Synchronous VHDL and DFF

Announcements

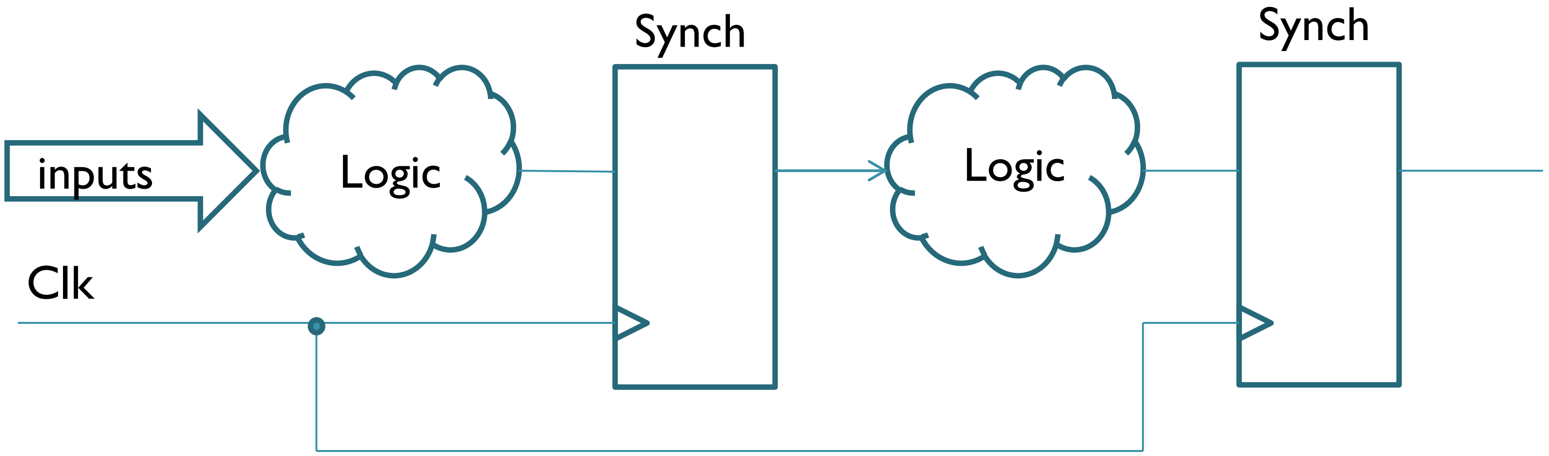
- No homework due this week
- HW#8 is posted
- Reminder:
  - You have to pass lab to pass the course
  - Many students did not submit a report for labs 2-4
  - Even if you do not submit a report, you need to submit your signoff sheet
- Reading assignment:
  - Ch. 6 section 2

Synchronous VHDL

- Up until this point all VHDL has resulted in combinatorial circuitry
  - What is the characteristic of combinational logic?

- Combinational logic has limited application in digital systems
  - Almost all digital systems are synchronous
  - What is the characteristic of synchronous logic?

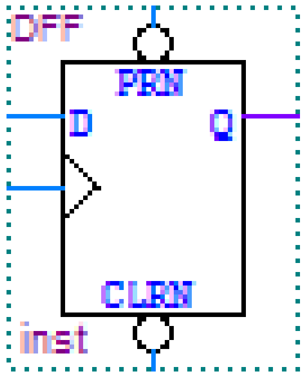
Synchronous System



- The logic between synchronous stages is combinational
- On the clock edge the output from the combinational stages is latched (or stored) in the synchronous logic (registers)

Flip – Flop

- Basic memory element
- Has two states; 0 or 1
- Needs a clock edge or a non-synchronous preset or clear signal to change states
- Flip – Flops are interconnected to form memory structures that store multiple bits of information as a unit



VHDL clock signal

- Synchronous elements need a clock signal for a state change
  - Should be put in an If statement within a process
  - Must be in the sensitivity list
    - Clock and Reset are only signals in sensitivity list for a clocked process
    - If a clocked process is sensitive to any other signal, it is written incorrectly

Clock Signal (con’t)

- Two ways to write
  - IF (clk’EVENT) AND (clk=‘1’) THEN
    - This is a clock signal named clk that is active on the rising edge
    - Can be used with data type BIT and STD_LOGIC
  - IF (RISING_EDGE(clk)) THEN
    - Can only be used with STD_LOGIC

Reset Signals

- All clocked elements in a system should have a reset signal
  - Why?
  - Reset can put device in '1' or '0' state
  - Can be active high or active low
- Two types of reset

Two types of Reset

- Asynchronous – reset independent of clock

```
ex: PROCESS(clk, reset_n) IS
   BEGIN
       IF (reset_n ='0') THEN
        state <= '0'
       ELSIF (RISING_EDGE(clk)) THEN
          state <= <clocked value>;
       END IF;

 END PROCESS;
```

