INDIAN INSTITUTE OF INFORMATION TECHNOLOGY VADODARA - INTERNATIONAL CAMPUS DIU

Practical Work

Name : Pooja
Roll No : 202311063
Subject : Computer Organisation and Architecture
Course : CS268
Branch : CSE
Batch : 2023 - 2027

EXPERIMENT-8

Aim:- Design and implementation of an advance pipeline architecture in verilog.

Prerequisites:- A fair understanding of pipeline concept and verilog language.

Theory:- It is four stage pipeline:-

1. **Stage 1 (Instruction Fetch/Decode):** Fetches operands from a register bank and decodes the instruction inputs (e.g., source registers, function code, destination register, and memory address).
2. **Stage 2 (Execution):** Performs the specified operation (e.g., addition, subtraction, multiplication) based on a function code.
3. **Stage 3 (Destination Register):** Writes the execution result back to the specified register in the register bank.
4. **Stage 4 (Write Back):** Stores the result in a memory bank at the specified address.

To avoid overlapping of consecutive stages two clock signal are used as shown:-

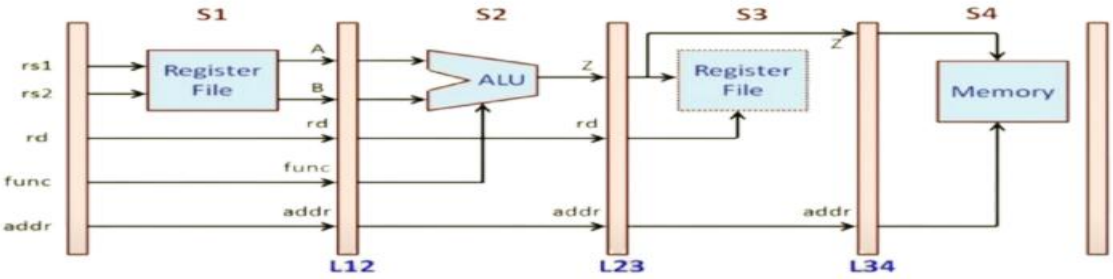
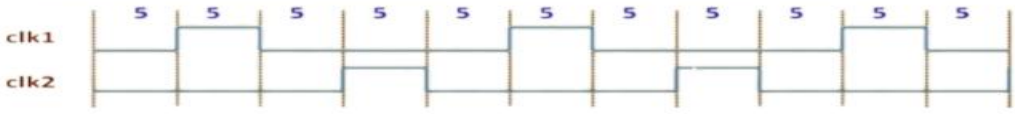


Fig.-1

Clocking Issue in Pipeline

- It is important that the consecutive stages be applied suitable clocks for correct operation.
- Two options:
  a) Use master/slave flip-flops in the latches to avoid race condition.
  b) Use non-overlapping two-phase clock for the consecutive pipeline stages.



Main code:-

C:/Users/pooja/project_5/project_5.srcs/sources_1/new/pipe.v

```
1  module pipe(zout,rs1,rs2,rd,func,addr,clk1,clk2);
2  input[3:0] rs1,rs2,rd;
3  input clk1,clk2;
4  input [3:0] func;
5  input [7:0] addr;
6  output [15:0] zout;
7  reg [15:0] l1a,l1b,l2z,l3z;
8  reg [7:0] l1add,l2add,l3add;
9  reg [3:0] l1func,l1rd,l2rd;
10 reg [15:0] regbank [15:0];
11 reg [15:0] membank[255:0];
12
13 assign zout=l3z;
14 always @(posedge clk1)
15 begin
16 l1a <= #2 regbank[rs1];
17 l1b <= #2 regbank[rs2];
18 l1func <= #2 func;
19 l1rd <= #2 rd;
20 l1add <= #2 addr;
21 end
22 always @(posedge clk2) begin
23 case(func)
24 0: l2z <= #2 l1a+l1b;
25 1: l2z <= #2 l1a-l1b;
26 2: l2z <= #2 l1a*l1b;
27 3: l2z <= #2 l1a;
28 4: l2z <= #2 l1b;
29 5: l2z <= #2 l1a&l1b;
30 6: l2z <= #2 l1a|l1b;
31 7: l2z <= #2 l1a^l1b;
```

```
32  8: l2z <= #2 ~l1a;
33  9: l2z <= #2 ~l1b;
34  10: l2z <= #2 l1a>>1;
35  11: l2z <= #2 l1b<<1;
36  default: l2z<=16'hxxx; endcase
37  l2rd <= #2 l1rd;
38  l2add <= #2 l1add;
39  end
40  always @(posedge clk1)
41  begin
42  regbank[l2rd] <=#2 l2z;
43  l3z <=#2 l2z;
44  l3add <=#2 l2add;
45  end
46  always @(posedge clk2) begin
47  membank[l3add]=#2 l3z; end
48  endmodule
```

Testbench code:-

C:/Users/pooja/project_5/project_5.srcs/sim_1/new/pipe_tb.v

```
1  module simuli();
2  reg[3:0] rs1;
3  reg[3:0] rs2;
4  reg[3:0] rd;
5  reg[3:0] func;
6  reg[7:0] addr;
7  wire [15:0] zout;
8  integer k, i;
9  reg clk1, clk2;
10 pipe pipe(zout,rs1,rs2,rd,func,addr,clk1,clk2);
11 initial clk1=1;
12 initial clk2=0;
13 always #10 clk1=~clk1;
14 always #10 clk2=~clk2;
15 initial
16 for(k=0;k<15;k=k+1)
17 begin
18 pipe.regbank[k]=k;
19 end
20 initial begin
21 $dumpfile("pipe.vcd");
22 $dumpvars;
23 #20 rs1=6; rs2=1; rd=10; addr= 125; k=addr; func = 2; #20 rs1=9; rs2=8; rd=12; addr=126;k=addr;
24 func = 3; #20 rs1=2; rs2=4; rd=13; addr= 125;k=addr; func = 4; #60 $finish ;
25 end
26 endmodule
```

GTK Waveform:-

Schematic Diagram:-