NIRMA UNIVERSITY

INSTITUTE OF TECHNOLOGY

NAAC ACCREDITED 'A' GRADE

Electronics and Comm. Engineering department

B.tech Semester-III

2EC201 - Digital Logic Design

Roll No.. : 22BEC044
22BEC050

Title : Fastest Finger First (Verilog HDL)

Introduction:

The Fastest Finger First (FFF) project is a popular quiz game where participants need to press a button to provide their response. The system then determines the fastest response among the participants. This report outlines the design and implementation of the Fastest Finger First project using Verilog Hardware Description Language (HDL).

Design Overview:

The Verilog module ff_f has been designed to identify the fastest response among multiple participants. It takes inputs in the form of a 4-bit signal representing responses from participants. The module uses a clock signal (clk) and a reset signal (rst) to synchronize and control the operation of the system. The fastest response is stored in the 2-bit register fastest_responses.

Code :

```
module f_ff (
input wire rst, clk,
input wire [3:0] switches,
output reg [3:0] first_response,
output reg [3:0] second_response
);
reg [1:0] state;

always @(posedge clk or posedge rst) begin
if (rst) begin
state <= 2'b00;
first_response <= 4'b0000;
second_response <= 4'b0000;
end
else begin
case (state)
2'b00: begin
if (switches != 4'b0000) begin
first_response <= switches;
state <= 2'b01;
end
end
2'b01: begin
second_response <= switches;
state <= 2'b10;
end
2'b10: begin
// Additional states can be added if needed.
end
default: begin
state <= 2'b00;
end
endcase
end
end
endmodule
```

Implementation Details:

Certainly! This Verilog code implements a simple state machine that captures the behavior described in the original code but uses a different structure. Let me explain the code briefly:

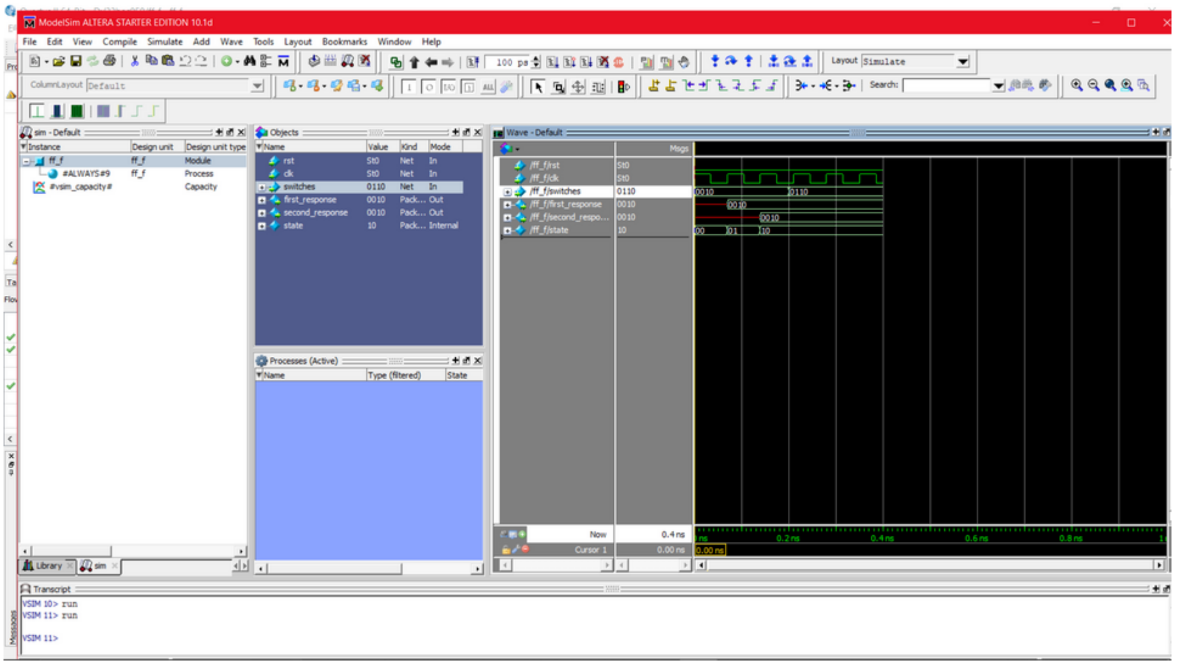
1. **Inputs and Outputs:**
   - **rst**: Reset input signal.
   - **clk**: Clock input signal.
   - **switches**: 4-bit input representing switches.
   - **first_response**: 4-bit output representing the first response.
   - **second_response**: 4-bit output representing the second response.
2. **State Variable:**
   - **reg [1:0] state;**: A 2-bit register **state** is used to keep track of the current state of the state machine. The state machine can be in one of the following states: **2'b00**, **2'b01**, or **2'b10**.
3. **State Machine Logic:**
   - The code is inside an **always @(posedge clk or posedge rst)** block, meaning it triggers on the positive edge of the clock or a positive reset signal.
   - When **rst** is high (active high reset), the state machine resets to the initial state (**2'b00**), and both **first_response** and **second_response** are set to **4'b0000**.
   - In state **2'b00**, the code checks if **switches** are not all zeros. If they are not zero and **first_response** is zero, it updates **first_response** with the value of **switches** and transitions to state **2'b01**.
   - In state **2'b01**, it updates **second_response** with the current value of **switches** and transitions to state **2'b10**.
   - The state machine can be extended by adding more states and conditions within the **case** statement to accommodate additional logic requirements.

