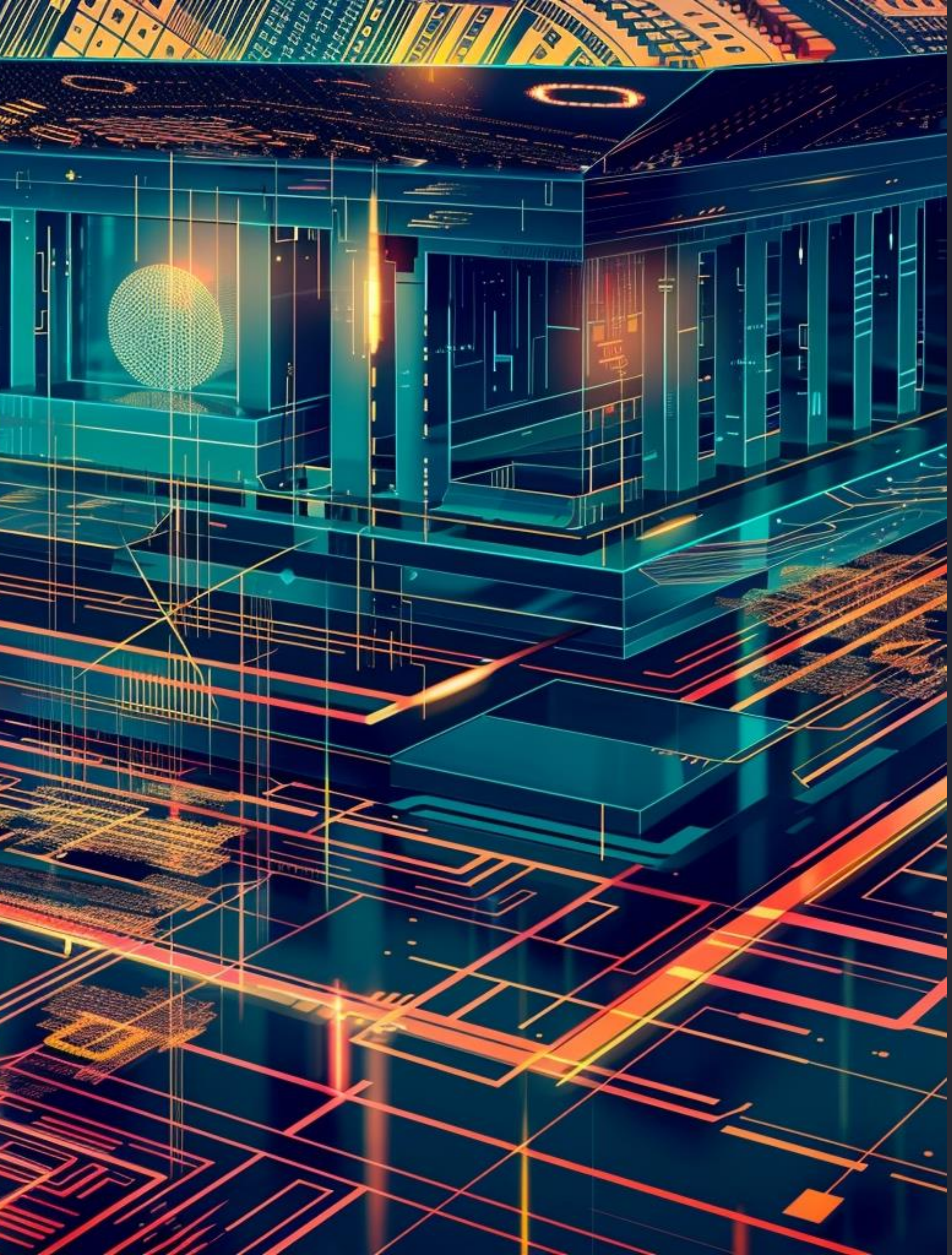


DSD Lab

Habib Ullah
habib.ullah@uetpeshawar.edu.pk

Department of Computer Systems Engineering,
University of Engineering & Technology, Peshawar, Pakistan.

Lab 5 : Implementation of an 8-bit Ring Counter

Ring Counter

- A ring counter is a type of shift register where the output of the last flip-flop is fed back to the input of the first flip-flop.
- It creates a circular pattern of bits.
- The counter is initialized with a single '1' and the rest '0's (e.g., 0001).
- On each clock cycle, the '1' shifts to the next bit position.
- The last bit is fed back to the first bit.

