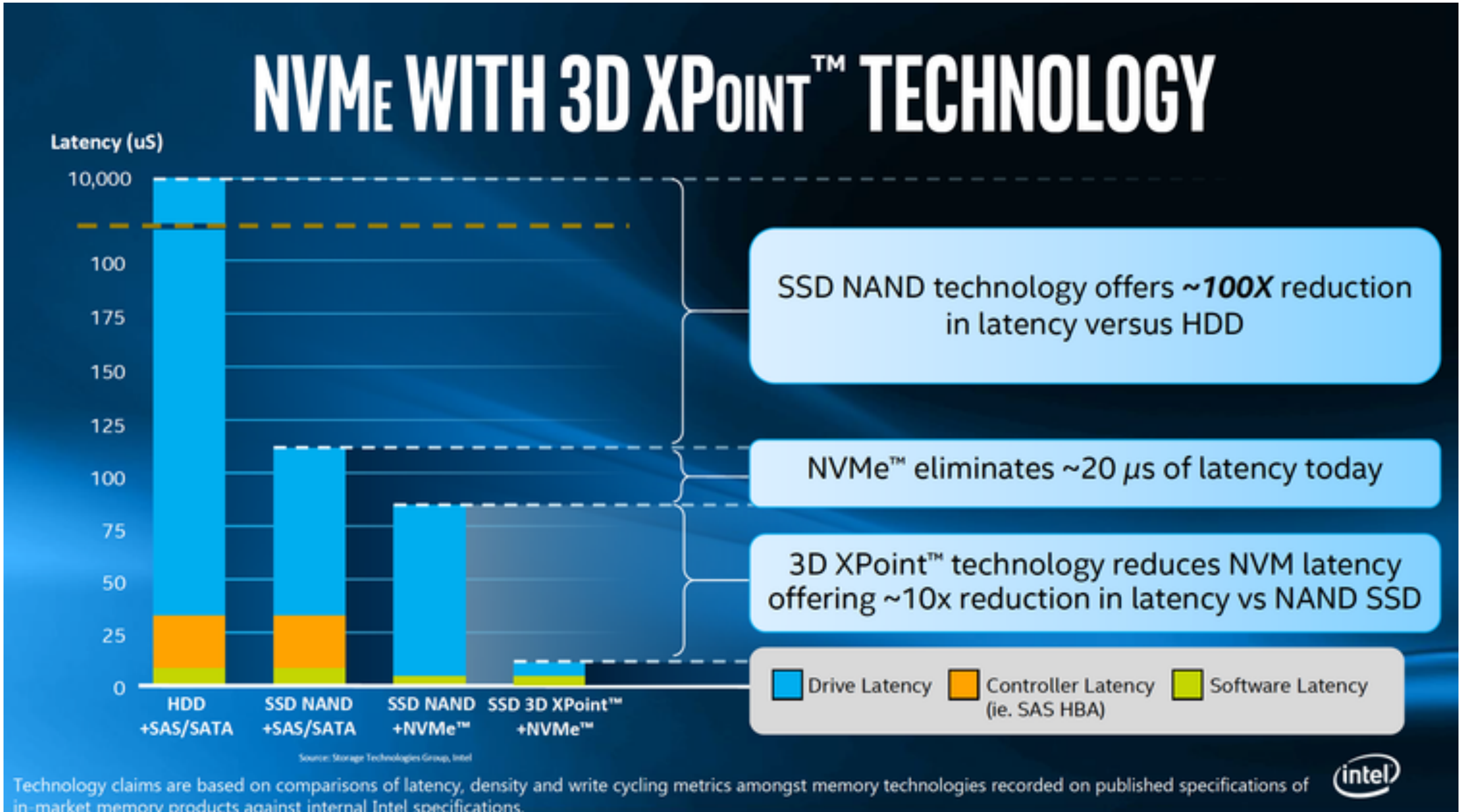
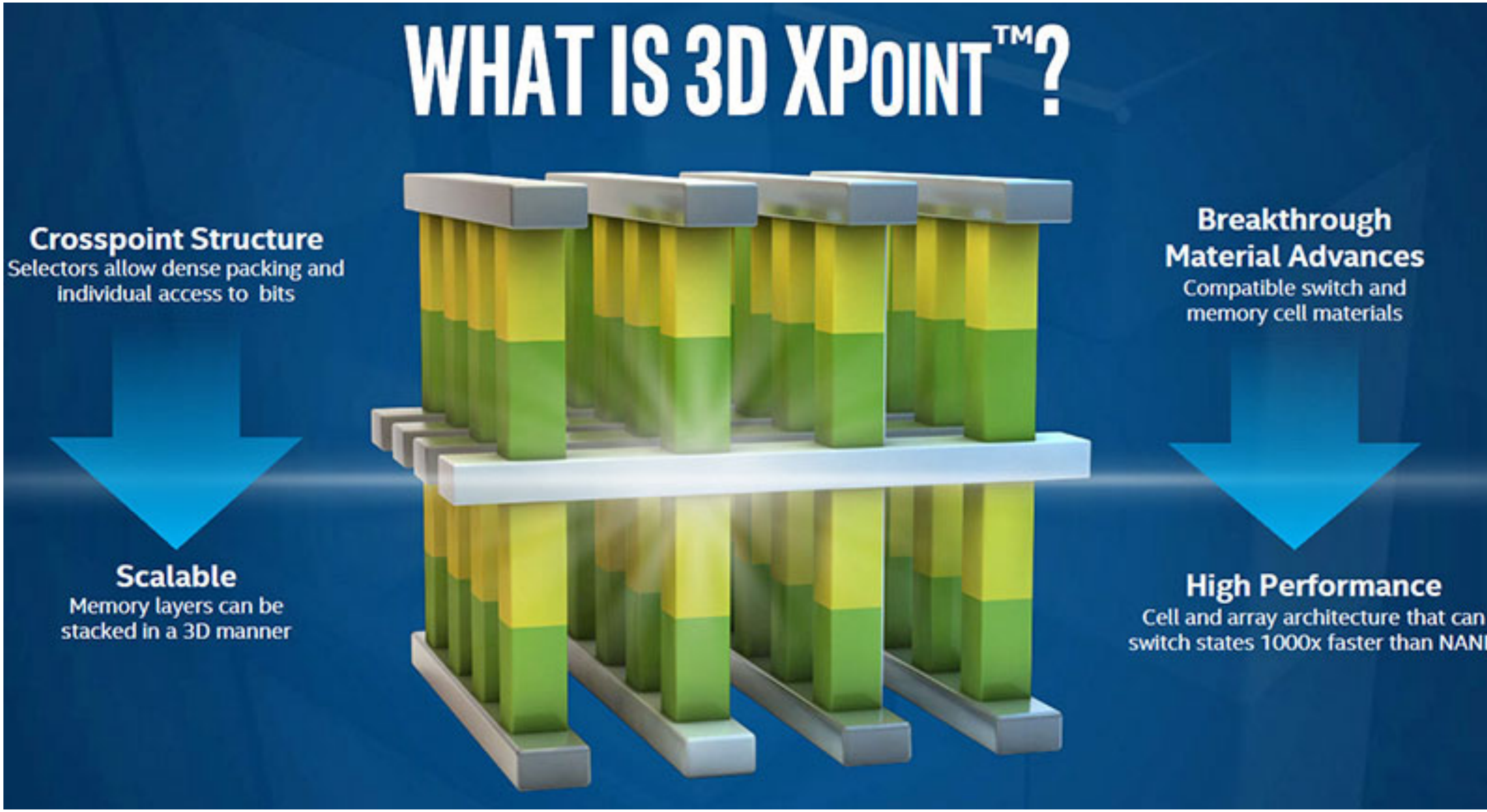
Chapter 4 Sequential Circuits

