INDIAN INSTITUTE OF INFORMATION TECHNOLOGY VADODARA - INTERNATIONAL CAMPUS DIU

Practical Work

Name : **Kshitiz Gangwar**
Roll No : 202311045
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:-

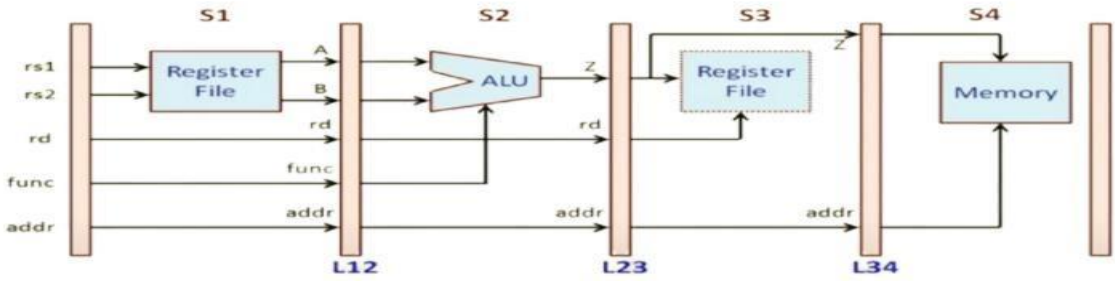
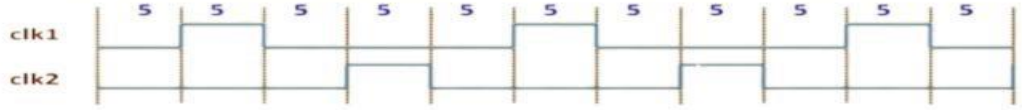


Fig.-1

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:-

Project Summary | simuli.v | pipe.v

C:/Users/KSHITIZ GANGWAR/project_2/project_2.srcs/sources_1/new/pipe.v

```
module pipe(zout,rs1,rs2,rd,func,addr,clk1,clk2);
input[3:0] rs1,rs2,rd;
input clk1,clk2;
input [3:0] func;
input [7:0] addr;
output [15:0] zout;
reg [15:0] l1a,l1b,l2z,l3z;
reg [7:0] l1add,l2add,l3add;
reg [3:0] l1func,l1rd,l2rd;
reg [15:0] regbank [15:0];
reg [15:0] membank[255:0];

assign zout=l3z;
always @(posedge clk1)
begin
l1a <= #2 regbank[rs1];
l1b <= #2 regbank[rs2];
l1func <= #2 func;
l1rd <= #2 rd;
l1add <= #2 addr;
end
always @(posedge clk2) begin
case(func)
0: l2z <= #2 l1a+l1b;
1: l2z <= #2 l1a-l1b;
2: l2z <= #2 l1a*l1b;
3: l2z <= #2 l1a;
4: l2z <= #2 l1b;
5: l2z <= #2 l1a&l1b;
6: l2z <= #2 l1a|l1b;
7: l2z <= #2 l1a^l1b;
5: l2z <= #2 l1a&l1b;
6: l2z <= #2 l1a|l1b;
7: l2z <= #2 l1a^l1b;
8: l2z <= #2 ~l1a;
9: l2z <= #2 ~l1b;
10: l2z <= #2 l1a>>1;
11: l2z <= #2 l1b<<1;
default: l2z<=16'hxxx; endcase
l2rd <= #2 l1rd;
l2add <= #2 l1add;
end
always @(posedge clk1)
begin
regbank[l2rd] <=#2 l2z;
l3z <=#2 l2z;
l3add <=#2 l2add;
end
always @(posedge clk2) begin
membank[l3add]=#2 l3z; end
endmodule
```

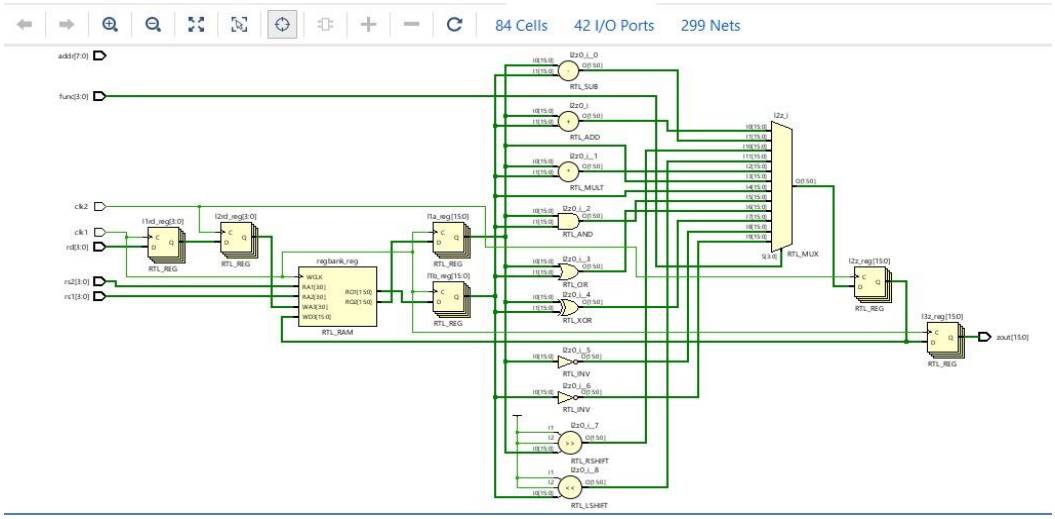
Testbench code:-

Project Summary | simuli.v | pipe.v

C:/Users/KSHITIZ GANGWAR/project_2/project_2.srcs/sources_1/new/simuli.v

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

Schematic Diagram:-



GTK Waveform:-