

ENGN2219/COMP6719

Computer Systems & Organization

Convener: Shoaib Akram

shoaib.akram@anu.edu.au



Australian
National
University




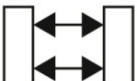
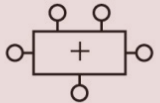

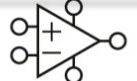


Plan: Week 3

Week 2: Logic gates & Combinational logic

Week 2: Multiplexers, ALU, and decoders

This Week: Boolean algebra (equation minimization)

This Week: Sequential circuits (state & memory)

Application Software		Programs
Operating Systems		Device Drivers
Architecture		Instructions Registers
Micro-architecture		Datapaths Controllers
Logic		Adders Memories
Digital Circuits		AND Gates NOT Gates
Analog Circuits		Amplifiers Filters
Devices		Transistors Diodes
Physics		Electrons

**Broadening our horizon
"one layer at a time"**

Sequential Circuits

- Sequential logic has memory
- The output of sequential logic depends on both the current input values and the *state* of the system
- What is *state*?
 - A set of bits called *state variables* constitute *state*
 - *They inform us everything about the past we need to know to predict the future*

What is State?

- Consider a **very** simple traffic light controller
 - RED, YELLOW, GREEN
 - How many bits do I need to remember the state of the traffic light?
 - We need a minimum of two state variables (or bits) to store the state of the traffic light
- *To remember/store the state of the system (traffic light), we need an element with memory*

Storing One Bit

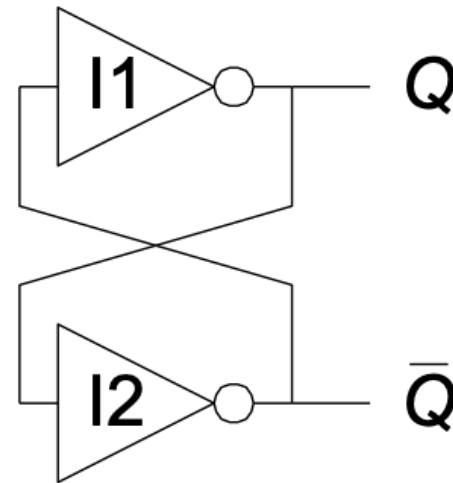
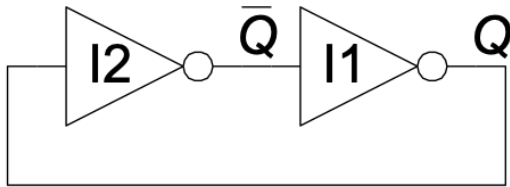
One bit represents two possible states. To store one bit, we need

- An element with two stable states
- The *ability to change the state*

Let us focus on the “*ability to change state*” later and first find the bistable element

Memory

The fundamental building block of memory is a bistable element with two stable states



Cross-coupled inverters: A pair of inverters connected in a loop

Analysis of Bistable Element

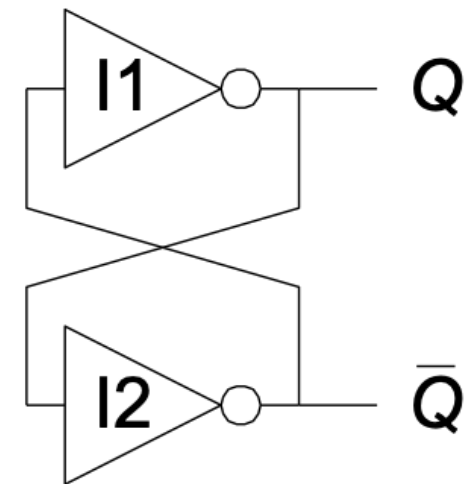
- How should we analyze circuits with cyclic paths?
 - When the circuit is switched on, Q is either 0 or 1 (scenarios)
 - Show that output is stable (consequence – A)
 - Show that Q and Q' are complements of each other (consequence – B)

Scenario # 1: Q = 0 (FALSE)

- I2 receives 0, Q' = 1: B is satisfied
- Q' = 1, Q = 0, consistent with our original assumption and hence stable: A is satisfied

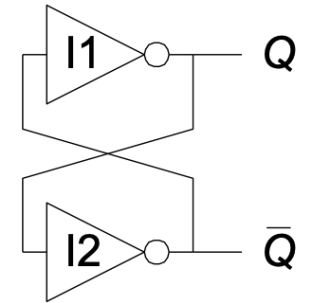
Scenario # 2: Q = 1 (TRUE)