- **Part 1 - Storage Elements and Analysis**
  - Introduction to sequential circuits
  - Types of sequential circuits
  - Storage elements
    - Latches
    - Flip-flops
  - Sequential circuit analysis
    - State tables
    - State diagrams
  - Circuit and System Timing
- **Part 2 - Sequential Circuit Design**
  - Specification
  - Assignment of State Codes
  - Implementation

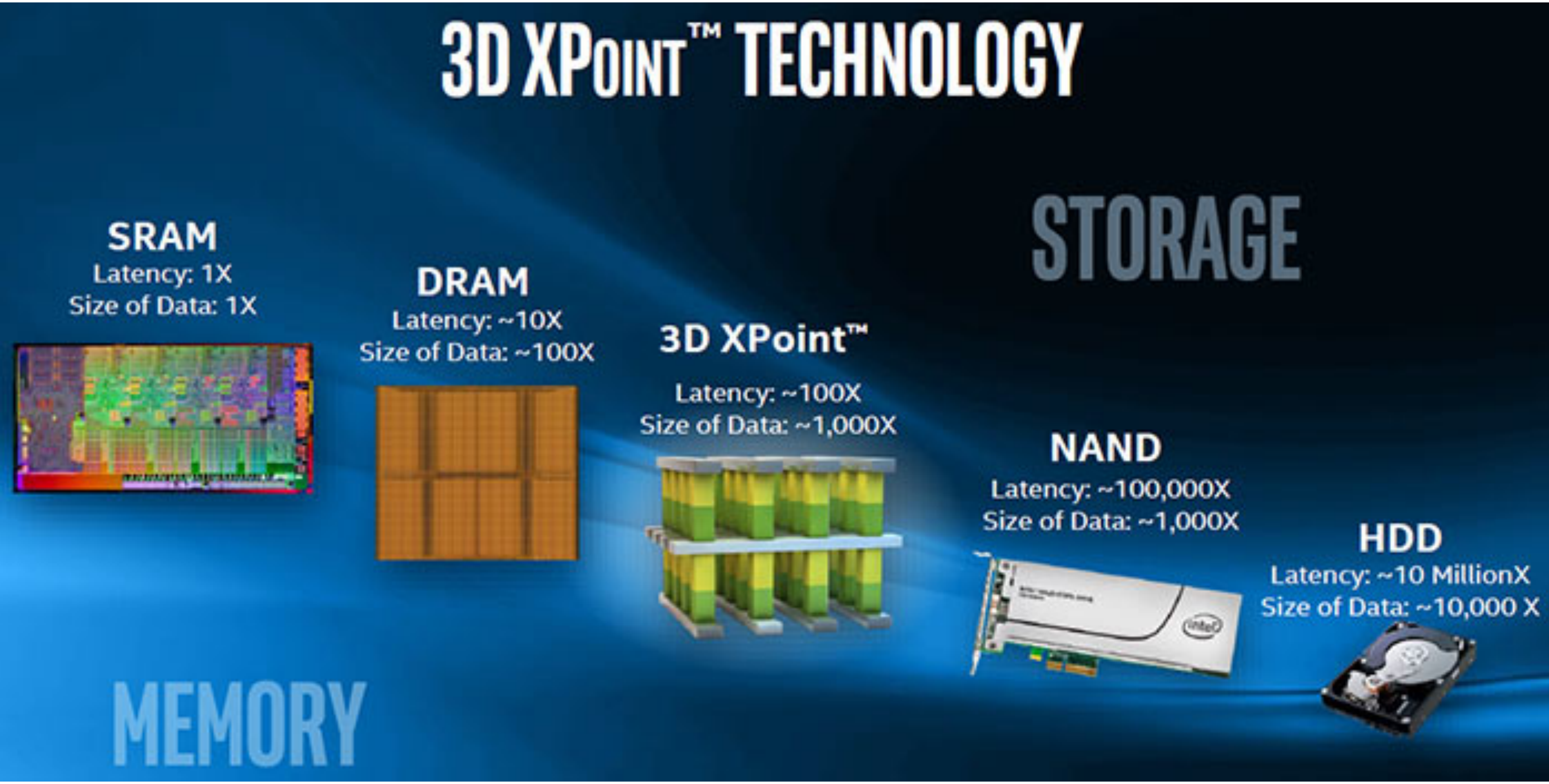
Intel&Micron 3D X-Point Memory



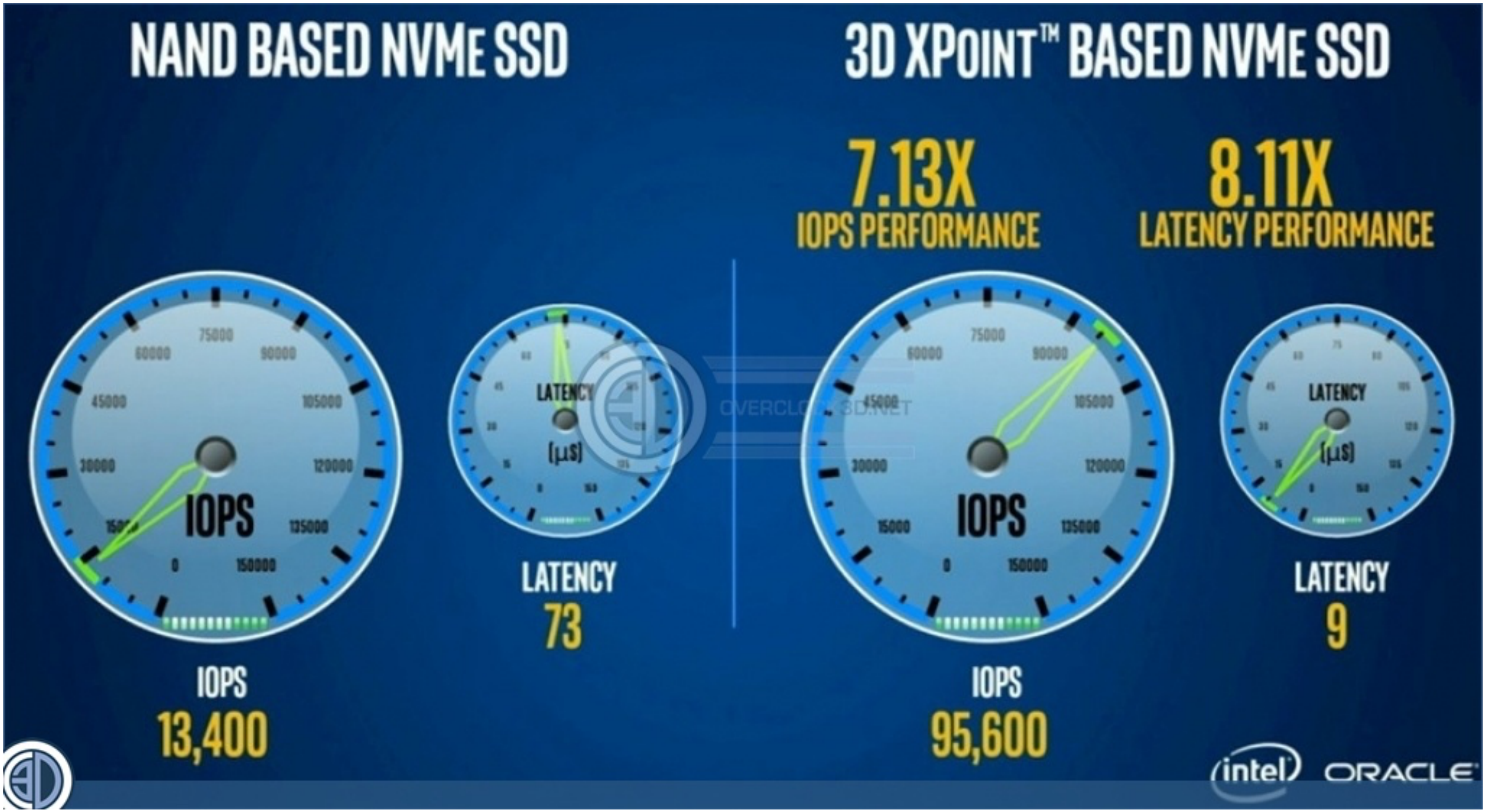
Intel&Micron 3D X-Point Memory



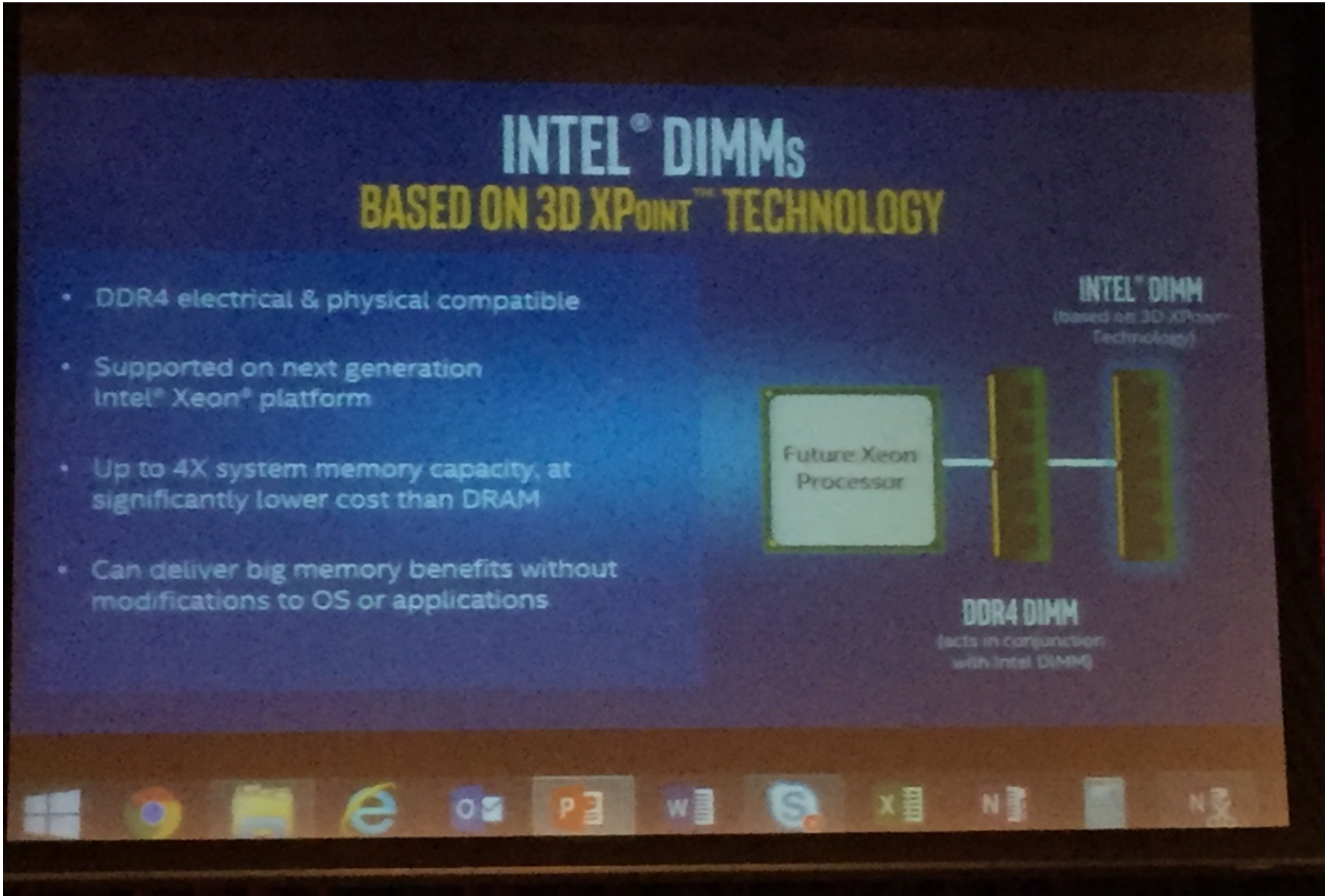
Intel&Micron 3D X-Point Memory



Intel&Micron 3D X-Point Memory



Intel&Micron 3D X-Point Memory



Overview

- **Part 1 - Storage Elements and Analysis**
  - **Introduction to sequential circuits**
  - **Types of sequential circuits**
  - **Storage elements**
    - **Latches**
    - **Flip-flops**
  - **Sequential circuit analysis**
    - **State tables**
    - **State diagrams**
  - **Circuit and System Timing**
- **Part 2 - Sequential Circuit Design**
  - **Specification**
  - **Assignment of State Codes**
