Team Members:
Saveetha Venkatesan (SXV200028), Vishnuvaradhan Moganarengam (VXM210090)

EEDG6302: Microprocessor and Embedded Systems

Wednesday Lab Report

Part_4 Task: Designing and simulating Data Hazard Stall (DHS) and Branch Detect module of the MCU.

Week 4 Summary:

The goal of this week's task is to verify the Verilog Data Hazard Stalls (DHS) and Branch Detect modules individually.

We have written Verilog module for Data Hazard Stall for MCU. As given in the MCU module Top view, it takes Address bit A, Address bit B, Destination Address, and MUX_A, MUX_B select bits, Read_Write signal as input and generates DHS value as output. DHS is computed by performing 'OR' operation between HA and HB which are the registers for storing the intermediate values. The values are calculated by performing series of operations such as OR operation between DA bits etc.

We have written Verilog module for Branch Detection for MCU. As given in description it takes BS (MC from MUXC) as input and generates Branch Detect as output. Intially Br_detect is assigned with value 0. Then Bitwise OR is performed with two bits of BS and then Negation operation is performed. The Output of this operation is given to Br_detect.

Problems Encountered During Design:

- We had to check with other modules for the exact input and output dependent names are similar.

1. What is the difference between active low and active high control signals?

When a digital circuit signal is described as "active low," it means that the signal will perform its intended function when the logic level of the signal is zero. To activate an active-low pin, it is necessary to connect the pin to ground to "pull" it LOW. Conversely, when using an active-high pin, the HIGH voltage (usually 3.3V or 5V) is connected to the pin. When the input signal is in the low state, the function becomes active and is executed, whereas it becomes inactive and completed when the input signal is in the high state. If there is a chip controlling the output, the connector will include a "chip enable" pin labelled as CE.

Team Members:
Saveetha Venkatesan (SXV200028), Vishnuvaradhan Moganarengam (VXM210090)

2. What is the difference between positive-edge triggered and negative-edge triggered control signals?

Positive edge triggered and negative edge triggered are two types of clock signal triggering mechanisms used in digital systems.

Positive edge triggered describes a system that updates its circuit state or samples data when the clock signal's rising edge occurs. The clock signal usually takes the form of a square wave, with a high voltage level representing the active state (logic level 1). Upon the transition from low to high voltage level of the clock signal, the circuit updates its state or samples data according to the input values at that moment. Positive edge triggering is commonly used in synchronous digital systems that utilize clock signals to synchronize the operation of multiple circuits.

Negative edge triggered refers to a system that updates its circuit state or samples data when the clock signal's falling edge occurs. The clock signal transitions from a high to a low voltage level, and the circuit updates its state or samples data according to the input values at that moment. Negative edge triggering is less frequent than positive edge triggering but can be beneficial in cases where a circuit's propagation delay is asymmetrical, and a negative edge-triggered flip-flop can offer more precise timing.

3. What is the difference between a latch and a flip-flop?

Feature	**Latch**	**Flip-Flop**
Triggering	Level-sensitive	Edge-sensitive
Output	Transparent (continuously updates)	Latches output on clock edge
Timing	Can have timing violations	Minimizes timing violations
Usage	Used for simple storage and short delays	Used for synchronization and control of system timing
Control	Controlled by enable signal	Controlled by clock signal
Types	SR latch, D latch, JK latch	SR flip-flop, D flip-flop, JK flip-flop, T flip-flop

4. Why do all registers need reset signal? When is reset usually applied to the microprocessors? Explain in 2 to 3 lines

To ensure a known state and recover from any undefined behavior, all registers require a reset signal. A missing reset signal may lead to unpredictable initial states and incorrect operation of the system. During system initialization or when an error or unexpected condition arises, the microprocessor is typically reset.

The reset signal initializes the microprocessor and its peripherals to a known state and provides a way to recover from any unexpected or undefined behavior. Upon applying the reset signal, the microprocessor begins executing from its reset vector, typically located at a fixed memory address in the system.

5. Some flip-flops or latches have preset input. If it is activated, the module will be filled with all ones. Can we have "preset" signal instead of reset in the MCU? Explain in 2 to 3 lines?

It is possible to have a "preset" signal in addition to a "reset" signal in a microcontroller unit (MCU). A preset signal sets all the flip-flops or latches in the MCU to a specific state, typically all ones, when it is activated. This can be useful in certain applications where it is desirable to initialize the MCU to a known state before normal operation begins. However, the presence of a preset signal will depend on the specific design of the MCU and its intended use.

6. Imagine there is no hardware or software solution for hazards in the MCU. Explain why running following codes leads to an incorrect result after entering the pipeline?

		1	2	3	4	5	6
MOVA R1, R5	R1 ← R5	IF	DOF	EX	WB		
ADD R2, R1, R6	R2 ← R1 + R6		IF	DOF	EX	WB	
ADD R3, R1, R2	R3 ← R1 + R2			IF	DOF	EX	WB

At first cycle, the instruction is fetched and decoded in second cycle, during third cycle it executes the instruction and only at fourth cycle the R5 value is updated in R1. But, at fourth cycle the second instruction is executed, that is, values of R1 and R6 are added, here R1 value isnt updated, it uses the old value. Also, at fourth cycle, while third instruction is decoded, it fetches the updated R1 value from first instruction, but will use old value of R2, since R2 is not WB for second instruction. Therefore, the output results will be incorrect.

Team Members:
Saveetha Venkatesan (SXV200028), Vishnuvaradhan Moganarengam (VXM210090)

7. Another hardware solution for data hazard is called "data forwarding". Explain it briefly in 3 to 5 lines.

Data forwarding, also called bypassing, is a hardware solution that resolves data hazards in digital circuits. When an instruction cannot execute due to the required data not being available in a register, a stall occurs in the pipeline. With data forwarding, the data is forwarded directly from the output of one functional unit to the input of another functional unit, bypassing the register where the data would usually be stored. This allows the subsequent instruction to access the required data before it is written to the register, avoiding the stall in the pipeline and improving the overall performance of the system.