- 4-bit Ring Counter 0001 → 0010 → 0100 → 1000 → 0001

Clock Divider

A circuit that reduces the frequency of a clock signal by a specific factor.

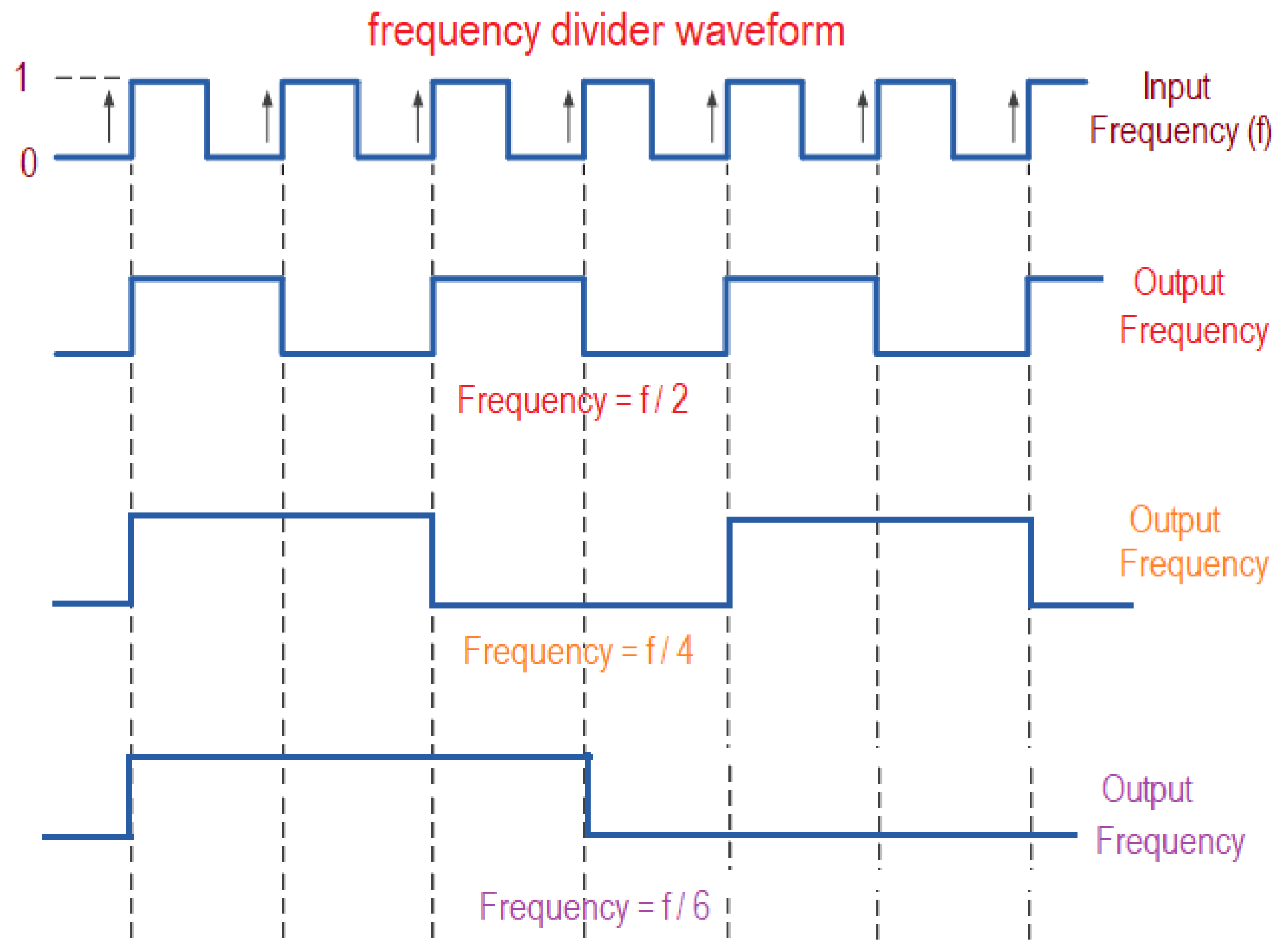
Input Frequency (Fin)

