Anubhav Gupta
U80332699

Abhishek Gaur
U42265597

LAB 2

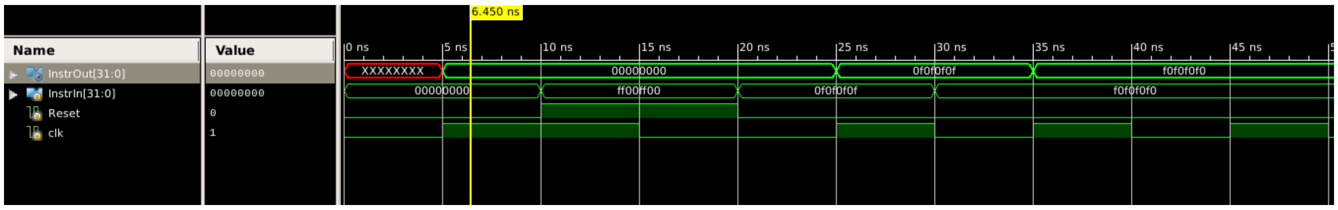
We have designed 3 stage 32-bit pipe-lined data path using different building blocks. We have attached a block diagram of our 3 stage 32-bit pipe-lined data path.

Description of different blocks of the diagram are:--------

1. Stage 1 FlipFlop :-

Its a D type of flipflop. It has 32-bit InstrIn , 1-bit clock and 1-bit reset as inputs and 32-bit InstrOut as output. When reset is low(0) and at positive edge of clock whatever be the input it is reflected as output. When reset is high(1) the output is low(0).

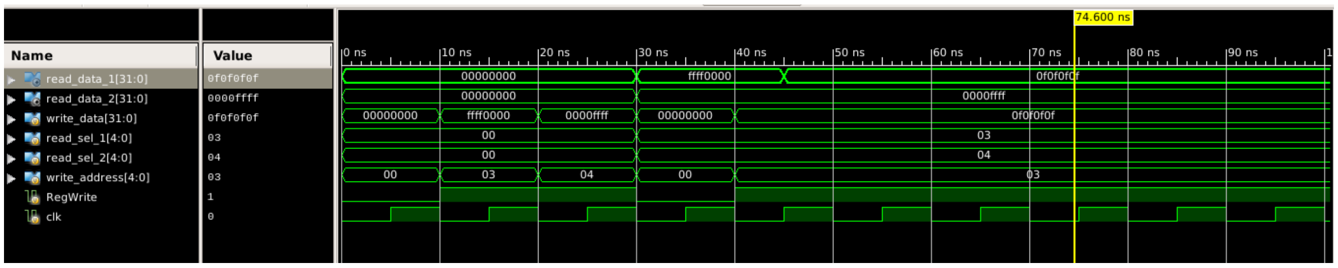
Following is the timing diagram for FlipFlop:



2. **Register File :-**

It is a register file which stores data at a particular address. We can read data from register file too. It has 5-bit readselect1, readselect2,writeselect, 1-bit writeenable and 32-bit writedata as inputs. Whereas it has 32-bit readdata1 and readdata2 as outputs.
When writeenable is 1 the register file will store data whatever is there in writedata in the location of writeselect but there should be positive clock edge.

Following is the timing diagram for Register File:



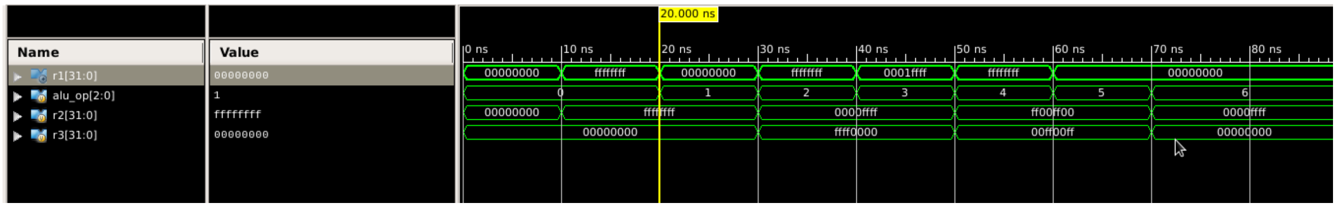
Anubhav Gupta
U80332699

Abhishek Gaur
U42265597

3. ALU :-

ALU block is used for different operations. It has 32-bit R2, R3 and 3-bit ALUop as inputs and 32-bit R1 as output. It performs different operations like addition, subtraction, SLT, etc.

