**Lecture 5**

- Introduction - Sequential Logic Design
- Storing One Bit
  - SR Latch
  - D flip-flop

---

**Digital Design**

**Sequential Logic Design - Controllers: Bit Storage**

- Flight-attendant call-button system.
  - Pressing Call turns on the light, which stays on after Call is released.
  - Pressing Cancel turns off the light.

---

First (failed) attempt at using feedback to store a bit.

---

**Basic SR latch**

- 2 cross-coupled NOR gates
- 2 inputs: Set & Reset
- 2 outputs: Q' and Q
- Feedback loop

---

SR latch when S=0 and R=1.

- Q(next) = 0
SR latch when S=0 and R=0.
\[ Q(\text{next}) = Q \]

SR latch when S=1 and R=0.
\[ Q(\text{next}) = 1 \]

Flight attendant call-button system using a basic SR latch.

What happens when you Set (S=1) & Reset (R=1) at the same time?
S=1 and R=1 causes problems –
Q oscillates when SR return to 00.

Possible Solution:
Introduce circuitry to prevent S=R=1

Q eventually settles to either 0 or 1, due to race condition.

Will it Work?
Conceptually: YES ... S and R can't both be 1 in this sample circuit.
Reality: NOT ALWAYS... due to the delay of the inverter and AND gate.
Digital Design
Sequential Logic Design -- Controllers: Bit Storage

Gate Delays: can cause $S=R=1$.

Glitches: Temporary values on signals caused by gate delays.

Digital Design
Sequential Logic Design -- Controllers

Level-sensitive SR latch = SR latch + enable input C.

Digital Design
Sequential Design -- Controllers: Synchronous Circuits

$MHz, GHz$ -> speeds of a processor
- Time is measured by clocks
- A clock is used to coordinate & synchronize events
- Clock cycle time

Digital Design
Sequential Design -- Controllers: Synchronous Circuits

Timing Input Changes:
- Circuit inputs (i.e. $S$ & $R$) should only change while $Clk=0$, such that latch inputs will be stable when $Clk=1$.

Digital Design
Sequential Logic Design -- Controllers: Bit Storage

But How Do We Prevent $S=R=1$ once & for all?!?!
(caveat: assumes you are using $C$ responsibly)

Make a D (data)-Latch
A problem with latches - through how many latches will \( Y \) propagate for each pulse of \( \text{Clk}_A \)? For \( \text{Clk}_B \)?

**What the heck?!?!?**

**Digital Design**

**Sequential Logic Design - Controllers: Bit Storage**

**Sequential Design - Controllers: Latch: Signal Propagation**

**Digital Design**

**Sequential Logic Design - Controllers: D-Latch**

**Digital Design**

**Sequential Logic Design - Controllers: Bit Storage**

**Digital Design**

**Sequential Logic Design - Controllers: Registers**

**Digital Design**

**Sequential Logic Design - Controllers: Bit Storage: Flip-Flop**
Through how many flip-flops will Y propagate for each pulse of Clk_A? For Clk_B?

One flip-flop exactly per pulse, for either clock signal.

Positive (shown on the left) and negative (right) edge-triggered D flip-flops. The sideways triangle input represents an edge-triggered clock input.

Latch versus Flip-Flop Timing

Call Cancel Q D /Q(next)

0 0 0 0
0 0 1 1
0 1 0 0
0 1 1 0
1 0 0 1
1 0 1 0
1 1 0 0
1 1 1 1

Flight attendant call-button system:

Call button
Cancel button

Flight attendant call-button system

(a) implemented using a D flip-flop.

Increasingly-better bit storage blocks, leading to the D flip-flop.