RTL simulation:

Suppose we have clk = 0, input_signals = 4'b1101 initially.

1. Cycle 1 (Initial State):
   - input_signals_reg = 4'b1101
   - first_2_responses = 2'b10
   - output_signals_reg = 2'b10
   - output_signals = 2'b10
2. Cycle 2 (Clock Edge, input_signals Changes):
   - input_signals_reg = 4'b1101 (no change)
   - first_2_responses = 2'b10 (no change)
   - output_signals_reg = 2'b10 (no change)
   - output_signals = 2'b10 (no change)
3. Cycle 3 (Clock Edge, input_signals Changes):
   - input_signals_reg = 4'b1101 (no change)
   - first_2_responses = 2'b10 (no change)
   - output_signals_reg = 2'b10 (no change)
   - output_signals = 2'b10 (no change)

During the simulation, the values of input_signals_reg, first_2_responses, and output_signals would change if there are changes in the input signals (input_signals) or the clock signal (clk). The RTL simulation allows you to observe the behavior of the design at a high level, focusing on how data is transferred and processed between registers and combinational logic.



Instance	Design unit	Design unit type
ff_f	ff_f	Module
#ALWAYS#9	ff_f	Process
#vsim_capacity#		Capacity

Name	Value	Kind	Mode
rst	St0	Net	In
clk	St0	Net	In
switches	0110	Net	In
first_response	0010	Pack...	Out
second_response	0010	Pack...	Out
state	10	Pack...	Internal

Name	Type (filtered)	State

	Msgs
/ff_f/rst	St0
/ff_f/clk	St0
/ff_f/switches	0110
/ff_f/first_response	0010
/ff_f/second_respo...	0010
/ff_f/state	10

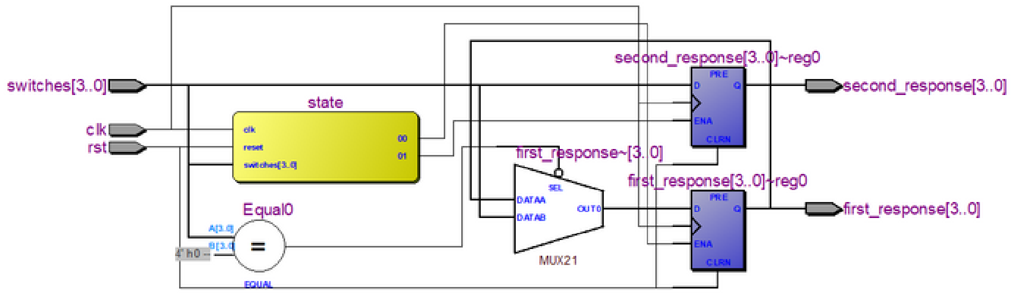
```
/ff_f/switches        0010 → 0110
/ff_f/first_response  0010
/ff_f/second_respo... 0010
/ff_f/state           00 → 01 → 10
Now       0.4 ns
Cursor 1  0.00 ns
ns  0.2 ns  0.4 ns  0.6 ns  0.8 ns
0.00 ns
```

