3rd electrical

Computer department
Computer organization project (II)

Group no.:41

Submitted by:

Aya alaa ghonaim
Aya ahmed mohamed
Aya sayed fouad
Aya kamal shawky
Marwa magdy ahmed
Marwa mostafa kaoud

[PIPELINED MIPS PROCCESSOR]

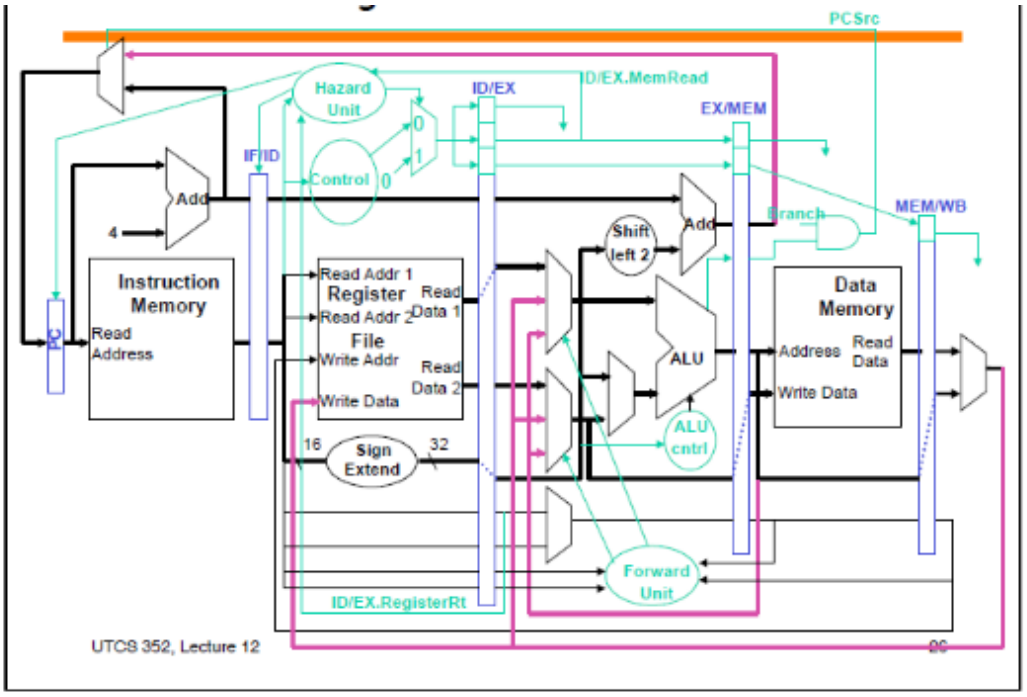
The report generally shows the hardware design of the pipelined mips processor by Verilog HDL under modelsim edit and simulation tool ;taking into consideration :data hazards , memory hazards and control hazards that would take place due to overlapping between pipelining stages and the implementation or synthesis of the processor design using Xilinx Synthesis Tool

Contents:

1. Processor complete design (schematic)
2. implementation codes
  2.1 verilog modules for each processor element
  2.2 verilog module for the whole processor (complete module)
3.test cases used in testing the behavior of the processor
 3.1 text files for each test case in both assembly and binary code
 3.2.screenshots for the outputs(wave simulation) of each test case
4.synthesis of the Verilog HDL design using xillnix tool
 4.1 synthesis for each datapath element module
 4.2 synthesis for the whole design

1. Processor complete design (schematic)

This is the final design we reached; the design also covers data hazards , control hazards and memory hazards .



2. implementation codes:

Attached a text file (" verilog-pipeline-project.txt ") contains the complete verilog code we reached.

3.test cases used in testing the behavior of the processor:

Attached a text files contains the complete test cases used explaining the reason of using each test case and what it really tests on the processor ex.alu forwarding , branch stalls and so on.

```
000000_01000_01001_00001_00000_100000 //add $1,$8,$9  //15+14=29
000000_00001_00101_00010_00000_100010 //sub $2,$1,$5 //29-1=28
000000_00001_00010_00001_00000_100000 //add $3,$1,$2  //29+28=57
000000_00011_00011_00101_00000_100101 //or $5,$3,$3  // 57|57=57
000000_00011_00011_00110_00000_100100  //and $6,$3,$3  // 57&57=57
```

```
100011_00100_00001_0000000000000001 //lw $1,1($4) // $1=>8
100011_00100_00010_0000000000000000 //lw $2,2($4)
000000_00001_00010_00011_00000_100000 //add $3,$1,$2
000000_00011_00101_01000_00000_100000 //add $8,$3,$5
```

```
101011_00010_00001_0000000000000010 //sw $1,2 ($2)
100011_00101_00010_0000000000000000 //lw $2,0 ($5)
000000_00011_01000_00101_00000_100101 //or $5,$3,$8
000000_00010_00001_01000_00000_100000 //add $8,$2,$1
000000_00000_00111_00100_00010_000000//sll $4,$7,2
000000_00001_00101_00110_00000_100100  //and $6,$1,$5
000000_01110_00001_00010_00000_100010 //sub $2,$14,$1  15-1=14
```

