MINI PROJECT:

IMPLEMENTATION OF COMPLETE PROCESSOR

Student Name :
Jeevan Sammeswar

Guide:
Dr. Rajesh Kedia

Details about project :

- The project was aimed to design a 32 bit processor
- The architecture used in the processor is MIPS
- The processor is designed using VHSIC Hardware Description Language and Xilinx-Vivado software .
- It contains 2KB size instruction block which stores instructions , 10KB size memory to store data.
- It has 32-bit 32 registers
- The ALU performs operations on signed integers .
- Its has Dot product unit which can perform dot product on two 4D vectors after loading all 8 elements to register block of dot product unit and simultaneously it performs 4 multiplications and store the result in memory block.

Instructions and their clock cycles

The instruction and memory blocks have 1 clock cycle delay ,so after every operations the control unit wait for 1 clock cycle .

Operation	Cycles
And	2
Or	2
Xor	2
Addi	2
Add	2
Sub	2
Mult	33
Div	33
Set less than	2
Shift left logical	2
Shift right logical	2
Load word	2
Store word	2
Branch on equal	2
Jump to address	2

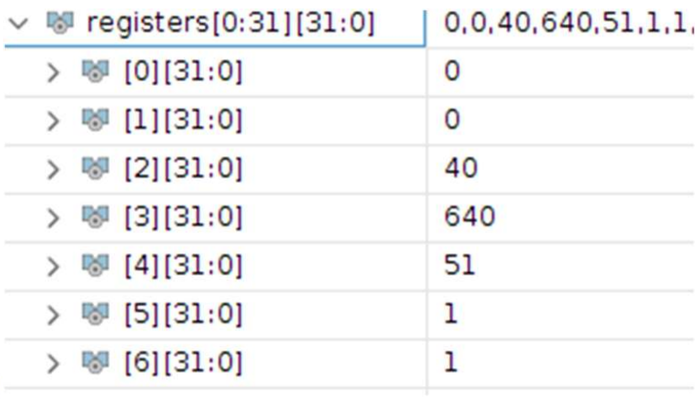
Dlw	2
Dsw	33

Example :

Algorithm

```
int x=40 , sum=40 ,result;
if(x>50)
{
        sum=sum+40;
}
else
{  sum= sum<<4 ;  }
result= sum ;
```

