Counters

Dineth Mudalige

- Keep asking questions in the Zoom chat
- I will answer the questions while you implement the circuits
- Give feedback anytime
- All codes are available in the LMS as zip file

Recap on counters

- Let us recall session 3 on sequential logic circuits

1. Asynchronous counters
2. Synchronous counters

Recap on 4-bit bidirectional counter

```
1  module four_bit_synchronous_counter
2  (
3      input logic clk, rstn, up_down,
4      output logic [3:0] count
5  );
6
7      always_ff @(posedge clk) begin
8          if (!rstn) count <= 0;
9          else if (up_down == 0) begin
10             count <= count + 1'b1;
11         end else if (up_down == 1) begin
12             count <= count - 1'b1;
13         end
14     end
15
16 endmodule
```

Can you make this counter a upcounter?

4-bit up counter

```
module up_counter
(
    input logic clk, rstn,          // Clock and active-low reset
    output logic [3:0] count        // 4-bit output count
);

// Sequential logic for the 4-bit counter
  always_ff @(posedge clk) begin
    if (!rstn)
        count <= 4'b0000;           // Reset the counter to 0 when rstn is low
    else
        count <= count + 1'b1;      // Increment the counter on each clock edge
end

endmodule
```

Overflow resets the counter

Mod-10 counter

```
module mod_10_counter
(
    input logic clk, rstn,          // Clock and active-low reset
    output logic [3:0] count        // 4-bit output count (0 to 9)
);

// Sequential logic for the Mod-10 counter
  always_ff @(posedge clk) begin
    if (!rstn)
        count <= 4'b0000;           // Reset the counter to 0 when rstn is low
    else if (count == 4'd9)
        count <= 4'b0000;           // Reset the counter when it reaches 9
    else
        count <= count + 1'b1;      // Increment the counter on each clock edge
end

endmodule
```

Parameterised N-mod counter

```
module param_mod_counter #(parameter N = 10)
(
    input logic clk, rstn,          // Clock and active-low reset
    output logic [$clog2(N)-1:0] count // Output count based on modulus N
);

// Sequential logic for the N-mod counter
  always_ff @(posedge clk or negedge rstn) begin
    if (!rstn)
        count <= 0;                 // Reset the counter to 0 when rstn is low
    else if (count == N-1)
        count <= 0;                 // Reset the counter when it reaches N-1
    else
        count <= count + 1'b1;      // Increment the counter on each clock edge
  end

endmodule
```

Digital Clock

- Can you think of a way to make a digital clock using parameterized counters?

Seconds (ones place): Mod-10 (0 to 9)
Seconds (tens place): Mod-6 (0 to 5)
Minutes (ones place): Mod-10 (0 to 9)
Minutes (tens place): Mod-6 (0 to 5)

Digital Clock Implementation

```
module digital_clock
(
    input logic clk, rstn,              // Clock and active-low reset
    output logic [3:0] sec_ones,        // Seconds (ones place)
    output logic [2:0] sec_tens,        // Seconds (tens place)
    output logic [3:0] min_ones,        // Minutes (ones place)
    output logic [2:0] min_tens        // Minutes (tens place)
);

logic sec_clk, min_clk, next_min_clk;

// Instance of Mod-10 counter for seconds ones place
param_mod_counter #(.N(10)) sec_ones_counter (
    .clk(clk),
    .rstn(rstn),
    .count(sec_ones)
);

// Generate clock for tens of seconds (increments when ones place
// reaches 9)
assign sec_clk = (sec_ones == 9);

// Instance of Mod-6 counter for seconds tens place
param_mod_counter #(.N(6)) sec_tens_counter (
    .clk(sec_clk),
    .rstn(rstn),
    .count(sec_tens)
);
```

Digital Clock Implementation

```
// Generate clock for ones of minutes (increments when tens of seconds
assign min_clk = (sec_tens == 5);

// Instance of Mod-10 counter for minutes ones place
param_mod_counter #(.N(10)) min_ones_counter (
    .clk(min_clk),
    .rstn(rstn),
    .count(min_ones)
);

// Generate clock for tens of minutes (increments when ones of minutes
assign next_min_clk = (min_ones == 9);

// Instance of Mod-6 counter for minutes tens place
param_mod_counter #(.N(6)) min_tens_counter (
  .clk(next_min_clk),
    .rstn(rstn),
    .count(min_tens)
);
endmodule
```

Digital Clock Implementation

Can you extend the nested counter to implement the counting of hours?

Hours (ones place): Mod-10 (0 to 9)
Hours (tens place): Mod-3 (0 to 2, for a 24-hour clock)

Issue

What is the main problem that you see in this design if you want to measure a time interval?

Comparator to see whether a counter has reached a certain value to start the next counter

Down counters

- Comparing zeros is easy

```
module down_counter #(
    parameter int N = 10 // Default count value if not specified
)(
    input logic clk,        // Clock signal
    input logic rstn,       // Active-low reset
    output logic done,      // Output goes high when count reaches 0
    output logic [$clog2(N):0] count  // Current count value
);

    // Sequential logic for the down counter with auto-reset
    always_ff @(posedge clk or negedge rstn) begin
        if (!rstn) begin
            count <= N;
            done <= 0;
        end
        else if (count == 0) begin
            count <= N;  // Restart from N once it reaches 0
            done <= 1;
        end
        else begin
            count <= count - 1;
            done <= 0;
        end
    end

endmodule
```

Nested Down Counter

```
// Tens of seconds down counter (0-5)
down_counter #(.N(5)) tens_counter (
    .clk(sec_done),
    .rstn(rstn),
    .done(tens_done),
    .count(tens_count)
);

// Minutes down counter (0-1 for 2-minute timer)
down_counter #(.N(1)) min_counter (
    .clk(tens_done),
    .rstn(rstn),
    .done(min_done),
    .count(min_count)
);
```

Practical Scenario

- Alarm switches on when the down counter is finished. Both alarm and counting clears after pressing reset