

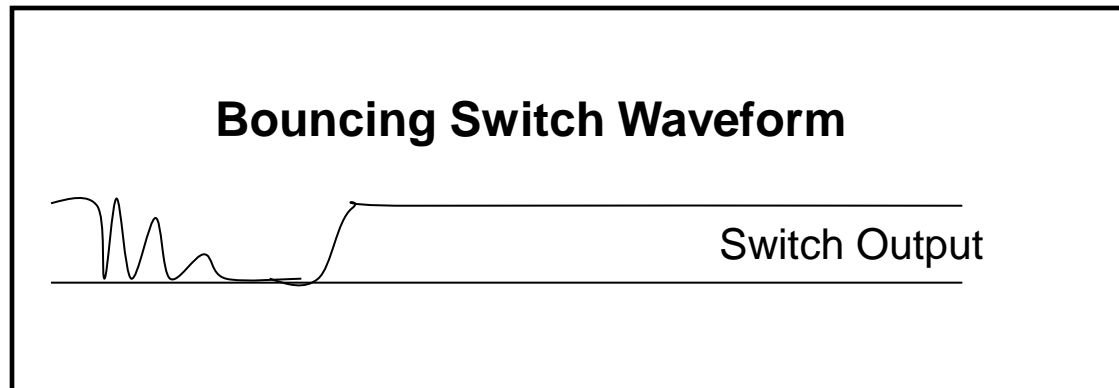
---

CSE 20221: Logic Design

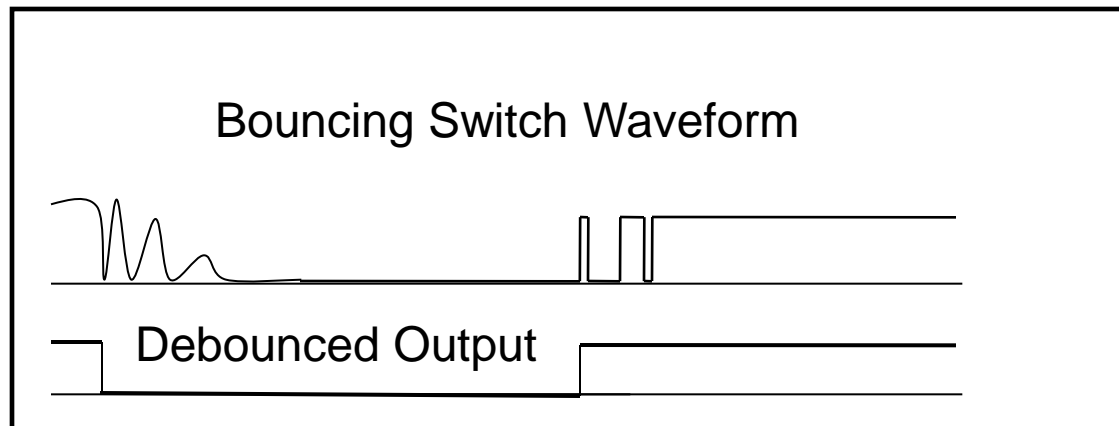
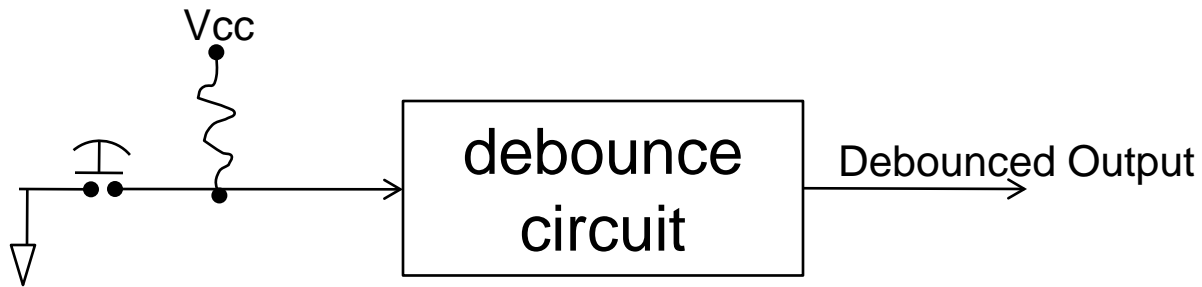
# Switch Debouncing, Handshaking

# Mechanical Switches

- Mechanical switches bounce when closing or opening.
- Multiple pulses will occur
- Counting switch closures or detecting change in the state of the switch will cause errors.



