CS221: Digital Design

Sequential Logic Design (Latch & FF)

A. Sahu

Dept of Comp. Sc. & Engg.

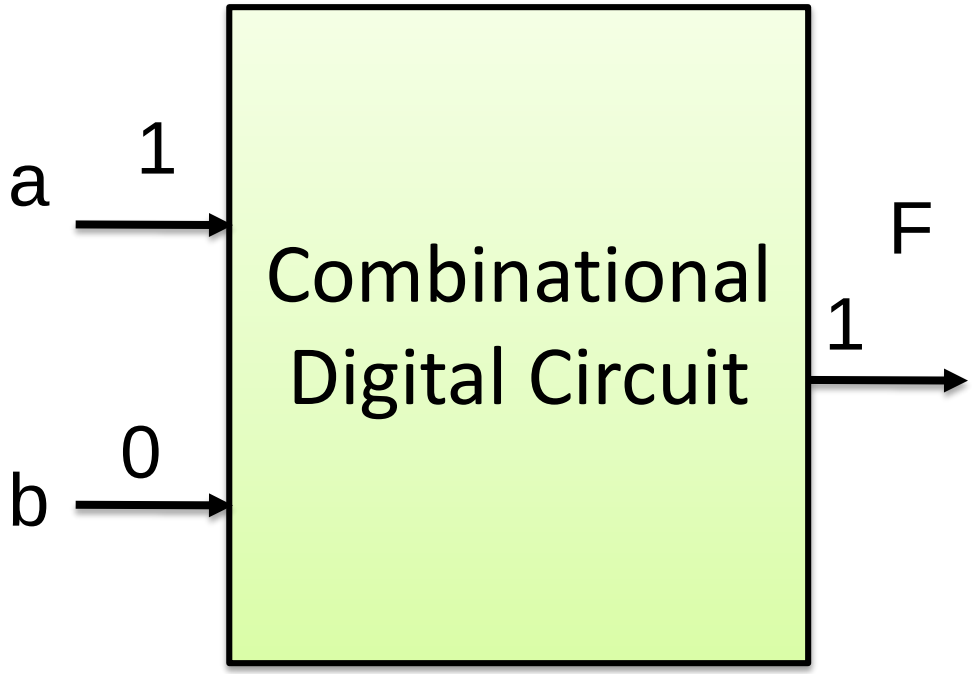
Indian Institute of Technology Guwahati

Outline

- Combinational Vs Sequential Logic Design
- Design a **flip-flop**, that stores one bit
  - RS latch
- Stabilizing RS latch : Level Sensitive
- Clocked Latch : Flip Flop- Edge Sensitive
- D, JK, T flip flops
- Characterization Table and Equation
  - RS, D, JK and T Flip flop

Combinational Vs Sequential Logic

- Combinational circuit
  - **Output depends on present input**
  - Examples: F (A,B,C), FA, HA, Multiplier, Decoder, Multiplexor, Adder, Priority Encoder



Y = F (a,b)

Propagation delay

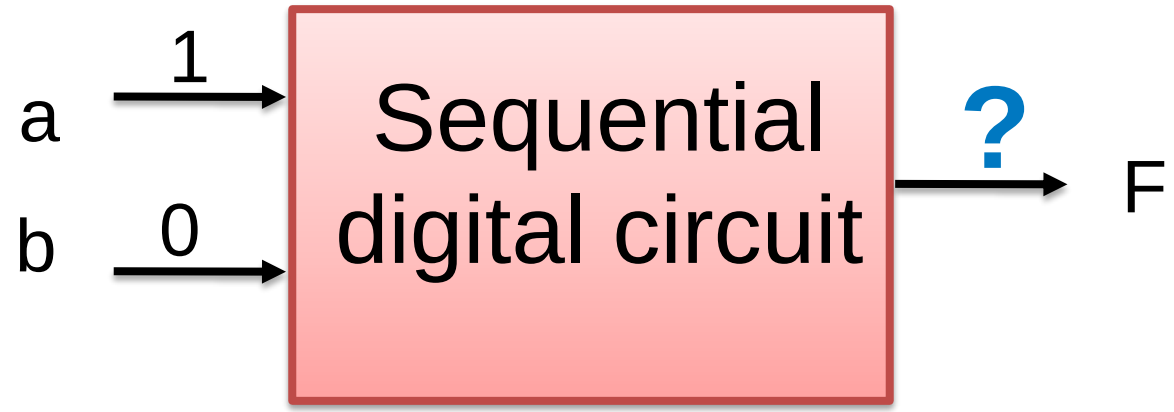
$Y(t+t_{pd})=F(a(t), b(t))$

Combinational Vs Sequential Logic

- Sequential circuit
  - Output depends not just on present inputs
  - But also on past sequence of inputs (State)
    - Stores bits, also known as having “state”

Combinational Vs Sequential Logic

- Simple example: a circuit that counts up in binary

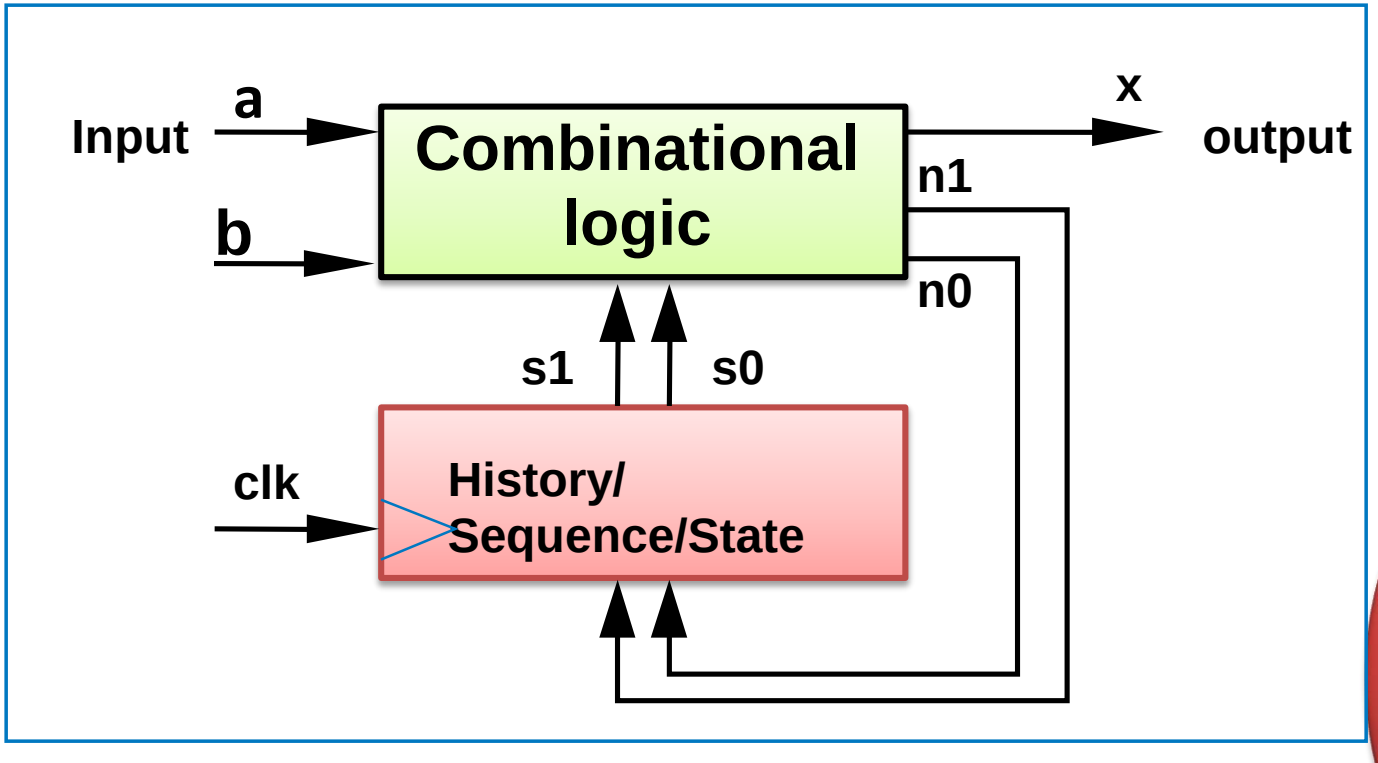


Must know sequence of past inputs to know output

Y (t) = F (a(t),b(t), H)