Register File contents

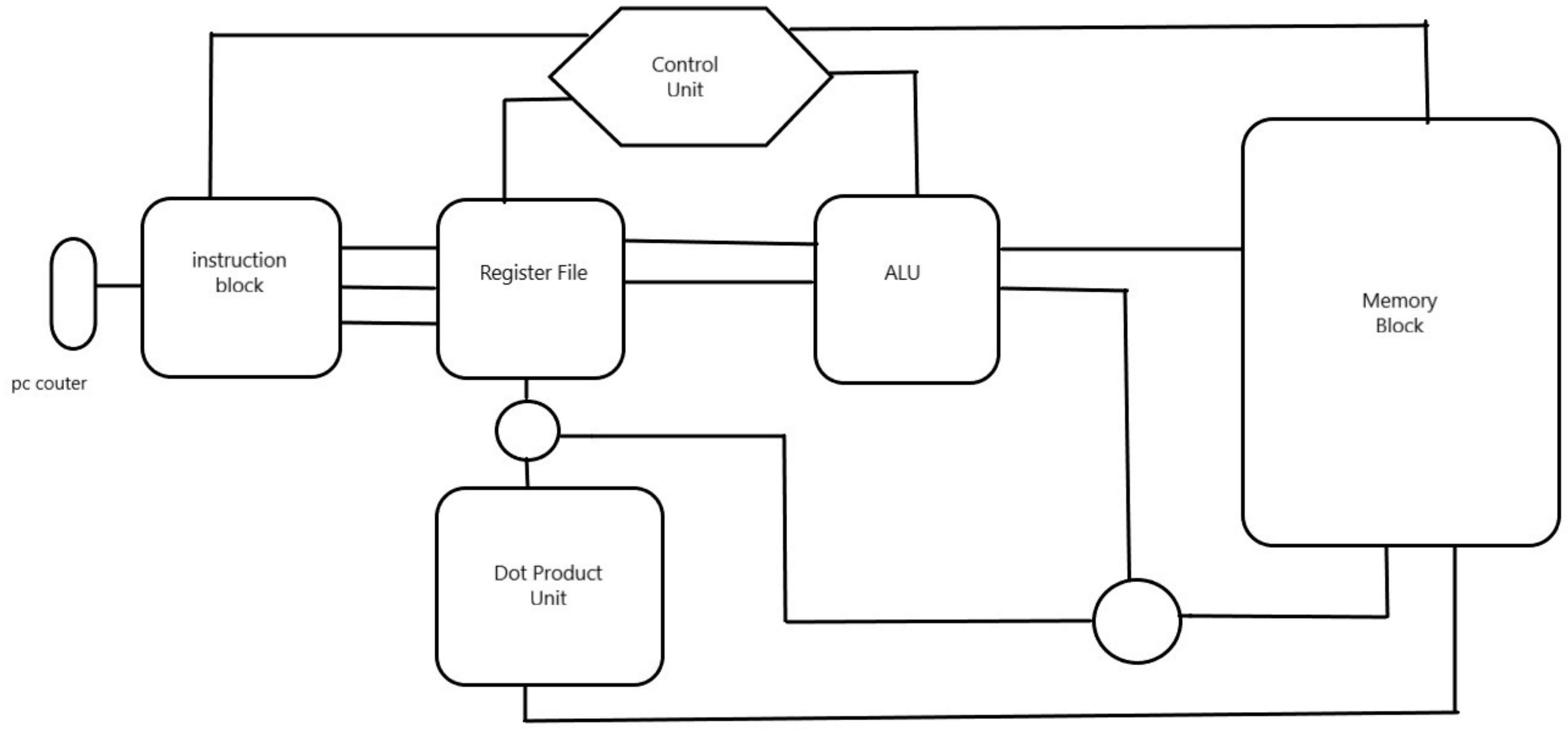


registers[0:31][31:0]	0,0,40,640,51,1,1,
[0][31:0]	0
[1][31:0]	0
[2][31:0]	40
[3][31:0]	640
[4][31:0]	51
[5][31:0]	1
[6][31:0]	1

