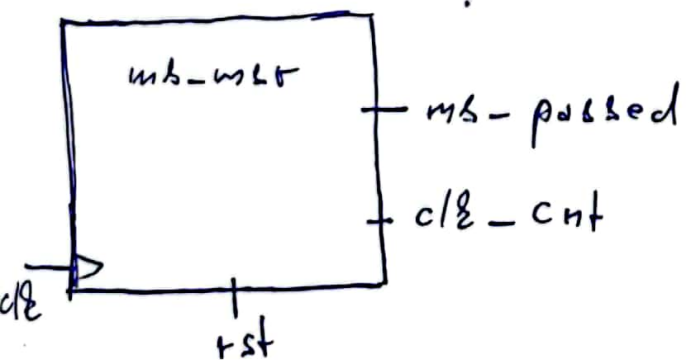
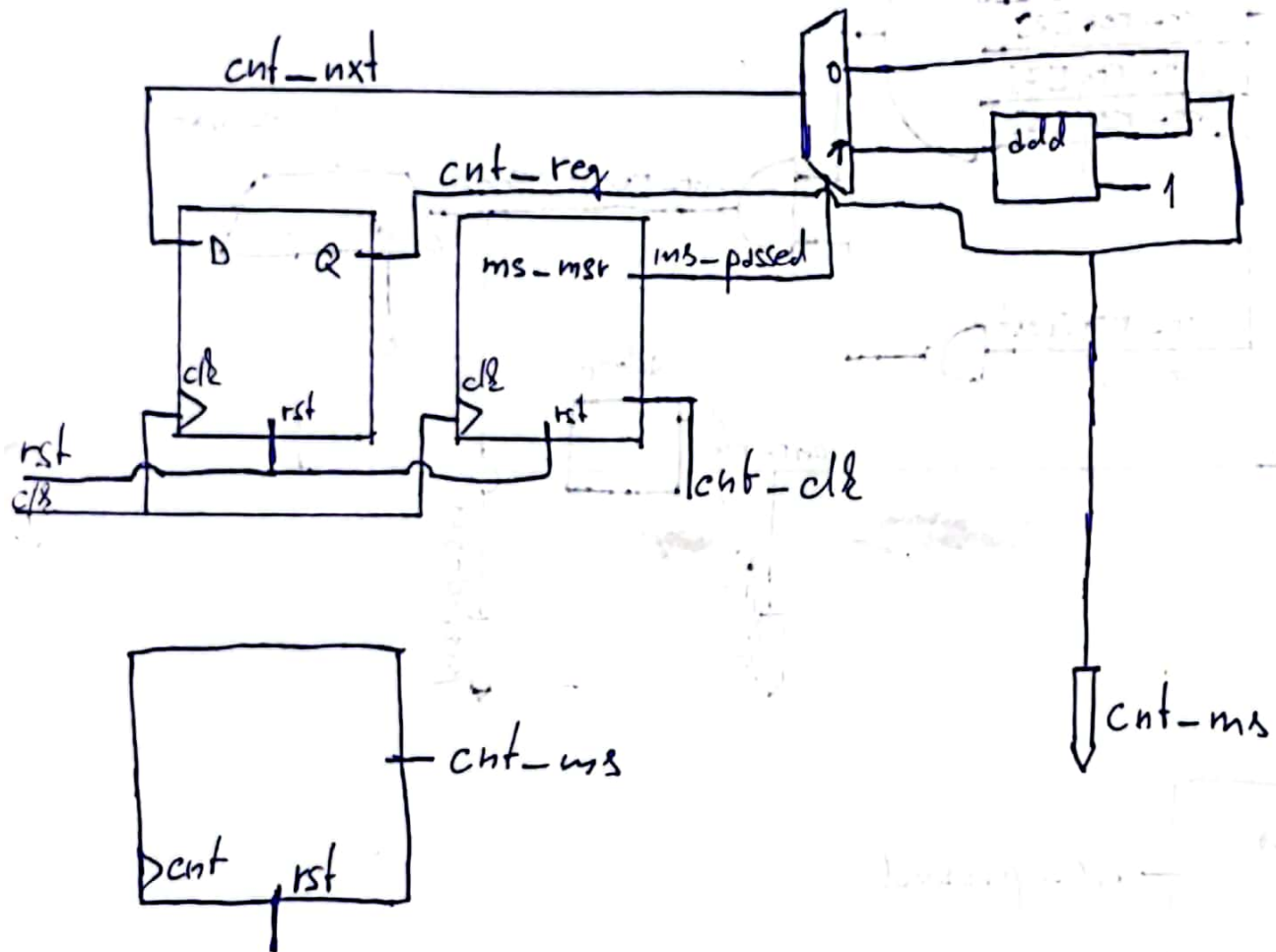