# Debouncing Circuit

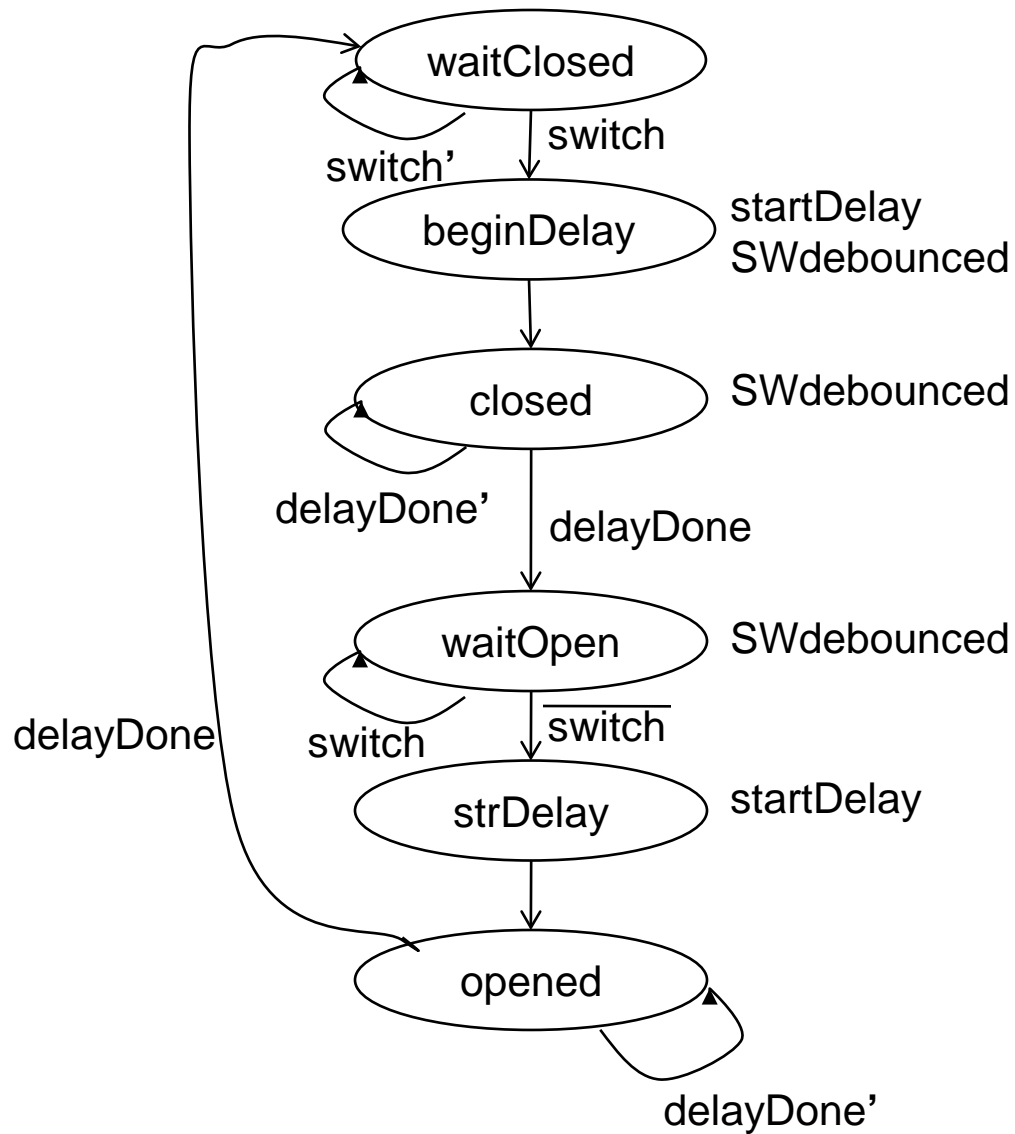


# Design of Debounce Circuit

---

- Detect change of the switch state
- Delay for a period greater than the maximum bounce time
- Bounce time varies depending on the switch
- View on Oscilloscope and do a statistical analysis
- Try 10 to 50 msec delay