  - **Implementation**

Sequential Circuits

- **The output of a combinational circuit depends solely upon the input. The implication is that combinational circuits have no *memory*.**
- **We need circuits whose output depends upon both the input of the circuit and its previous state. In other words, we need circuits that have *memory*.**

Three Characteristics

For a device to serve as a memory, it must have three characteristics:

- The device must have two stable states
- There must be a way to read the state of the device
- There must be a way to set the state at least once.

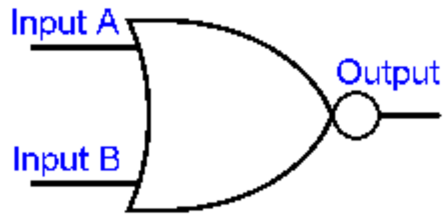
Using Feedback Technique

- **So far, the logical flow in the circuits we've studied has been from input to output. Such a circuit is called *acyclic*.**

- **It's possible to produce circuits with memory using the digital logic gates we've already seen.**

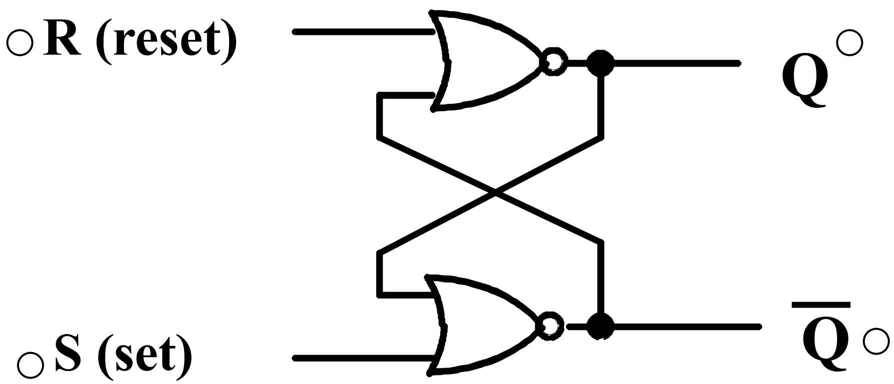
- **We introduce a circuit in which the output is fed back to the input, giving the circuit memory.**

The NOR gate



- $F=\overline{A+B}$
- How about we cross-couple two NOR gates?

