Stall Unit

Overview;

The Stall Unit in this code is responsible for detecting and handling stall conditions in a processor pipeline. Stalls occur when there are data hazards, specifically when a read-after-write (RAW) hazard is detected. The Stall Unit ensures that the pipeline does not progress until the hazard is resolved. Implementation

The Stall Unit is implemented in the `Fwd_Flush_Stall_Unit` module. It takes various inputs related to branch prediction (`br_taken`), current instruction (`inst`), previous instruction (`inst1`), and read enable signals (`rd_en_1`). The module provides outputs `stall` and `sel_rd1`/`sel_rd2`, which determine whether to stall the pipeline and which registers to read from.

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 calculates `raddr1`, `raddr2`, and `waddr` based on instruction fields.It checks for data hazards by comparing `raddr1` and `raddr2` with `waddr`. If a hazard is detected, it sets `stall` to 1 and ensures that `sel_rd1` and `sel_rd2` are both 0.

- If no hazard is detected, `stall` is set to 0, and `sel_rd1` and `sel_rd2` are determined based on `raddr1` and `raddr2` matching `waddr`.

Flush Unit

Overview

The Flush Unit in this code is responsible for handling branch mispredictions by flushing the pipeline. When a branch misprediction is detected, it discards instructions in the pipeline that should not 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
```

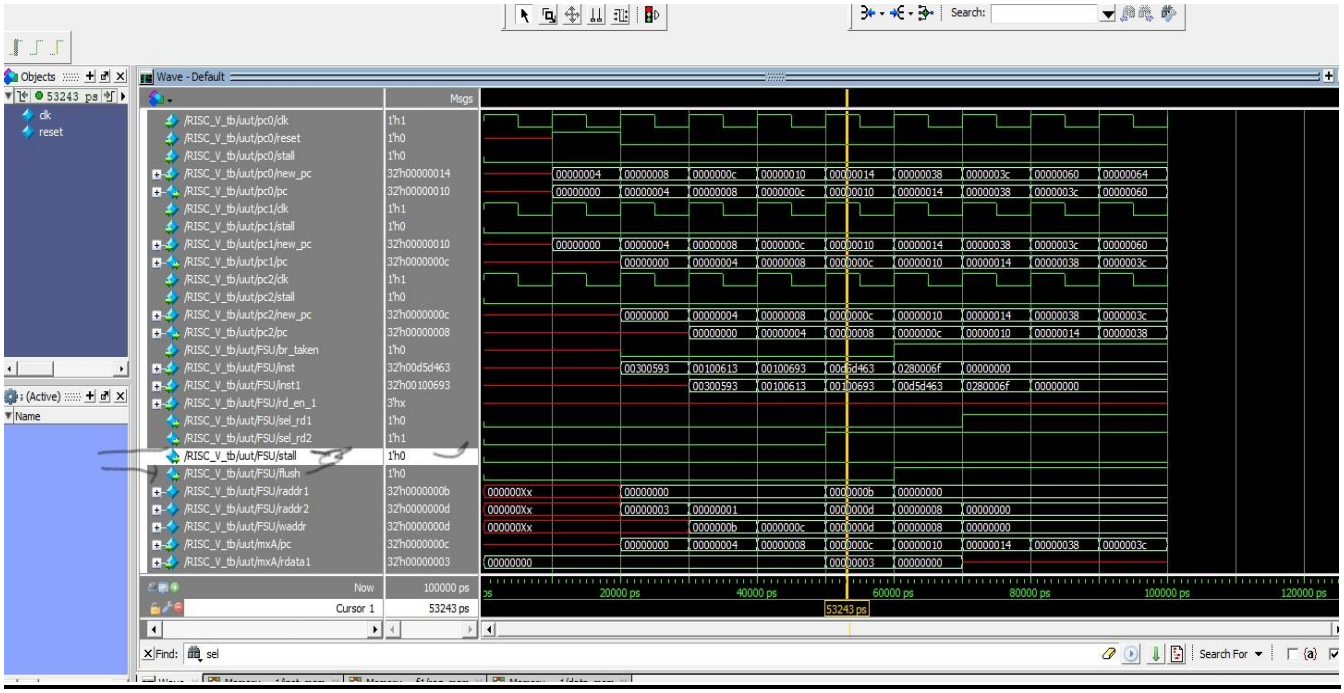
Logic