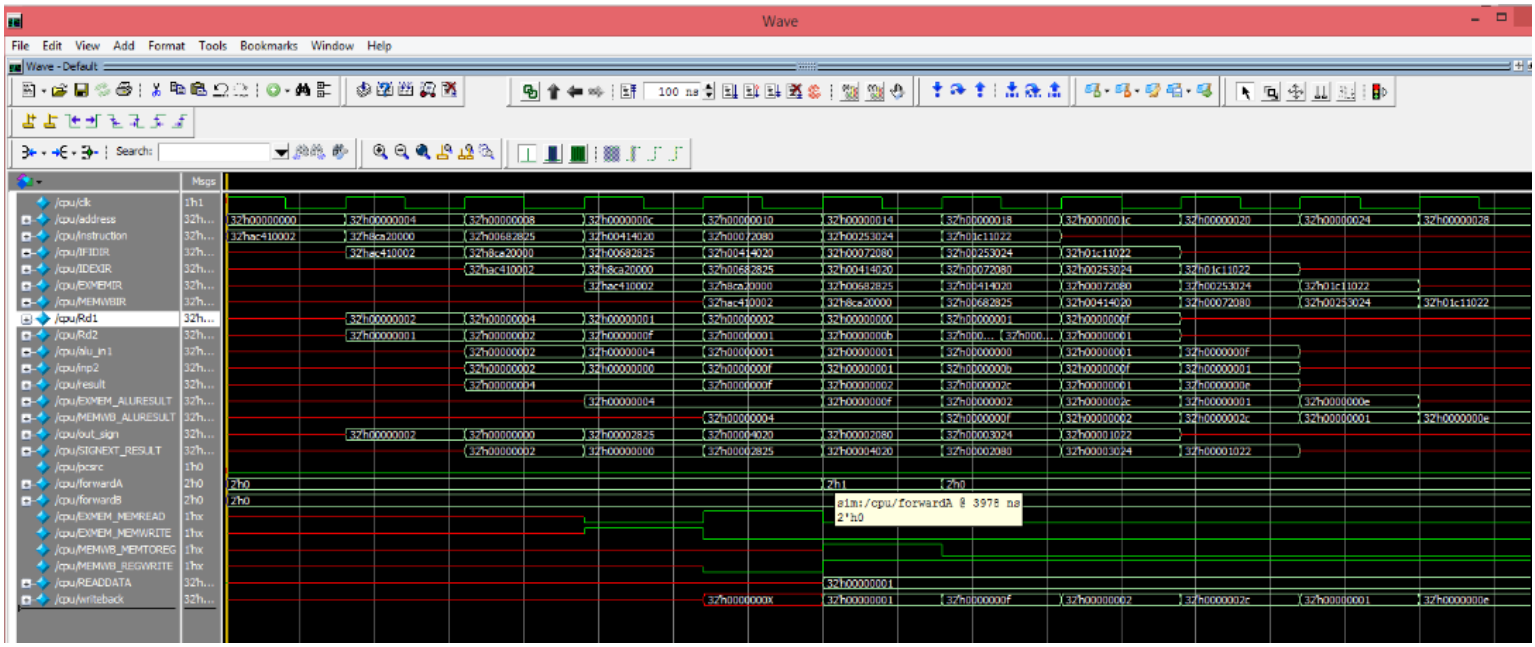
```
/*test case 5 :all hazards */
100011_00011_00100_0000000000000100  //lw $4,4($3);
000000_00100_00001_00010_00000_100010 //sub $2,$4,$1; lw , sub>>mem hazard
000000_00010_00010_00101_00000_100000 //add $5,$2,$2; sub , add >>data hazard
000100_00011_00100_0000000000000010   //beq $3,$4,2;   branch taken >>control hazard
000000_00111_00110_00110_00000_100101  //or $6,$7,$6; >> not to be excuted
101011_00011_00001_0000000000000000   //sw $1,0($3); >>not to be excuted
000000_00011_00011_00001_00000_100000 //add $3,$5,$4; >>branch destiniation


/* test case 6:lw followed by a branch */
100011_00011_00010_0000000000001000 //lw $2,4($3);
000100_00010_00011_0000000000000010 //beq $2,$3,1; >>branch taken
000000_00111_00110_00110_00000_100101 //or $6,$7,$6;
000000_01010_01010_01001_00000_100000 //add $9,$10,$10;
```

3.1 test cases outputs :

1. Test case 1: tests all instruction without hazards and branches

101011_00010_00001_0000000000000010 //sw $1,2 ($2)
100011_00101_00010_0000000000000000 //lw $2,0 ($5)
000000_00011_01000_00101_00000_100101 //or $5,$3,$8
000000_00010_00001_01000_00000_100000 //add $8,$2,$1
000000_00000_00111_00100_00010_000000 //sll $4,$7,2
000000_00001_00101_00110_00000_100100 //and $6,$1,$5
000000_01110_00001_00010_00000_100010 //sub $2,$14,$1 15-1=14

Output on wave simulation:



Register file :

```
Rdata[32'd0] <= 0;

Rdata[32'd1] <= 1;

Rdata[32'd2] <= 2;

Rdata[32'd3] <= 1;

Rdata[32'd4] <= 1;

Rdata[32'd5] <= 1;

Rdata[32'd6] <= 80;

Rdata[32'd7] <= 11;

Rdata[32'd8] <= 15;
Rdata[32'd9] <= 14;
Rdata[32'd10] <= 9;
Rdata[32'd11] <= 5;
Rdata[32'd12] <= 55;
Rdata[32'd13] <= 35;
Rdata[32'd14] <= 6;
```