- Synchronous – Reset only occurs on clock edge

```
ex: PROCESS(clk)
    BEGIN
       IF (RISING_EDGE(clk)) THEN
           IF (reset_n = '0') THEN
                     state <= '0';
            ELSE
                     state <= <clocked value>;
            END IF;
       END IF;
END PROCESS;
```

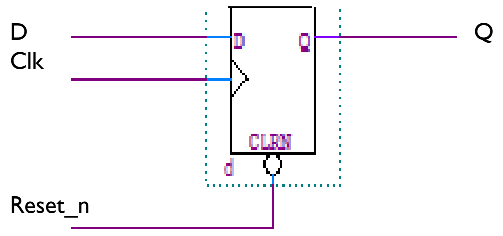
Nested If

Important To Remember

- In Asynchronous
  - Reset condition is the IF
  - Clock condition is the ELSIF
  - NO ELSE
    - All logic is nested under elseif
- In Synchronous
  - Clock condition is the IF
  - NO ELSE to go with the clock's IF
  - The reset is nested under the clock IF
    - All logic is in the ELSE

Flip-Flop VHDL

- Write the VHDL entity and architecture for a D flip-flop with an asynchronous reset



Flip-Flop Variations

- Change the architecture to make the reset synchronous

Flip-Flop Variations

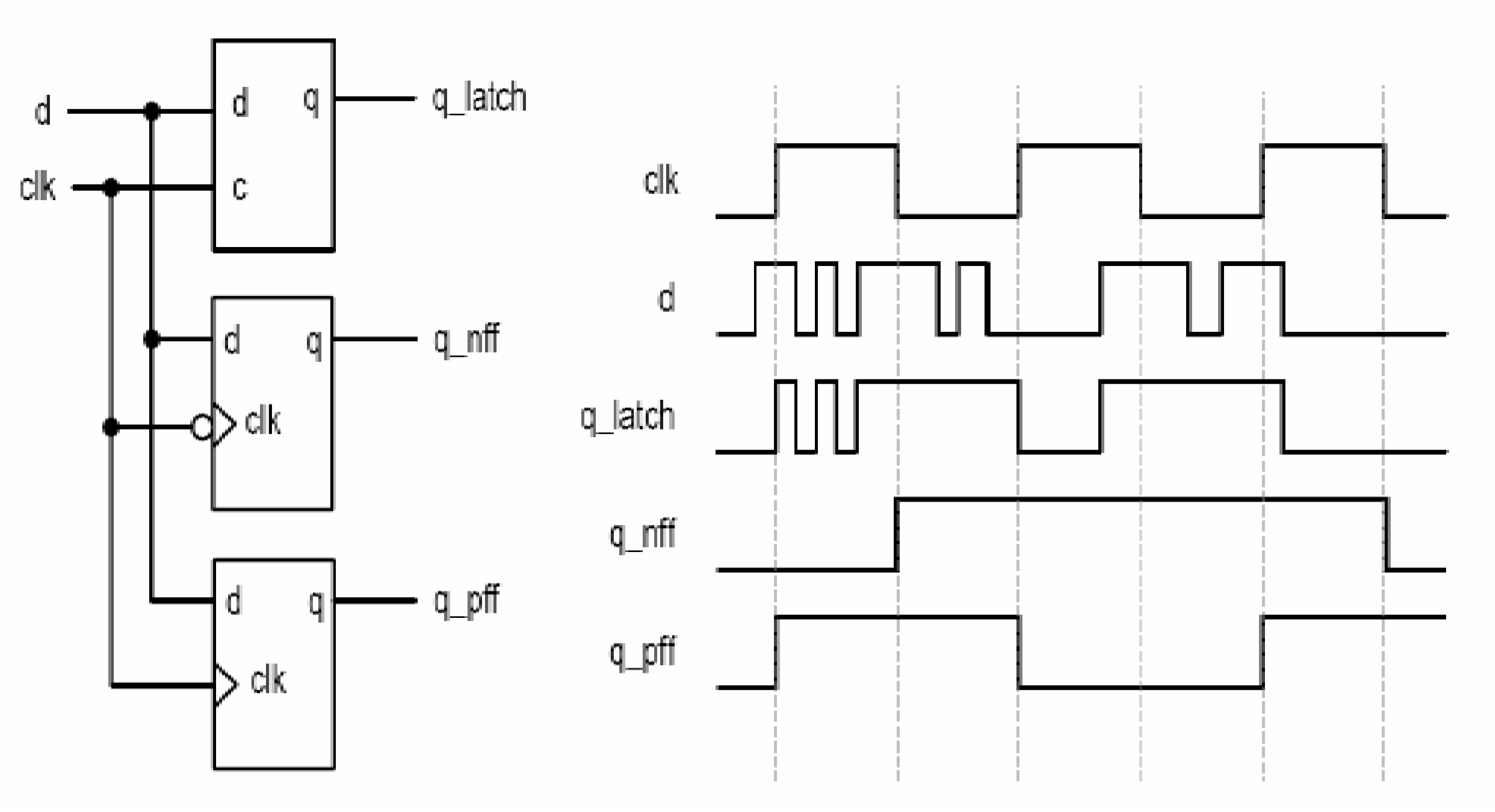
- Change the architecture to add a QN output. Use an internal signal.

