Introduction

- Sequential circuit
  - Output depends not just on present inputs (as in combinational circuit), but on past sequence of inputs
  - Simple example: a circuit that counts up in binary
- In this chapter, we will:
  - Design a new building block, a flip-flop, that stores one bit
  - Combine that block to build multi-bit storage—a register
  - Describe the sequential behavior using a finite state machine
  - Convert a finite state machine to a controller—a sequential circuit having a register and combinational logic

Example Needing Bit Storage

- Flight attendant call button
  - Press call: light turns on
  - Stays on after button released
  - Press cancel: light turns off
  - Logic gate circuit to implement this?

First attempt at Bit Storage

- We need some sort of feedback
  - Does circuit on the right do what we want?
    - No: Once Q becomes 1 (when S=1), Q stays 1 forever—no value of S can bring Q back to 0

Bit Storage Using an SR Latch

- Does the circuit to the right, with cross-coupled NOR gates, do what we want?
  - Yes! How did someone come up with that circuit? Maybe just trial and error, a bit of insight...
Example Using SR Latch for Bit Storage

• SR latch can serve as bit storage in previous example of flight-attendant call button
  – Call=1: sets Q to 1
  – Cancel=1: resets Q to 0

• But, there's a problem...

Problem with SR Latch

• Problem
  – If S=1 and R=1 simultaneously, we don't know what value Q will take

Solution: Level-Sensitive SR Latch

• Add enable input "C" as shown

Clock Signals for a Latch

• How do we know when it’s safe to set C=1?
  – Most common solution – make C pulse up/down
  – C=0: Safe to change X, Y
  – C=1: Must not change X, Y

Clocks

• Clock period: time interval between pulses
  – Above signal: period = 20 ns

• Clock cycle: one such time interval
  – Above signal shows 3.5 clock cycles

• Clock frequency: 1/period
  – Above signal: frequency = 1 / 20 ns = 50 MHz
  – 1 Hz = 1/s
Level-Sensitive D Latch

- SR latch requires careful design to ensure SR=11 never occurs
- D latch relieves designer of that burden
  - Inserted inverter ensures R always opposite of S

D latch symbol

Problem with Level-Sensitive D Latch

- D latch still has problem (as does SR latch)
  - When Clk=1, through how many latches will a signal travel?
  - Depends on for how long Clk=1
  - Clk_A -- signal may travel through multiple latches
  - Clk_B -- signal may travel through fewer latches
  - Hard to pick Clk that is just the right length
  - Can we design bit storage that only stores a value on the rising edge of a clock signal?

D Flip-Flop

- Flip-flop: Bit storage that stores on clock edge, not level
- One design -- master-servant
  - Two latches, output of first goes to input of second, master latch has inverted clock signal
  - So master loaded when Clk=0, then servant when Clk=1
  - When Clk changes from 0 to 1, master disabled, servant loaded with value that was at D just before Clk changed -- i.e., value at D during rising edge of Clk

D Latch vs. D Flip-Flop

- Latch is level-sensitive: Stores D when Clk=1
- Flip-flop is edge triggered: Stores D when Clk changes from 0 to 1
  - Saying "level-sensitive latch," or "edge-triggered flip-flop," is redundant
  - Two types of flip-flops -- rising or falling edge triggered

D latch symbol

D Flip-Flop

- Solves problem of not knowing through how many latches a signal travels when Clk=1
  - In figure below, signal travels through exactly one flip-flop, for Clk_A or Clk_B
  - Why? Because on rising edge of Clk, all four flip-flops are loaded simultaneously -- then all four no longer pay attention to their input, until the next rising edge. Doesn't matter how long Clk is 1.
Flight-Attendant Call Button Using D Flip-Flop

- D flip-flop will store bit
- Inputs are Call, Cancel, and present value of D flip-flop, Q
- Truth table shown below

| Call | Cancel | Q  | D
|------|--------|----|----
| 0    | 0      | 0  | 0  
| 0    | 0      | 1  | 1  
| 0    | 1      | 0  | 0  
| 1    | 0      | 1  | 1  
| 1    | 0      | 1  | 1  
| 1    | 1      | 1  | 1  

Preserve value; if Q=0, make D=0; if Q=1, make D=1

Cancel -- make D=0

Call -- make D=1

Let’s give priority to Call -- make D=1

Circuit derived from truth table, using Chapter 2 combinational logic design process

Bit Storage Summary

- We considered increasingly better bit storage until we arrived at the robust D flip-flop bit storage

**Design Challenge 1**

- Design Challenge
  - Design a 4x2 priority encoder using AND, OR, and NOT gates

Due Next Lecture (as announced in class)
1 point extra credit (Homework)

**Design Challenge 2**

- Design Challenge
  - Compare the behavior of a D latch and D flip-flop for the following input pattern. Assume each internal latch is initially a 0. Complete the timing diagram assuming gates have a small non-zero delay.

Due Next Lecture (as announced in class)
1 point extra credit (Homework)