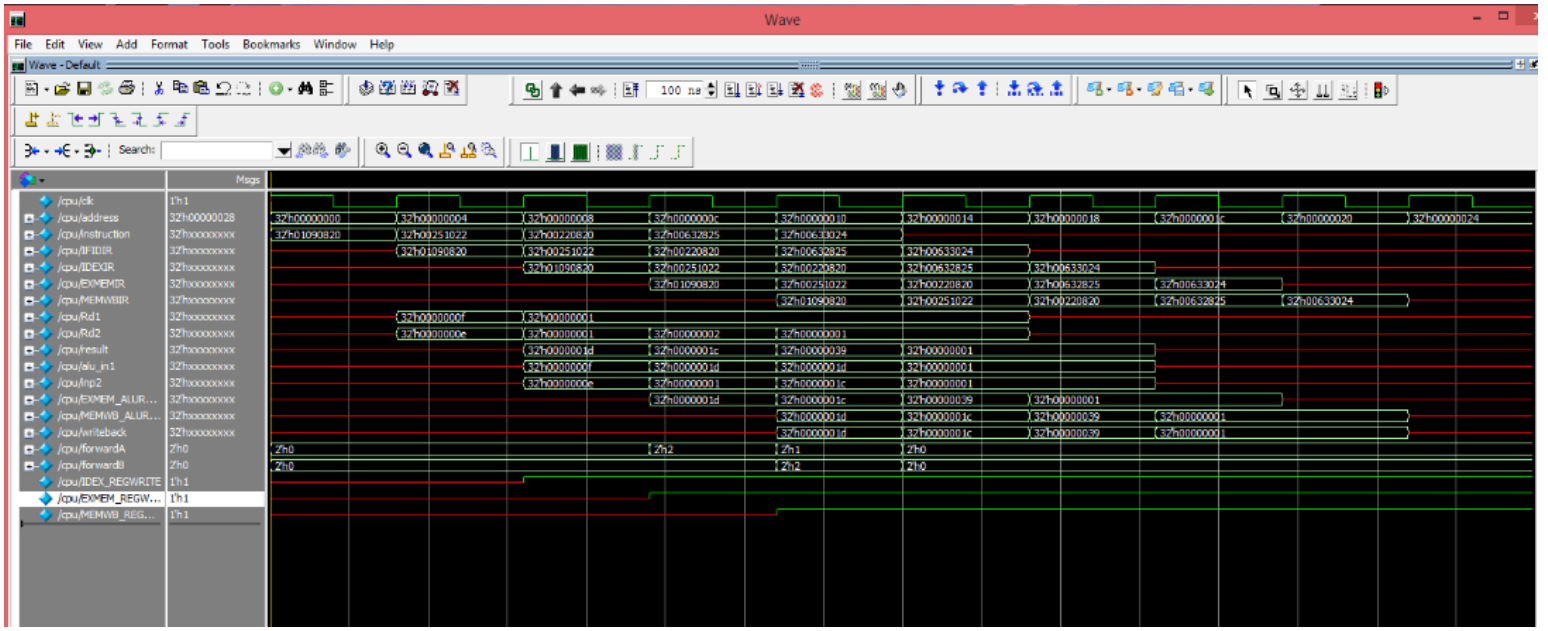
Data memory:

```
Data_arr[32'd0]<=5;
Data_arr[32'd1]<=10;
Data_arr[32'd2]<=8;
Data_arr[32'd3]<=15;
Data_arr[32'd4]<=12;
Data_arr[32'd5]<=11;
Data_arr[32'd6]<=9;
Data_arr[32'd7]<=8;
Data_arr[32'd8]<=6;
Data_arr[32'd9]<=13;
Data_arr[32'd10]<=5;
Data_arr[32'd11]<=11;
Data_arr[32'd12]<=16;
Data_arr[32'd13]<=5;
Data_arr[32'd14]<=14;
Data_arr[32'd15]<=15;
```

Test case 2: tests the alu forwarding (data hazard)

000000_01000_01001_00001_00000_100000 //add $1,$8,$9 //15+14=29
000000_00001_00101_00010_00000_100010 //sub $2,$1,$5 //29-1=28 >>forwarding $1
000000_00001_00010_00001_00000_100000 //add $3,$1,$2 //29+28=57
000000_00011_00011_00101_00000_100101 //or $5,$3,$3 // 57|57=57 >>forwarding $3
000000_00011_00011_00110_00000_100100 //and $6,$3,$3 // 57&57=57

Output on wave simulation:

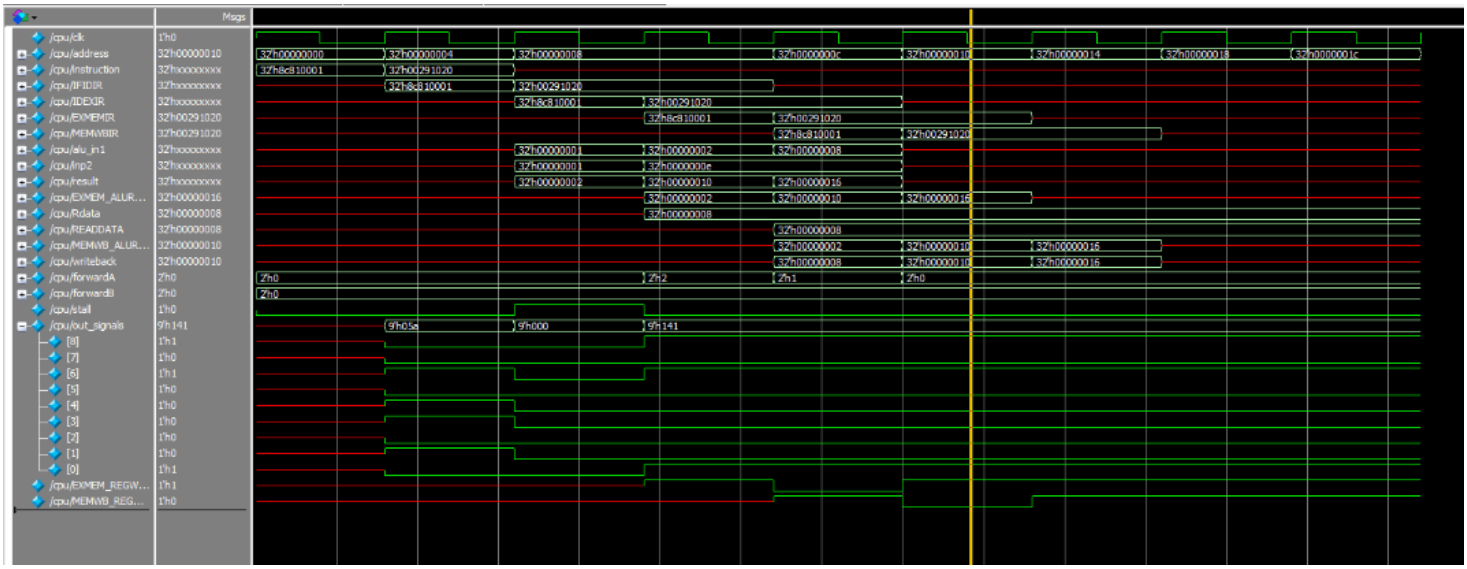


Test case 3 :tests memory hazards (pipeline stalls) and data hazard

```
100011_00100_00001_0000000000000001 //lw $1,1($4) // $1=>8
100011_00100_00010_0000000000000000 //lw $2,2($4)
000000_00001_00010_00011_00000_100000 //add $3,$1,$2
000000_00011_00101_01000_00000_100000 //add $8,$3,$5
```

1. Stalls add $3,$1,$2 until data is written in $2 And then forwarding from data memory to alu
   2.forwarding $3 to alu for the last instruction

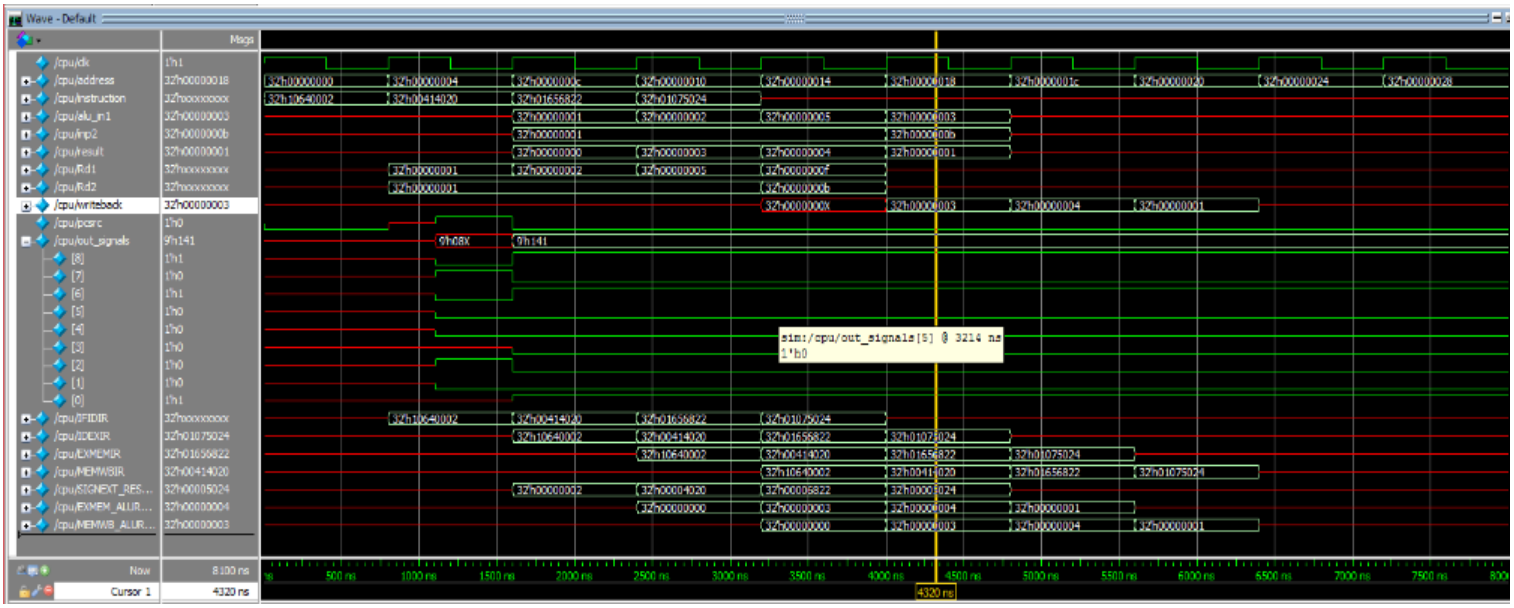
Output on wave simulation:



Test case 4:tests branch taken and stalls (control hazards)

```
000100_00011_00100_0000000000000010 //beq $3,$4,2 taken
000000_00010_00001_01000_00000_100000 //add $8,$2,$1
000000_00001_00101_00110_00000_100100  //and $6,$1,$5
000000_01011_00101_01101_00000_100010 //sub $13,$11,$5
000000_01000_00111_01010_00000_100100  //and $10,$8,$7
```

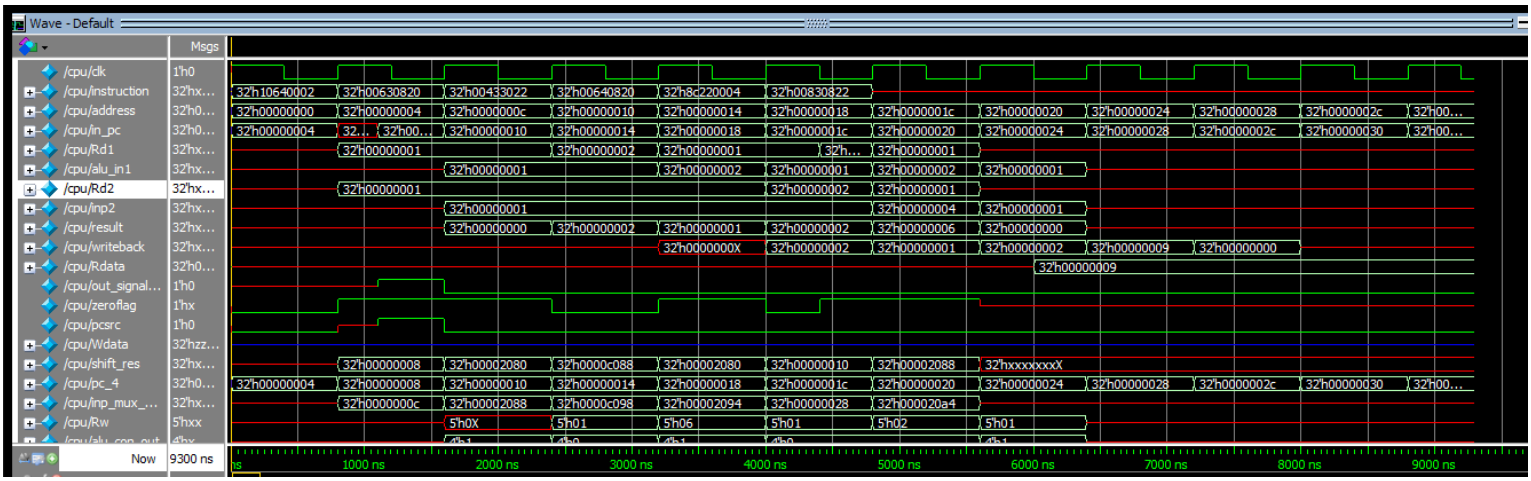
Output on wave simulation:



Test case 5:

```
000100_00101_00110_0000000000000010   //beq $5,$6,2;  branch not taken //1!=80
000000_00011_00100_00001_00000_100000 //add $1,$3,$4;  1+1>>2
100011_00001_00010_0000000000000100   //lw $2,4($1);  $2>>>9
000000_00100_00011_00001_00000_100010 //sub $1,$4,$3; 1-1=0
```

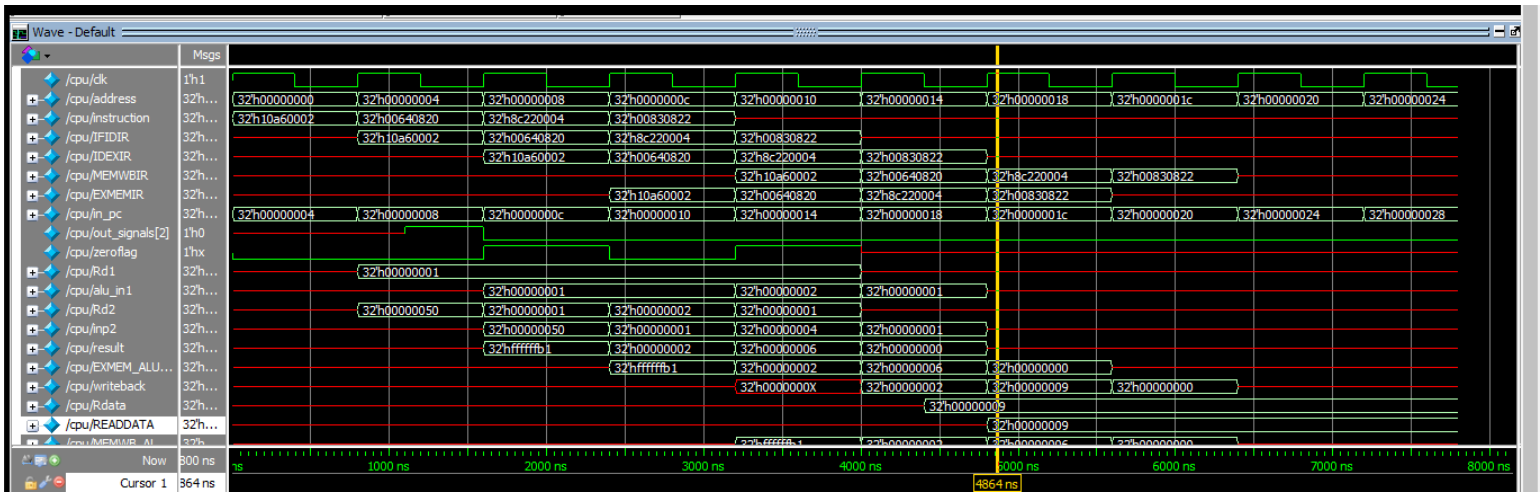
Output on wave simulation:



Test case 6:

```
000100_00011_00100_0000000000000010   //beq $3,$4,2; 1=1
000000_00011_00011_00001_00000_100000 //add $1,$3,$3;****//el wr reg hysawi zero
000000_00000_00011_00001_00110_000000 //sll $1,$3,6;****
000000_00010_00011_00110_00000_100010 //sub $6,$2,$5; 2-1=1
000000_00011_00100_00001_00000_100000 //add $1,$3,$4; 1+1=2
100011_00001_00010_0000000000000100   //lw $2,4($3); >>9
000000_00100_00011_00001_00000_100010 //sub $1,$4,$3; 1-1=0
```

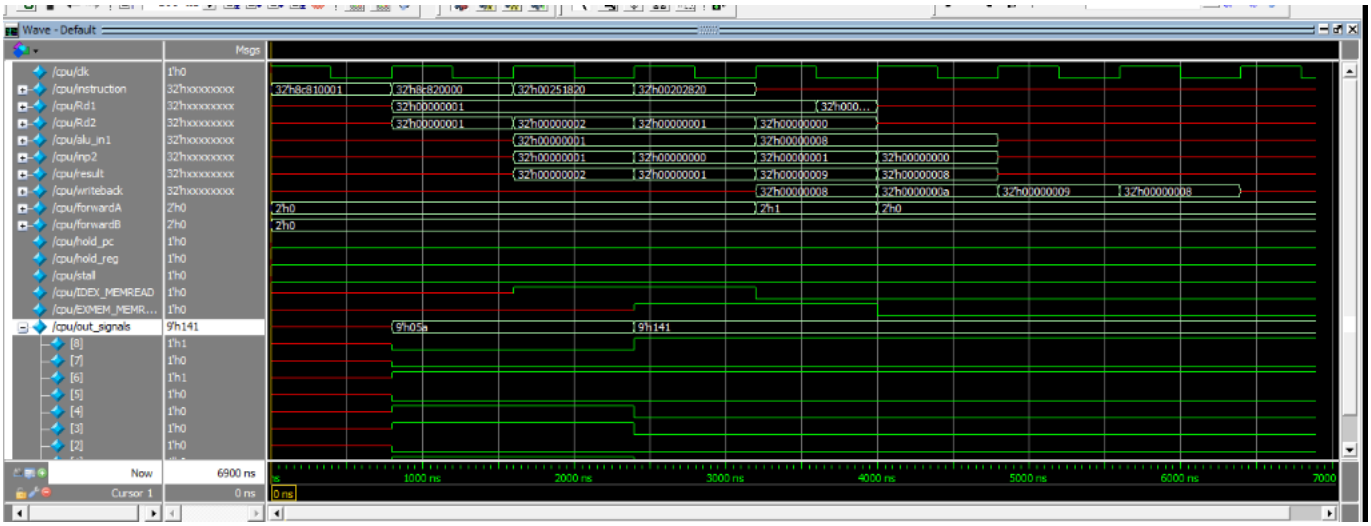
Output on wave simulation:



Test case 7:

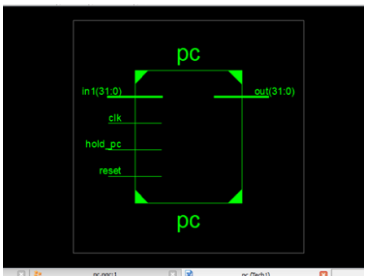
```
100011_00100_00001_0000000000000001 //lw $1,1$4 //load 8 in $1
100011_00100_00010_0000000000000000 //lw $2,0$4 //load 10 in $2
000000_00001_00101_00011_00000_100000 //add $3,$1,$5// $3=8+1
000000_00001_00000_00101_00000_100000 //add $5,$1,$0 // $5=8+0
```

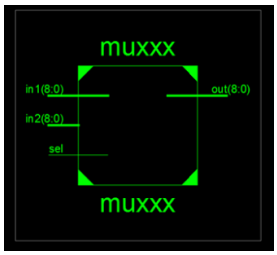
Output on wave simulation:

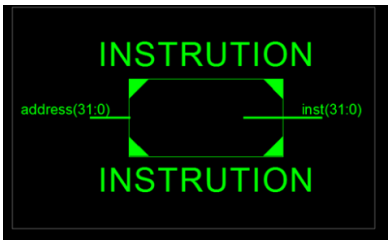


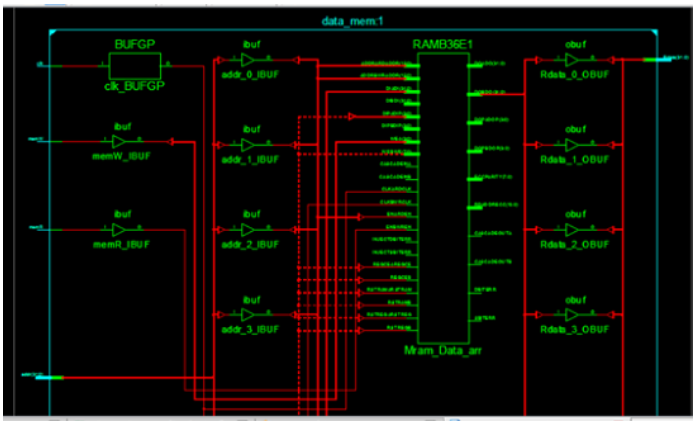
4.synthesis of the Verilog HDL design using xillnix tool:

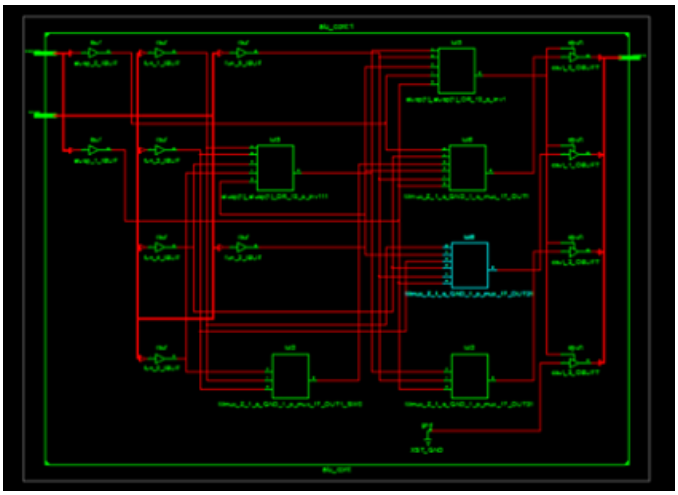
4.1 synthesis for each datapath element module :

