2014 Spring CSE140L

Digital Systems Laboratory

Lecture #3

by

Dr. Choon Kim

CSE Department, UCSD
chk034@eng.ucsd.edu
Introduction to Sequential Logic
Combinational vs. Sequential Logic
Clock Signal

Question: What is it?

CLK

How to get a clock on DE1 board?
D Flip-Flop

Inputs               Output
Clock Enable

Asynchronous Clear

<table>
<thead>
<tr>
<th>CLR</th>
<th>CE</th>
<th>D</th>
<th>CK</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

CLK = 0

CLK = 1

Lecture #3
T Flip-Flop

Asynchronous Clear

Inputs

<table>
<thead>
<tr>
<th>CLR</th>
<th>CE</th>
<th>D</th>
<th>C</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>Q</td>
</tr>
</tbody>
</table>

Output

Clock Enable

0 1 1 ~Q

0 1 0 Q
D-FF Timing

 CLK

 CLK

 D

 Q

 ~Q

 t

 t

 t

 t

 t

 t

 t

 t

 t

 Lecture #3
Timing issues

- Tpd
- Slack
- Required Time
- Actual time
Specify settings for the Classic Timing Analyzer. Use the Assignment Editor for individual timing assignments. Note: These settings affect the Classic Timing Analyzer only. To specify TimeQuest Timing Analyzer settings, use the TimeQuest Timing Analyzer (Timing Analysis Settings menu).

### Delay Requirements

- **tsu:**
  - Value: 
  - Unit: ns

- **tco:**
  - Value: 15
  - Unit: ns

- **tld:**
  - Value: 
  - Unit: ns

- **th:**
  - Value: 
  - Unit: ns

### Clock Settings

- **Default required fmax:**
  - Value: 
  - Unit: MHz

### Description

Specifies the maximum acceptable clock setup time for the input (data) pin. The setup time is the length of time for which data that feeds a register via its data or enable input(s) must be present at an input pin before the clock signal that clocks the register is asserted at the clock pin.
Input Timing Constraints

- **Setup time**: $t_{\text{setup}} = \text{time before the clock edge that data must be stable (i.e. not changing)}$
- **Hold time**: $t_{\text{hold}} = \text{time after the clock edge that data must be stable}$
- **Aperture time**: $t_a = \text{time around clock edge that data must be stable (} t_a = t_{\text{setup}} + t_{\text{hold}})$
Timing/Frequency of Sequential Circuit

- The **minimum** delay from register R1 through the combinational logic to R2 determines the **maximum** frequency.

\[
T_c \geq t_{pcq} + t_{pd} + t_{\text{setup}}
\]

\[F_{\text{max}} = 1/T_c\]

e.g. if \(T_c \geq 6.7\) ns,
\[F_{\text{max}} = 1/6.7\text{ns} = 149.25\text{MHz}\]
Signal A is given as input.

Time Steps | A | B | C | D  
--- | --- | --- | --- | ---  
0 | 0 | X | X | X  
1 | 1 | 0 | X | X  
2 | 0 | 1 | 0 | X  
3 | 1 | 0 | 1 | 0  
4 | 1 | 1 | 0 | 1  
5 | 0 | 1 | 1 | 0  
6 | 0 | 0 | 1 | 1  
7 | 1 | 0 | 0 | 1
Counter using T-FF (Down counting)

CLK

Reset \( A(0) = B(0) = C(0) = 0 \)

Time | C   | B   | A   |
-----|-----|-----|-----|
0    | 0   | 0   | 0   |
1    | 1   | 1   | 1   |
2    | 1   | 1   | 0   |
3    | 1   | 0   | 1   |
4    | 1   | 0   | 0   |
5    | 0   | 1   | 1   |
6    | 0   | 1   | 0   |
7    | 0   | 0   | 1   |

Lecture #3
Up Counter

How to make an up counter using TFF?
000 -> 001 -> 010 -> ....... -> 111 -> 000

**Hint:**
Use ~Q instead of Q to connect to the clk port of next stage TFF
How about **Behavioral modeling**?

- So far we have designed a counter using **structural-level modeling** (i.e., schematics of TFF chain, gates, etc).
- Can we design the same circuit using **behavioral-level modeling**?
  (i.e., similar to SW language, for example,
  ```
  if (input occurs...)
  
  counter_out = counter_out + 1; // Up counting case...
  counter_out = counter_out - 1; // Down counting case...
  
  )
  ```

**Answer:** **Yes, we can do it with a HDL, like Verilog.** In fact Verilog allows you to design LAB#2 circuit in **any level** of modeling(behavioral, structural, ...etc.). Try behavioral-level modeling!
module module_name ( port specification... );

local wires, variables declaration;  
task, function declaration;  

continuous assignments ;  
procedural blocks;  
instantiation of modules ;  
instantiation of primitives /UDP;  

endmodule
Introduction to Verilog HDL

Rule for handling possible incorrect information found on documents:

Electronics, CAD Technology and Verilog HDL keep changing continuously. Therefore some information contained in the following documents (or any document in general) may be **obsolete, incorrect** or **not working**. In our CSE140L class, the way to verify the correctness of an information is to test it by simulating, compiling and testing the result on our FPGA DE1 board using our CAD SW (Altera Quartus II Web Edition Software v9.0 Service Pack 2).

1. **[Quartus II example practice using Verilog]**:  *(Note: You can do this example without understanding of Verilog.)* Follow the instructions described in tut_quartus_intro_verilog_de1
2. **[Verilog Tutorial for beginner]**: Follow Verilog short tutorial to make yourself familiar with this new design entry methodology. Altera’s Introduction to Verilog   Altera’s Verilog HDL Basics
3. **[References]**: intro_verilog   Altera’s Recommended HDL Coding Styles
4. **[Recommended Verilog books]**  Note that so many good Verilog books are available in the world and each book is different in various ways. In the end you yourself should find a book best suitable for your own unique situation:
   A Verilog HDL Primer (Third Edition) -- 2005 by J. Bhasker  
   Verilog HDL (2nd Edition) -- 2003 by Samir Palnitkar
5. **[Quick Reference Card]**: Two-page Card   Multiple-page Card
6. In addition, there are numerous Verilog information & tutorials available on the web in doc, book, & video format. The Verilog on wiki is an excellent place to learn the language. It also lists many tutorial links that student may visit and learn.