CSE140: Components and Design Techniques for Digital Systems

Tajana Simunic Rosing
Where we are now...

• What we are covering today:
  – Sample problems from old midterms

• **CAPEs are out!!!**  [https://cape.ucsd.edu/students/](https://cape.ucsd.edu/students/)
  – Your feedback is very important, please take the time to fill out the survey. We read your feedback carefully and use it for future courses. 😊
  – If at least 250 students do CAPES, I will drop the lowest HW grade!
  – If at least 300 students do CAPES, I will also drop the lowest QUIZ!

• Deadlines:
  – HW7 due now; solutions posted Wed 3:30pm
  – Exam#3 during the last day of class
    • 80 minutes long, comprehensive
    • Bring one 8 ½ x 11” paper with handwritten notes, but nothing else
    • Sample midterm 3 & solutions are posted from 2015 & 2012 (some problems do not apply)
    • Extra TA/tutor office hours starting on Monday through Thursday at 12pm
  – Optional bonus final exam:
    • Out Mon. 12/7; due Friday 12/11; gives max 1% bonus on your grade
    • Start from C-code, go all the way into circuits
Sample problems
RTL Examples: Reaction Timer

On reset \((rst)\) reaction timer waits for 10 sec before turning on light \((len=1)\). It measures the length of time \(rtime\) (ms) until a user presses the button \(B\)

- If reaction slower than 2 sec, output \(slow=1\) and \(rtime=2000\)
RTL Examples: Reaction Timer

On reset ($rst$) reaction timer waits for 10 sec before turning on light ($len=1$). It measures the length of time $rtime$ (ms) until a user presses the button $B$

If reaction slower than 2sec, output $slow=1$ and $rtime=2000$
Fast sum of 16 32-bit registers

Only for registers with value>0
Fast sum of 16 32-bit registers

> only values > φ

\[ y = \frac{\sum_{i=1}^{16} \text{32bit reg}}{16} \]

\[ \text{sld} = 1 \]
C-Code to RTL

Inputs: byte a, byte b, bit go
Outputs: byte gcd, bit done

while(1) {
    while( !go );
    done = 0;
    while( a != b ) {
        if( a > b ) {
            a = a - b;
        } else {
            b = b - a;
        }
    }
    gcd = a;
    done = 1;
}
C code to RTL

Step 1 - Capture a high-level state machine

*Inputs:* go (bit), a, b (8 bits)
*Outputs:* done (bit), gcd (8 bits)
*Local Registers:* a_reg (8 bits), b_reg (8 bits)
C code to RTL

Step 2 - Create a datapath
C code to RTL

Step 3 - Connect the datapath to a controller

Controller

Datapath

go

a

b

a_ld

a_sel

b_ld

b_sel

gcd_ld

a_eq_b

a_gt_b

done

gcd

Sources: TSR, Katz, Boriello, Vahid
Hot water detector

Outputs a warning when average temp over the past 4 samples exceeds a user defined value; \textit{clr} disables the system

• Inputs (32 bit): $CT$ – current temp; $WT$ – warning temp
• Output : $W$ – high if hot temperature; stays on until \textit{clr} pressed again
Hot water detector

Outputs a warning when average temp over the past 4 samples exceeds a user defined value; clr disables the system

- Inputs (32 bit): CT – current temp; WT – warning temp
- Output: W – high if hot temperature; stays on until clr pressed again
Hot water detector cont.
Going from FSM to implementation
<table>
<thead>
<tr>
<th></th>
<th>S₀</th>
<th>S₁</th>
<th>S₂</th>
<th>S₃</th>
<th>S₄</th>
<th>S₅</th>
<th>S₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Timing diagram

- D-FF is negative edge triggered; Tsu=2ns, Th=1ns, Tpd=3ns
- D latch is high sensitive; Td=1ns
- Clk: period 12ns, 25% duty; rising edge starts at 1ns
- Delay gates = #inputs * 1ns
- At t=0; DFF has a 0, D has a 1
- D-FF is negative edge triggered; $T_{su}=2\text{ns}$, $T_h=1\text{ns}$, $T_{pd}=3\text{ns}$
- D latch is high sensitive; $T_d=1\text{ns}$
- Clk: period $12\text{ns}$, 25% duty; rising edge starts at $1\text{ns}$
- Delay gates = $\#\text{inputs} \times 1\text{ns}$
- At $t=0$; DFF has a 0, D has a 1
FSM design

- Output Z=1 when the input sequence 00 01 10 11 is recognized, else z =0. Draw the state diagram.
Latch problem

- Transistor level implementation of an SR latch
Latch problem

Replace NOR with transistors

NOR gate
End
### FSM design example – pattern recognizer

<table>
<thead>
<tr>
<th>Present State $Q_1Q_0$</th>
<th>Next State $AB$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$AB = 00$</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td>00</td>
</tr>
<tr>
<td>10</td>
<td>00</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
</tr>
</tbody>
</table>