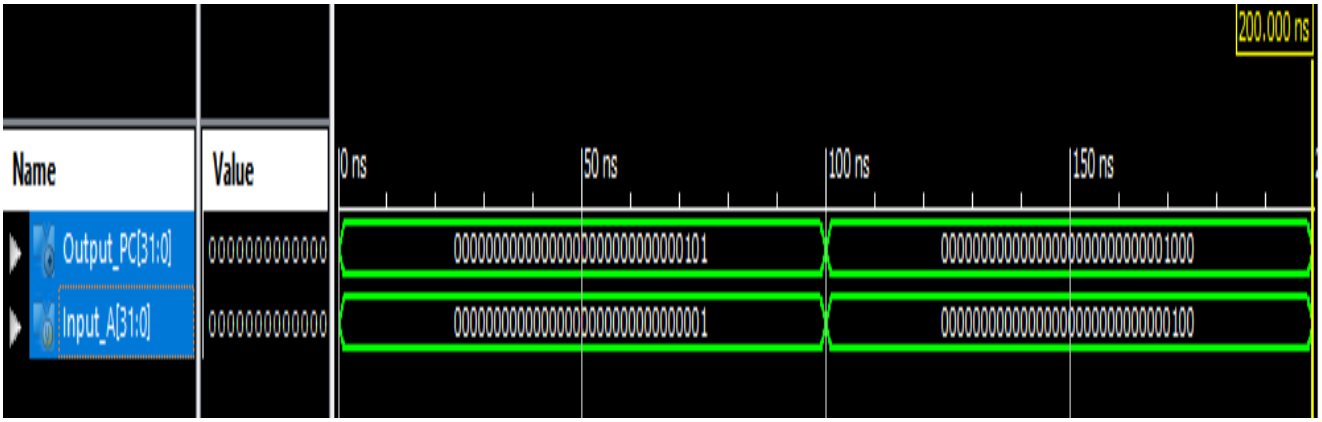
- **ALU Control Unit**



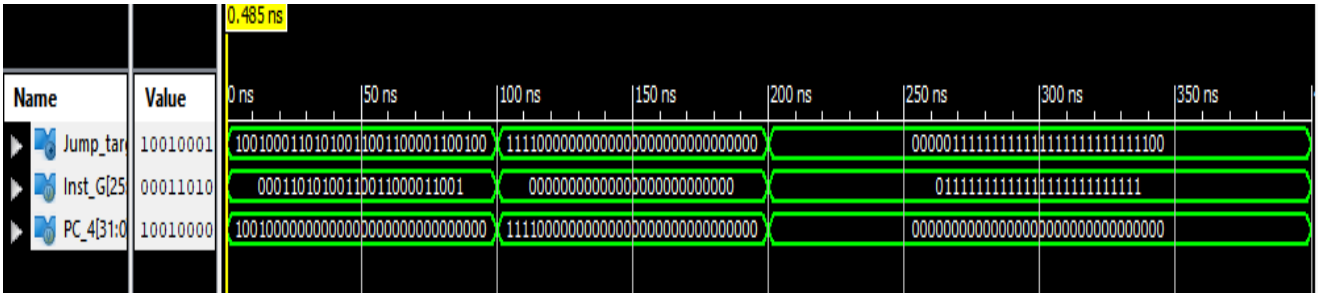
- **Adder Branch to compute Branch Instruction Address**