H is History/Sequence/State

Sequential Circuit



Where to Store this History

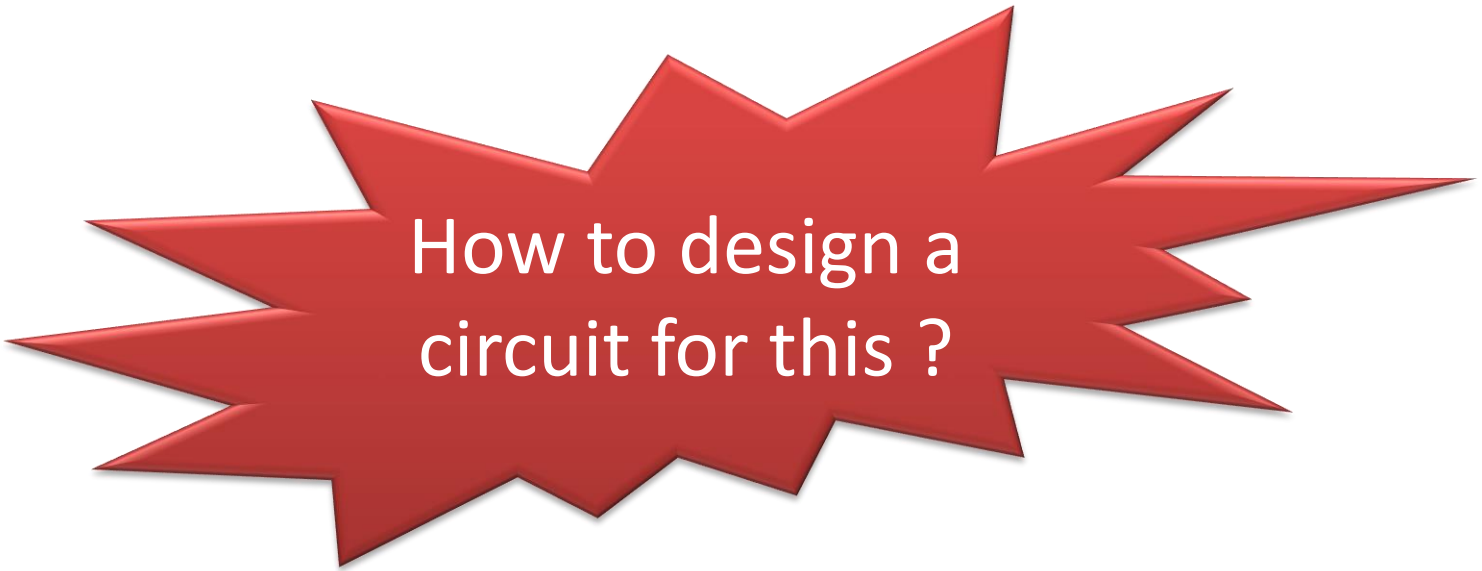
(Memory Element)

Y (t) = F (a(t), b(t), H)

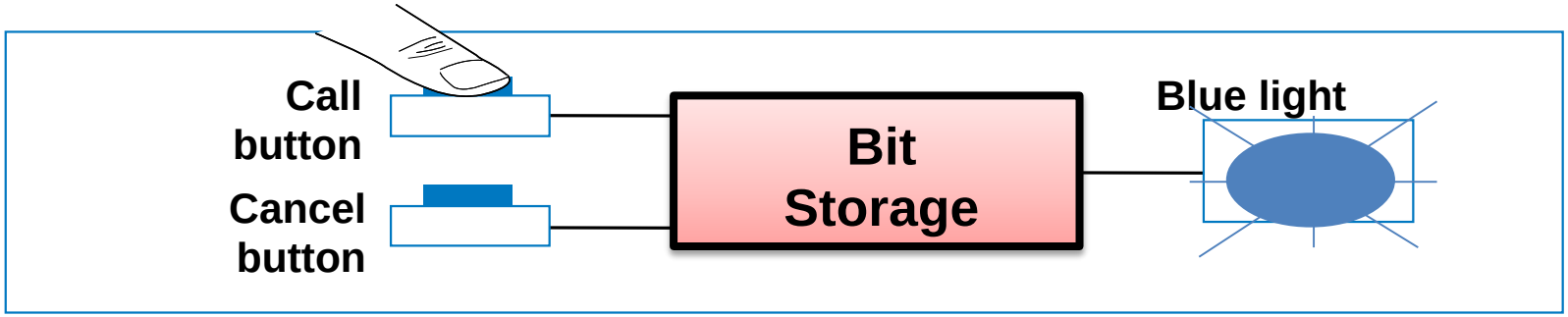
H is History/Sequence/State

Example Needing Bit Storage

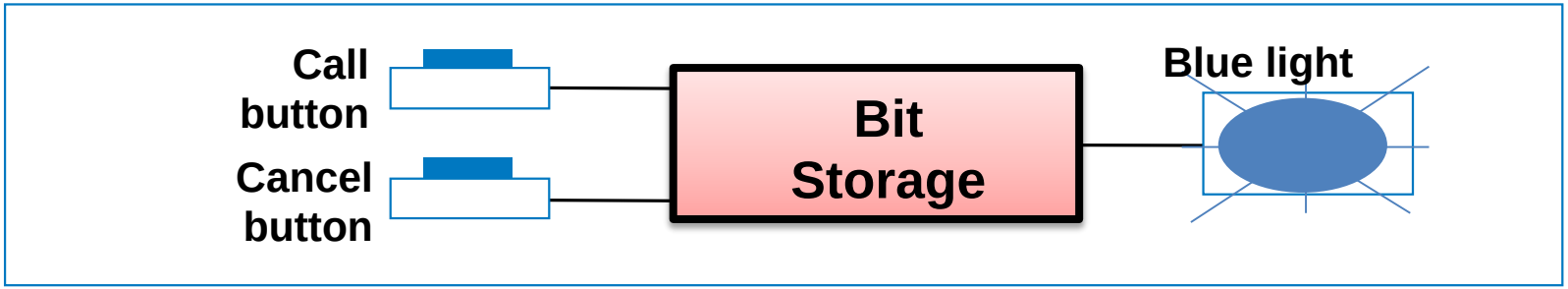
- Example: Flight attendant call button
  - Press call: light turns on
    - ***Stays on*** after button released
  - Press cancel: light turns off