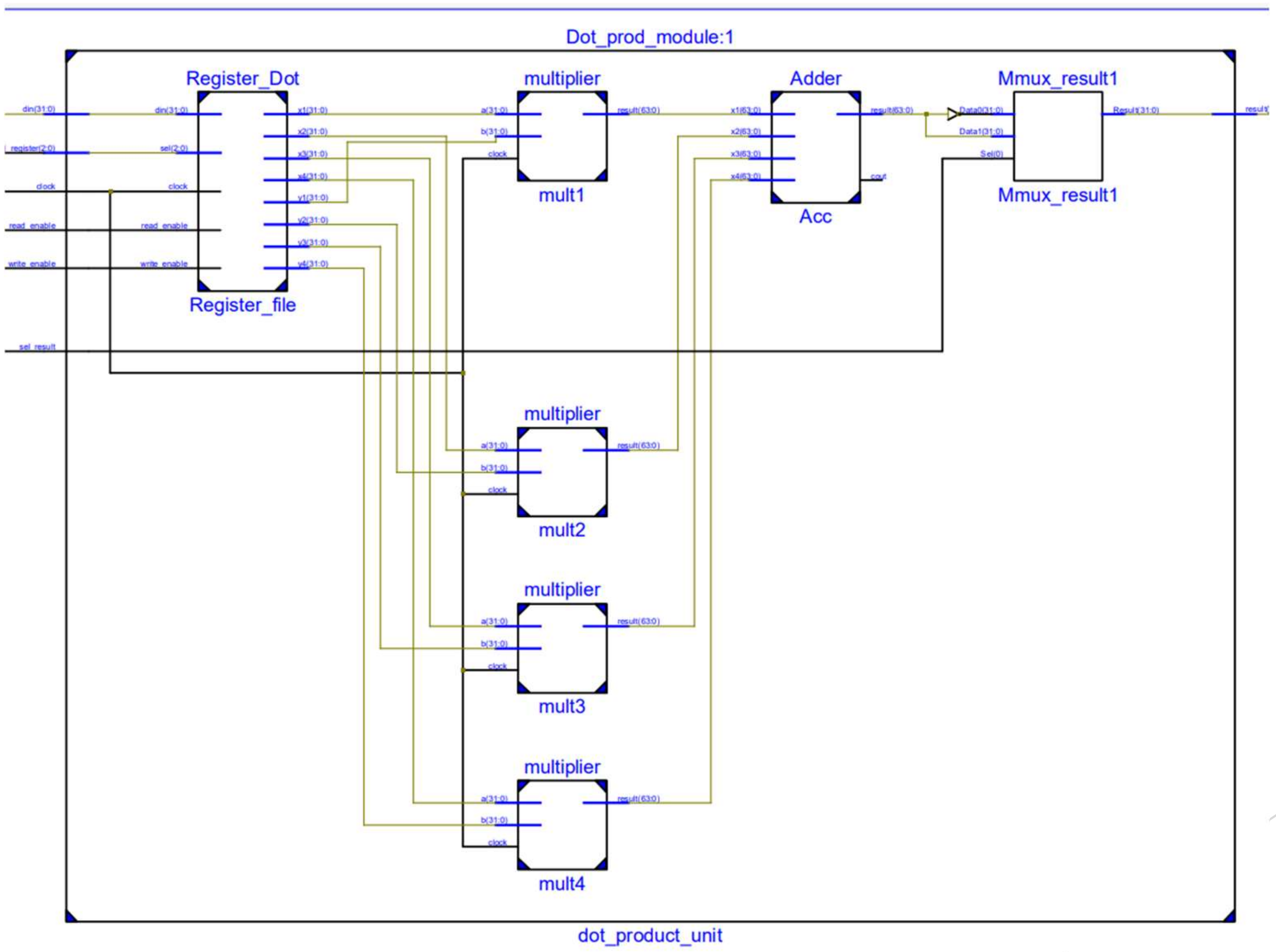
Instructions :

```
Addi  r0 ,r2 , 40
sw    fp  ,r2 , 8
Addi  r0 ,r3 ,40
sw    fp , r3 ,12
Addi r0  ,r4 ,51
Addi r0  ,r5 ,1
slt     r2 ,r4 ,r6
beq  r6  ,r5 , 2
Add  r3 ,r2 ,r3
jump       36
sll     r3 ,r3 ,4
sw   fp  ,r3 1,6
```

Block diagram of processor



Lets look inside Dot Product Unit ….



Dot product operation :

Dot product of two 4D vectors : [x1 , x2 , x3 , x4] ● [y1 , y2 , y3 , y4]

Without Dot product

```
lw  $x1 , 0(addr)
….
lw $x4 , 12(addr)
lw $y1 , 16(addr)
….
lw $y4 , 28(addr)

Mult $x1 , $y1 , $10
….
Mult $x4 , $y4 , $13

Add $10 , $11 , $15
….
Add $15 , $13 , $15

sw  $15, 0(addr)
```

With Dot Product

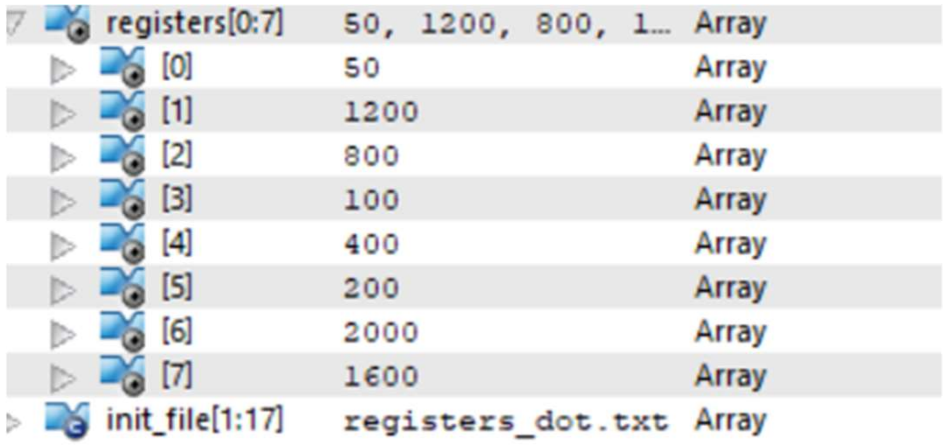
```
Dlw $0 , 0(addr)
…..
Dlw $3 , 12(addr)
Dlw $4 , 16(addr)
…..
Dlw $7 , 28(addr)

Dsw1    0(addr)
Dsw2    4(addr)
```

