Week 6 Discussion
Outline

Last week discussion:
- Latches
- Flip flops

This week:
- Registers
- Counters
- Finite State Machines
- Timing constraints
Registers

Basic register: It has to hold / store the bits.
Registers

Shift register: It shifts the bits with every clock pulse.
Registers

Exercise: You have to do a serial transfer from shift register A to shift register B when ‘shift control’ signal is 1 while maintaining information in A.
Registers

Application: Serial adder
Counters

Asynchronous counter: The flip flops involved are not connected to the same clock.

Exercise: Draw the waveforms for $A_0$, $A_1$, $A_2$, $A_3$
Counters
Counters

Synchronous counter: All flip flops are connected to the same clock.

Example:

Exercise: List the transitions starting from 000.
Finite State Machine

An FSM consists of

- set of states
- set of inputs / outputs
- initial state
- set of transitions

An FSM can be visualized with a ‘State Diagram’.
Finite State Machine

A simple example:

(a) Circuit diagram

(b) State table

(c) State diagram
Finite State Machine

Moore machine: Output depends on the state of the system
Finite State Machine

Mealy machine: Output depends on the state as well as inputs.
Finite State Machine
Finite State Machine
Finite State Machine

Exercise: Design a circuit that detects a sequence of three or more consecutive 1’s in a string of bits coming through serial input line.
Finite State Machine

<table>
<thead>
<tr>
<th>Present State</th>
<th>Input ( x )</th>
<th>Next State</th>
<th>Output ( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A ) ( B )</td>
<td></td>
<td>( A ) ( B )</td>
<td></td>
</tr>
<tr>
<td>0 0</td>
<td>0</td>
<td>0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 0</td>
<td>1</td>
<td>0 1</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>0</td>
<td>0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>1</td>
<td>1 0</td>
<td>0</td>
</tr>
<tr>
<td>1 0</td>
<td>0</td>
<td>0 0</td>
<td>0</td>
</tr>
<tr>
<td>1 0</td>
<td>1</td>
<td>1 1</td>
<td>0</td>
</tr>
<tr>
<td>1 1</td>
<td>0</td>
<td>0 0</td>
<td>1</td>
</tr>
<tr>
<td>1 1</td>
<td>1</td>
<td>1 1</td>
<td>1</td>
</tr>
</tbody>
</table>
Finite State Machine

\[ D_A = Ax + Bx \]

\[ D_B = Ax + B'x \]

\[ y = AB \]
Finite State Machine
Timing constraints

A sequential element has an aperture time around the clock edge, during which input must be stable for the flip-flop to produce a well-defined output.

The aperture is defined by a setup time ($t_{\text{setup}}$) and hold time ($t_{\text{hold}}$), before and after the clock edge respectively.

Output may start to change after contamination delay ($t_{\text{ccq}}$)

Output must settle to the final value within propagation delay ($t_{\text{pcq}}$)
Timing constraints

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

\[ t_{ccq} + t_{cd} \geq t_{hold} \]
Timing constraints

We assumed that the clock reaches all components at exactly same time. In reality, wires from clock source to different components may be of different lengths resulting in a slight variation in clock edges. This is called clock skew.
Timing constraints

In the worst case, R1 receives the latest skewed clock and R2 receives the earliest skewed clock, leaving as little time as possible for data to propagate between the registers.

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

In the worst case, R1 receives an early skewed clock, $CLK_1$, and R2 receives a late skewed clock, $CLK_2$. The data zips through the register and combinational logic but must not arrive until a hold time after the late clock.

$$t_{ccq} + t_{cd} \geq t_{hold} + t_{skew}$$
Timing constraints
Thank you!