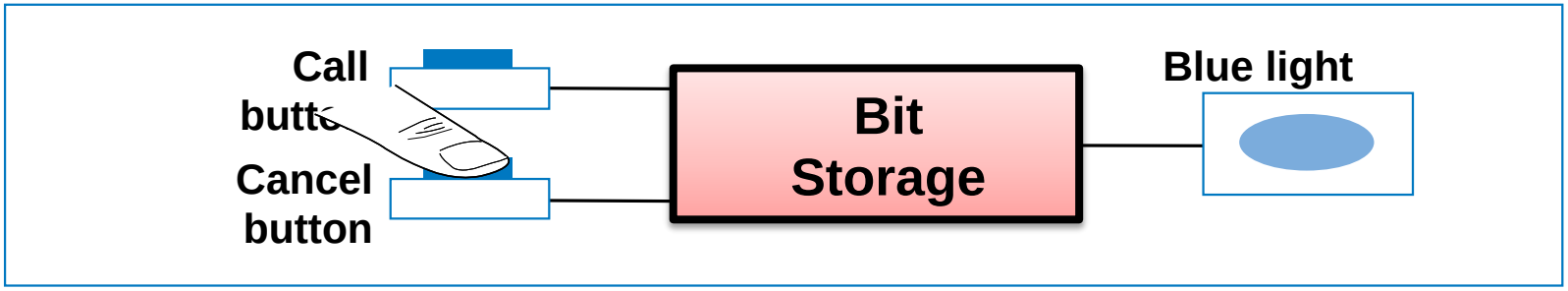
Example Needing Bit Storage



1. Call button pressed – light turns on



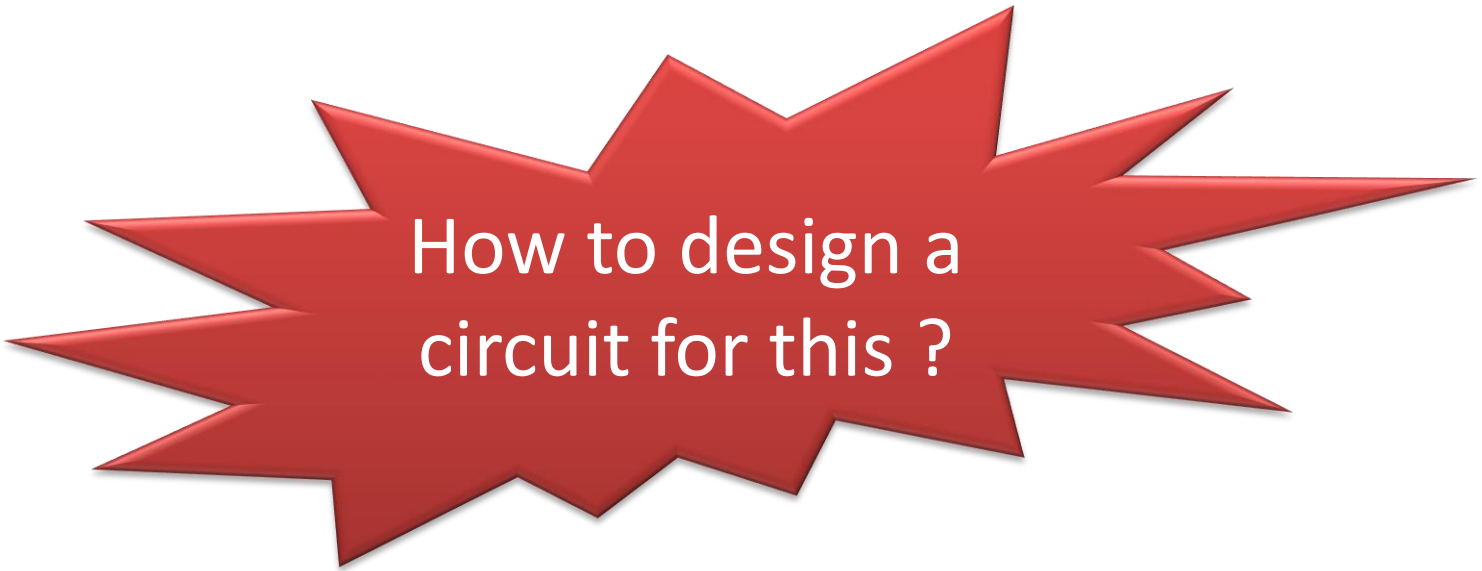
2. Call button released – light stays on



3. Cancel button pressed – light turns off

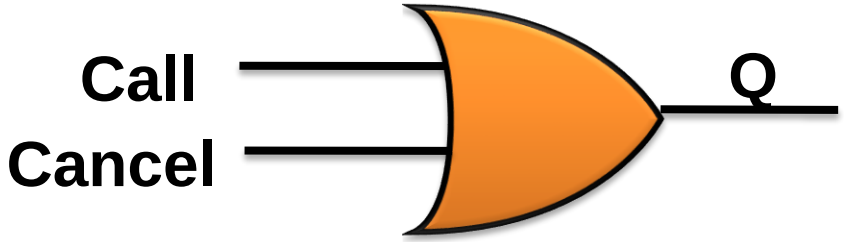
Example Needing Bit Storage

- Flight attendant call button
  - Press call: light turns on
    - ***Stays on*** after button released
  - Press cancel: light turns off



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?

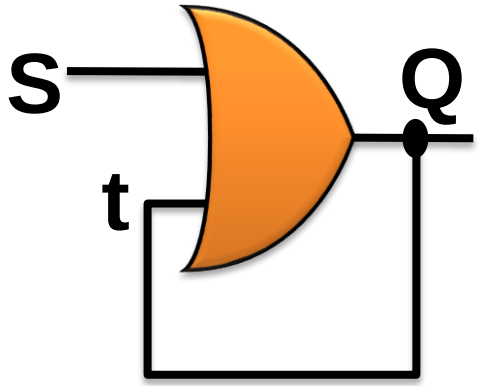


Doesn't work. Q=1 when Call=1, but doesn't stay 1 when Call returns to 0

Need some form of "feedback" in the circuit

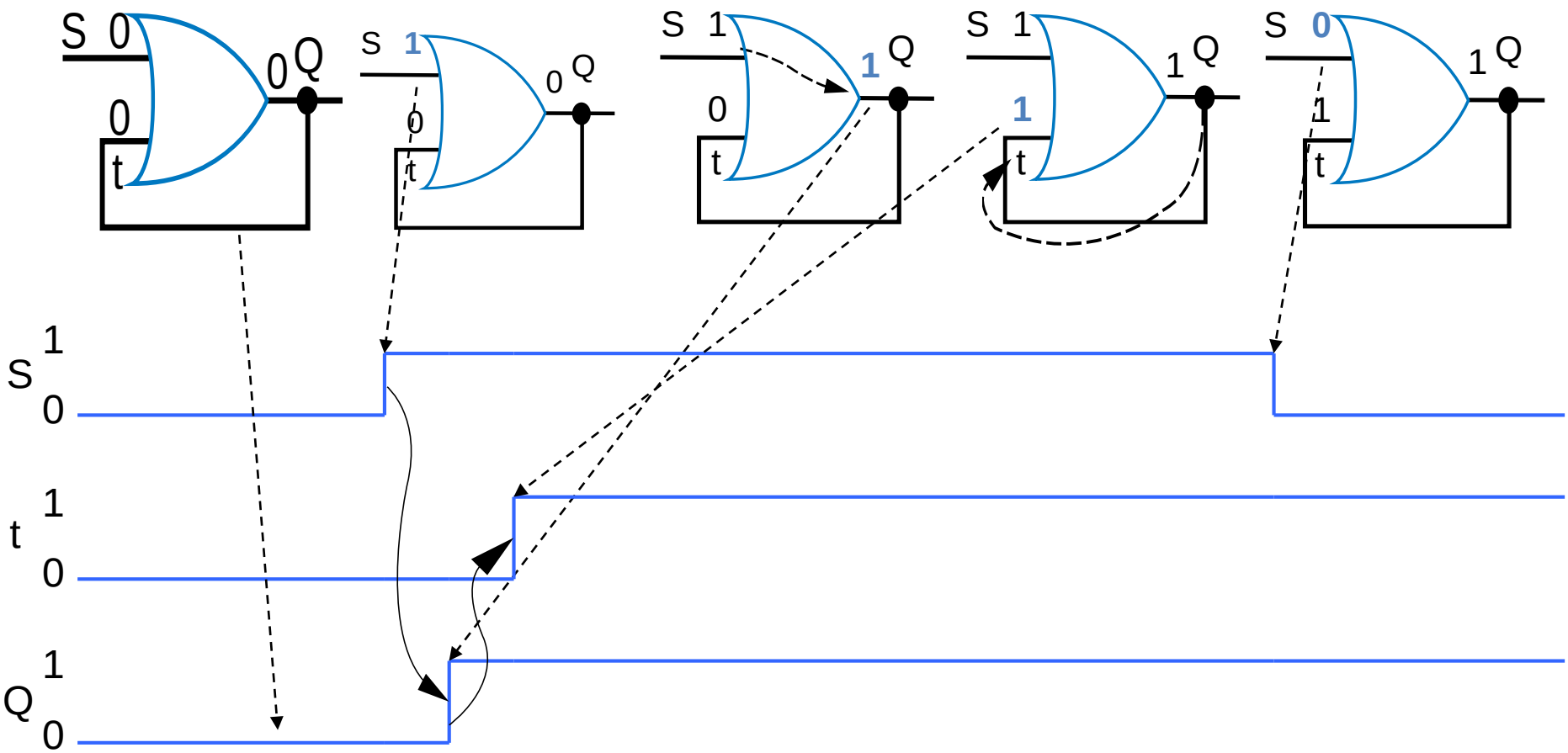
First attempt at Bit Storage

- We need some sort of feedback
  - Does circuit this circuit 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