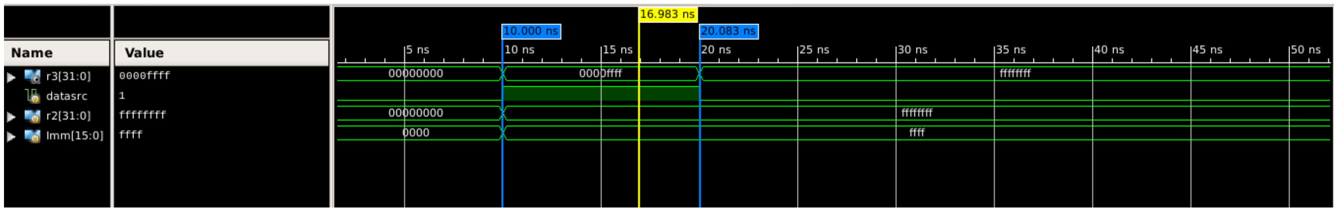
Following is the timing diagram for ALU:



4. MUX:-

It is used for selection of data on which ALU will perform operation. It has 32-bit R2, 16-bit Imm and Datasrc as inputs. As Imm is 16-bit it also converts 16-bit Imm to 32-bit Imm. Thus as per Datascr it selects which input should be given to ALU.

Following is the timing diagram for MUX:



Testing of Register File :-

Step 1: We gave writedata and writeselect as inputs. No values where given to readdata. We made regwrite as 1. Thus register file stores data which is there in writedata on positive edge of clock.

```
write_data = 32'hffff0000;
read_sel_1 = 0;
read_sel_2 = 0;
write_address = 5'b00011;
RegWrite = 1;
```

Step 2: We gave writedata and writeselect as inputs but this time the write address was different than the previous one.. No values where given to readdata. We made regwrite as 1. Thus register file stores data which is there in writedata on positive edge of clock.

```
write_data = 32'h0000ffff;
read_sel_1 = 0;
read_sel_2 = 0;
write_address = 5'b00100;
RegWrite = 1;
```

Step 3: In this step we tried to read data from both the locations where we stored the data in previous steps. We gave inputs to read_sel_1 and read_sel_2 as the addresses of the locations. Thus we got output for both the locations.

```
write_data = 0;
read_sel_1 = 5'b00011;
read_sel_2 = 5'b00100;
write_address = 0;
RegWrite = 0;
```

Step 4: In this step we tried to write data and read data simultaneously. The address at which we trying to write data is the same as the previous location. Thus when we read data from the same location simultaneously we get the updated value not the previous value which was there in the location.

```
write_data = 32'h0f0f0f0f;
read_sel_1 = 5'b00011;
read_sel_2 = 5'b00100;
write_address = 5'b00011;
RegWrite = 1;
```