Latches

- A basic rule of combinatorial VHDL is :
  - If you assign a value to a signal in the if, you must assign a value to that signal in the elsif (if applicable) and the else
  - Likewise, if you assign a value to a signal in one case, you must assign a value to that signal in all cases
- Why does this rule exist?

What are latches and Why are they bad?

- A latch is level-sensitive – as long as clk is high, q_latch follows d
- The flip-flop is edge-sensitive – q only follows d on the rising or falling edge of clk



Latches (con’t)

- To synthesize a D flip flop, use a clocked process

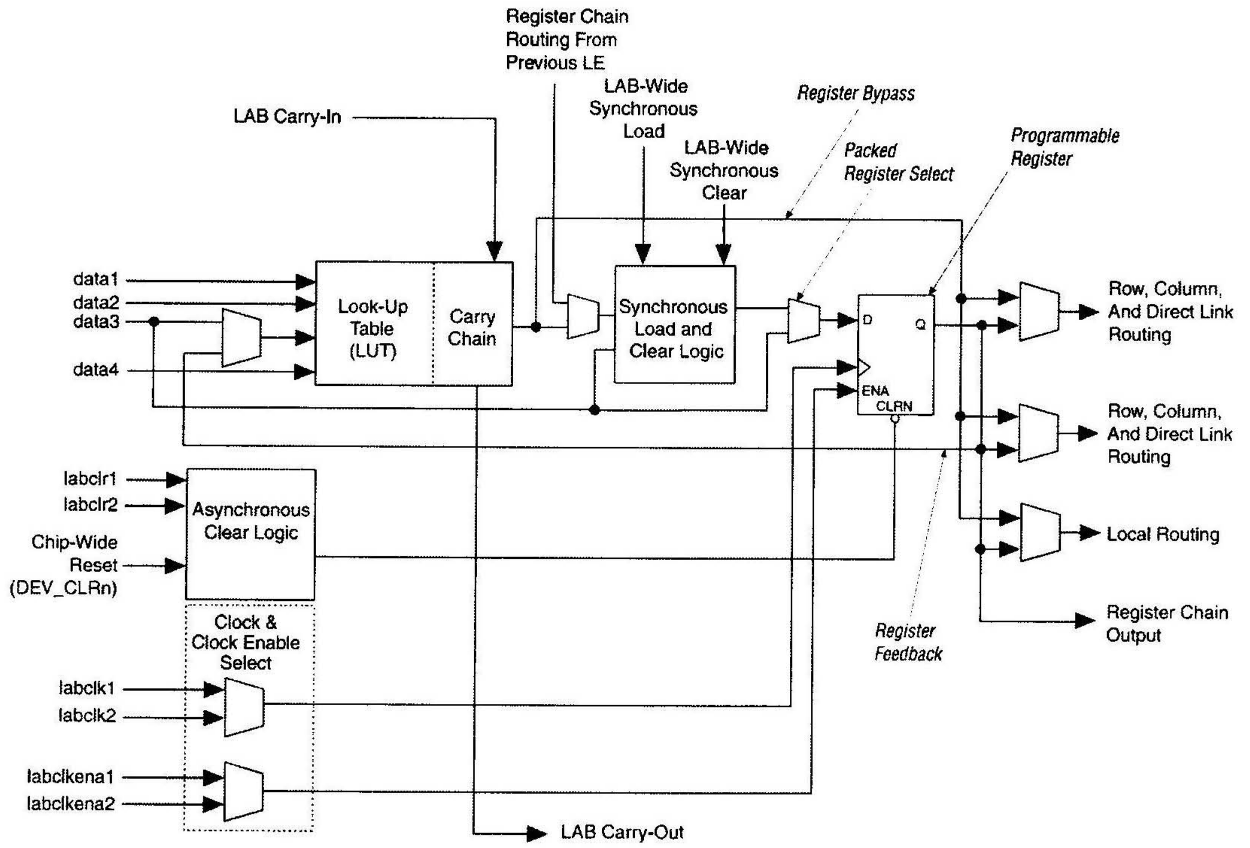
```
dff : process(clk)
    begin
        if (clk’event) and (clk=‘1’) then
              q<= d;
        end if;
end process;
```

- Each logic element in the FPGA has a flip-flop built into the hardware
- All of the flip flops in the FPGA are optimized and clocked by a global clock signal
- Properly coded flip flops are implemented in the hardware

Latches (con't)

Figure 2–2 shows a Cyclone II LE.