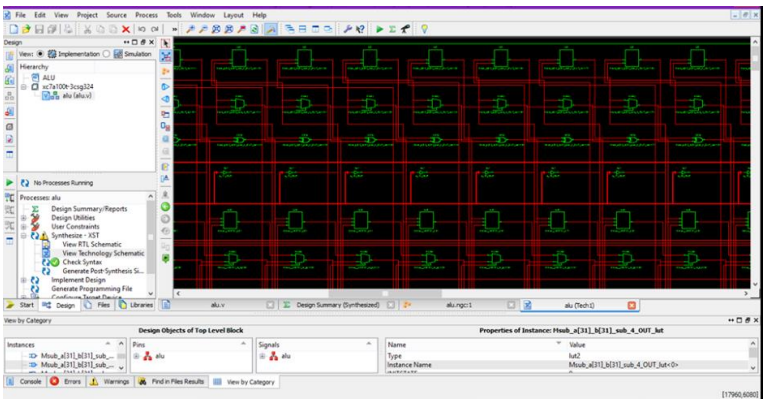


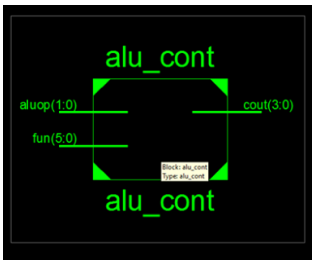


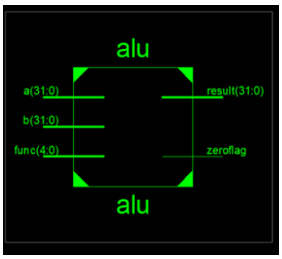


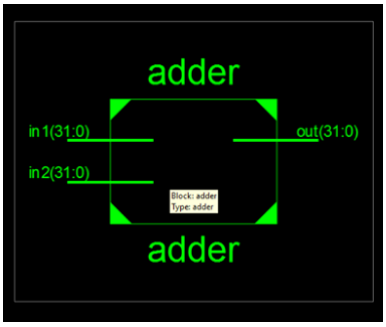


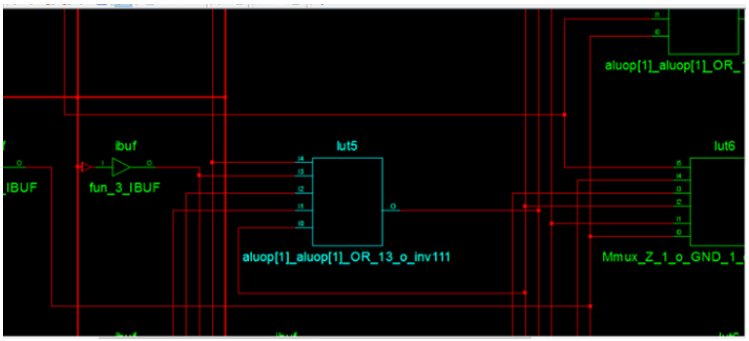


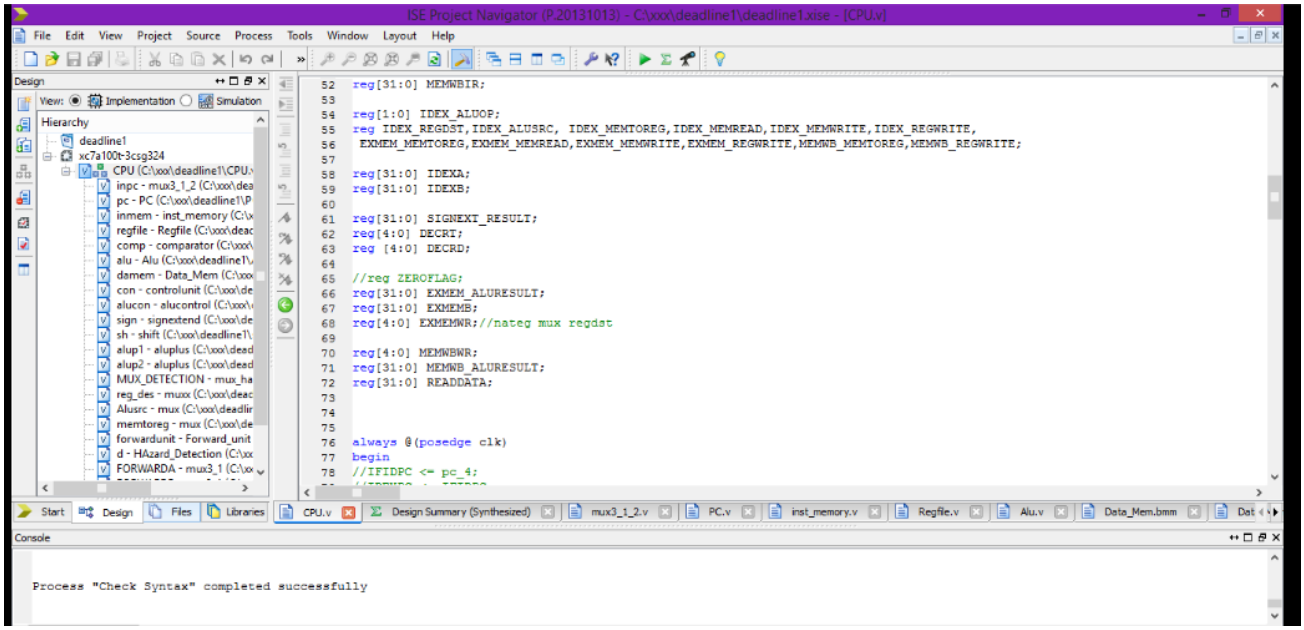











ISE Project Navigator (P.20131013) - C:\xxx\deadline1\deadline1.xise - [CPU.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
deadline1
xc7a100t-3csg324
CPU (C:\xxx\deadline1\CPU.v
inpc - mux3_1_2 (C:\xxx\dea
pc - PC (C:\xxx\deadline1\P
inmem - inst_memory (C:\x
regfile - Regfile (C:\xxx\deac
comp - comparator (C:\xxx\
alu - Alu (C:\xxx\deadline1\
damem - Data_Mem (C:\xxx
con - controlunit (C:\xxx\de
alucon - alucontrol (C:\xxx\
sign - signextend (C:\xxx\de
sh - shift (C:\xxx\deadline1\
alup1 - aluplus (C:\xxx\dead
alup2 - aluplus (C:\xxx\dead
MUX_DETECTION - mux_ha
reg_des - muxx (C:\xxx\deac
Alusrc - mux (C:\xxx\deadlir
memtoreg - mux (C:\xxx\de
forwardunit - Forward_unit
d - HAzard_Detection (C:\xx
FORWARDA - mux3_1 (C:\xx
52 reg[31:0] MEMWBIR;
53
54 reg[1:0] IDEX_ALUOP;
55 reg IDEX_REGDST,IDEX_ALUSRC, IDEX_MEMTOREG,IDEX_MEMREAD,IDEX_MEMWRITE,IDEX_REGWRITE,
56 EXMEM_MEMTOREG,EXMEM_MEMREAD,EXMEM_MEMWRITE,EXMEM_REGWRITE,MEMWB_MEMTOREG,MEMWB_REGWRITE;
57
58 reg[31:0] IDEXA;
59 reg[31:0] IDEXB;
60
61 reg[31:0] SIGNEXT_RESULT;
62 reg[4:0] DECRT;
63 reg [4:0] DECRD;
64
65 //reg ZEROFLAG;
66 reg[31:0] EXMEM_ALURESULT;
67 reg[31:0] EXMEMB;
68 reg[4:0] EXMEMWR;//nateg mux regdst
69
70 reg[4:0] MEMWBWR;
71 reg[31:0] MEMWB_ALURESULT;
72 reg[31:0] READDATA;
73
74
75
76 always @(posedge clk)
77 begin
78 //IFIDPC <= pc_4;
Start Design Files Libraries
CPU.v Design Summary (Synthesized) mux3_1_2.v PC.v inst_memory.v Regfile.v Alu.v Data_Mem.bmm
Console
Process "Check Syntax" completed successfully