Output Frequency (Fout)

Counter Value (N)

$$N = \frac{F_{in}}{2 \times F_{out}}$$

$$N = \frac{100{,}000{,}000}{2 \times 1} = 50,\ 000{,}000$$



Tasks

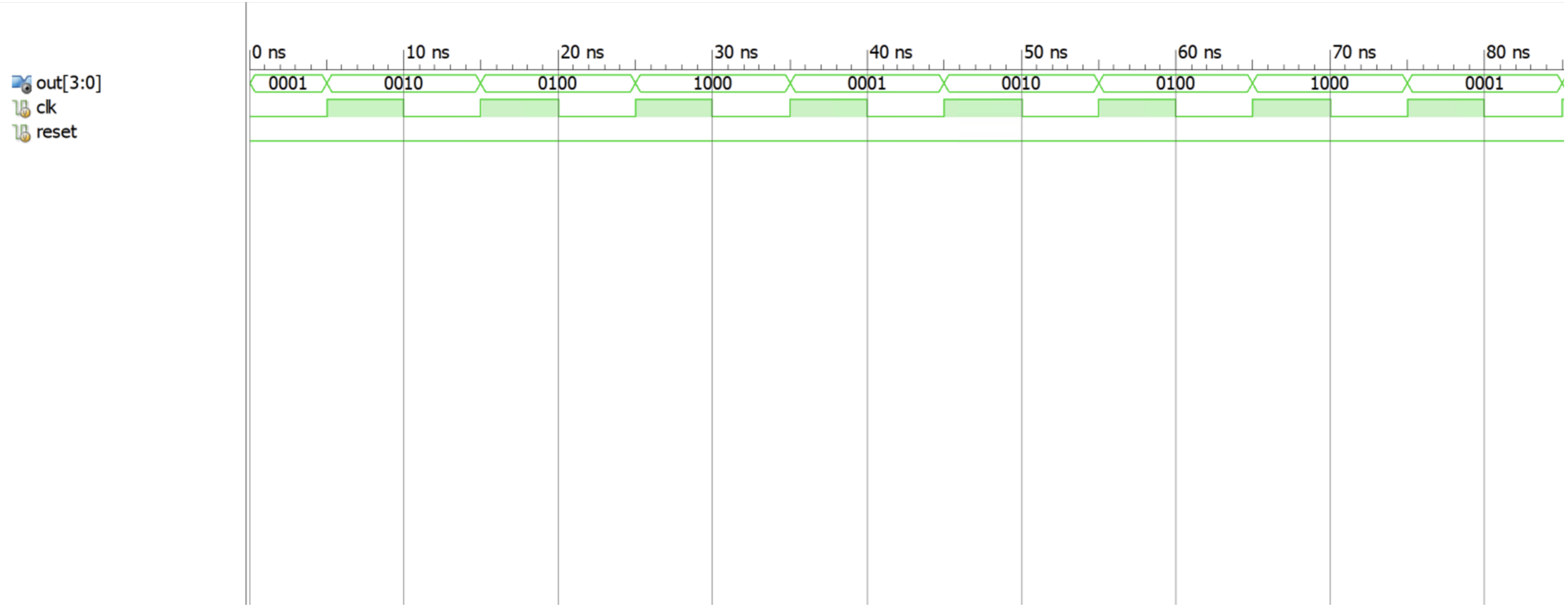
1. Design and simulate a 4-bit ring counter.
2. Design and simulate an 8-bit ring counter.
3. Design and implement a digital circuit that divides a 100 MHz clock frequency into a 1 Hz frequency. The 1 Hz clock will be used as the input clock to an 8-bit Ring Counter. The output of the Ring Counter should be connected to 8 LEDs on the Mimas V2 FPGA Board. The LEDs should turn on and off one by one from left to right with an interval of 1 second.

