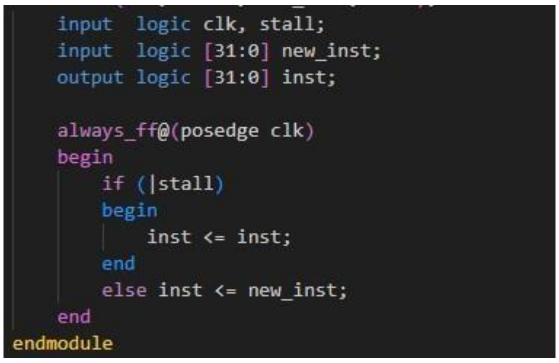
**2020-CE-42**
**Aneeza Sabir Butt**

## Stall Unit

**Overview**
In this code, the Stall Unit is in charge of identifying and managing stall conditions in a processor pipeline. Data risks, more precisely the detection of a read-after-write (RAW) hazard, cause stalls. The pipeline won't move forward until the risk has been eliminated, thanks to the Stall Unit's oversight.

**Implementation**
The `Fwd_Flush_Stall_Unit` module implements the Stall Unit. It receives a number of inputs, including read enable signals ({rd_en_1}), previous instruction ({inst1}), current instruction ({inst{), and branch prediction ({br_taken}). The outputs {stall} and {sel_rd1}/{sel_rd2} of the module decide which registers to read from and whether to stall the pipeline.

**Logic:**



```
    input  logic clk, stall;
    input  logic [31:0] new_inst;
    output logic [31:0] inst;

    always_ff@(posedge clk)
    begin
        if (|stall)
        begin
            inst <= inst;
        end
        else inst <= new_inst;
    end
endmodule
```

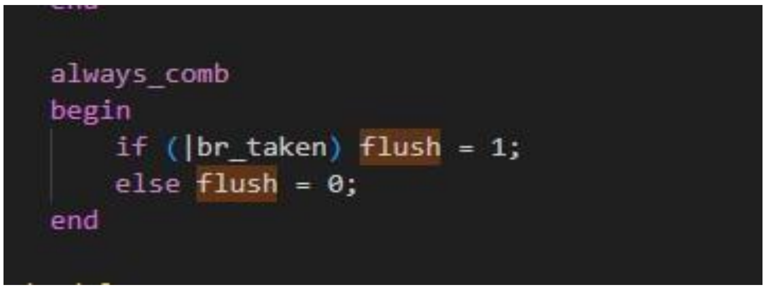
The module computes `raddr1`, `raddr2`, and `waddr` by analyzing instruction fields. It examines data hazards by comparing `raddr1` and `raddr2` with `waddr`. If a hazard is identified, it sets `stall` to 1 and ensures that both `sel_rd1` and `sel_rd2` are set to 0.

Conversely, if no hazard is found, `stall` is set to 0, and the values of `sel_rd1` and `sel_rd2` are determined by checking if `raddr1` and `raddr2` match `waddr`.

## Flush Unit

**Overview**
The Flush Unit in this code manages branch mispredictions by clearing the pipeline. Upon detecting a branch misprediction, it eliminates instructions in the pipeline that shouldn't have been executed.

**Implementation**
The Flush Unit is also implemented within the `Fwd_Flush_Stall_Unit` module. It has a single output, `flush`, which is set to 1 when a branch misprediction is detected.