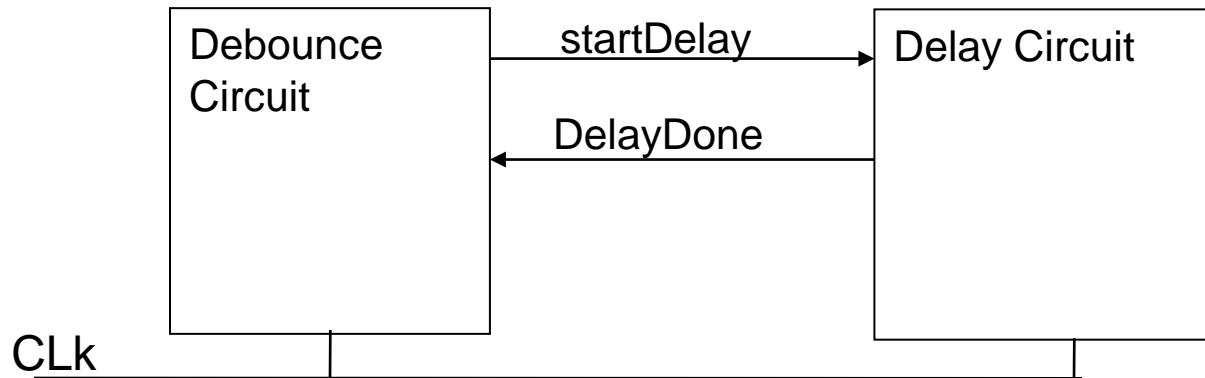
# Debounce – State Diagram



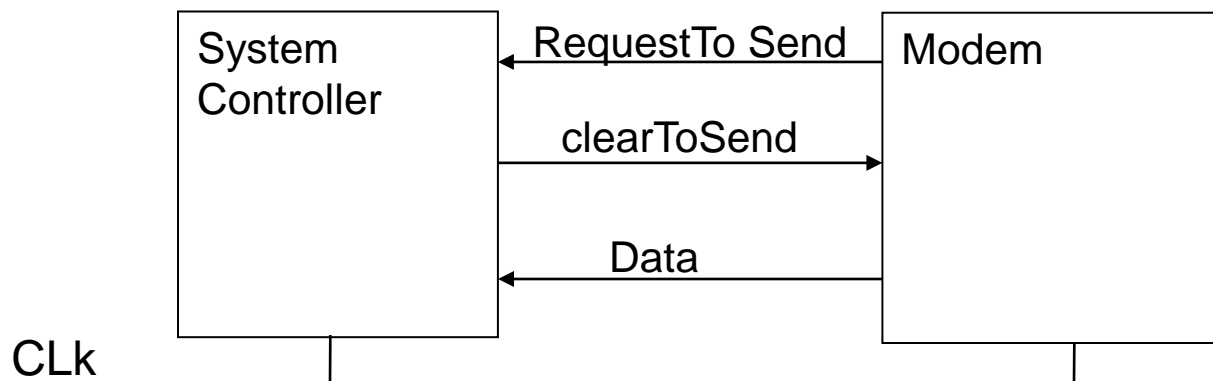
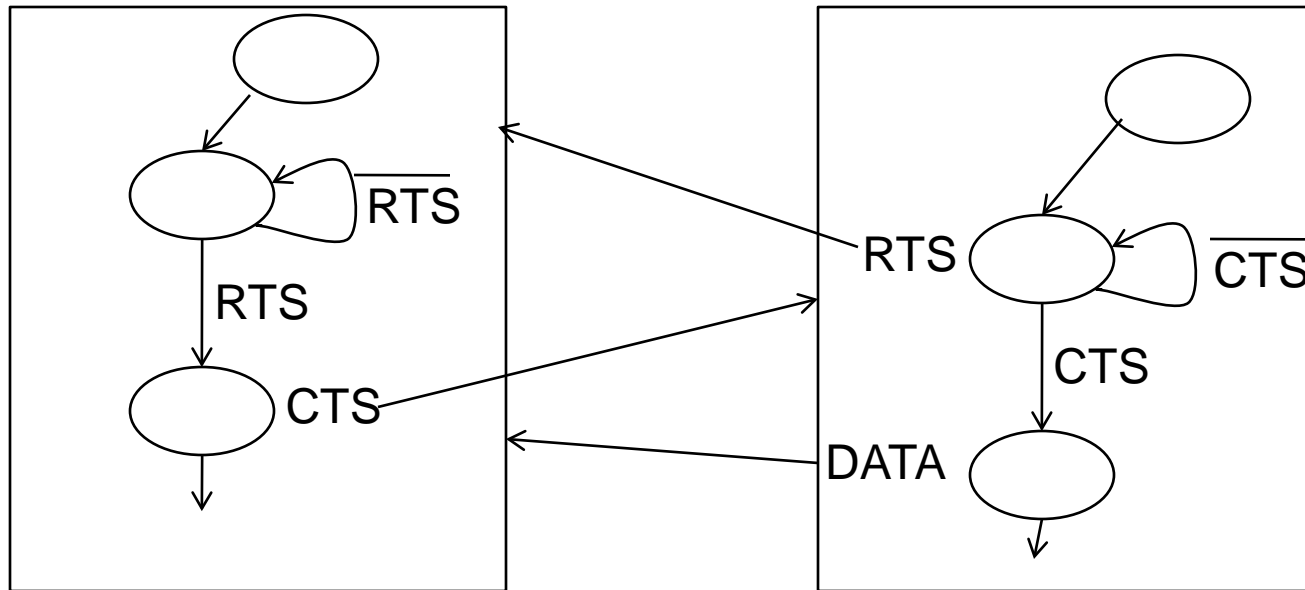
# Handshaking For Debounce Circuit

