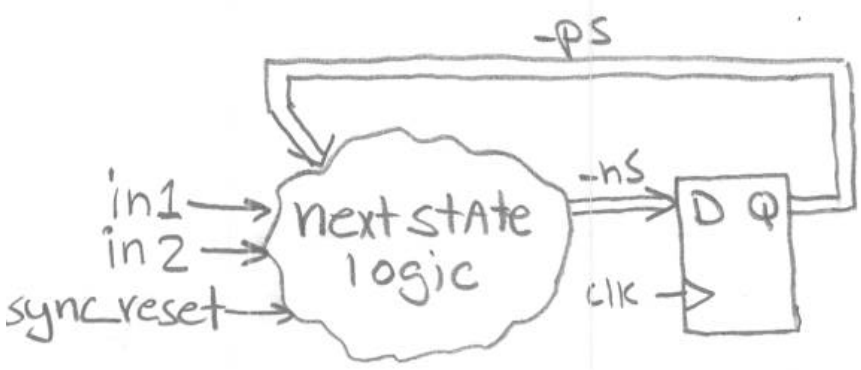
Asynchronous versus Synchronous Resets

- Reset is needed for:
  - forcing the ASIC into a sane state for simulation
  - initializing hardware, as circuits have no way to self-initialize
- Reset is usually applied at the beginning of time for simulation
- Reset is usually applied at power-up for real hardware
- Reset may be applied during operation by watchdog circuits

Synchronous Resets

- Reset is sampled only on the clock edge
- Reset applied as any other input to the state machine

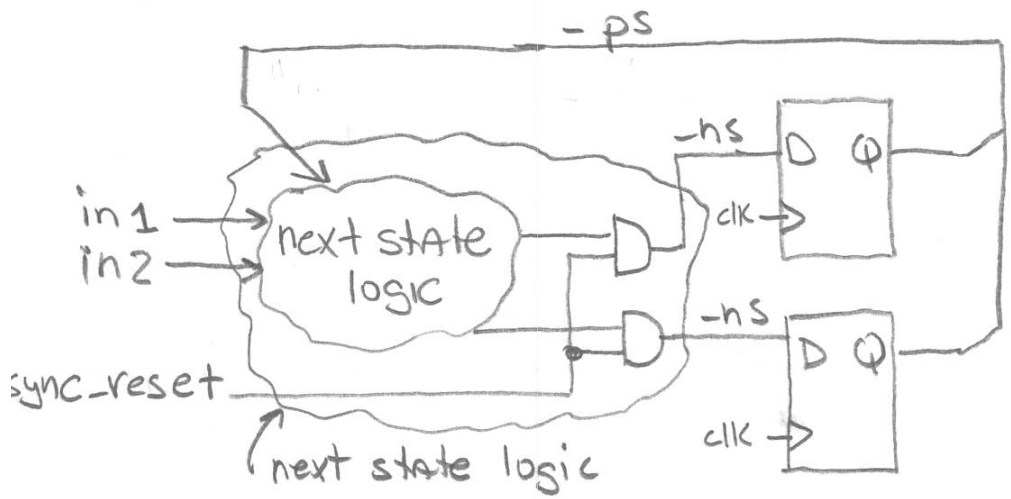


Synchronous Resets

- Sync reset advantages
  - The flip-flop is less complex, thus smaller in area
  - Circuit is completely synchronous
  - Synchronous resets provide filtering for the reset line
- Sync reset disadvantages
  - Combinatorial logic grows and may cancel out the benefit
  - Reset buffer tree may have to be pipelined to keep all resets occurring within the same clock cycle
  - May need to pulse stretch reset so its is wide enough to be seen at a clock rising edge
  - Requires a clock to be present if reset is to occur
  - If internal tri-state buffers are present, separate asynchronous reset may still be required
  - Reset signal may take the fastest path to flip-flops

