Digital Design Report

Dhawal - EE24BTECH11015
April 11, 2025

1 Question 1: 4-to-2 Priority Encoder

1.1 Approach

The priority encoder checks inputs from highest to lowest priority:

- Uses a series of `if-else` statements to implement priority
- Highest priority input `in[3]` is checked first
- If none are active, `valid` is set to 0
- Output encoding:
  - `in[3]` → `11`
  - `in[2]` → `10`
  - `in[1]` → `01`
  - `in[0]` → `00`

1.2 Verilog Code

```
module priority_encoder_4to2 (
    input [3:0] in,
    output reg [1:0] out,
    output reg valid
);

always @(*) begin
    if (in[3]) begin
        out = 2'b11;
        valid = 1'b1;
    end
    else if (in[2]) begin
        out = 2'b10;
        valid = 1'b1;
    end
    else if (in[1]) begin
        out = 2'b01;
        valid = 1'b1;
    end
    else if (in[0]) begin
        out = 2'b00;
        valid = 1'b1;
    end
    else begin
        out = 2'b00;
        valid = 1'b0;
    end
end

endmodule
```

2 Question 2: 4-bit Up Counter

2.1 Approach

The counter implements the following behavior:

- Asynchronous reset has highest priority (immediate effect)
- On rising clock edge when enabled, increments count
- Automatically wraps around from 15 (4'b1111) to 0
- Maintains value when not enabled
- Uses non-blocking assignments (`<=`) for sequential logic

2.2 Verilog Code

```
module counter_4bit (
    input clk,
    input reset,    // Active-high asynchronous reset
    input enable,   // Count enable
    output reg [3:0] count  // 4-bit counter output
);

// Counter logic
always @(posedge clk or posedge reset) begin
    if (reset) begin
        // Asynchronous reset (highest priority)
        count <= 4'b0000;
    end
    else if (enable) begin
        // Increment counter when enabled
        count <= count + 1'b1;
    end
    // Else: count maintains its value (implicit)
end

endmodule
```

3 Question 3: 8-bit Even Parity Generator

Approach

- Uses XOR reduction operator (`^data`)
- Combinational logic (no clock)
- Outputs 1 when odd number of 1's in input

3.1 Verilog Code

```
module even_parity_gen (
    input [7:0] data,
    output reg parity
);

always @(*) begin
    parity = ^data; // XOR reduction operator
end

endmodule
```