Example : Dot product operation

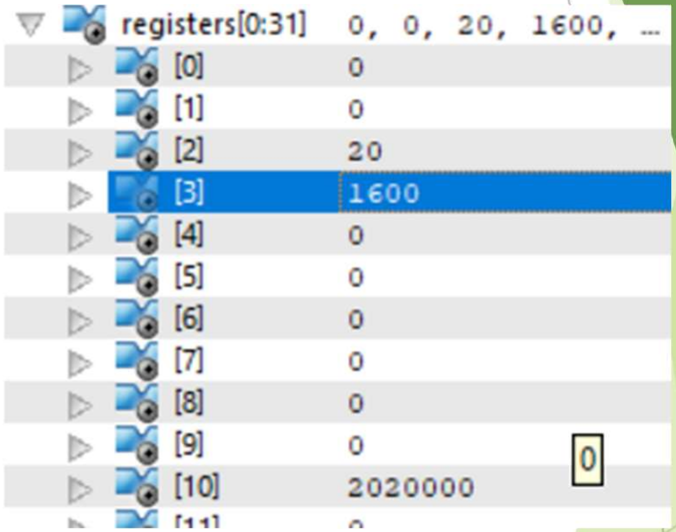
Vector A = [50, 1200, 800 , 100] , Vector B =[400 , 200 , 2000 , 1600]

Dot product register file contents



```
▽ registers[0:7]     50, 1200, 800, 1...  Array
  ▷ [0]              50                   Array
  ▷ [1]              1200                 Array
  ▷ [2]              800                  Array
  ▷ [3]              100                  Array
  ▷ [4]              400                  Array
  ▷ [5]              200                  Array
  ▷ [6]              2000                 Array
  ▷ [7]              1600                 Array
▷ init_file[1:17]    registers_dot.txt    Array
```

Main register file has the dot product result



```
▽ registers[0:31]   0, 0, 20, 1600, ...
  ▷ [0]             0
  ▷ [1]             0
  ▷ [2]             20
  ▷ [3]             1600
  ▷ [4]             0
  ▷ [5]             0
  ▷ [6]             0
  ▷ [7]             0
  ▷ [8]             0
  ▷ [9]             0              0
  ▷ [10]            2020000
```

Comparison

Normal procedure
8 load operations : 2 x 8 cycles
4 mult operations : 33 x 4 cycles
3 add operations : 2 x 3 cycles
1 sw operations : 1 x 2 cycle
Total cycles : 156

Dot product unit
8 load operations : 2 x 8 cycles
4 mult operation : 33 x 1 cycles
(we are performing 4 multiplications simultaneously , in the last cycle we are storing the result in memory)
Total cycles : 49