Task 1

```verilog
`timescale 1ns / 1ps

module ring_counter #(parameter WIDTH=4)
  (
   input clk,
   input reset,
   output reg [WIDTH-1:0] out=1
  );


  always @ (posedge clk) begin
      if (reset)
        out <= 1;
      else begin
       out <= out << 1;
       out[0] <= out[WIDTH-1];
      end
  end
endmodule
```

```verilog
`timescale 1ns / 1ps

module ring_counter_tb;

reg clk;
reg reset;

wire [3:0] out;

ring_counter #(4) uut (clk, reset, out);

always #5 clk = ~clk;

initial begin
reset = 0;
clk = 0;

#100;
$finish;

end
endmodule
```

Task 1 (Testbench)

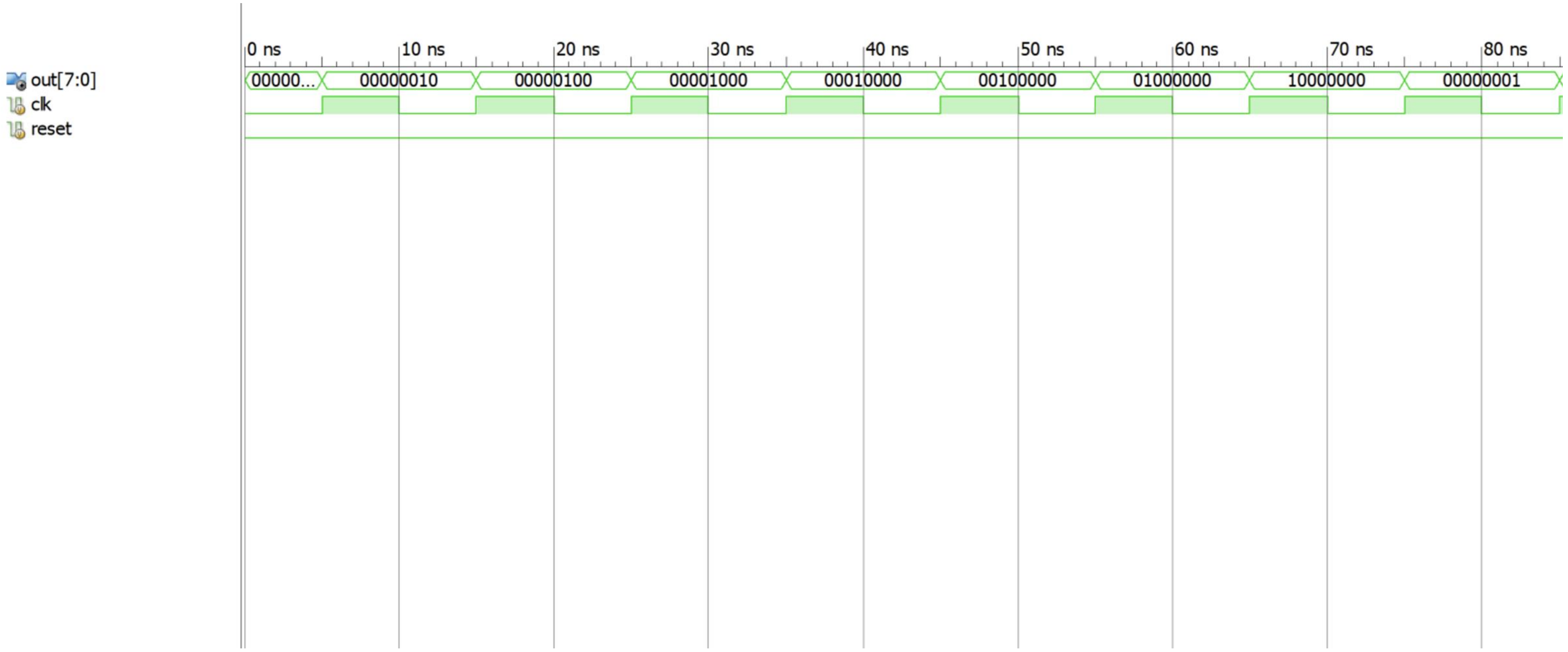