Figure 2–2. Cyclone II LE



Latches (con't)

- Latches are inferred when a signal is incompletely specified in an if-then-else statement or a case statement.
- The value of the signal is specified in one case but not in the other.
- In the case that the signal is not specified, the signal must maintain its value – INFERRING MEMORY
- The problem is there is no clock to properly clock the memory

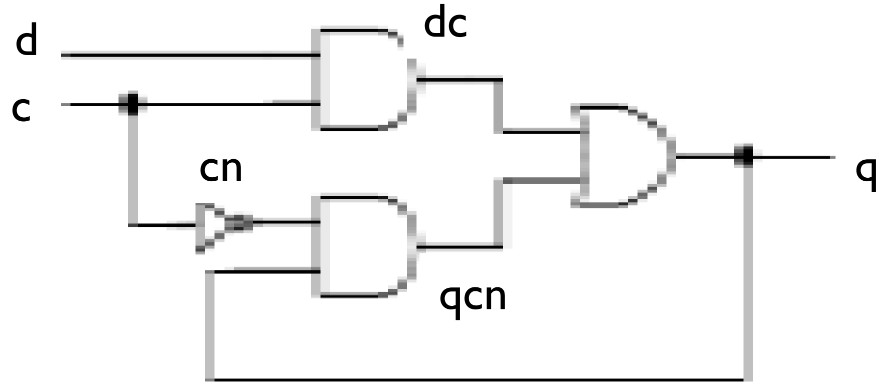
Latches (con't)

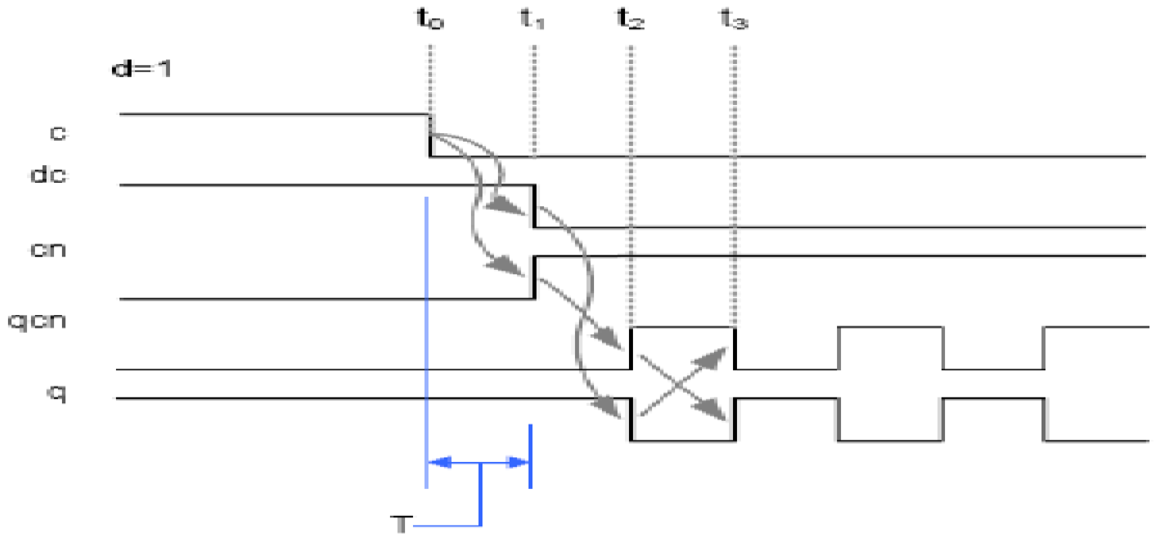
- There are no latch elements in the FPGA
- Inferred latches are implemented in the general FPGA gate logic.
- A feedback loop is created
- The static timing analysis will fail as the tools cannot analyze the feedback path.
  - Warning will be generated

Latches (con't)

- Latch Example

```
Latch : process (c, d)
   begin
       if (c = '1') then
           q_latch <= d;
      end if;
End process;
q <= q_latch;
```





Latches - Final

- Latches are bad and should be avoided
  - The synthesizer will warn you if you have inferred a latch
  - Do not ignore the warning.
  - Simple designs may still work, but more complex designs have been seen to fail