## Handshaking

The process of a system controller interacting with another controller or device through the use of control signals. A certain protocol between the interacting hardware is necessary in order to assure that data is not lost while a process is not yet completed.



# Handshaking for Serial Communication



# Verilog for Debouncer

```
module Debouncer(reset, switch, delayDone, clk, startDelay, SWdebounced);
    input reset, switch, delayDone, clk;
    output startDelay, SWdebounced;

    reg startDelay, SWdebounced;
    reg [2:0] state, nextState;
    parameter waitClosed = 3'b000, beginDelay = 3'b001, closed = 3'b010,
        waitOpen = 3'b011, strDelay = 3'b100, opened = 3'b101;

    always @(posedge clk)
    begin
        if (reset) begin
            state <= waitClosed;
            nextState <= waitClosed;
        end
        else
            state <= nextState;
    end
end
```



# Verilog for FSM

```
always @(state,switch,delayDone)
begin
    SWdebounced = state == beginDelay || state == closed ||
    state == waitOpen;
    startDelay = state == beginDelay || state == strDelay;
    case (state)
waitClosed : begin
    if (switch) nextState = beginDelay;
    else nextState = waitClosed;
    end
beginDelay : begin
    nextState = closed;
    end
closed : begin
    if ( delayDone) nextState <= waitOpen;
    else nextState <= closed;
    end
end
```

# Verilog for FSM continued

```
waitOpen : begin
    if (switch == 0) nextState <= strDelay;
    else nextState <= waitOpen;
end
strDelay : begin
    nextState <= opened;
end
opened : begin
    if (delayDone) nextState <= waitClosed;
    else nextState <= opened;
end
default : begin
    nextState = waitClosed;
end
endcase
end
endmodule
```