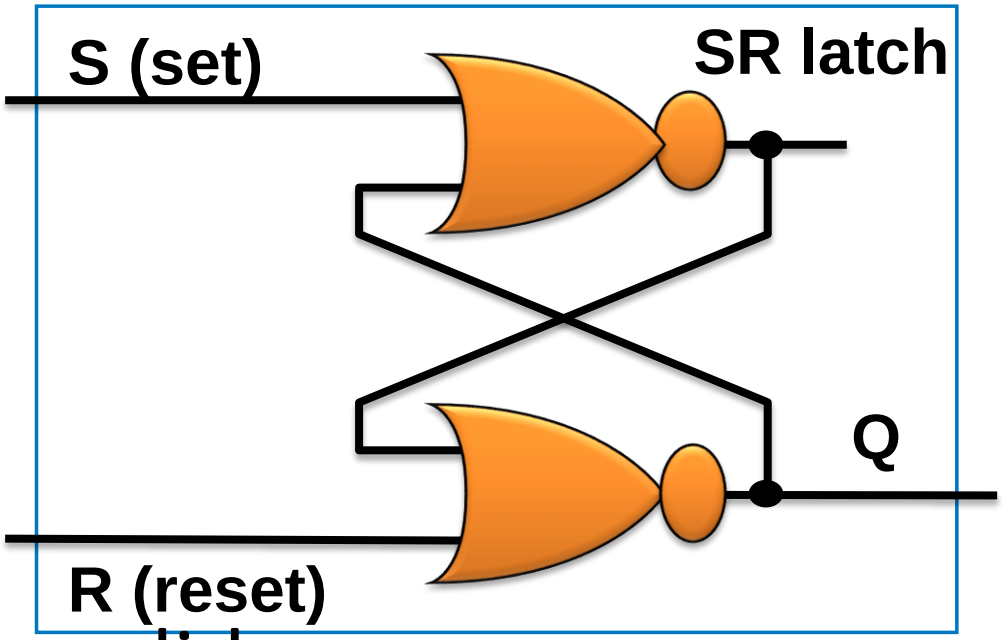
First attempt at Bit Storage



Once **Q** becomes 1 (when **S**=1), **Q** stays 1 forever – no value of **S** can bring **Q** back to 0

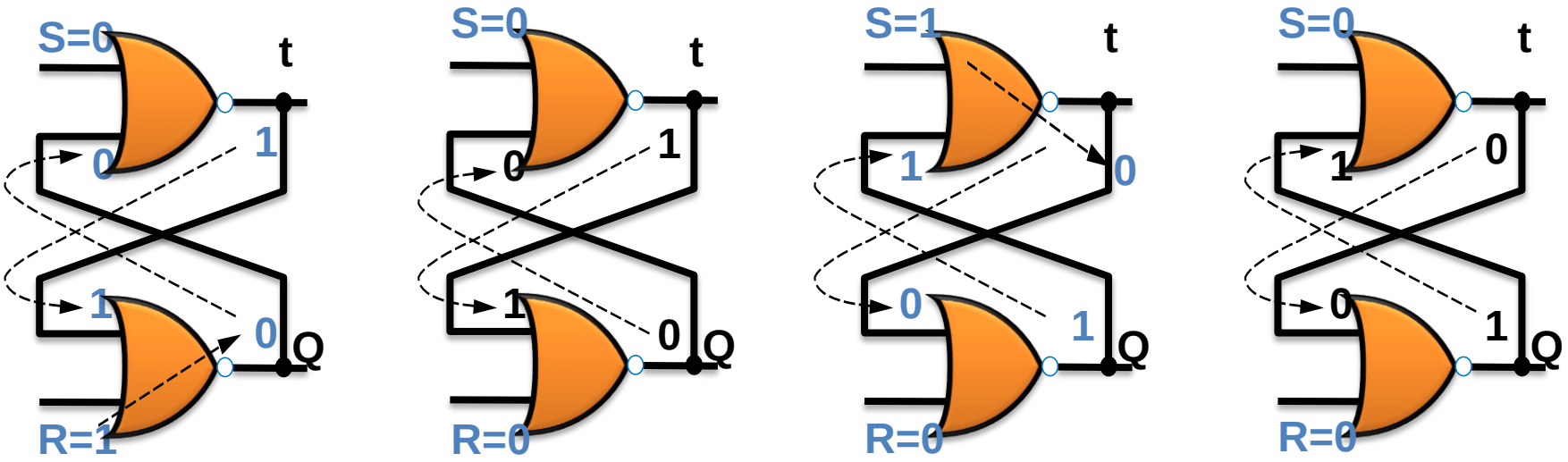
Bit Storage Using an SR Latch

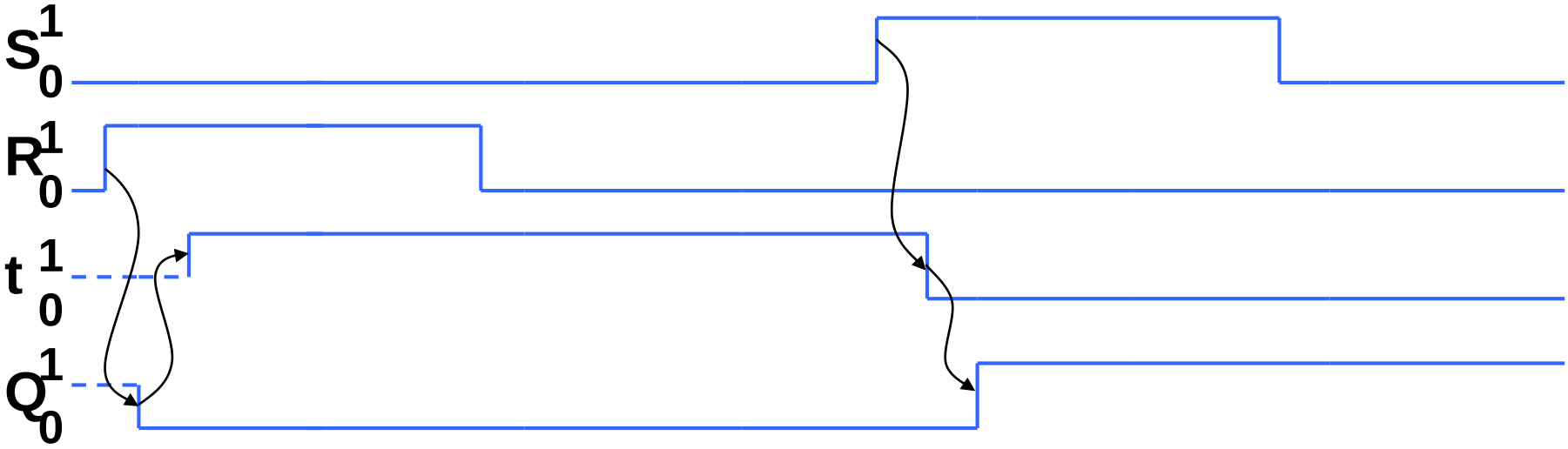
- With cross-coupled NOR gates
  - Does the circuit to the right ?
  - Do what we want?



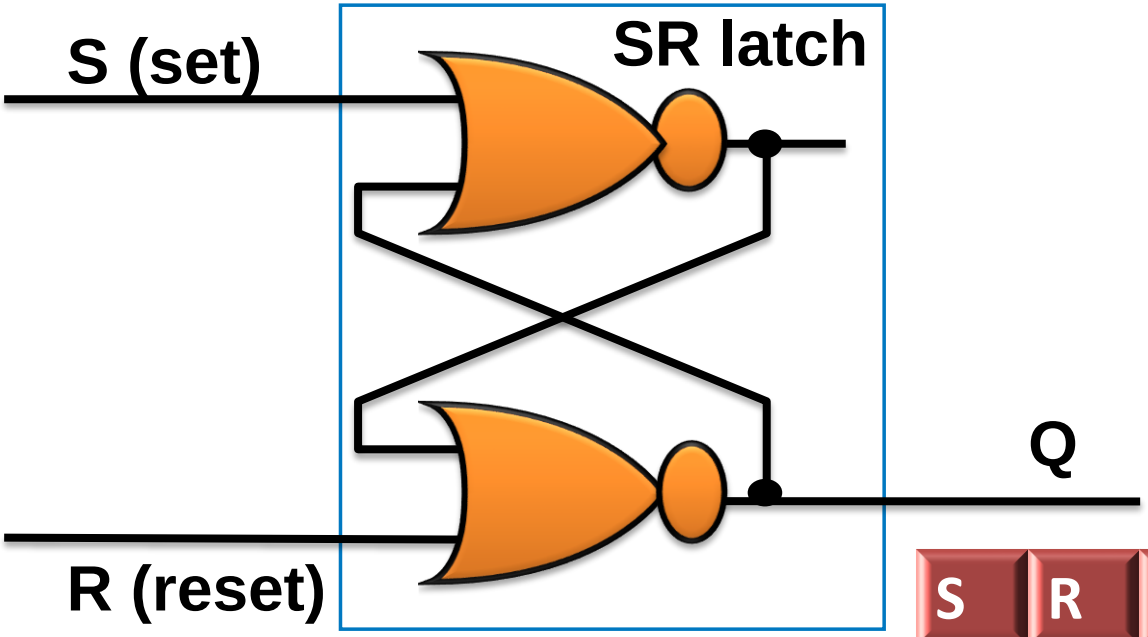
- Yes! How did someone come up with that circuit?
  - Maybe just trial and error, a bit of insight...

