2013 Fall CSE140L

Digital Systems Laboratory

Lecture #4

by

Dr. Choon Kim

CSE Department, UCSD
cchk034@eng.ucsd.edu
Introduction to Sequential Logic
Combinational vs. Sequential Logic
Clock Signal

Question: What is it?

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

Asynchronous Clear

Inputs               Output
<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>X</td>
<td>X</td>
<td>Q</td>
</tr>
</tbody>
</table>

(No change)

0 1 1 1

0 1 0 0

CLK = 0

CLK = 1
T Flip-Flop

Asynchronous Clear

Inputs               Output
Clock Enable

<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>

Asynchronous Clear

Clock Enable

Inputs

Output

 CLR  CE  D  C  Q
0 1 1 ~Q
0 1 0 Q
D-FF Timing

CLK

CLK

D

Q

\( t_{\text{setup}} \)

\( t_{\text{hold}} \)

\( t_{\text{cq}} \)

Lecture #4
Timing issues

- Tpd
- Slack
- Required Time
- Actual time
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.

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

\[
F_{max} = \frac{1}{T_c}
\]

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

<table>
<thead>
<tr>
<th>Time Steps</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Counter using T-FF (Down counting)

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

<table>
<thead>
<tr>
<th>Time</th>
<th>C</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Lecture #4
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;  // for mainly combinational circuit
procedural blocks;  // for mainly sequential circuit
instantiation of modules;  // for hierarchical design
instantiation of primitives /UDP;  // for built-in primitives

endmodule
Big question when you *synthesize*!

**Question:**
Can I use *any* construct defined in Verilog HDL definition in my design if my goal is to *synthesize* my design into chip?

**Answer:**
No. All the constructs of Verilog are simulatable. However, not all the constructs of Verilog are synthesizable. Each CAD synthesizer supports its own *synthesizable subset* of the Verilog constructs only.

Then, which constructs are supported by our Altera Quartus II synthesizer for synthesis?
You have to review and use the following Altera's synthesizable subset support list,

(****Remember a warning, "Warning about incorrect information on documents:" in previous slide. You still need to verify any information by testing on Quartus II SW & DE1 board HW.****)

- [Quartus II Verilog HDL Support](#)
- [Quartus II Support for Verilog 2001](#)

Also, following information may help you when coding Verilog in general.

- [Altera's Recommended HDL Coding Styles](#)
Introduction to Verilog HDL (Home work)

Warning about incorrect information 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 incorrect and not working.
For our FA13 CSE140L class, only way to verify the correctness of the information is to test any information by simulating, compiling and testing using our CAD SW(Altera Quartus II Web Edition Software v9.0 Service Pack 2) on DE1 board.

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.

3. [Manual]: intro_verilog_manual

4. [Quick Reference Card]: Two-page Card  Multiple-page Card

5. 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.
More Verilog Tutorial Sources

Slide Presentations:
1) Altera’s Introduction to Verilog
2) Altera's Verilog HDL Basics

Recommended Books:
1) Verilog HDL (2nd Edition) -- 2003 by Samir Palnitkar
2) A Verilog HDL Primer (Third Edition) -- 2005 by J. Bhasker

Question: What is SystemVerilog?