S-R Latch

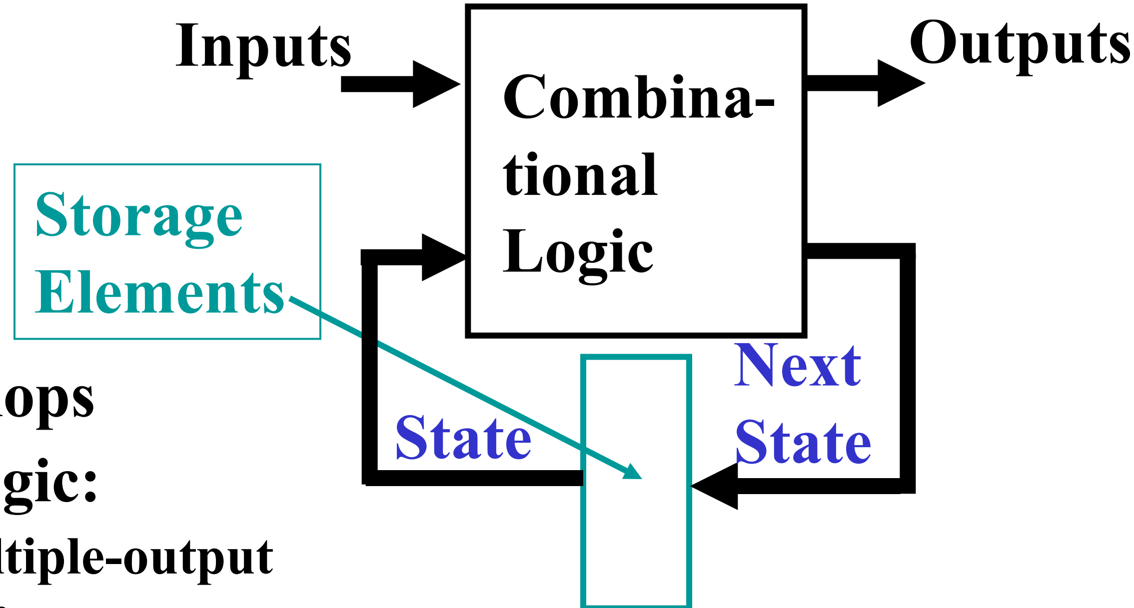


Explanation of Last Slide

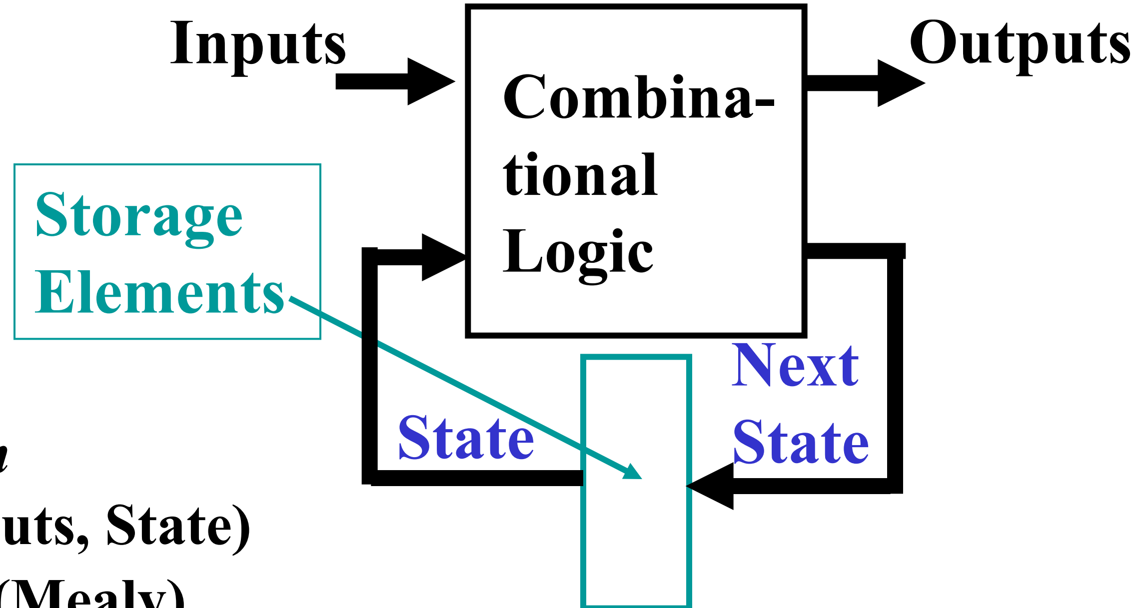
- The output of a NOR gate is true only when both inputs are false. ($F=\underline{X+Y}$)
- The output of each NOR gate is fed back to the input of the other.
- S sets the latch, causing Q to become true. R resets the latch.
- This means that if the output of one NOR gate is true, the output of the other must be false.
- Now press the S button. The output of the upper NOR gate, Q is forced to true, allowing the output of the lower NOR to become false.
- Press S again to turn it off. The output of the circuit is unchanged.
- Examine the circuit to understand why. What has happened is that we have stored the value of S. Turning S on and off again does not change the output.
- With S off, turn R on, then off again. What happens? Why.

Introduction to Sequential Circuits

- A Sequential circuit contains:
  - Storage elements: Latches or Flip-Flops
  - Combinatorial Logic:
    - Implements a multiple-output switching function
    - Inputs are signals from the outside.
    - Outputs are signals to the outside.
    - Other inputs, State or Present State, are signals from storage elements.
    - The remaining outputs, Next State are inputs to storage elements.



Introduction to Sequential Circuits



- Combinatorial Logic
  - *Next state function*
    Next State = f(Inputs, State)
  - *Output function* (Mealy)
    Outputs = g(Inputs, State)
  - *Output function* (Moore)
    Outputs = h(State)
- Output function type depends on specification and affects the design significantly

Types of Sequential Circuits

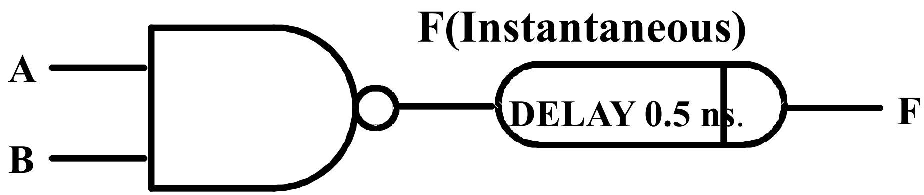
- Depends on the times at which:
  - storage elements observe their inputs, and
  - storage elements change their state
- Synchronous
  - Behavior defined from knowledge of its signals at discrete instances of time
  - Storage elements observe inputs and can change state only in relation to a timing signal (clock pulses from a clock)
- Asynchronous
  - Behavior defined from knowledge of inputs at any instant of time and the order in continuous time in which inputs change
  - If clock just regarded as another input, all circuits are asynchronous!
  - Nevertheless, the synchronous abstraction makes complex designs tractable!

Discrete Event Simulation

- **In order to understand the time behavior of a sequential circuit we use <u>discrete event simulation</u>.**
- **Rules:**
  - **Gates modeled by an <u>ideal</u> (instantaneous) function and a <u>fixed gate delay</u>**
  - **Any <u>change in input values</u> is evaluated to see if it causes a <u>change in output value</u>**
  - **Changes in output values are scheduled for the fixed gate delay after the input change**
  - **At the time for a scheduled output change, the output value is changed along with any inputs it drives**

Simulated NAND Gate

- Example: A 2-Input NAND gate with a 0.5 ns. delay:

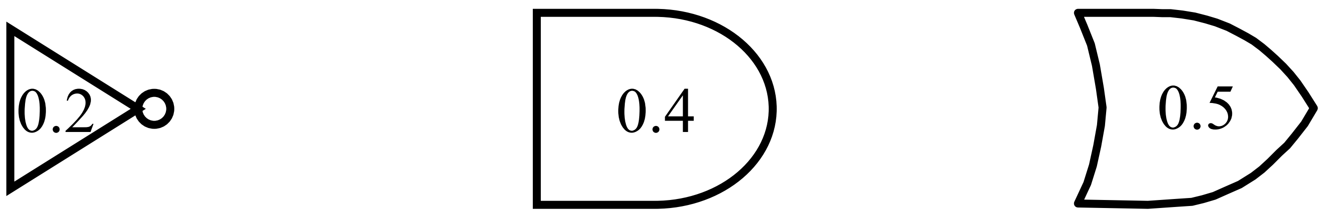


- Assume A and B have been 1 for a long time
- At time t=0, A changes to a 0 at t= 0.8 ns, back to 1.