Bit Storage Using an SR Latch





Function Table of SR Latch

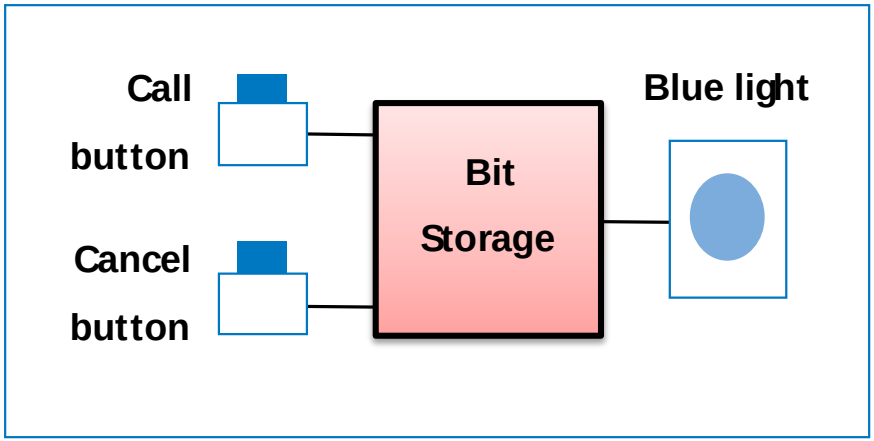


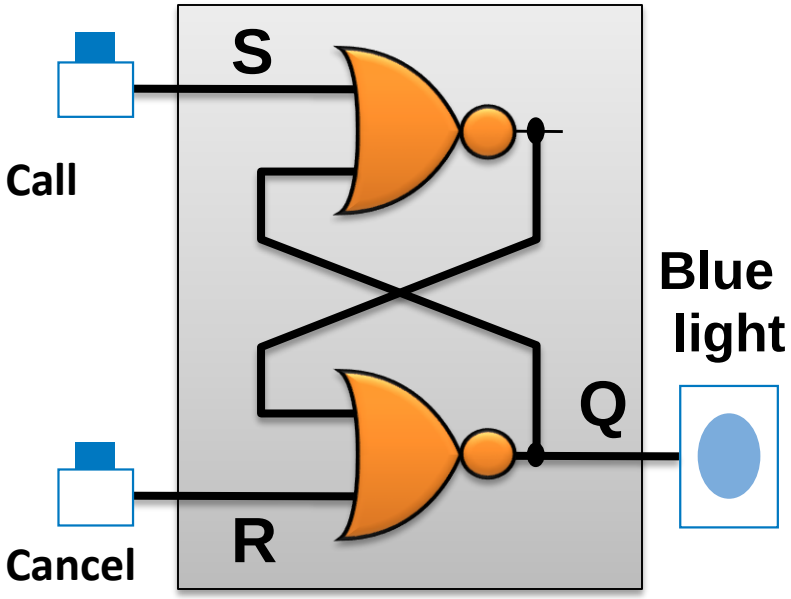
S	R	Q	Q’	
1	0	1	0	
0	0	1	0	After S=1, R=0
0	1	0	1	
0	0	0	1	After S=0, R=1
1	1	0	0	Forbidden

Set ***: make out put 1***

Reset ***: make out put 0***

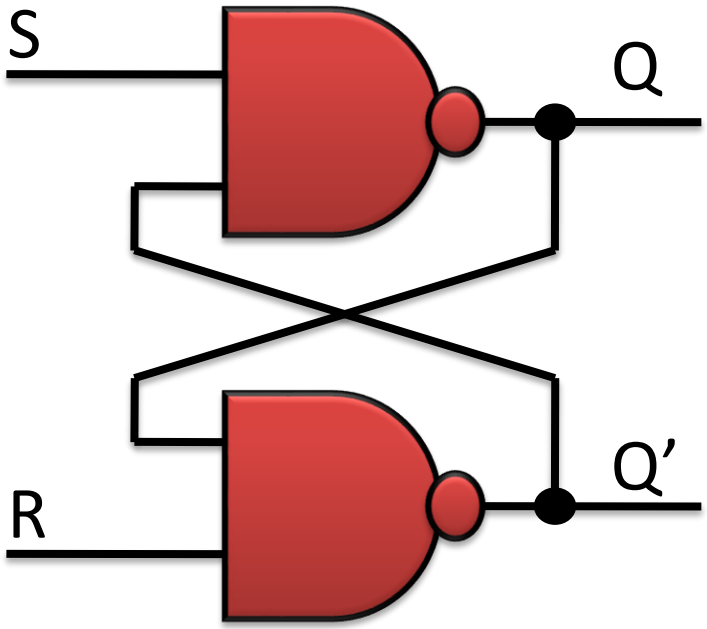
Example Using SR Latch for Bit Storage





- SR latch can serve as bit storage of flight-attendant call button : ☺ ☺ ☺ Great
  - Call=1 : sets Q to 1
    - Q stays 1 even after Call=0
  - Cancel=1 : resets Q to 0
- **But, there's a problem...**

SR Latch with NAND Gates



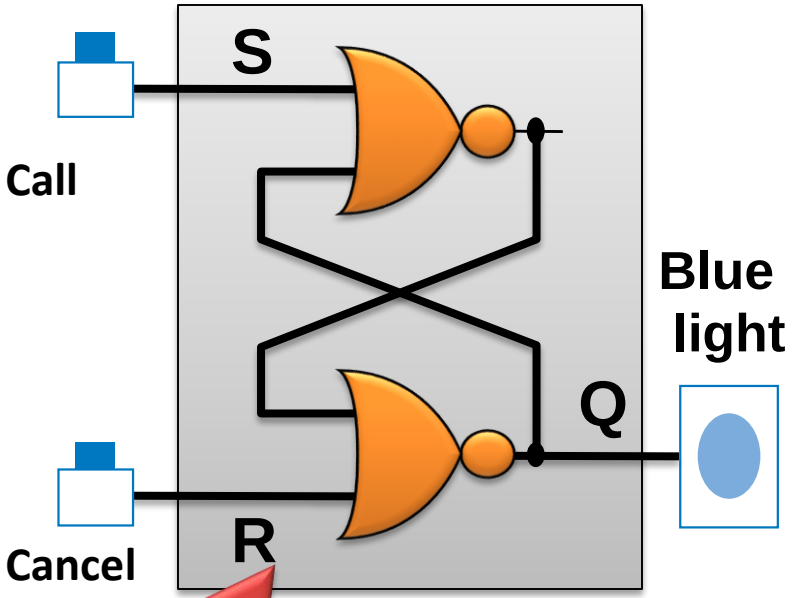
S	R	Q+
0	0	1*0* (Unpredictable)
0	1	1
1	0	0
1	1	Q

Opposite to SR Latch with NOR Gates

Set will do Q=0 and Reset will Q=1

Problem with SR Latch: SR=11

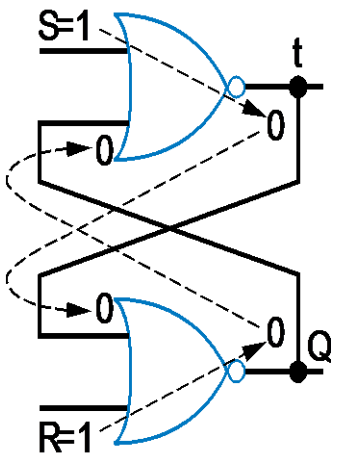
- Problem
  - If S=1 and R=1 simultaneously, we don't know what value Q will take (Unpredictable)