- I2 receives 1, Q' = 0: B is satisfied
- Q' = 0, Q = 1, consistent with our original assumption and hence stable: A is satisfied



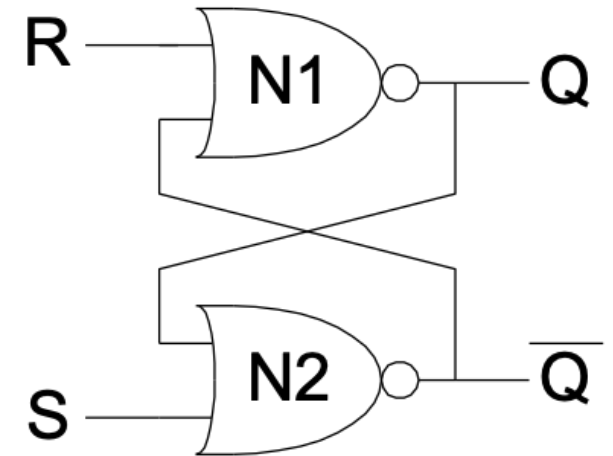
Bistable Element: Observations

- Bistable element has two stable states
 - Can store one bit of information
- The value of Q reveals about past, necessary to explain the future behavior
 - If $Q = 0$, it will remain 0 forever
 - If $Q = 1$, it will remain 1 forever
- Q' is an acceptable choice for storing the state variable
- When power is first applied, the state of the bistable element is unpredictable
 - Bistable element is *not practical* because the user lacks inputs to control state
 - We need something else!



SR Latch

- Two cross-coupled NOR gates
- The state can be controlled with S/R inputs
 - S = Set to 1
 - R = Reset to 0



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

NOR gate revision

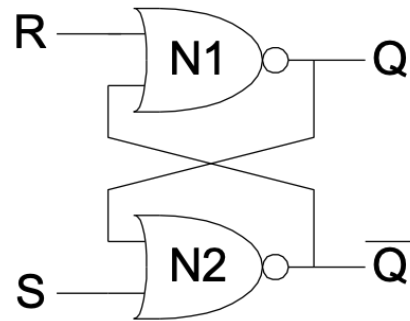
- When both inputs are 0, the output is 1
- If any of the inputs is 1, the output is 0

SR Latch: Analysis

SR latch has inputs (unlike the cross-coupled inverters)
Four scenarios in the truth table (Sim = Simultaneous)

Scenario	S	R	Q	Q'
Sim-0	0	0		
Reset	0	1		
Set	1	0		
Sim-1	1	1		

Whiteboard: SR Latch



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

S	R	Q
0	0	
0	1	
1	0	
1	1	

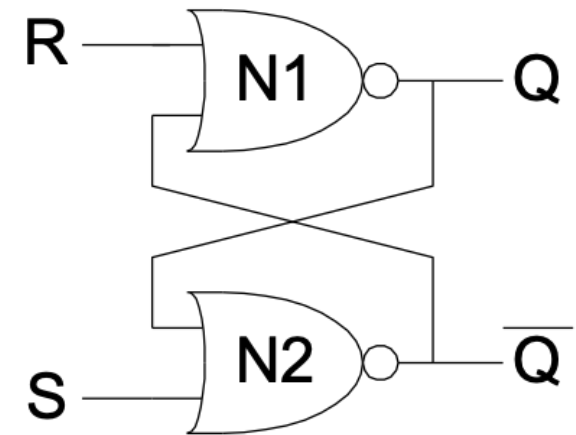
SR Latch: Analysis

Scenario # 1 (**Reset**): R = 1, S = 0

- N1 sees at least one **TRUE** (1) input
 - Q = **FALSE** (0)
- N2 sees both Q and S **FALSE**
 - Q' = **TRUE** (1)

Scenario # 2 (**Set**): R = 0, S = 1

- N1 sees 0 and Q'
 - What is Q'?
- N2 sees at least one **TRUE** input
 - Q' = **FALSE** (0)
- Revisit N1 (R = 0 and Q' = 0)
 - Q = **TRUE** (1)