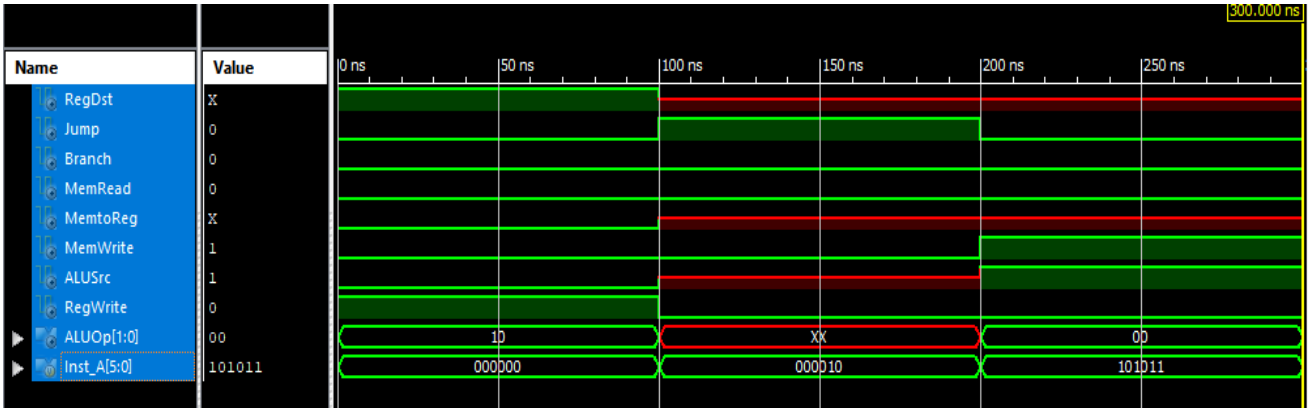
- **PC**



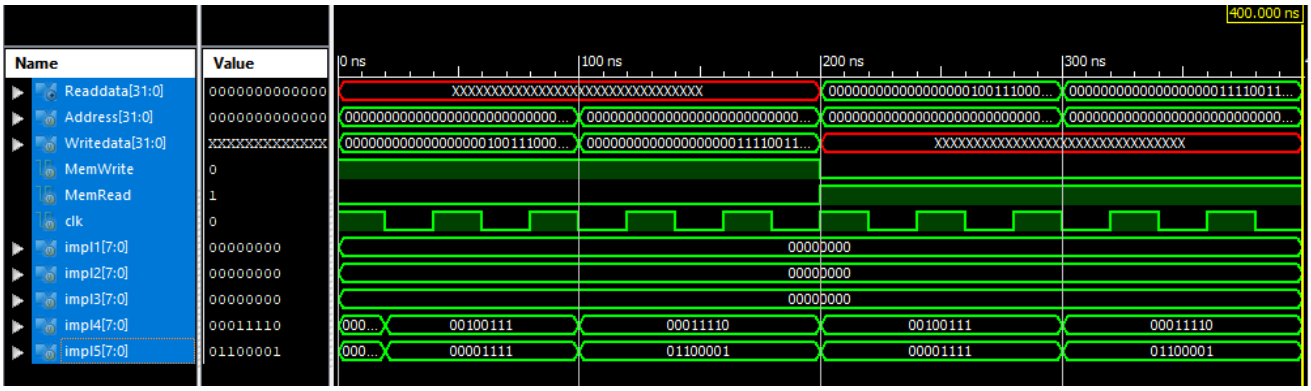
Jump



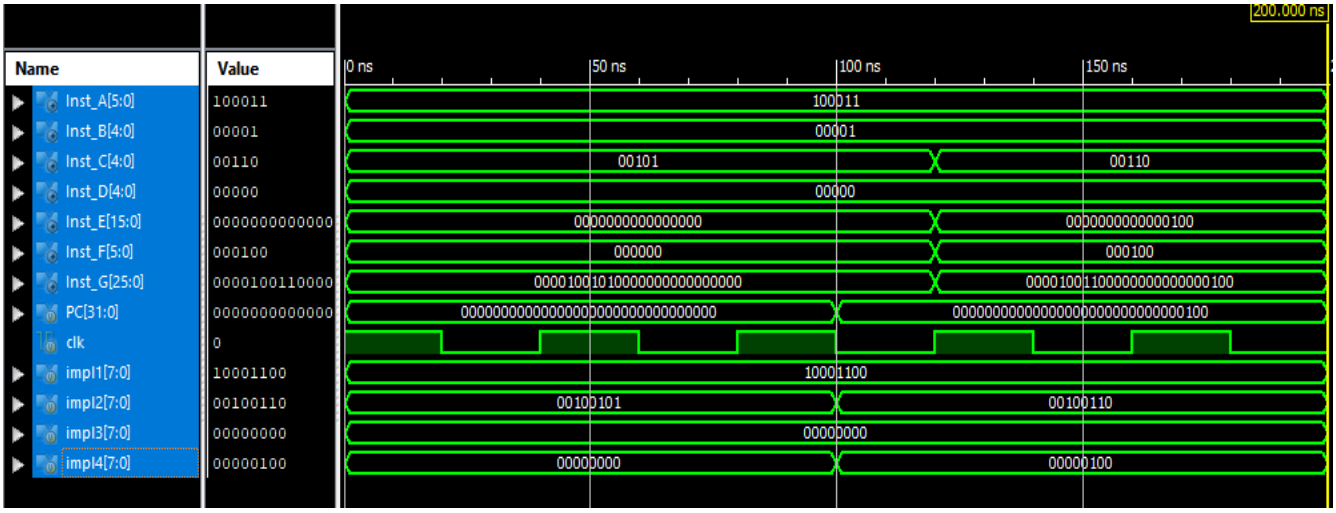
- ## Control Unit



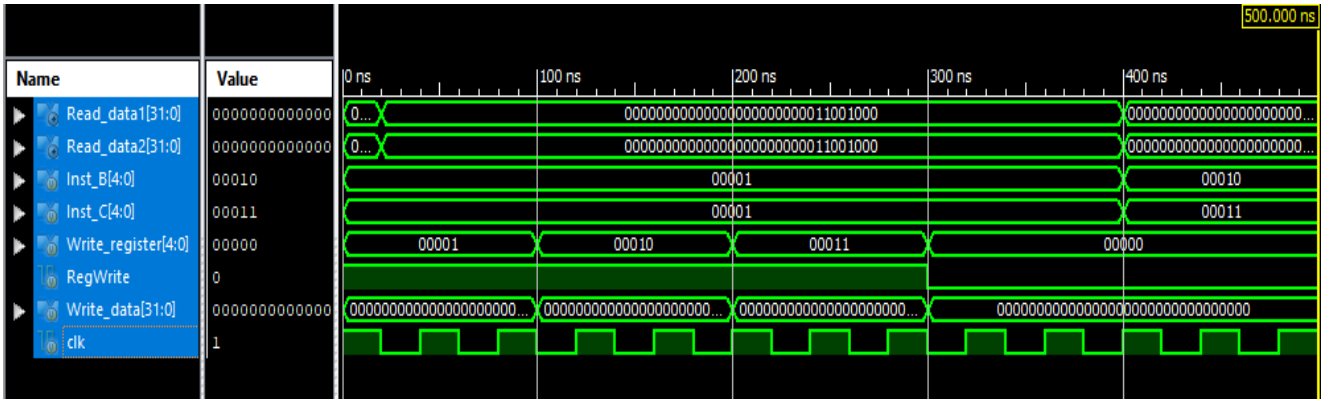
- ## Data Memory



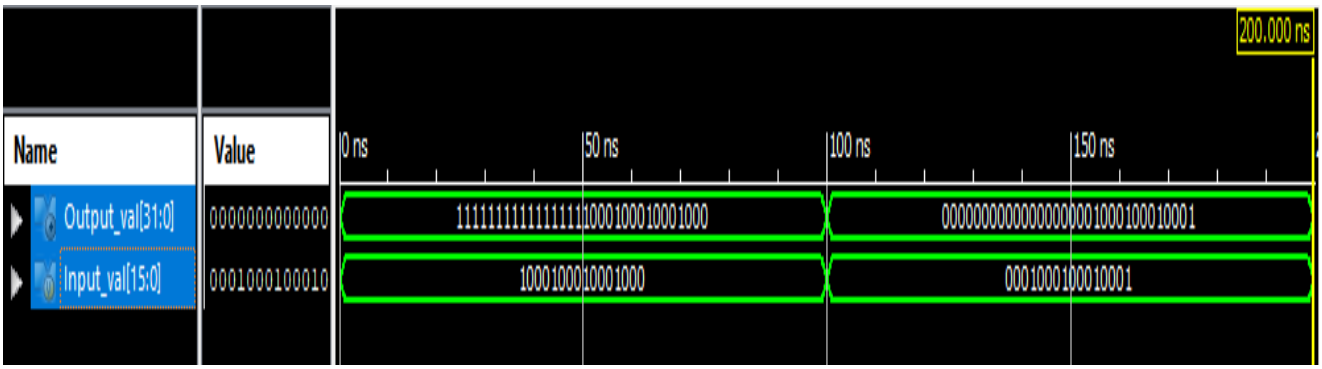
- **Instruction Memory**



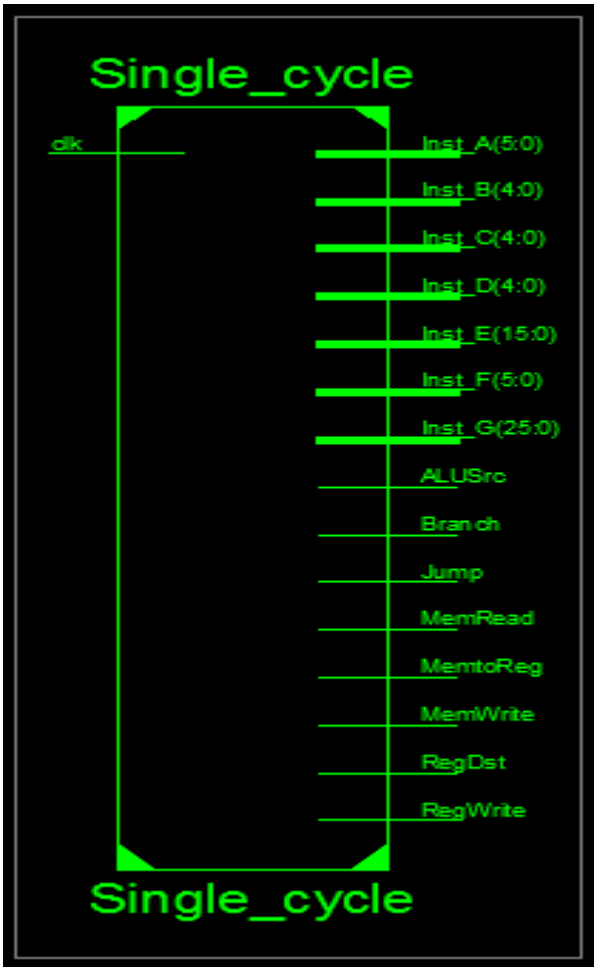
- **Register File**