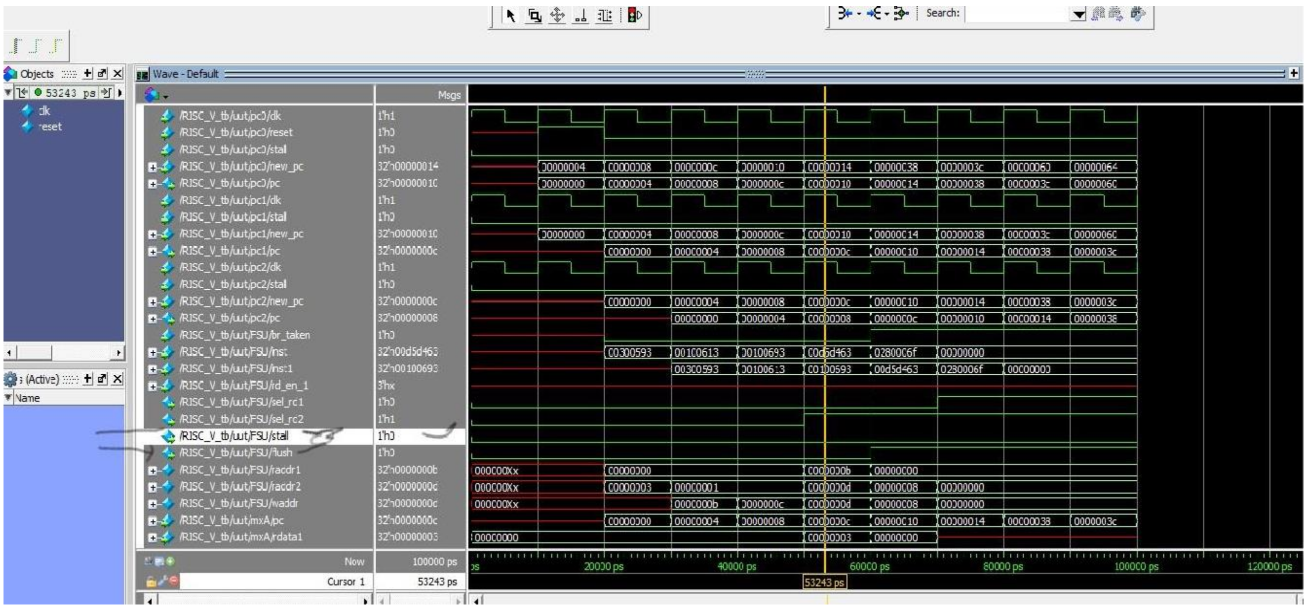
```
    always_comb
    begin
        if (|br_taken) flush = 1;
        else flush = 0;
    end
```

**Logic**
The `br_taken` input is examined by the Flush Unit to see if there has been a branch misprediction. It sets {flush} to 1 if {br_taken} is true, indicating a misprediction. If not, {flush} is configured to 0.

Together, these three units make sure that the pipeline runs smoothly, manage data risks, forward data when it can, and flush the pipeline when needed to correct branch mispredictions.

The diagram of the stall and flush in the test code question is displayed below:

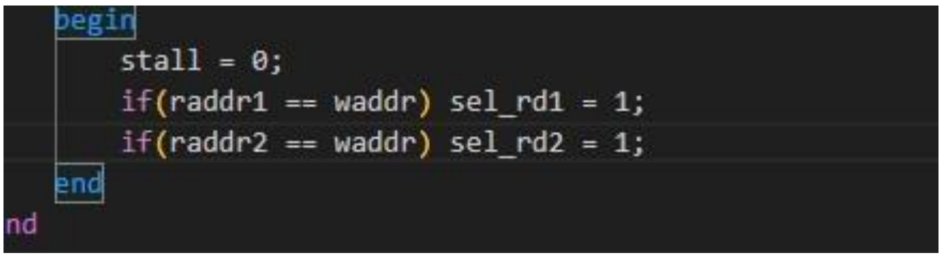
For factorial this diagram shown that there is no stall and flush occur in this code



## Forwarding Unit

**Overview**
The Forwarding Unit in this code facilitates the transfer of data from the execution stage to the decode stage within a processor pipeline. This aids in resolving data hazards and enhancing pipeline efficiency by minimizing delays.

**Implementation**
The Forwarding Unit is also implemented within the `Fwd_Flush_Stall_Unit` module. It utilizes the same inputs and outputs as the Stall Unit but serves a different purpose: forwarding data, when possible, to avoid stalls.