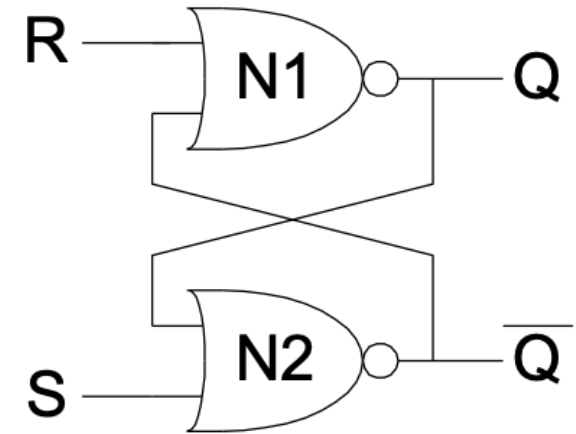
SR Latch: Analysis

Scenario # 3 (Sim-1): R = 1, S = 1

- N1 sees at least one TRUE (1) input
 - Q = FALSE (0)
- N2 sees at least one TRUE (1) input
 - Q' = FALSE (0)

Scenario # 4 (Sim-0): R = 0, S = 0

- N1 sees 0 and Q'
 - What is Q'?
- N2 sees 0 and Q
 - What is Q?
- **We are stuck!**
 - Wait: remember the cross-coupled inverter? Q can be 0 or 1 😊



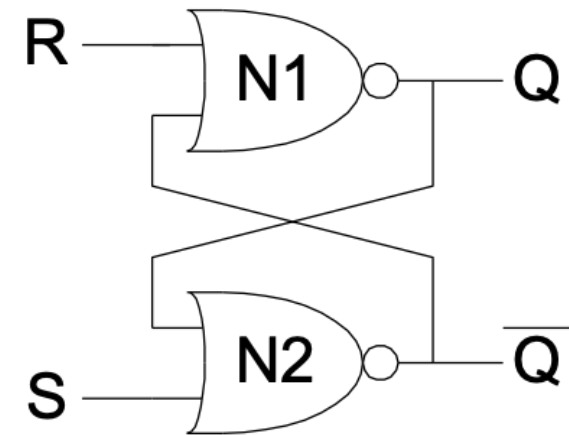
SR Latch: Analysis

Scenario # 4-A (Sim-0): R = 0, S = 0, Q = 0

- N2 sees S = 0 and Q = 0
 - $Q' = 1$
- N1 sees one TRUE (1) input
 - Q = 0 (hindsight: Q is indeed 0 as assumed)

Scenario # 4-B (Sim-0): R = 0, S = 0, Q = 1

- N2 sees Q = 1
 - $Q' = \text{FALSE}$
- N1 receives two FALSE inputs
 - Q = 1 (hindsight: Q is indeed 1 as assumed)



SR Latch: Analysis

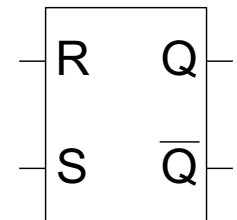
SR latch has inputs (unlike the cross-coupled inverters)
Four scenarios in the truth table (Sim = Simultaneous)

Scenario	S	R	Q	Q'
Sim-0	0	0	Q_{prev}	Q'_{prev}
Reset	0	1	0	1
Set	1	0	1	0
Sim-1	1	1	0	0

SR Latch: Observations

- SR latch is a bistable element but it's state can be controlled
 - To **set** a bit means to make it **TRUE**. To **reset** is to make it **FALSE**
- Q accounts for the entire history of past inputs
 - All prior patterns of setting and resetting are irrelevant
 - The most recent set/reset event predicts the future behavior of the SR latch
- Asserting both set/reset to **1** does not make sense
 - Neither intuitively nor physically
 - The circuit outputs **0** on Q and Q' which is inconsistent
 - We need something else!

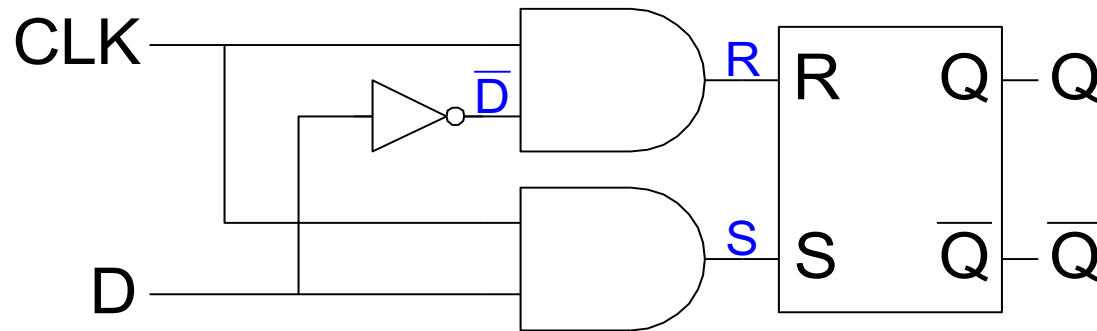
SR Latch
Symbol



D Latch

- Two drawbacks of SR latch
 - Strange behavior when $S = 1$ and $R = 1$
 - S and R inputs serve two roles: *what* the state is and *when* the state changes
- Designing sequential circuits is easier when we have control over *when* the state changes
- The **D latch** overcomes the two drawbacks
 - A data input (**D**) controls what the next state should be
 - The clock input (**CLK**) controls when the state should change