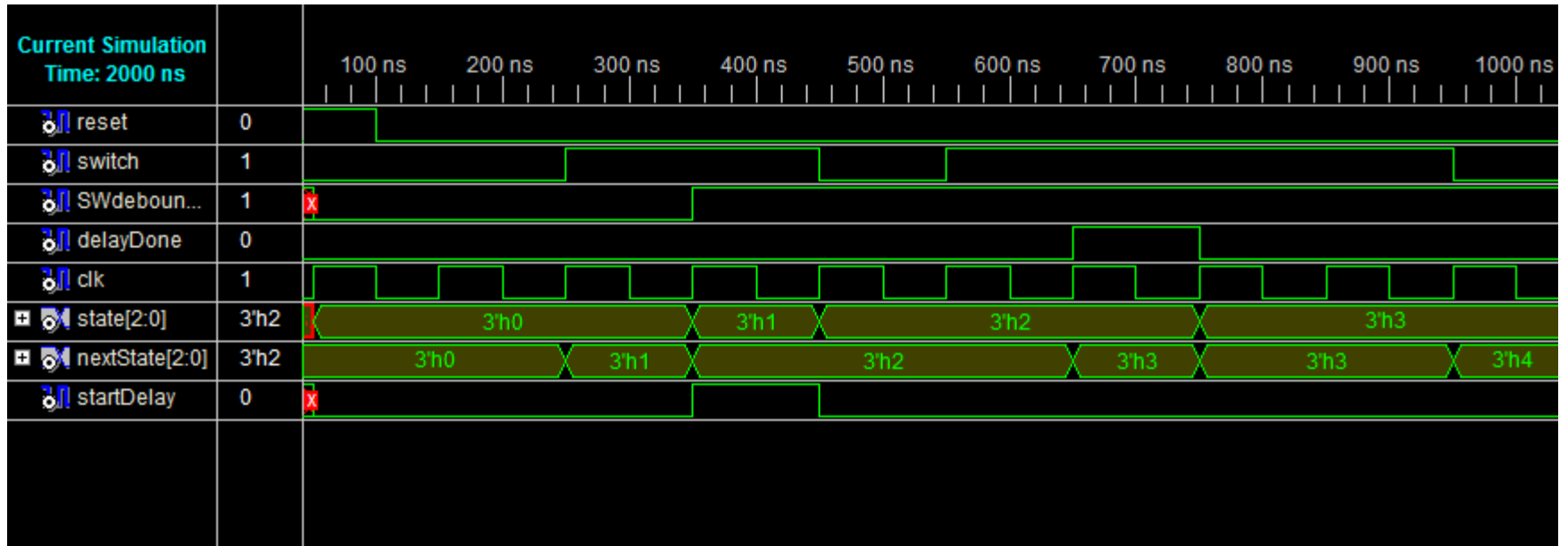
# Verilog for Delay

```
module timeDelay(clear,clk,start,alarm);
    input clear,clk,start;
    output alarm;
    reg alarm;
    parameter terminalCount = 3;//Sets delay time

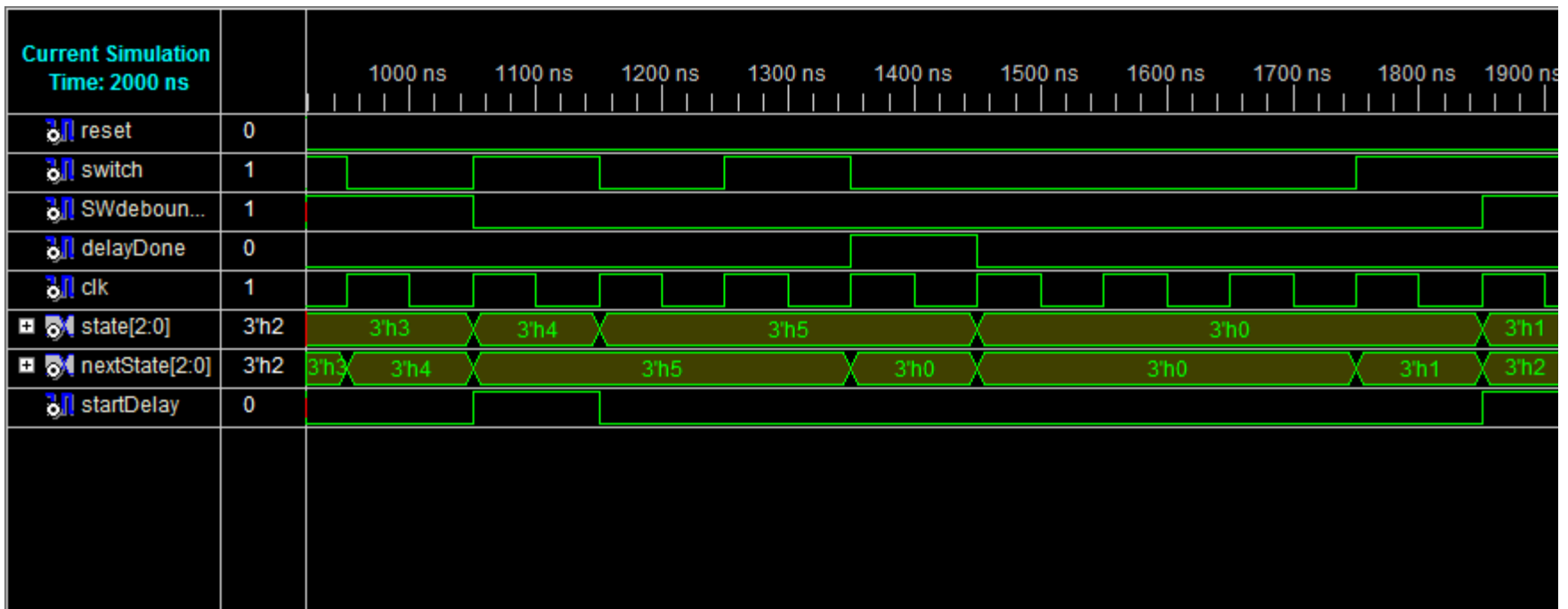
    reg [12:0] countValue;

    always @(posedge clk)
    begin
        alarm = 0;
        if (clear) countValue <= 0;
        else if (start) countValue <= 1;
        else if (countValue > 0)
            if (countValue == terminalCount) begin
                countValue <= 0;
                alarm = 1;
            end
            else countValue <= countValue + 1;
        end
    end
endmodule
```