t (ns)	A	B	F(I)	F	Comment
$-\infty$	1	1	0	0	A=B=1 for a long time
0	$1 \Rightarrow 0$	1	$1 \Leftarrow 0$	0	F(I) changes to 1
0.5	0	1	1	$1 \Leftarrow 0$	F changes to 1 after a 0.5 ns delay
0.8	$1 \Leftarrow 0$	1	$1 \Rightarrow 0$	1	F(Instantaneous) changes to 0
0.13	1	1	0	$1 \Rightarrow 0$	F changes to 0 after a 0.5 ns delay

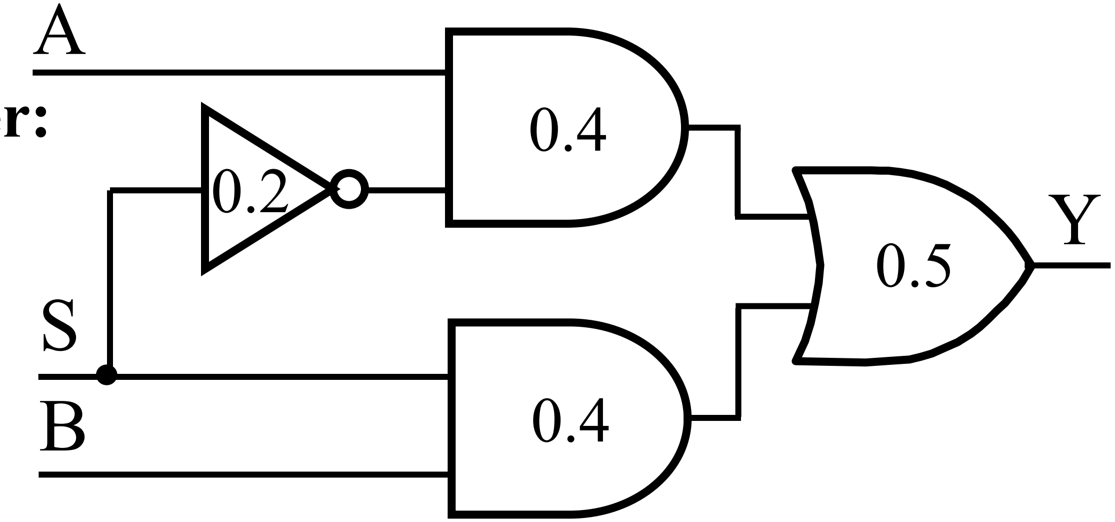
Gate Delay Models

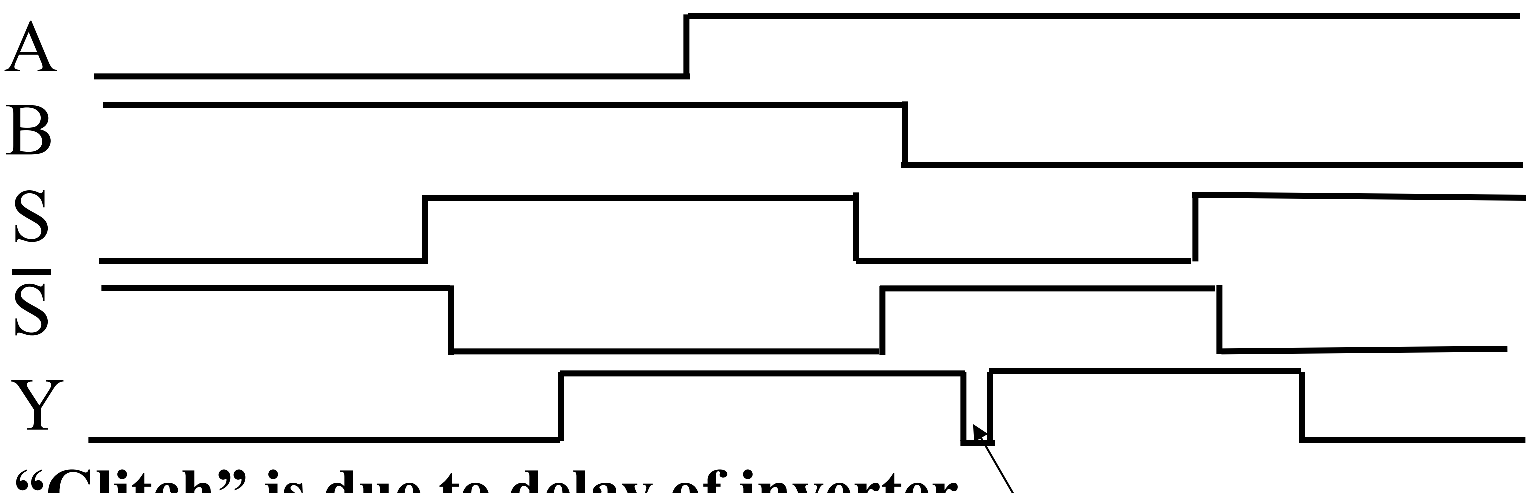
- **Suppose gates with delay *n* ns are represented for *n* = 0.2 ns, *n* = 0.4 ns, *n* = 0.5 ns, respectively:**



Circuit Delay Model

- **Consider a simple 2-input multiplexer:**
- **With function:**
  - **Y = A for S = 0**
  - **Y = B for S = 1**