D Latch



Scenario # 1: CLK = 0, S = 0, R = 0, D = X

- The value of D is irrelevant
- $Q = Q_{\text{prev}}$ (remember the old value)

Latch is opaque (blocks new data from flowing to Q)

Scenario # 2: CLK = 1, D = 0, S = 0, R = 1

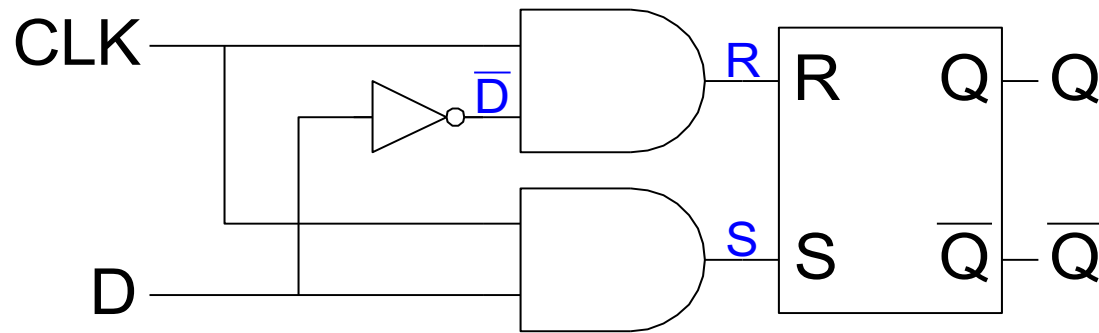
- The latch is reset

Latch is transparent (acts like a buffer)

Scenario # 3: CLK = 1, D = 1, S = 1, R = 0

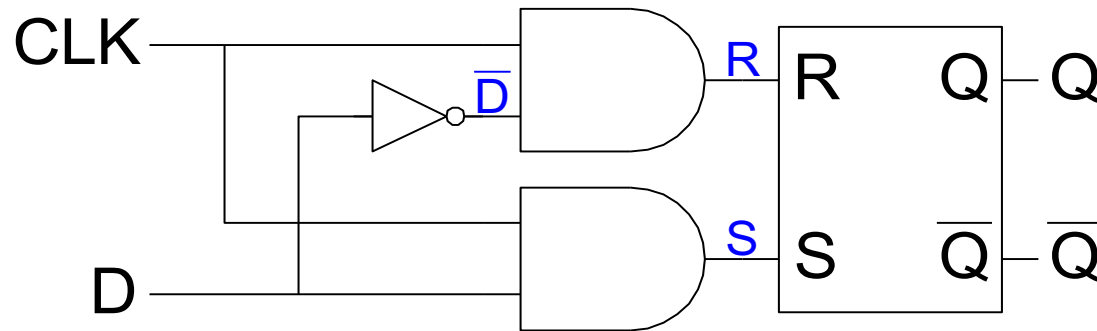
- The latch is set

Whiteboard: D Latch



S	R	Q
0	0	Q_{prev}
0	1	0
1	0	1
1	1	0

D Latch: Truth Table

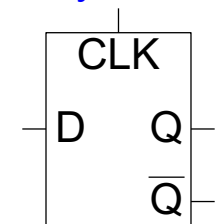


Scenario	CLK	D	Q	Q'
Opaque	0	X	Q_{prev}	Q'_{prev}
Transparent/0	1	0	0	1
Transparent/1	1	1	1	0

D Latch: Observations

- D latch is a *level-triggered* or a *level-sensitive* circuit
 - Reacts to the level (0 or 1) of the CLK input
- D latch avoids the awkward case of both S and R asserted
- D latch changes its state continuously when CLK = 1

D Latch
Symbol



Designing correct & efficient sequential circuits becomes easier when the state changes only at a specific instant in time instead of changing continuously. We need something else!

D Flip-Flop

- What is the problem with D latch?
 - The output changes continuously when CLK = 1
- We need a circuit element that *samples* the data input when CLK changes from 0 to 1 (or 1 to 0)
 - This process is called sampling at the edge of a clock input
- The D flip-flop is called *edge-triggered*
- At all other times, the D flip-flop simply remembers its state