The Flush Unit checks the `br_taken` input to determine if a branch misprediction has occurred. If `br_taken` is true, indicating a misprediction, it sets `flush` to 1. Otherwise, `flush` is set to 0.

These three units work together to ensure proper pipeline operation, handle data hazards, forward data when possible, and respond to branch mispredictions by flushing the pipeline when necessary.

Below is the shown diagram of stall and flush in test code questasim

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 is responsible for forwarding data from the execution stage to the decode stage in a processor pipeline. This helps in resolving data hazards and improving pipeline performance by reducing stalls.

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.

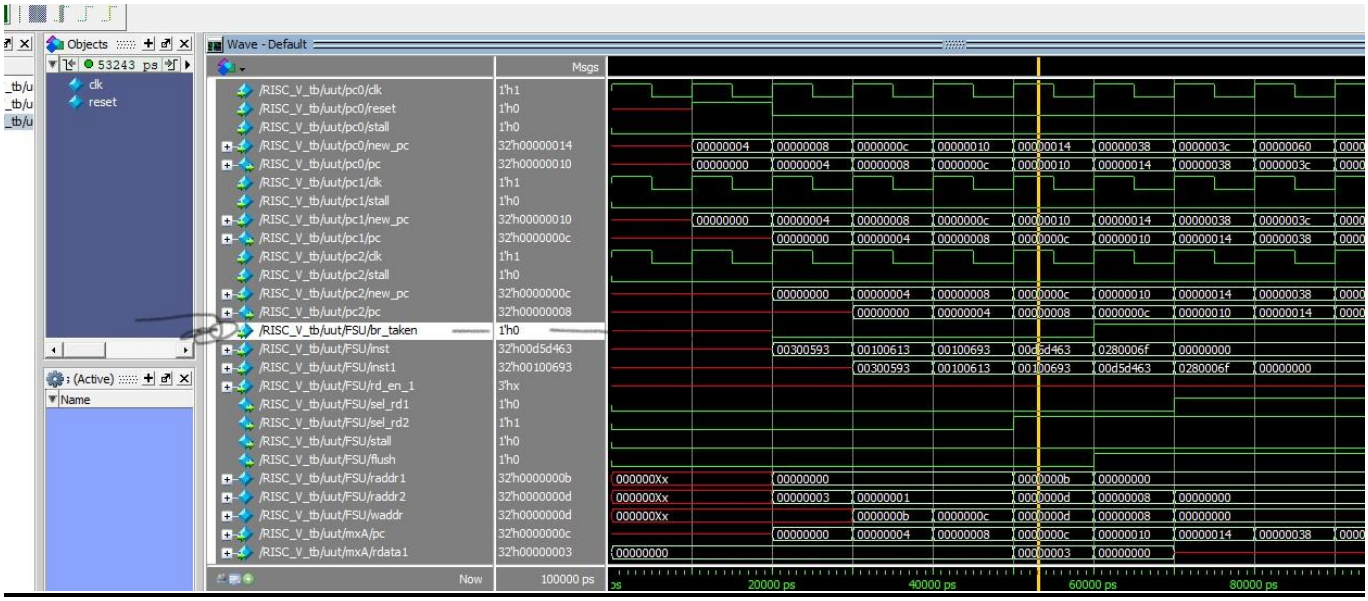
Logic

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

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:


Objects
53243 ps
clk
reset
Wave - Default
Msgs
/RISC_V_tb/uut/pc0/clk
/RISC_V_tb/uut/pc0/reset
/RISC_V_tb/uut/pc0/stall
/RISC_V_tb/uut/pc0/new_pc
/RISC_V_tb/uut/pc0/pc
/RISC_V_tb/uut/pc1/clk
/RISC_V_tb/uut/pc1/stall
/RISC_V_tb/uut/pc1/new_pc
/RISC_V_tb/uut/pc1/pc
/RISC_V_tb/uut/pc2/clk
/RISC_V_tb/uut/pc2/stall
/RISC_V_tb/uut/pc2/new_pc
/RISC_V_tb/uut/pc2/pc
/RISC_V_tb/uut/FSU/br_taken
/RISC_V_tb/uut/FSU/inst
/RISC_V_tb/uut/FSU/inst1
/RISC_V_tb/uut/FSU/rd_en_1
/RISC_V_tb/uut/FSU/sel_rd1
/RISC_V_tb/uut/FSU/sel_rd2
/RISC_V_tb/uut/FSU/stall
/RISC_V_tb/uut/FSU/flush
/RISC_V_tb/uut/FSU/raddr1
/RISC_V_tb/uut/FSU/raddr2
/RISC_V_tb/uut/FSU/waddr
/RISC_V_tb/uut/mxA/pc
/RISC_V_tb/uut/mxA/rdata1
(Active)
Name
Now
100000 ps
20000 ps
40000 ps
60000 ps
80000 ps