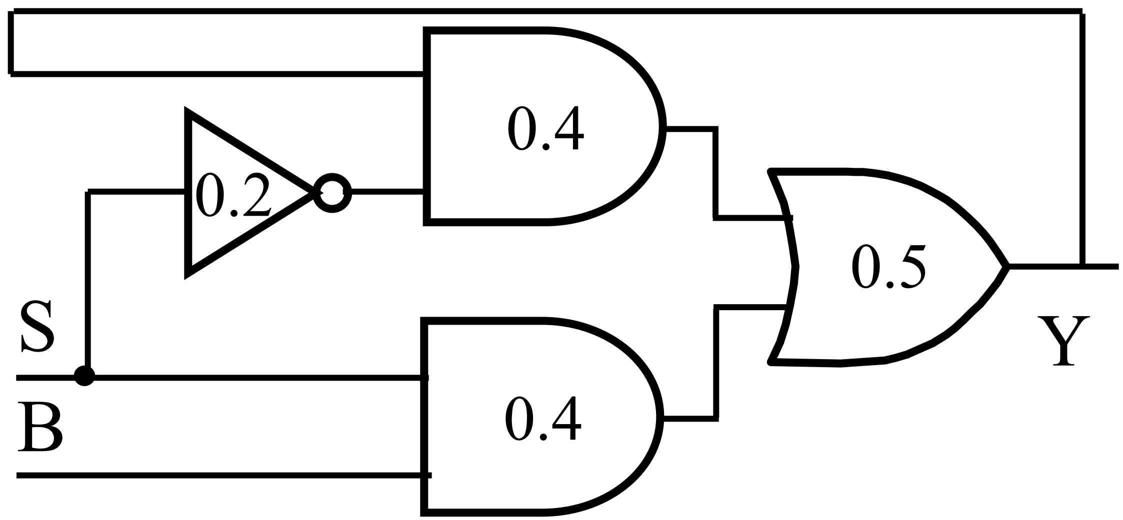


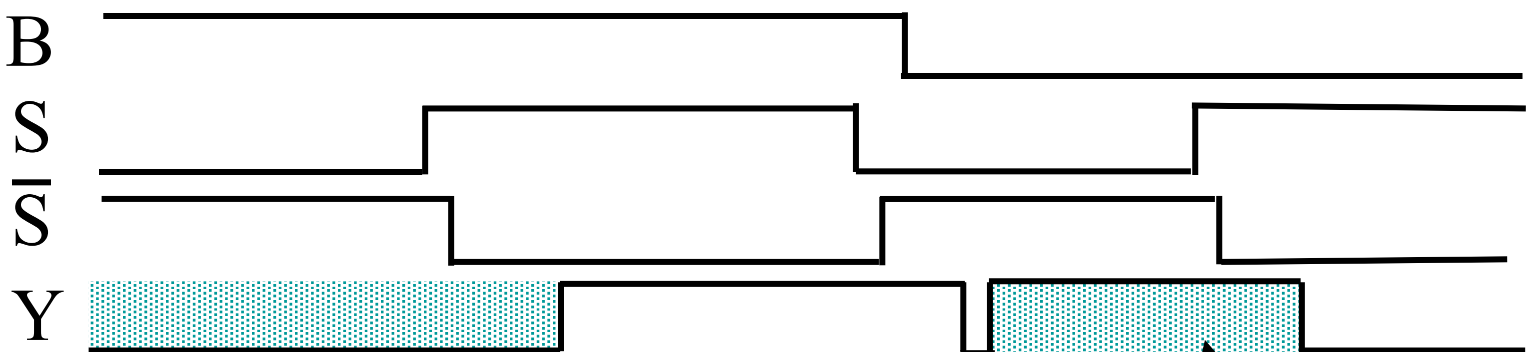


- **"Glitch" is due to delay of inverter**

Storing State

- **What if A connected to Y?**
- **Circuit becomes:**
- **With function:**
  - **Y = B for S = 1, and Y(t) dependent on Y(t – 0.9) for S = 0**





- **The simple <u>combinational circuit</u> has now become a <u>sequential circuit</u> because its output is a function of a time sequence of input signals!**

Y is stored value in shaded area

Storing State (Continued)

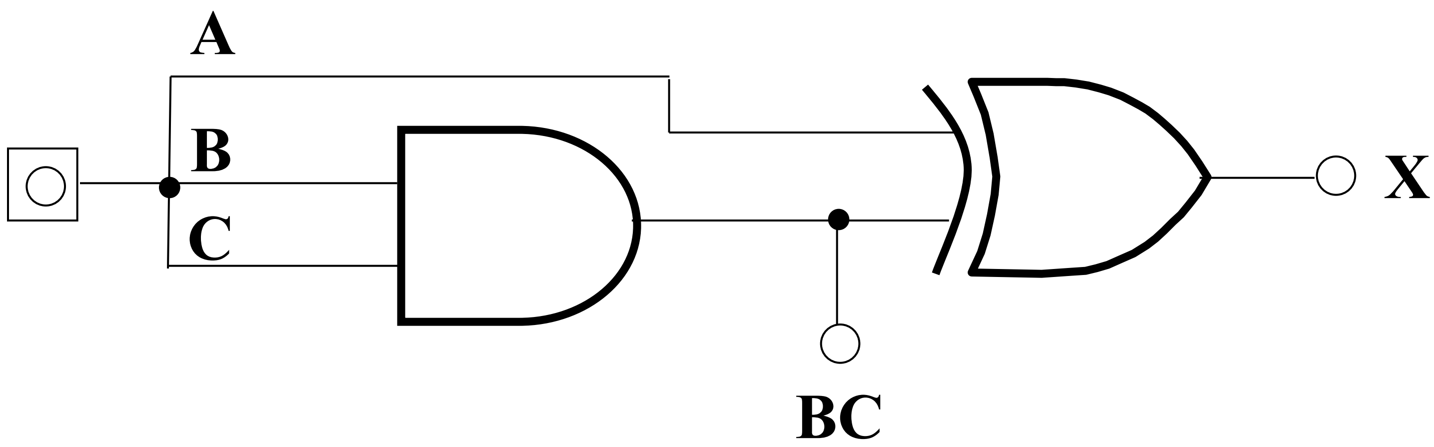
- **Simulation example as input signals change with time. Changes occur every 100 ns, so that the tenths of ns delays are negligible.**

Time ↓

B	S	Y	Comment
1	0	0	Y “remembers” 0
1	1	1	Y = B when S = 1
1	0	1	Now Y “remembers” B = 1 for S = 0
0	0	1	No change in Y when B changes
0	1	0	Y = B when S = 1
0	0	0	Y “remembers” B = 0 for S = 0
1	0	0	No change in Y when B changes

- **Y represent the <u>state</u> of the circuit, not just an output.**

Where Glitch From? (First Look)

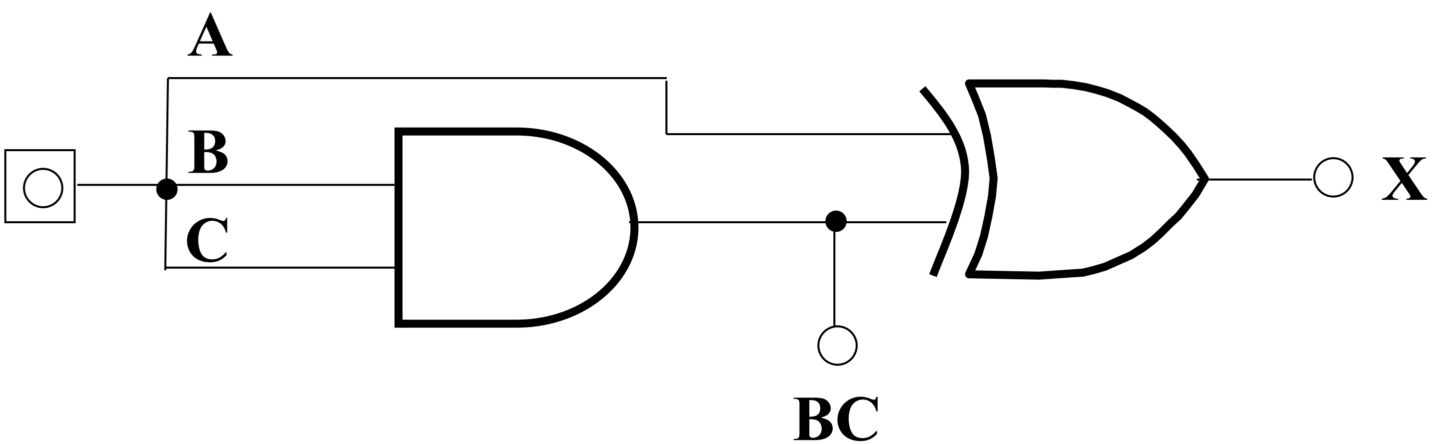


1. In ordinary circumstances, the three inputs A, B, and C would come from other circuits.

2. We've wired them all to one pushbutton to make a point.

3. If you study the circuit, you will see that the output should be zero or false regardless of the input. In Reality ?

Where Glitch From? (Second Look)

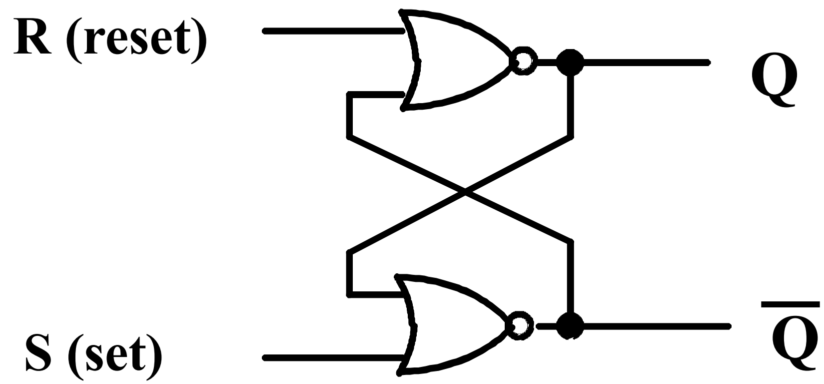


•A circumstance where timing dependencies can briefly cause incorrect output is called a hazard.