# Simulation Results



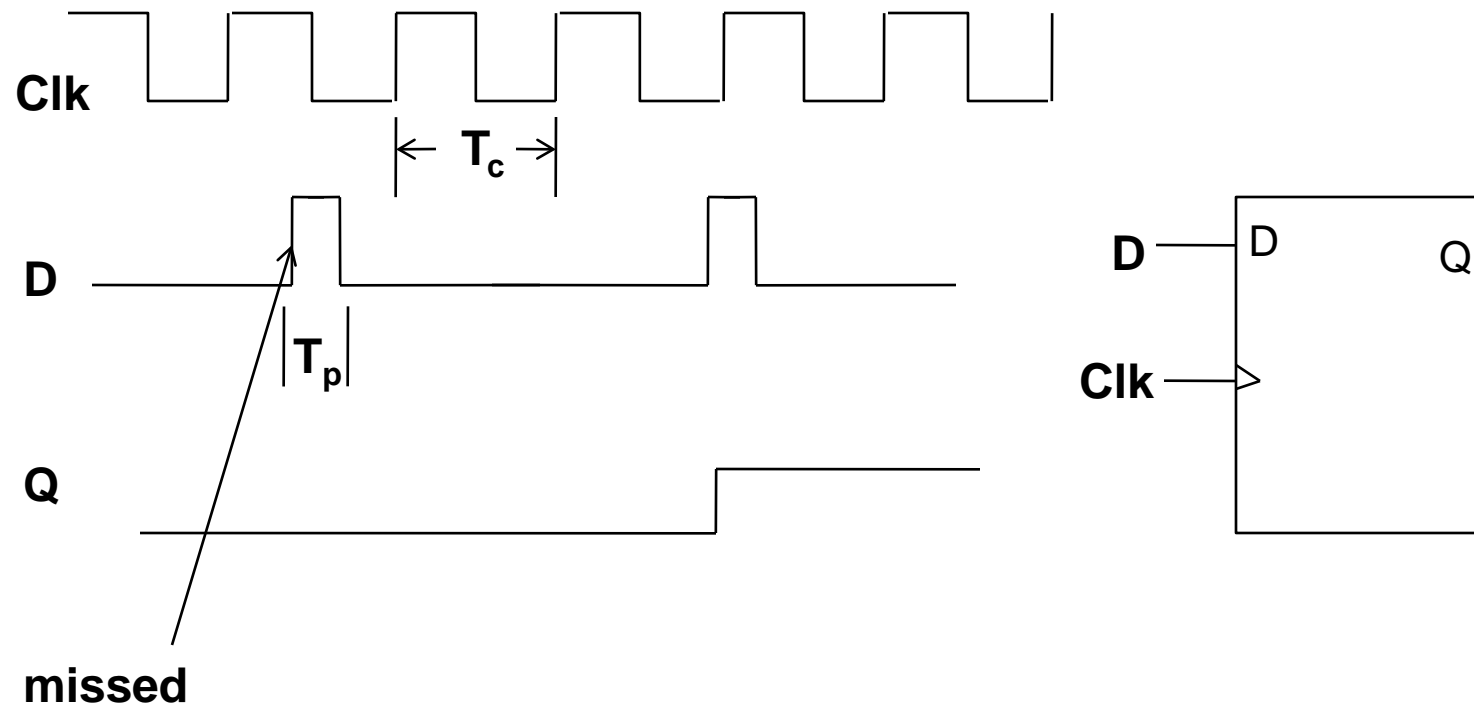
# Simulation Results continued



# Synchronization Problems and Solutions