HDL Synthesis Report for DPU

Macro Statistics

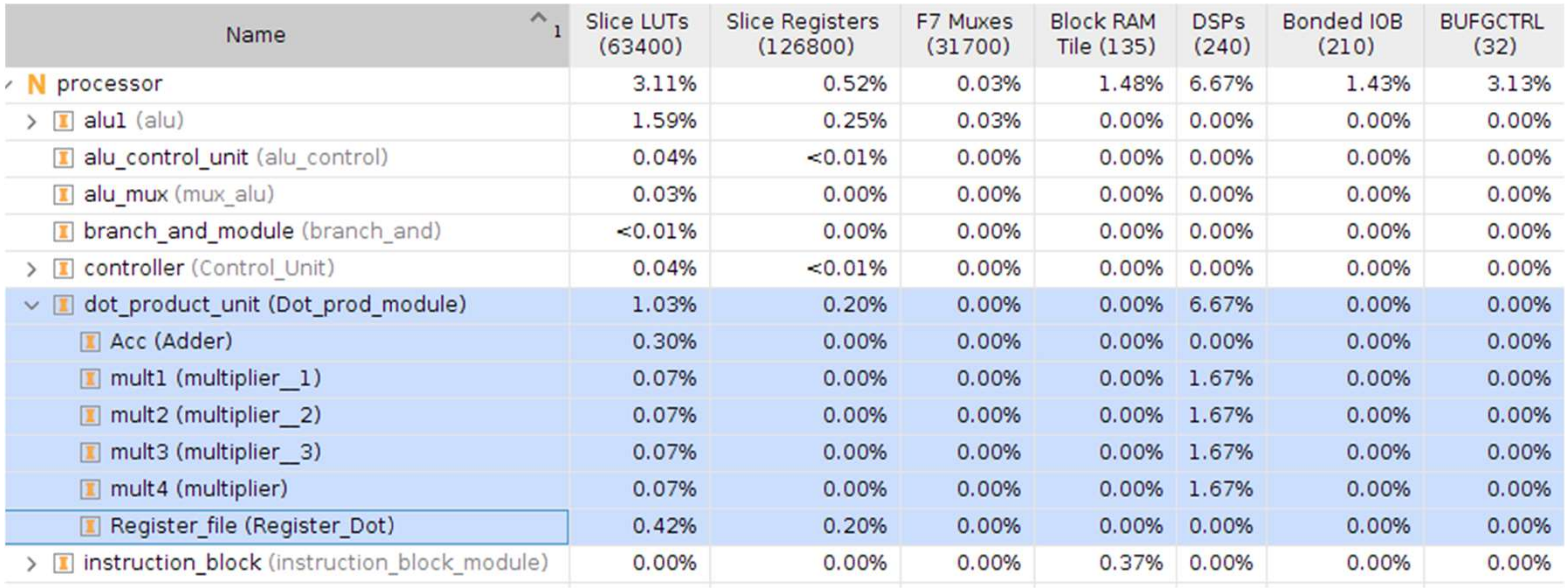
# Multipliers	: 4
32x32-bit multiplier	: 4
# Adders/Subtractors	: 3
64-bit adder	: 3
# Registers	: 8
32-bit register	: 8
# Multiplexers	: 9
32-bit 2-to-1 multiplexer	: 9

Performance gain

It has performed Dot product of two 4D vectors in 49 .

The percentage of clock cycles reduction is 68% .
With the cost of extra components and power

Dot Product Unit resource utilisation :