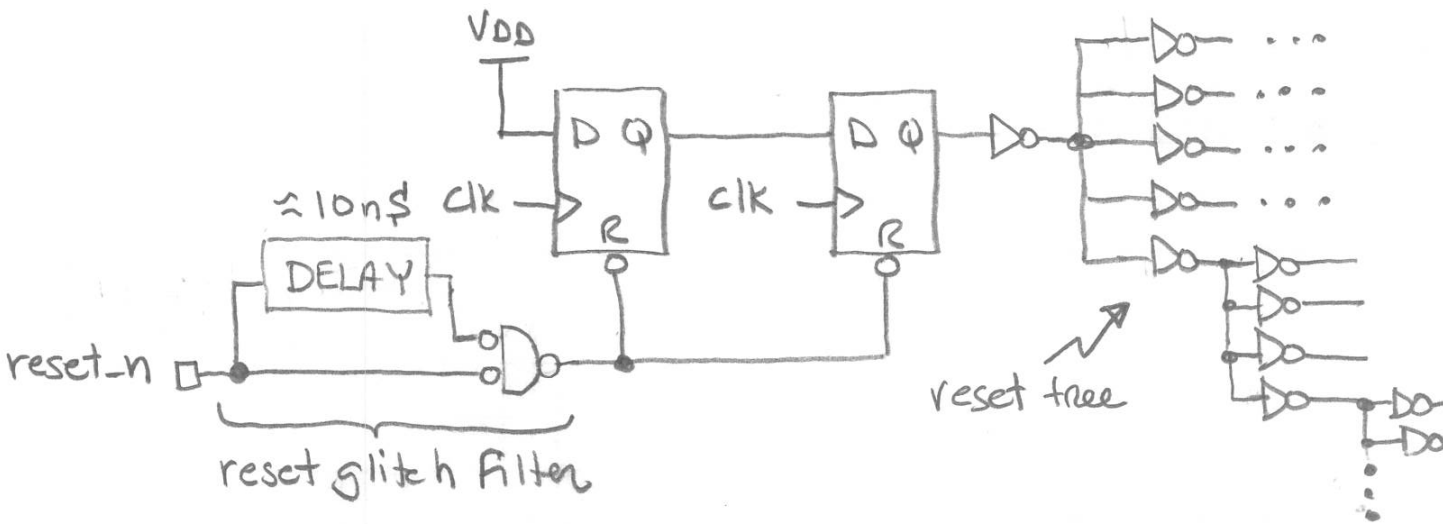
Asynchronous Resets

- Asynchronous reset advantages
  - Reset has priority over any other signal
  - Reset occurs with or without clock present
  - Data paths are always clear of reset signals
  - No coercion of synthesis tool needed for correct synthesis
- Asynchronous reset disadvantages
  - Reset deassertion to all flip-flops must occur in less than a clock cycle.
  - Reset line is sensitive to glitches at any time



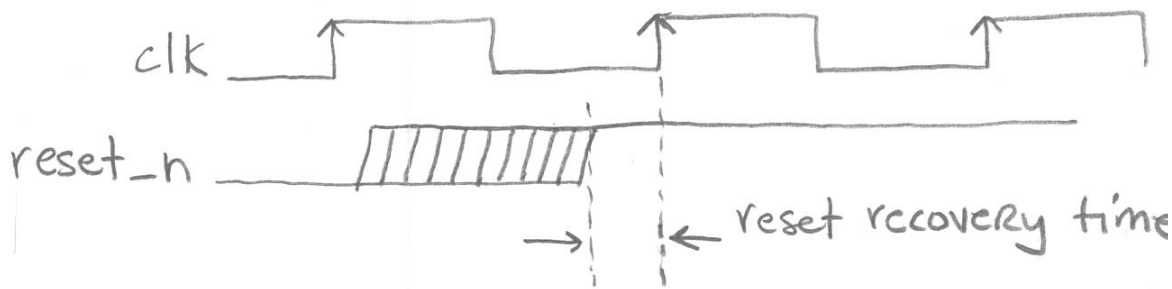
Asynchronous Resets

- Asynchronous reset synchronization circuit
  - Synchronization circuit required with asynchronous reset
  - Circuit will provide asynchronous reset and synchronous deassertion



Asynchronous Resets

- Reset tree
  - Routing and buffering of the reset tree almost as critical as the clock tree
  - Reset goes to every flip-flop, possibly 100's of thousands
  - Capacitive load is very large
  - Reset deassertion must happen within 1 clock cycle and allow time for reset recovery time



Resetting FPGAs

- FPGAs are initialized during configuration: all FFs, and memory
- ASICs must have an explicit reset to achieve initialization
- What happens here in both environments?

```
  %lecture_verilog/beamer/example_code/init_ffs.sv
module init_ffs (
  input   clk,
  output  q_out_b,
  input   d_in
  );
logic q_out = '0; //initalize flip flop
always_ff @ (posedge clk)
  q_out <= d_in;
assign q_out_b = q_out; //assign output
endmodule
```

- FPGA: FFs are initialized in hardware and simulation
- Hardware FFs get initialized by configuration
- Initialization statement resets simulation FFs

Resetting FPGAs

- What happens here in both environments? (cont.)

```
module init_ffs (
  input   clk,
  output  q_out_b,
  input   d_in
  );
logic q_out = '0; //initalize flip flop
always_ff @ (posedge clk)
  q_out <= d_in;
assign q_out_b = q_out; //assign output
endmodule
```

- ASIC: Synopsis design compiler says:

```
Warning: init_ffs.sv:8: The 'declaration initial assignment'
construct is not supported.  It will be ignored. (VER-104)
```

Resetting FPGAs

- Places for providing explicit reset
  - High availability application requiring hot reset
  - If clock drops out, need hard reset after PLL gets lock
  - Comm channel: if it drops out, how do you reset the hardware?
  - User push button reset
  - Partial configuration will require reset of new logic
  - IP (esp from ASIC sources) may require it
  - Migration to ASIC from FPGA
- Reset not needed for some logic (pipelines)
- Reset is always needed for any logic with feedback