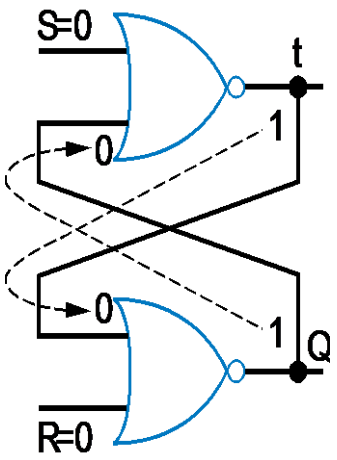


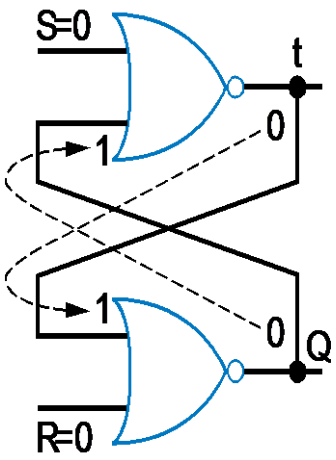
Race Condition :-

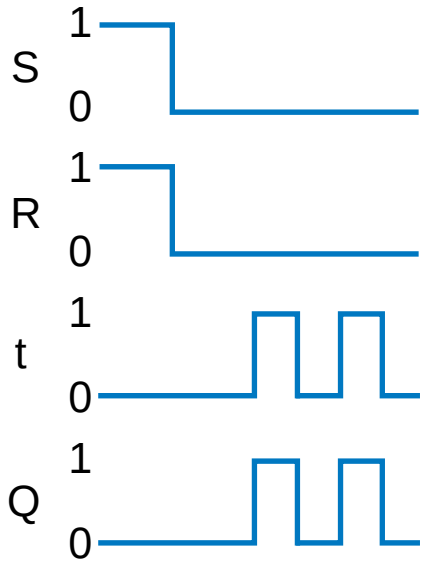
Who will win 0 or 1?
Eventually, what will be the value of Q?

Problem with SR Latch

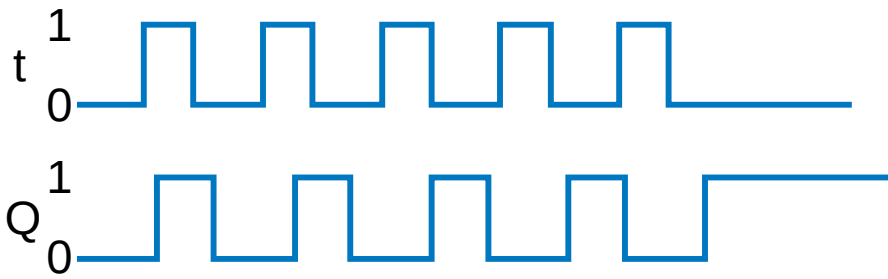






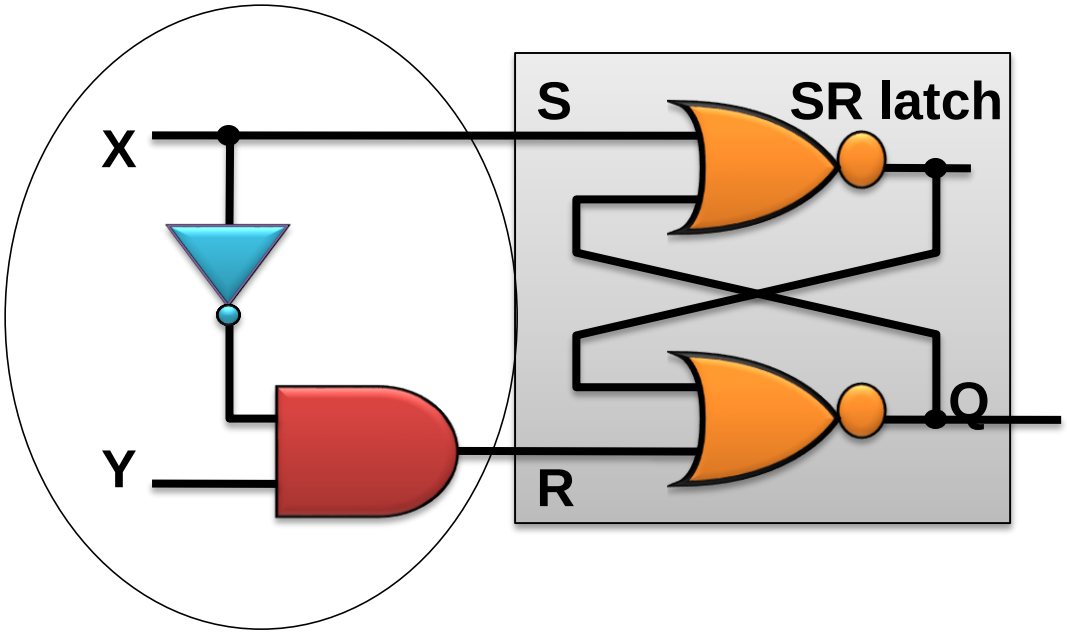


Q may oscillate. Then, because one path will be slightly longer than the other, Q will eventually settle to 1 or 0 – but we don't know which.



Solution to Race Condition

- We try to avoid S=1 and R=1 by the following circuit



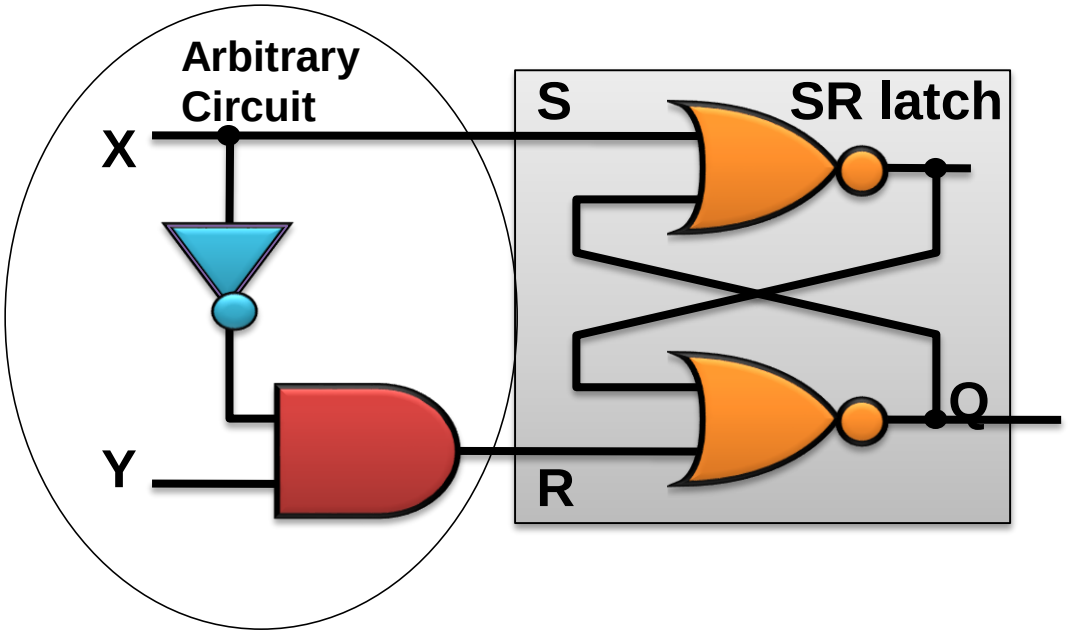
- Can we ensure value S=1 and R=1 will not happen at the same time?