Description of control logic for datapath :-

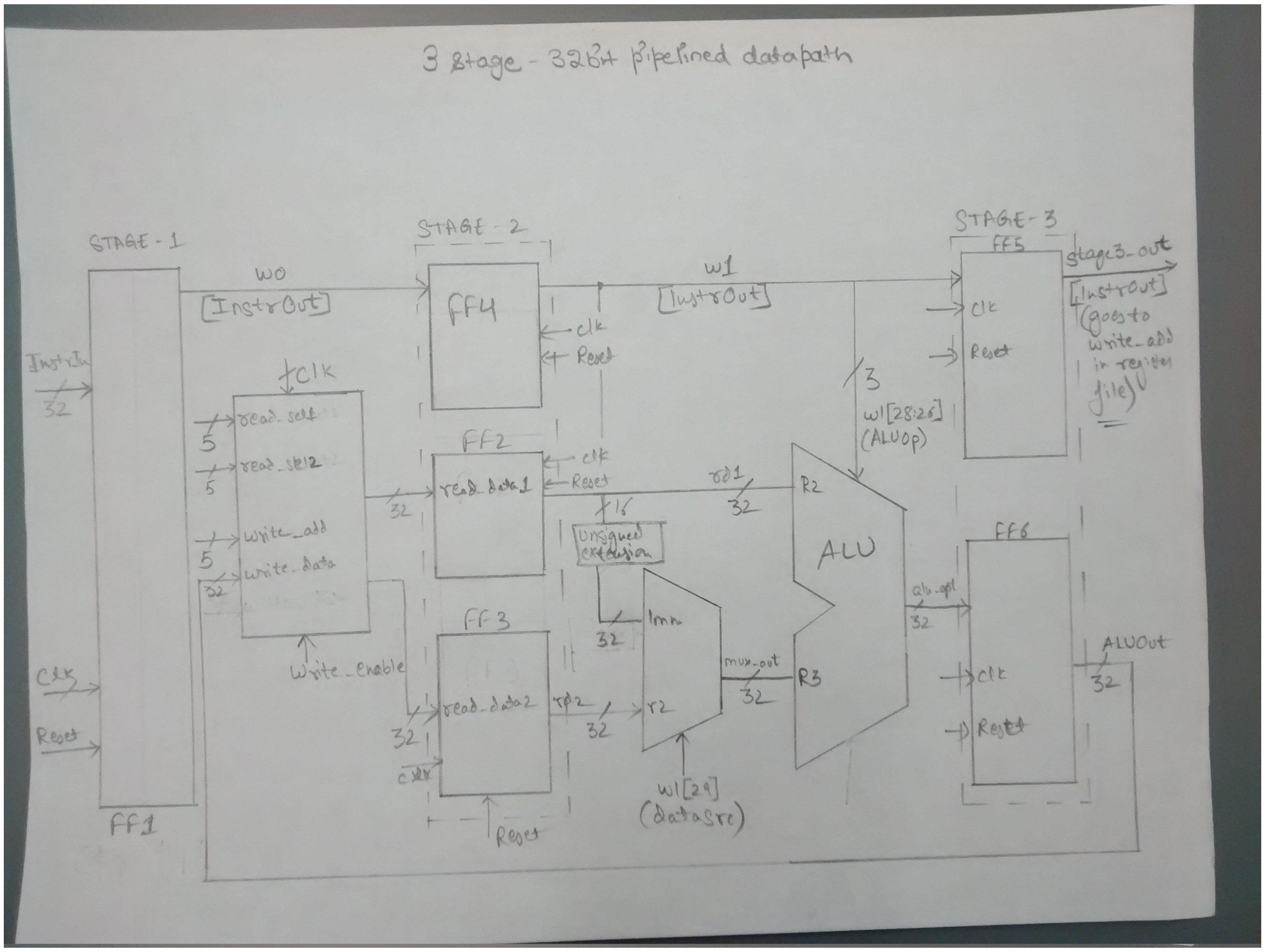
In the stage 1 the instruction is fetched and after one clock cycle it was passed to stage 2. After stage 2 of pipeline we extracted the 30th bit of instruction to determine whether the instruction is R type of I type. We used unsigned extension to convert Immediate value to 32-bits. In the same stage we extracted the bits 29th to 27th to determine the type of operation to be performed by the ALU. The output of the ALU and the whole instruction was passed through stage 3. The instruction passed was used to extract the address of the destination register whereas the ALU output served as the data to be written to the destination register. This output was written to the register file by enabling the write_enable bit.

One instruction took 3 clock cycles to generate the output but the incoming instructions didn't need to wait for all the 3 cycles because of the 3 stage pipeline. Hence each new instruction could be fetched at positive edge of clock cycle.

Anubhav Gupta
U80332699

Abhishek Gaur
U42265597

Block Diagram


3 stage - 32bit pipelined datapath
STAGE - 1
STAGE - 2
STAGE - 3
InstrIn
32
Clk
Reset
FF1
w0
[InstrOut]
clk
read_sel1
5
read_sel2
5
write_add
5
write_data
32
Write_enable
FF4
clk
Reset
FF2
read_data1
32
clk
Reset
FF3
read_data2
32
clk
Reset
rd2
32
w1
[InstrOut]
16
Unsigned extension
Imm
32
r2
32
w1[29]
(datasrc)
mux_out
32
3
w1[28:26]
(ALUop)
rd1
32
R2
R3
ALU
alu_opt
32
FF5
Clk
Reset
stage3_out
[InstrOut]
(goes to write_add in register file)
FF6
Clk
Reset
ALUOut
32