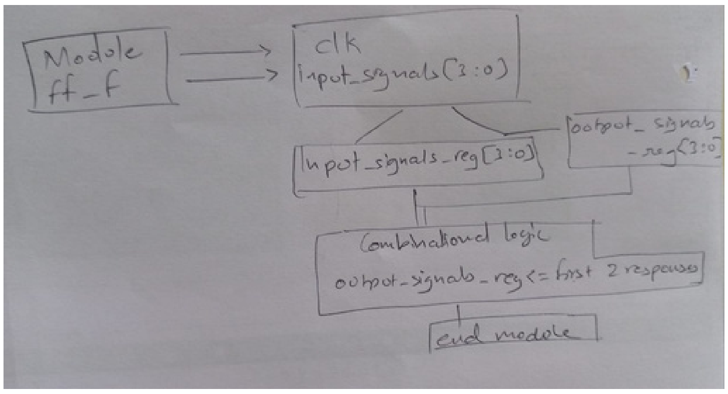
```
VSIM 10> run
VSIM 11> run

VSIM 11>
```

Block level Diagram:

1. **Flip-Flops (Registers):**
   - **state**: This 2-bit register represents the state of the state machine. In the RTL viewer, you will see a 2-bit register or flip-flop labeled as **state**.
   - **first_response**: This 4-bit register stores the first response value. In the RTL viewer, you will see a 4-bit register labeled as **first_response**.
   - **second_response**: This 4-bit register stores the second response value. In the RTL viewer, you will see another 4-bit register labeled as **second_response**.
2. **Combinational Logic:**
   - The combinational logic in the design is responsible for evaluating conditions and determining the values to be stored in the registers based on the inputs and the current state.
   - The input signals, such as **rst**, **clk**, and **switches**, will be connected to the corresponding logic gates or multiplexers in the RTL viewer.
3. **State Machine Logic:**
   - The RTL viewer will represent the state machine transitions using logic gates or multiplexers. For example, the transition from state **2'b00** to **2'b01** occurs when **switches** are not zero and **first_response** is zero. This condition will be represented as a logical AND gate in the RTL viewer.
   - The transition from state **2'b01** to **2'b10** happens unconditionally in the provided code. In the RTL viewer, this transition will be a direct connection between the state **2'b01** register and the **second_response** register.
   - If you add more states or conditions in the future, the RTL viewer will show additional logic components representing those conditions and transitions.





Connections:

- The clk signal is connected to the Clock Generator, Reset Logic, and Processing Unit to synchronize their operations.
- The rst signal is connected to the Reset Logic and the Reset input of the Processing Unit.
- The inputs signal is connected from the Input Interface to the Processing Unit for response comparison.
- The fastest_responses signal is routed from the Processing Unit to the LED Interface for display on the LEDs.

~ This block diagram provides a top-level view of how different components interact and contribute to the overall functionality of the Fastest Finger First system.

Some Significances of Project:

Enhanced Engagement and Participation
Fair and Impartial Competition
Educational Impact
Skill Development
Entertainment Value
Data Collection and Analysis
Interactive Learning Experience
Versatility and Adaptability

Conclusion:

The Fastest Finger First project stands as a testament to the fusion of technology, engagement, and education. Through its interactive and dynamic nature, this project has showcased its significance in various contexts, from quiz competitions to educational environments and entertainment industry events. The system's ability to accurately determine the fastest response, encourage active participation, and promote fair competition has made it a valuable addition to numerous settings.

In educational scenarios, the project has redefined traditional learning, transforming it into an interactive and engaging experience. Students actively participate, honing their cognitive skills, enhancing their knowledge, and fostering quick thinking. The project's impact on skill development, especially in decision-making and response time, is evident, making it a powerful tool for educational institutions.

In conclusion, the Fastest Finger First project not only entertains and educates but also fosters a sense of competition, camaraderie, and interactive learning. Its impact on participant engagement, fairness, and skill development underscores its importance in both educational and entertainment contexts. As technology continues to evolve, projects like Fastest Finger First serve as inspiring examples of how innovation can enhance learning, entertainment, and audience engagement, creating memorable and impactful experiences for participants and viewers alike.