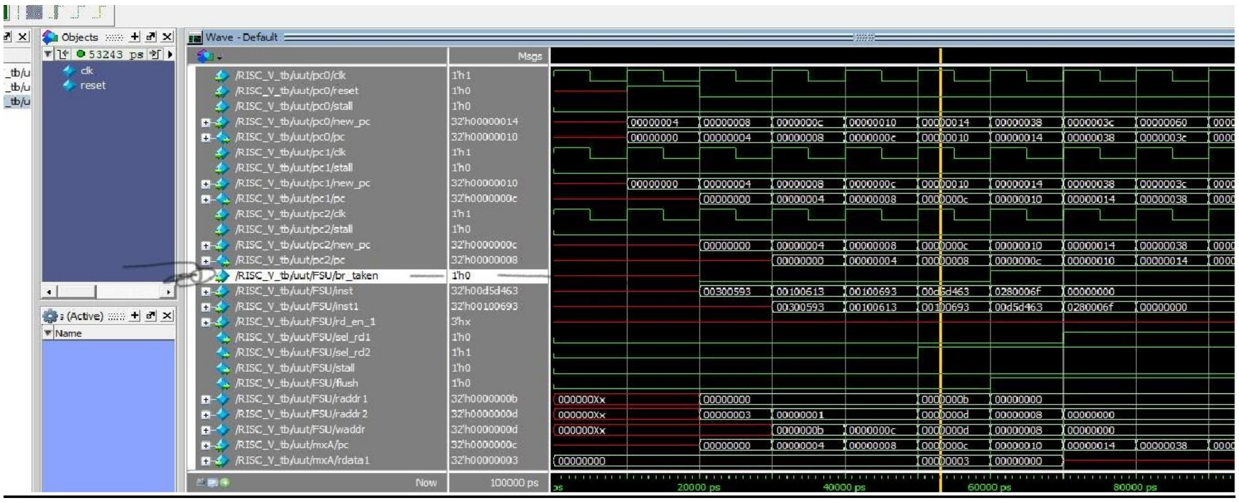
```
begin
    stall = 0;
    if(raddr1 == waddr) sel_rd1 = 1;
    if(raddr2 == waddr) sel_rd2 = 1;
end
nd
```

**Logic:**

Like the Stall Unit, the Forwarding Unit calculates `raddr1`, `raddr2`, and `waddr` based on instruction fields.

It checks for data hazards, and if no hazard is detected, it sets `sel_rd1` and `sel_rd2` to 1 for registers matching 'waddr', allowing data forwarding to occur.

Below is the shown diagram of forwarding:



| Wave - Default | Msgs |
|---|---|
| /RISC_V_tb/uut/pc0/clk | 1'h1 |
| /RISC_V_tb/uut/pc0/reset | 1'h0 |
| /RISC_V_tb/uut/pc0/stall | 1'h0 |
| /RISC_V_tb/uut/pc0/new_pc | 32'h00000014 |
| /RISC_V_tb/uut/pc0/pc | 32'h00000010 |
| /RISC_V_tb/uut/pc1/clk | 1'h1 |
| /RISC_V_tb/uut/pc1/stall | 1'h0 |
| /RISC_V_tb/uut/pc1/new_pc | 32'h00000010 |
| /RISC_V_tb/uut/pc1/pc | 32'h0000000c |
| /RISC_V_tb/uut/pc2/clk | 1'h1 |
| /RISC_V_tb/uut/pc2/stall | 1'h0 |
| /RISC_V_tb/uut/pc2/new_pc | 32'h0000000c |
| /RISC_V_tb/uut/pc2/pc | 32'h00000008 |
| /RISC_V_tb/uut/FSU/br_taken | 1'h0 |
| /RISC_V_tb/uut/FSU/inst | 32'h00d5d463 |
| /RISC_V_tb/uut/FSU/inst1 | 32'h00100693 |
| /RISC_V_tb/uut/FSU/rd_en_1 | 3'hx |
| /RISC_V_tb/uut/FSU/sel_rd1 | 1'h0 |
| /RISC_V_tb/uut/FSU/sel_rd2 | 1'h1 |
| /RISC_V_tb/uut/FSU/stall | 1'h0 |
| /RISC_V_tb/uut/FSU/flush | 1'h0 |
| /RISC_V_tb/uut/FSU/raddr1 | 32'h0000000b |
| /RISC_V_tb/uut/FSU/raddr2 | 32'h0000000d |
| /RISC_V_tb/uut/FSU/waddr | 32'h0000000d |
| /RISC_V_tb/uut/mxA/pc | 32'h0000000c |
| /RISC_V_tb/uut/mxA/rdata1 | 32'h00000003 |
| Now | 100000 ps |