•If the output of this figure were connected to the S input of an S-R latch. The latch could be set to true when it should not be. Storing an incorrect value in this way is called a glitch.

Basic (NOR) S – R Latch

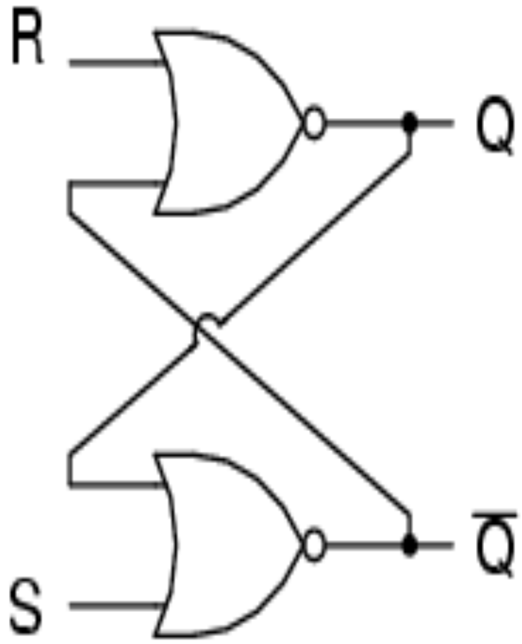
- **Cross-coupling two NOR gates gives the S – R Latch:**
- **Which has the time sequence behavior:**



Time ↓

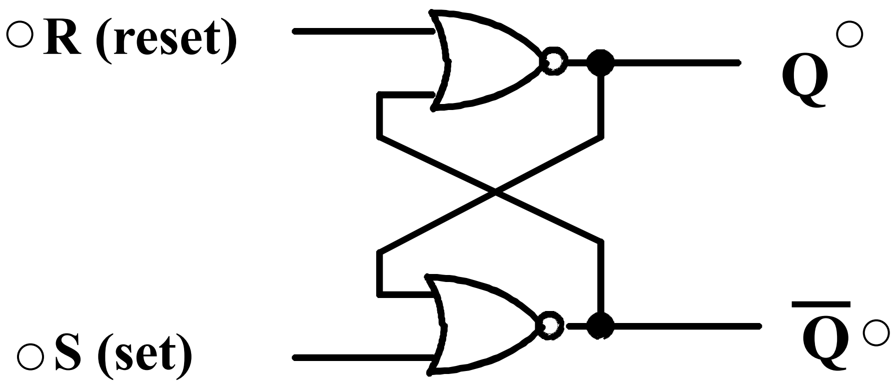
R	S	Q	$\overline{Q}$	Comment
0	0	?	?	Stored state unknown
0	1	1	0	"Set" Q to 1
0	0	1	0	Now Q "remembers" 1
1	0	0	1	"Reset" Q to 0
0	0	0	1	Now Q "remembers" 0
1	1	0	0	Both go low
0	0	?	?	Unstable!

S-R Latch



S	R	Q	$\overline{Q}$
0	0	latch	latch
0	1	0	1
1	0	1	0
1	1	0	0

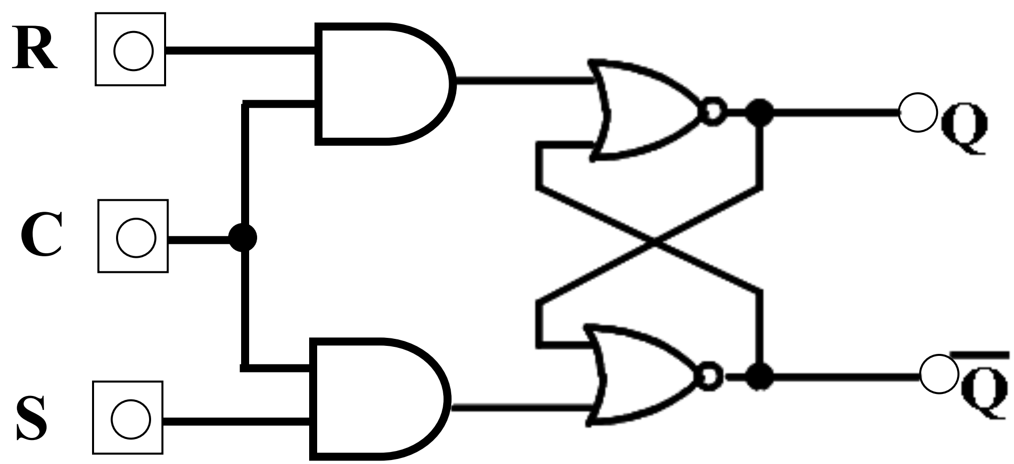
S-R Latch Explanation



How To Avoid Glitches ?

- In order to avoid glitches, we want to design storage elements that only accept input when ordered to do so.
- We will give the order only after the combinational circuits that compute the input to the storage device have had a chance to settle to their correct values.
- One way to do that is to interpose AND gates between the S and R inputs and the latch circuit.

Clocked S - R Latch



Explanation of Last Slide

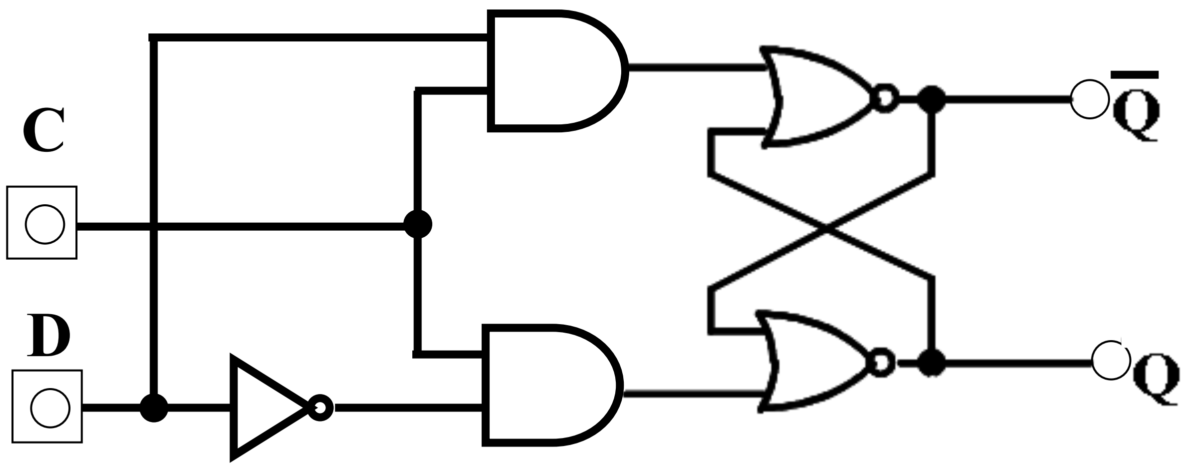
- Originally, S=R=C=0 and Q=0, Q'=1.
- Click R to turn it on, no change
- Click S to turn it on, no change
- Click R to turn it off, no change
- Click S to turn it off, no change
- Click C to turn it on, now this latch is enabled.
- Click S to turn it on, please note that Q becomes on and Q' becomes off
- Click S to turn it off, no change
- Click R to turn it on, please note that Q' becomes on and Q becomes off
- Click R to turn it off, no change
- Click R to turn it on, no change
- Click S to turn it on, please note that both Q and Q' become off, which means this Clocked S-R latch cannot fix S=R=1 input issue.