Task 2

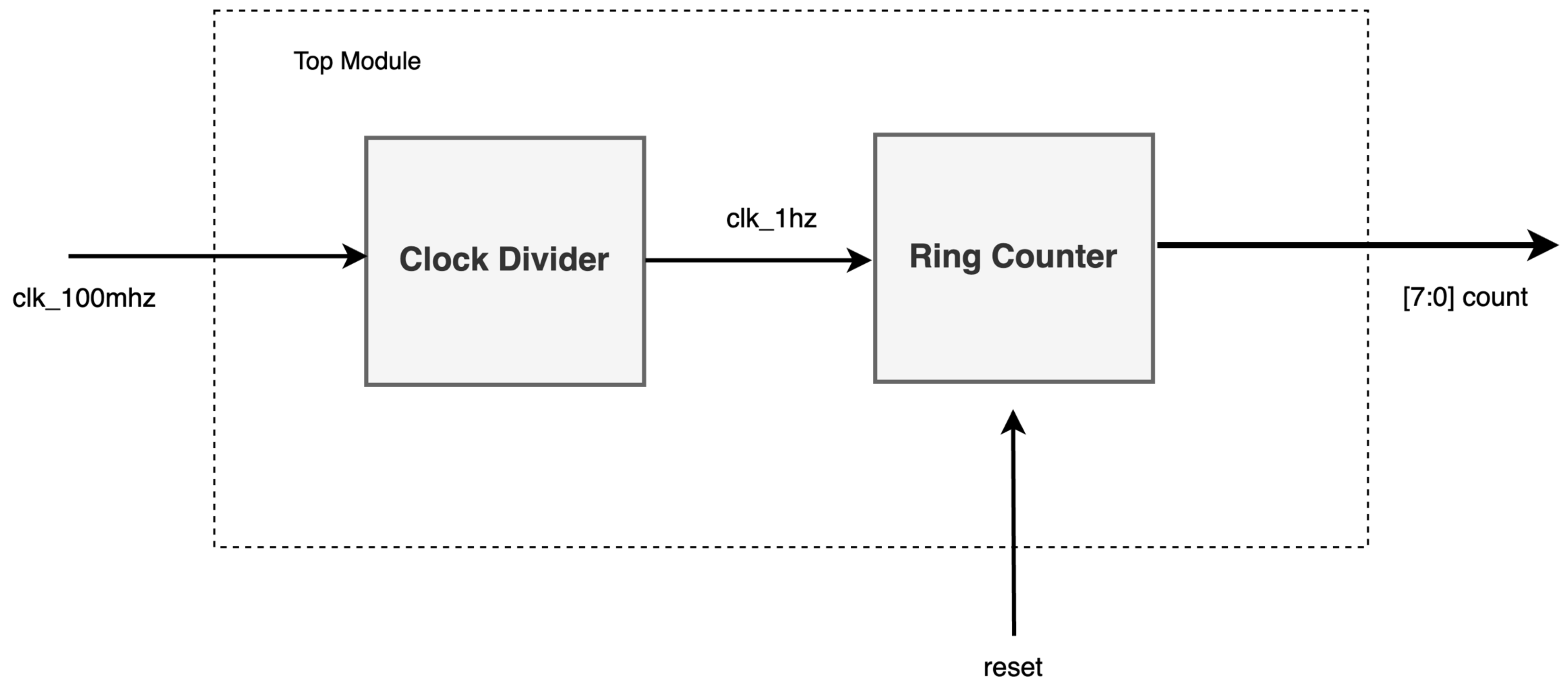
```
`timescale 1ns / 1ps

module ring_counter #(parameter WIDTH=4)
  (
   input clk,
   input reset,
   output reg [WIDTH-1:0] out=1
  );


  always @ (posedge clk) begin
      if (reset)
        out <= 1;
      else begin
       out <= out << 1;
       out[0] <= out[WIDTH-1];
      end
  end
endmodule
```

```
`timescale 1ns / 1ps

module ring_counter_tb;

reg clk;
reg reset;

wire [7:0] out;

ring_counter #(8) uut (clk, reset, out);

always #5 clk = ~clk;

initial begin
reset = 0;
clk = 0;

#100;
$finish;

end
endmodule
```

Task 2 (Testbench)