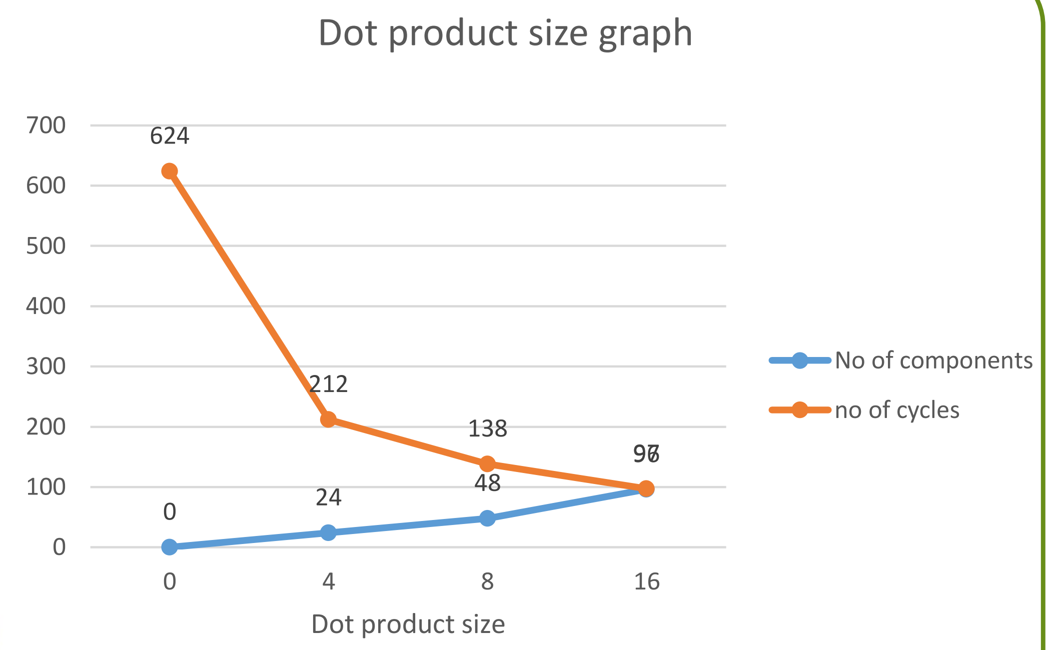


Name	Slice LUTs (63400)	Slice Registers (126800)	F7 Muxes (31700)	Block RAM Tile (135)	DSPs (240)	Bonded IOB (210)	BUFGCTRL (32)
processor	3.11%	0.52%	0.03%	1.48%	6.67%	1.43%	3.13%
alu1 (alu)	1.59%	0.25%	0.03%	0.00%	0.00%	0.00%	0.00%
alu_control_unit (alu_control)	0.04%	<0.01%	0.00%	0.00%	0.00%	0.00%	0.00%
alu_mux (mux_alu)	0.03%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
branch_and_module (branch_and)	<0.01%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
controller (Control_Unit)	0.04%	<0.01%	0.00%	0.00%	0.00%	0.00%	0.00%
dot_product_unit (Dot_prod_module)	1.03%	0.20%	0.00%	0.00%	6.67%	0.00%	0.00%
Acc (Adder)	0.30%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
mult1 (multiplier__1)	0.07%	0.00%	0.00%	0.00%	1.67%	0.00%	0.00%
mult2 (multiplier__2)	0.07%	0.00%	0.00%	0.00%	1.67%	0.00%	0.00%
mult3 (multiplier__3)	0.07%	0.00%	0.00%	0.00%	1.67%	0.00%	0.00%
mult4 (multiplier)	0.07%	0.00%	0.00%	0.00%	1.67%	0.00%	0.00%
Register_file (Register_Dot)	0.42%	0.20%	0.00%	0.00%	0.00%	0.00%	0.00%
instruction_block (instruction_block_module)	0.00%	0.00%	0.00%	0.37%	0.00%	0.00%	0.00%

- Here the dot product unit has 4 multiplier modules .
- They have utilized 6.67 % of DSPs
- Implementation of more multiplier will increase resource utilization significantly
- It will also need more registers to store data
- Here the dot product unit utilized 1.03 % of Slice LUTs