A problem with the clocked S-R latch

- **Note that clocking does not help with the problem of S=R=1. So, we still need an improved latch to overcome this problem.**

1. **Usually what we want to do with a storage device is store one bit of information.**
2. **The need for explicitly setting and resetting the latch adds complexity.**
3. **What we would really like is a circuit that has a data input D and a data output Q. When the clock signal is high, whatever appears on D should be stored in Q.**

Clocked D-Latch



Explanation of clocked D-Latch

- It has a data input, D, and a control input, C.
- The data input is connected through an AND gate to the S input of an S-R latch. It is also connected through an inverter and an AND gate to the R input.
- The other inputs of the two AND gates are connected to the C input of the circuit.
- If C is false, no signals reach the latch and its state remains unchanged.
- If C is true and D is true, the S input of the latch is true and the latch stores a value of true, which is equal to D.
- If C is true and D is false, the R input of the latch is driven through the inverter and a value of false, which is equal to D, is stored.

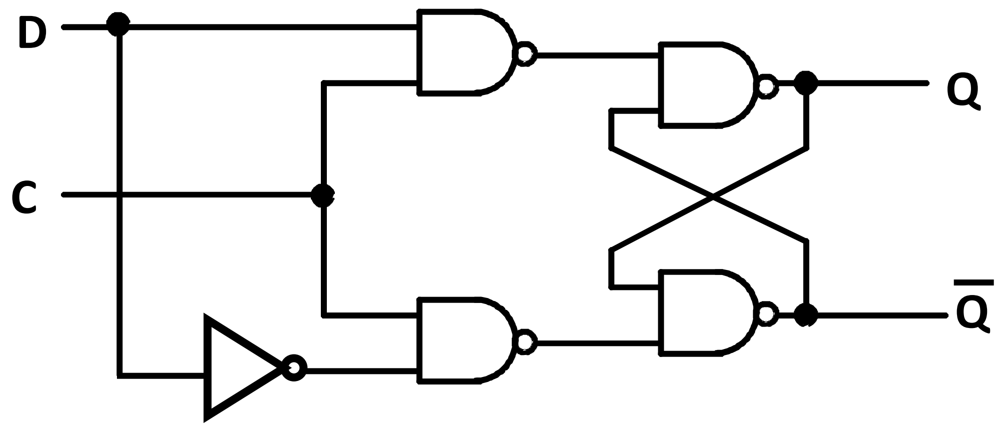
Clocked D-Latch Is Level Triggered

- **As long as the C input is true, changes to D are reflected in the output of the circuit.**

- **The clocked D-latch is a *level triggered* device. Whether it stores data depends upon *level* at C.**

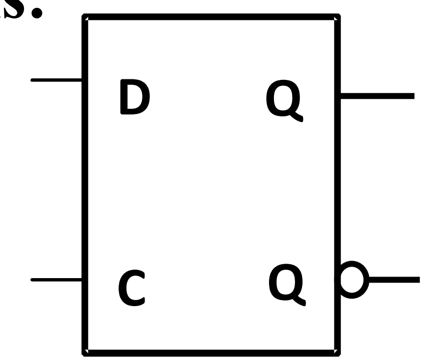
D Latch

- Adding an inverter to the S-R Latch, gives the D Latch:
- Note that there are no “indeterminate” states!



C	D	Q(t+1)	Comment
0	X	0	No change
1	0	0	Q=0, Reset state
1	1	1	Q=1, Set state

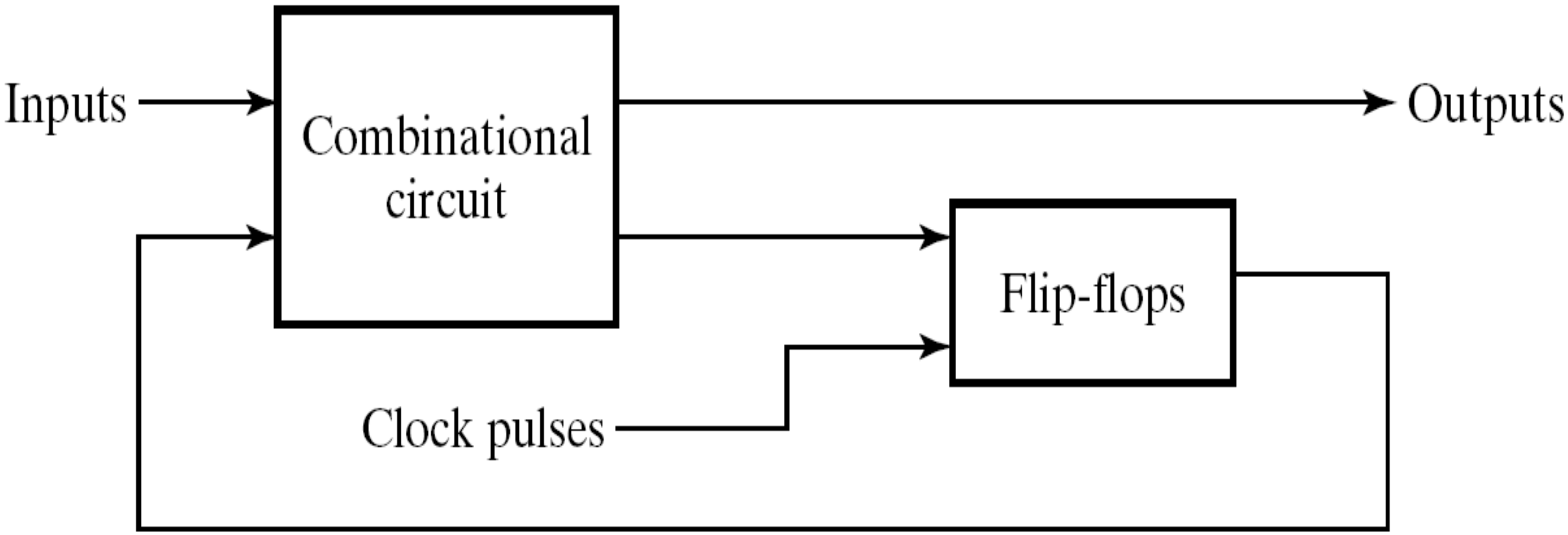
The graphic symbol for a D Latch is:



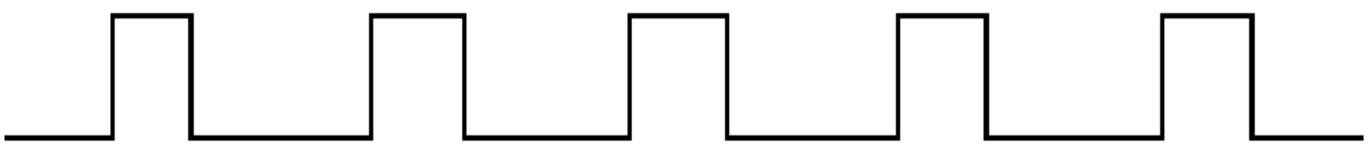
Problems with D-Latch

- **If D changes while C is true, the new value of D will appear at the output. The latch is *transparent*.**
- **If the stored value can change state more than once during a single clock pulse, the result is a hazard that might introduce a glitch later in the circuit.**
- **We must design the circuit so that the state can change only once per clock cycle.**

Figure 5-3



(a) Block diagram



(b) Timing diagram of clock pulses

How to remove the transparency ?

This can be accomplished by connecting two latches together. The left half of the circuit is the clocked D-latch from the previous section. The right half of the circuit is a clocked S-R latch.