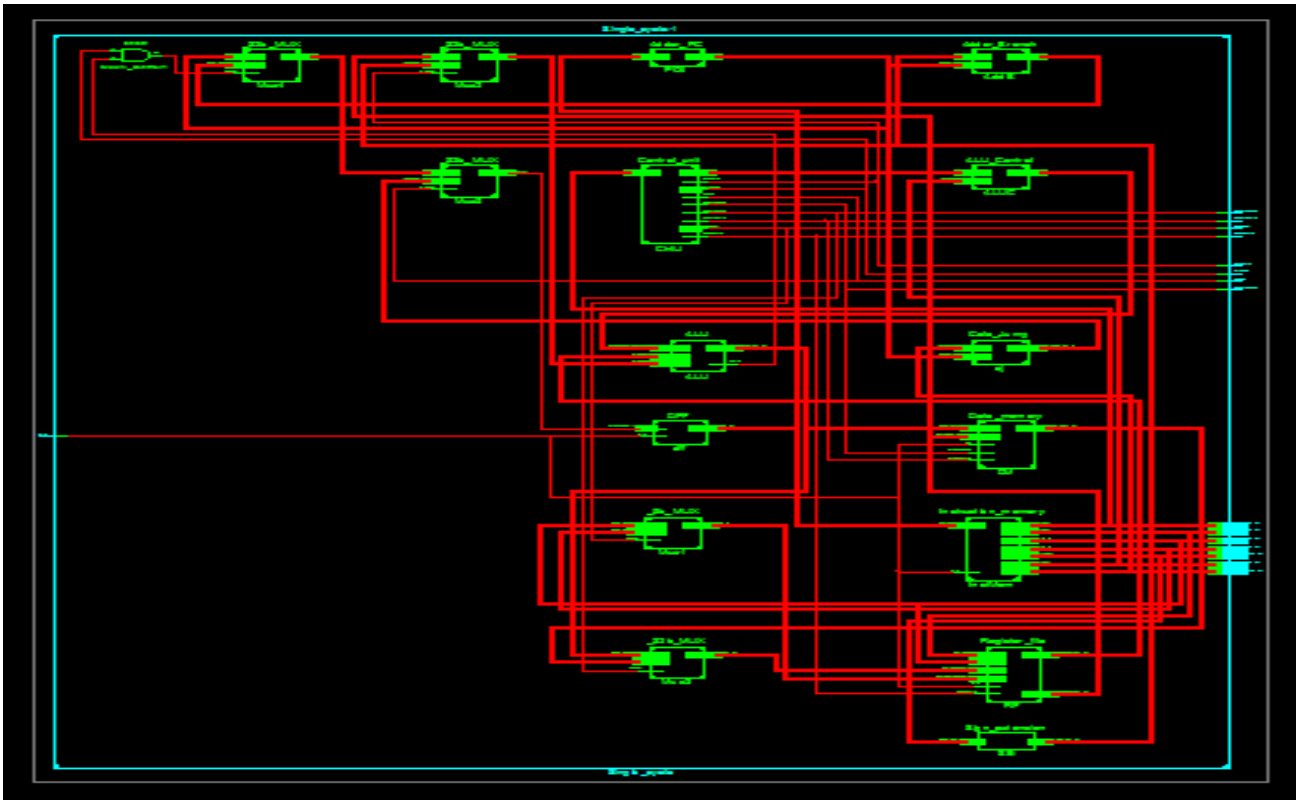


- **Sign Extension**

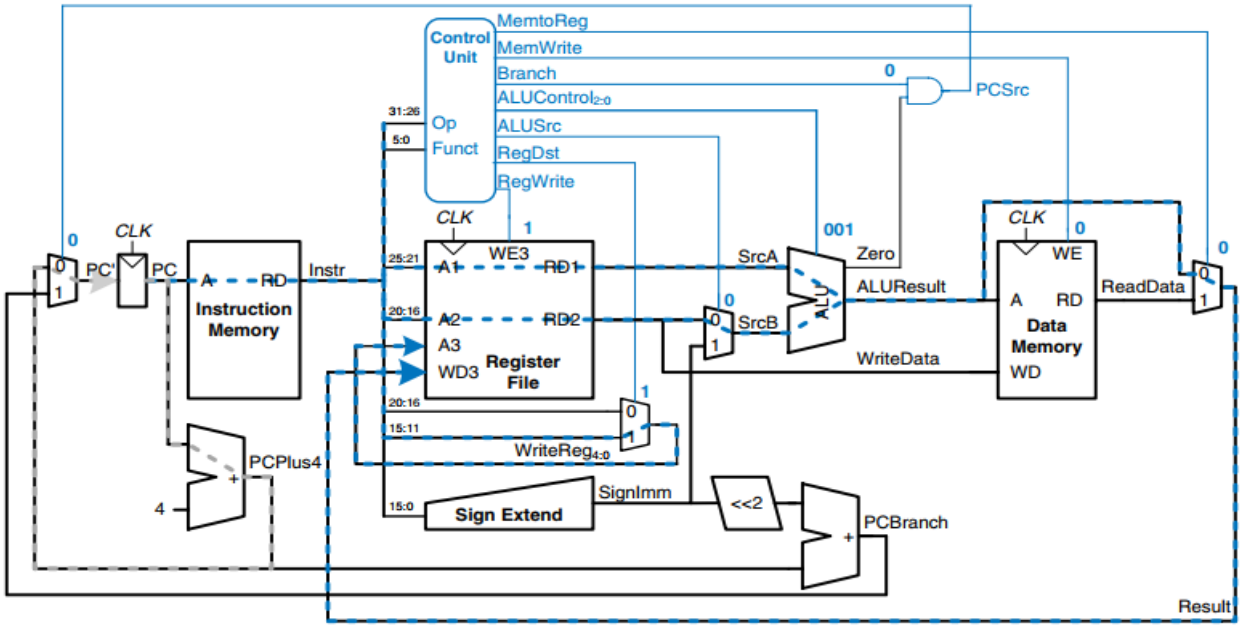


RTL Schematic of Single cycle MIPS :





Block Diagram of Single cycle MIPS



F) The output from FPGA are 4 LEDs DS3, DS4, DS5 &DS6 which we have given to the control signals of the design. For the types of instructions, the outputs are enabled correctly.

So, the design is verified with external stimuli given to FPGA Spartan-6.

G)

1) Top module should not have an empty sensitivity list, as this will result an error in the RTL schematic of the design.

2)Input & Output ports should be declared individually as, declaring multiple inputs/ outputs at the same time effects the readability of the code.

H) Conclusion:

32-bit Single Cycle MIPS processor with 20 Instructions set (add,sub,div,mult,or,not,slt,load,store,and,andi, ori,addi,subi,jump,branch) are being implemented & verified individually with testbench using Xilinx.

RTL Schematic of the Single cycle MIPS is obtained.

Results are obtained & verified through FPGA Spartan-6 & visualized through Chipscope Analyzer.`

Fig: Simulation results in FPGA Spartan-6