Problem with SR Latch

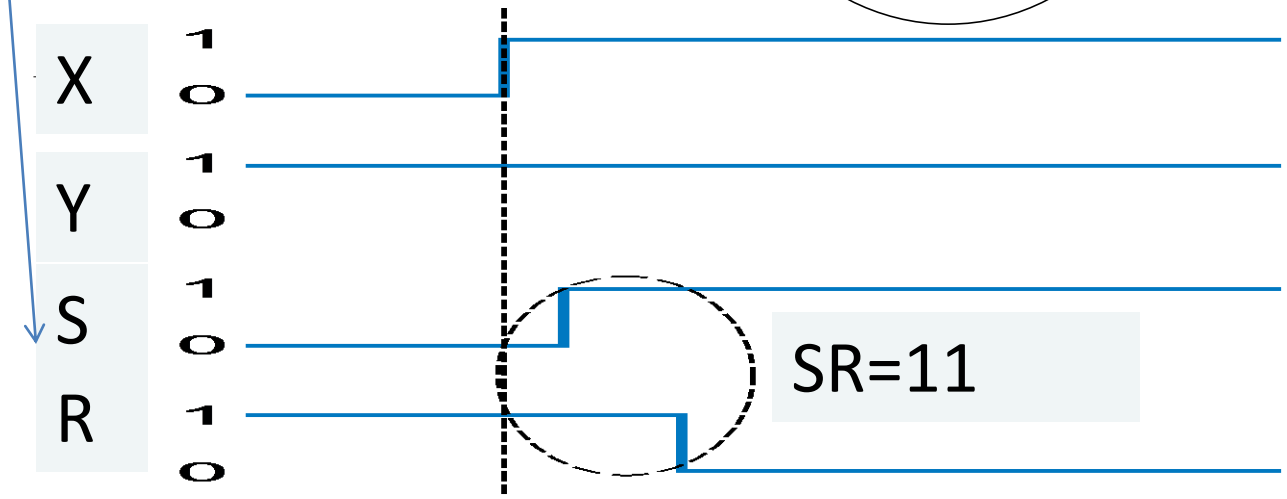
- Problem not just one of a user pressing two buttons at same time
- Can also occur even if SR inputs come from a circuit that supposedly never sets S=1 and R=1 at same time
  - But does, due to different delays of different paths

Problem with SR Latch

- X=1 and Y=1 set



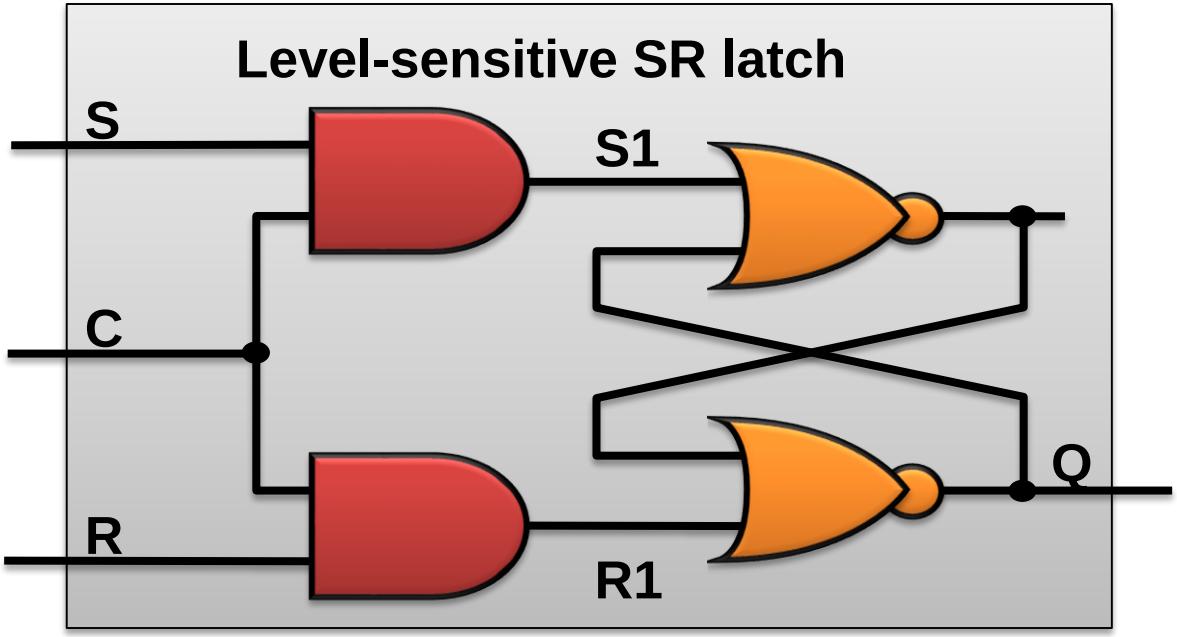
Yesterday it was my mistake

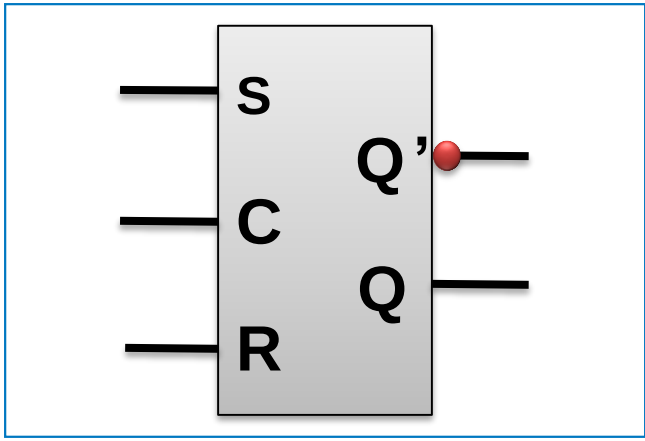


The longer path from X to R than to S causes SR=11 for short time – could be long enough to cause oscillation

Solution: Level-Sensitive SR Latch

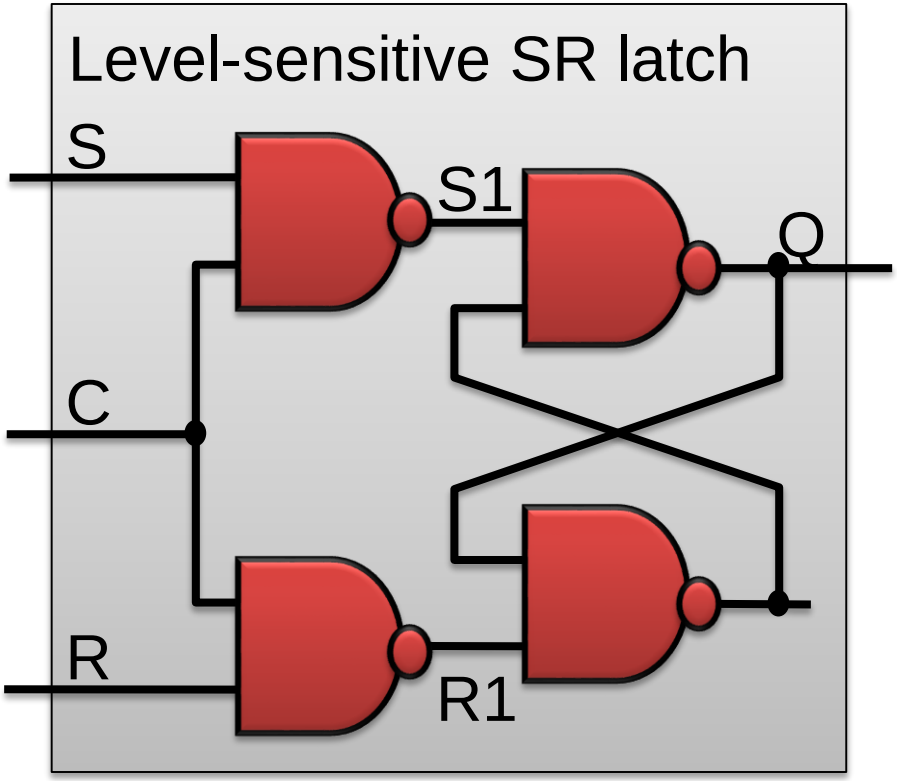
- Add enable input “C or En”
  - Only let S and R change when C =0 : Ensure circuit in front of SR never sets SR=11, except briefly due to path delays
  - Change C to 1 only after sufficient time for S and R to be stable
  - When C becomes 1, the stable S and R value passes through the two AND gates to the SR latch’s S1 R1 inputs.