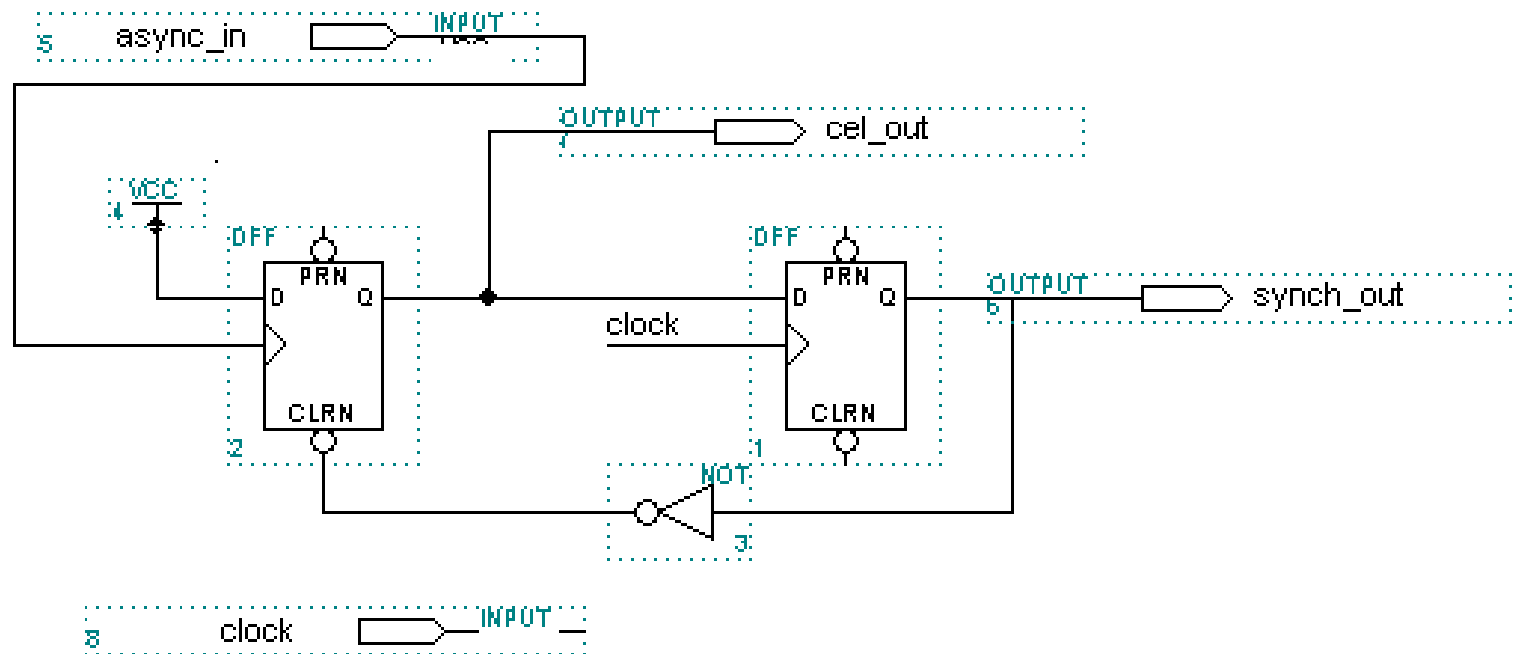
- The short pulse problem

Characterized by  $T_p < T_c$

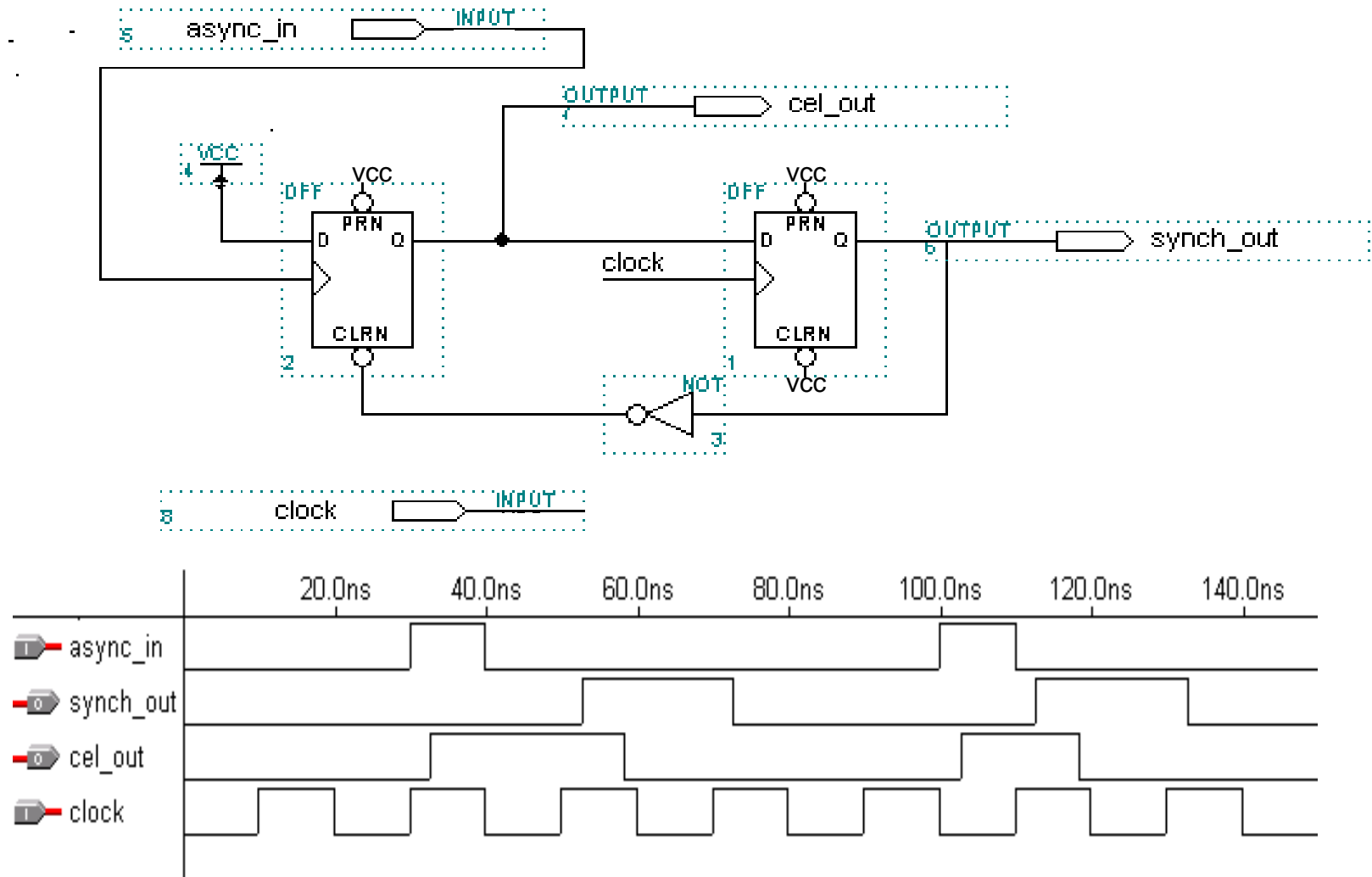


# Possible Solutions

1. Increase the clock frequency such that  $t_p > t_c$ .
2. Use a catching cell as shown below.



# Catching Cell Response





# Catching Cell – External Reset