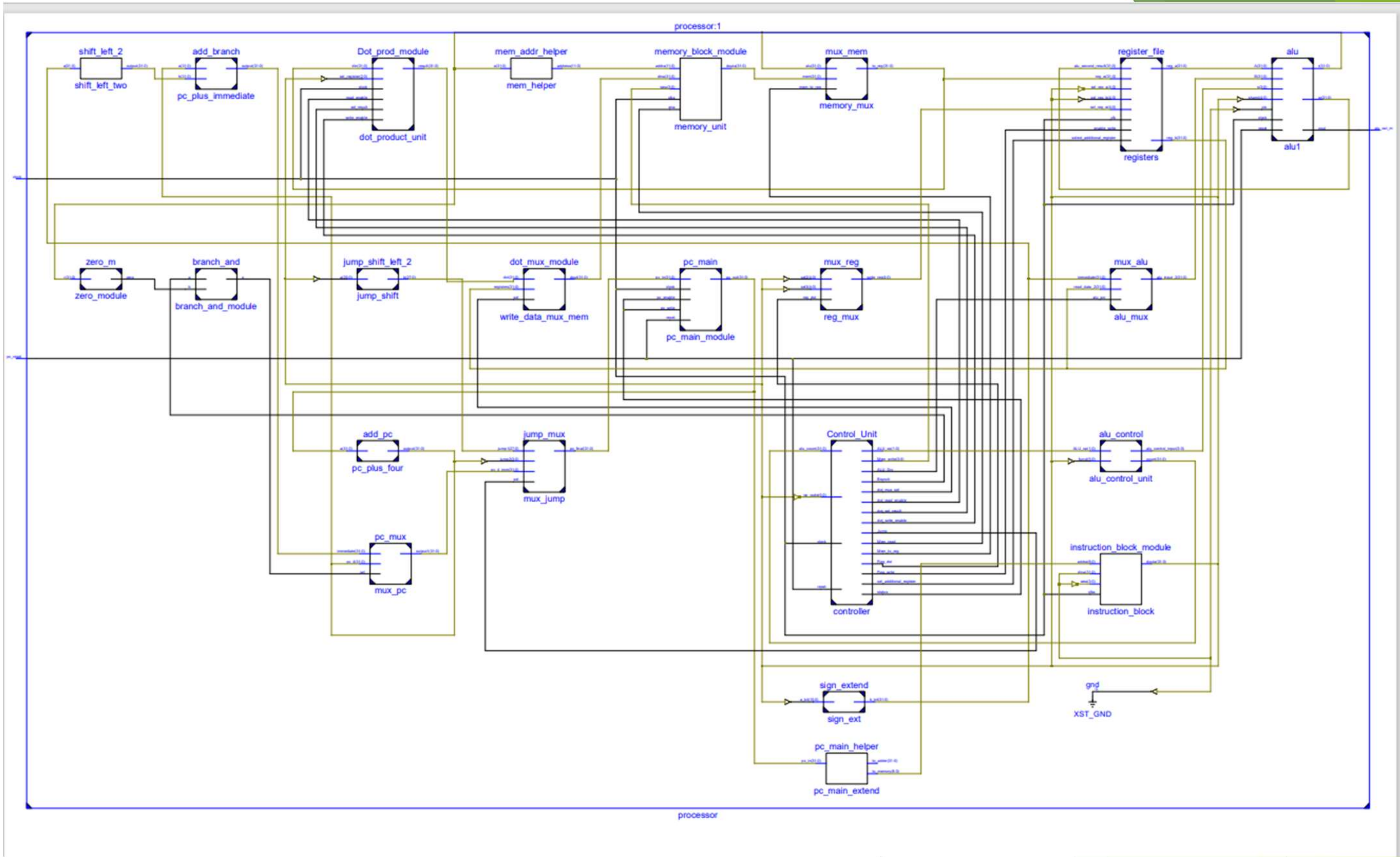
More about Dot product unit :

Results of performing Dot product operation on two 16D vectors with different sized Dot product units



Observations :

- If we increase the dot product size by 2 times , the number of components are twice compare to previous one.
- If Dot product size is 4 , there is a 66 % decrease in clock cycles
- If size is 8 , there is only 35 % decrease in cycles when compare to size 8 where the number of components are twice .
- Similarly there is 30 % decrease in clock cycles


processor:1
shift_left_2
shift_left_two
add_branch
pc_plus_immediate
Dot_prod_module
dot_product_unit
mem_addr_helper
mem_helper
memory_block_module
memory_unit
mux_mem
memory_mux
register_file
registers
alu
alu1
zero_m
zero_module
branch_and
branch_and_module
jump_shift_left_2
jump_shift
dot_mux_module
write_data_mux_mem
pc_main
pc_main_module
mux_reg
reg_mux
mux_alu
alu_mux
add_pc
pc_plus_four
jump_mux
mux_jump
Control_Unit
controller
alu_control
alu_control_unit
pc_mux
mux_pc
instruction_block_module
instruction_block
sign_extend
sign_ext
gnd
XST_GND
pc_main_helper
pc_main_extend
processor

Conclusion

- The project can be found at : JeevanIITH/Processor_final
- Resource book used : Computer organization and design
- I got very good insights of computer architecture , VHDL, VIVADO .
- Understood deep concepts of processor , some other digital circuits components .
- Able to develop other modules , add new components to existing ones.
- Plan to work on multicore processor .

THANK YOU