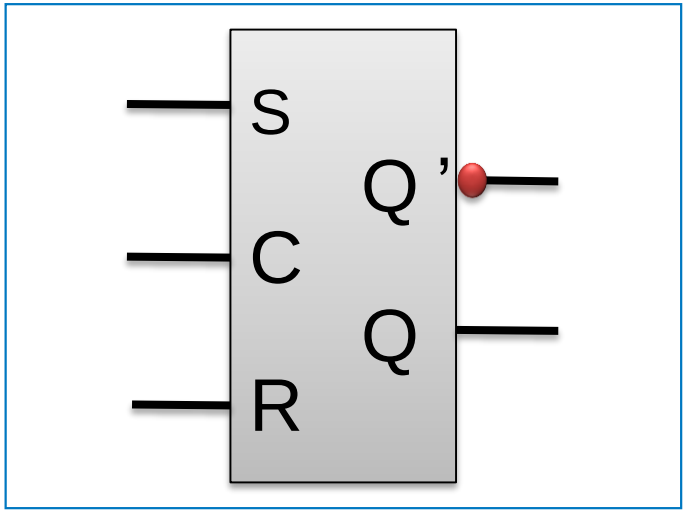




Level-sensitive SR latch symbol

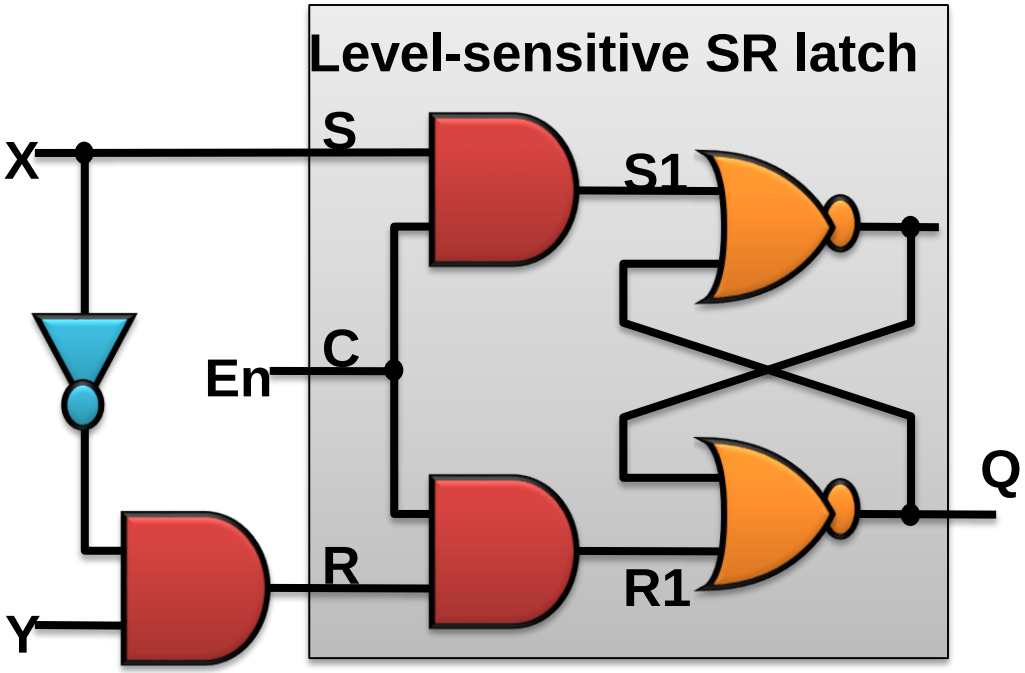
Level Sensitive: SR Latch with NAND Gate

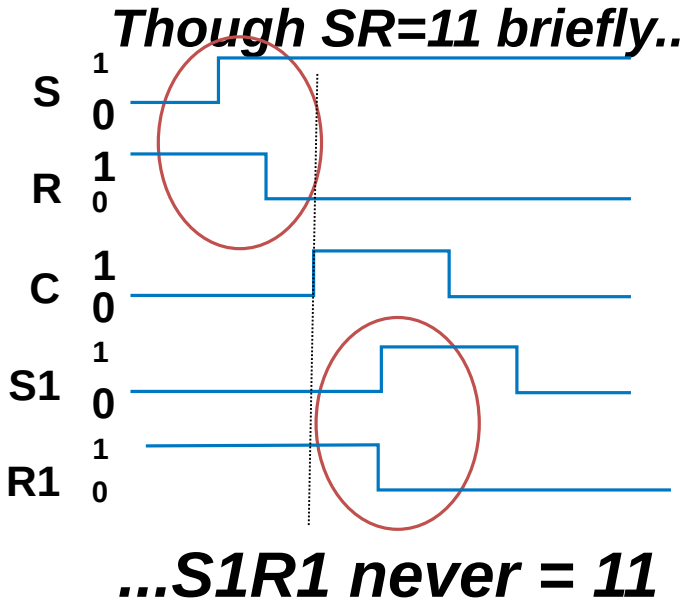




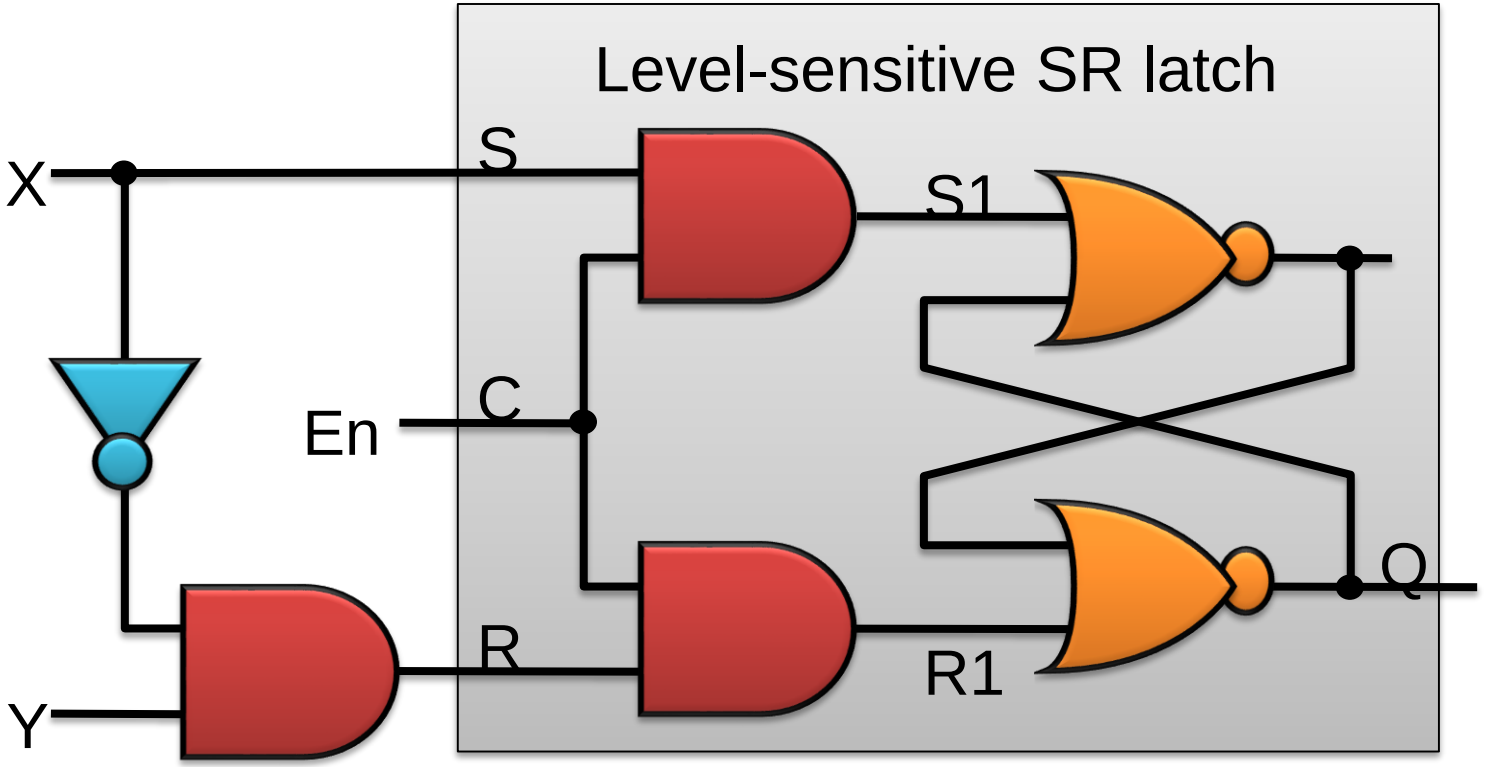
Level-sensitive SR latch symbol

Ensure S=1 and R=1 should not happened to Level-Sensitive SR Latch





Solution: Ensure, Stabilize, Store



Ensure Stage

Never Happens

SR=11

Stabilize Stage

When C=0

Stabilize SR and

Use when C=1

Store Stage

Store bit

Thanks