Task 3


Top Module
clk_100mhz
Clock Divider
clk_1hz
Ring Counter
[7:0] count
reset

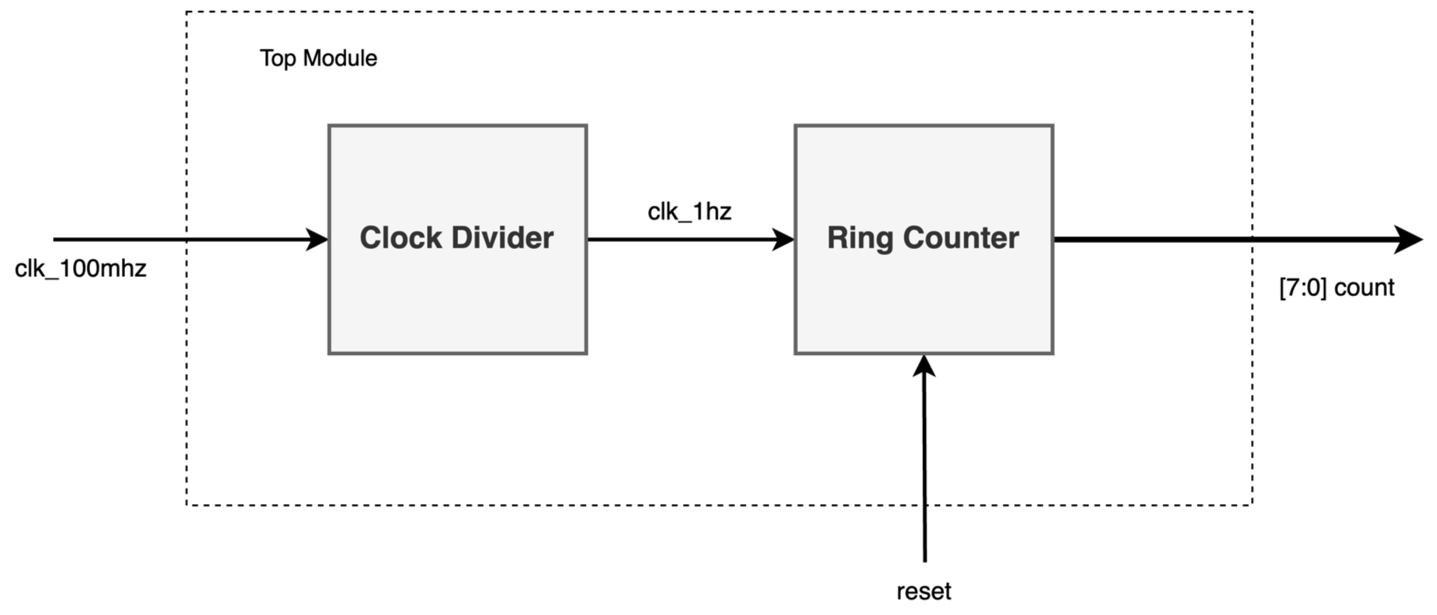
Task 3

```
`timescale 1ns / 1ps

module clock_divider (
    input clk,            // 100 MHz input clock
    output reg out_clk  // 1 Hz output clock
);

// Counter to divide 100 MHz to 1 Hz
reg [26:0] counter; // 27-bit counter

always @(posedge clk ) begin

 if (counter == 50_000_000 - 1) begin
   counter <= 0;
   out_clk <= ~out_clk;
  end else begin
     counter <= counter + 1;
  end
end

endmodule
```

```
`timescale 1ns / 1ps

module ring_counter #(parameter WIDTH=8)
  (
    input clk,
    input reset,
     output reg [WIDTH-1:0] out=1
  );


  always @ (posedge clk or posedge reset) begin
      if (reset)
        out <= 1;
      else begin
        out <= out << 1;
        out[0] <= out[WIDTH-1];
      end
  end
endmodule
```

Task 3 (Top Module)

```
`timescale 1ns / 1ps

module top(
    input clk, reset,
    output [7:0] count
    );

wire clk_1hz;

clock_divider U0(clk, clk_1hz);
ring_counter  U1(clk_1hz, ~reset, count);

endmodule
```



Task 3 (UCF)

```
NET "clk"         LOC = V10     | IOSTANDARD = LVCMOS33 | PERIOD = 100MHz ;
NET "reset"       LOC = M16     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST | PULLUP; #SW3

NET "count[7]"    LOC = P15     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;  #D1
NET "count[6]"    LOC = P16     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;  #D2
NET "count[5]"    LOC = N15     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;  #D3
NET "count[4]"    LOC = N16     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;  #D4
NET "count[3]"    LOC = U17     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;  #D5
NET "count[2]"    LOC = U18     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;  #D6
NET "count[1]"    LOC = T17     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;  #D7
NET "count[0]"    LOC = T18     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;  #D8
```