The diagram illustrates a digital circuit for a 16-bit counter. It features a D flip-flop on the left, which serves as the counter's register. The flip-flop's D input is connected to its Q output, and its clock input is labeled 'clk'. The Q output is connected to the 'cnt_reg' input of a 16-bit register block. The register block has two outputs: 'cnt_reg[12]' and 'cnt_reg[13]'. These outputs are connected to a 4-input AND gate. The AND gate's inputs are also labeled 'cnt_reg[9]', 'cnt_reg[8]', 'cnt_reg[7]', and 'cnt_reg[6]'. The output of this AND gate is connected to a 2-input AND gate. The other input of this second AND gate is the output of an inverter, which has 'cnt_reg[13]' as its input. The output of the second AND gate is connected to the 'cnt_reg = 5000' input of a 2-to-1 multiplexer. The multiplexer has two data inputs: '1' and '0'. The output of the multiplexer is connected to the 'cnt_next' input of the 16-bit register block. The register block also has a 'cnt_reg[13]' output, which is connected to an 'adder' block. The adder block has a 'cnt_reg[13]' input and a 'cnt-clk' output. The output of the adder is connected to the 'cnt_reg[13]' input of the register block. The output of the register block is connected to the 'cnt-clk' output of the adder. The output of the adder is also connected to a '1ms-pulse' output. The output of the register block is also connected to the 'cnt-clk' output of the adder.



ms_cnt



From:

0ns

To:

4,500,000ns

Get Signals

Radix ▾

🔍 🔍

100%

⏮ ⏭

↶ ↷

⏪ ⏩

⏴ ⏵

⏶ ⏷

⏸



```
1 //correct functionality for a timescale of ns/<pre>
2 module test;
3     initial begin
4         $dumpfile("dump.vcd");
5         $dumpvars(1, test);
6     end
7
8     reg clk, rst;
9     wire [35:0] cnt_ms;
10    parameter period = 200; //200ns periods correspond to 5 MHz
11
12    ms_cnt ms_cnt_1(
13        .clk(clk),
14        .rst(rst),
15        .cnt_ms(cnt_ms)
16    );
17
18    initial begin
19        rst = 1;
20        #10 rst = 0;
21    end
22
23    initial begin
24        repeat(45000)
25            #period/2 clk = ~clk;
26    end
27
28    endmodule
```

```
1 module ms_msr(
2     input clk, rst,
3     output ms_passed,
4     output [35:0] cnt_clk
5 );
6     reg [35:0] cnt_reg, cnt_nxt;
7
8     always @(*) begin
9         cnt_nxt = cnt_reg;
10    end
11
12    if(cnt_reg == 5000) begin
13        cnt_nxt = 0;
14    end
15    else begin
16        cnt_nxt = cnt_reg + 1;
17    end
18
19    always @(negedge clk or posedge rst) begin
20        if(rst == 1) begin
21            cnt_reg <= 0;
22        end
23        else begin
24            cnt_reg = cnt_nxt;
25        end
26    end
27
28    assign ms_passed = (cnt_reg == 5000);
29    assign cnt_clk = cnt_reg;
30 endmodule
31
32 module ms_cnt(
33     input clk, rst,
34     output [35:0] cnt_ms
35 );
36
37     reg [35:0] cnt_reg, cnt_nxt;
38     wire ms_passed;
39     wire [35:0] cnt_clk;
40
41     ms_msr ms_msr_1(
42         .clk(clk),
43         .rst(rst),
44         .ms_passed(ms_passed),
45         .cnt_clk(cnt_clk)
46     );
47
48     always @(*) begin
49         cnt_nxt = cnt_reg;
50    end
51    if(ms_passed) begin
52        cnt_nxt = cnt_reg + 1;
53    end
54    else begin
55        cnt_nxt = cnt_reg;
56    end
57
58    always @(negedge clk or posedge rst) begin
59        if(rst == 1) begin
60            cnt_reg <= 0;
61        end
62        else begin
63            cnt_reg = cnt_nxt;
64        end
65    end
66    assign cnt_ms = cnt_reg;
67 endmodule
```