



04 - Combinational and Sequential Logic Circuits

IT1206

Level I - Semester 1

4. Combinational & Sequential Logic Circuits

4.1 Adders

4.2 Decoders

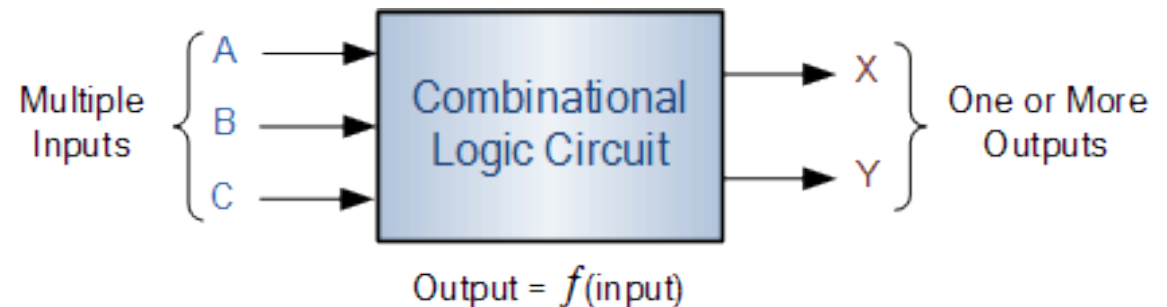
4.3 Multiplexers

4.4 Arithmetic Logic Unit

4.5 Flip Flop – SR,JK,D(Data)

Combinational Logic

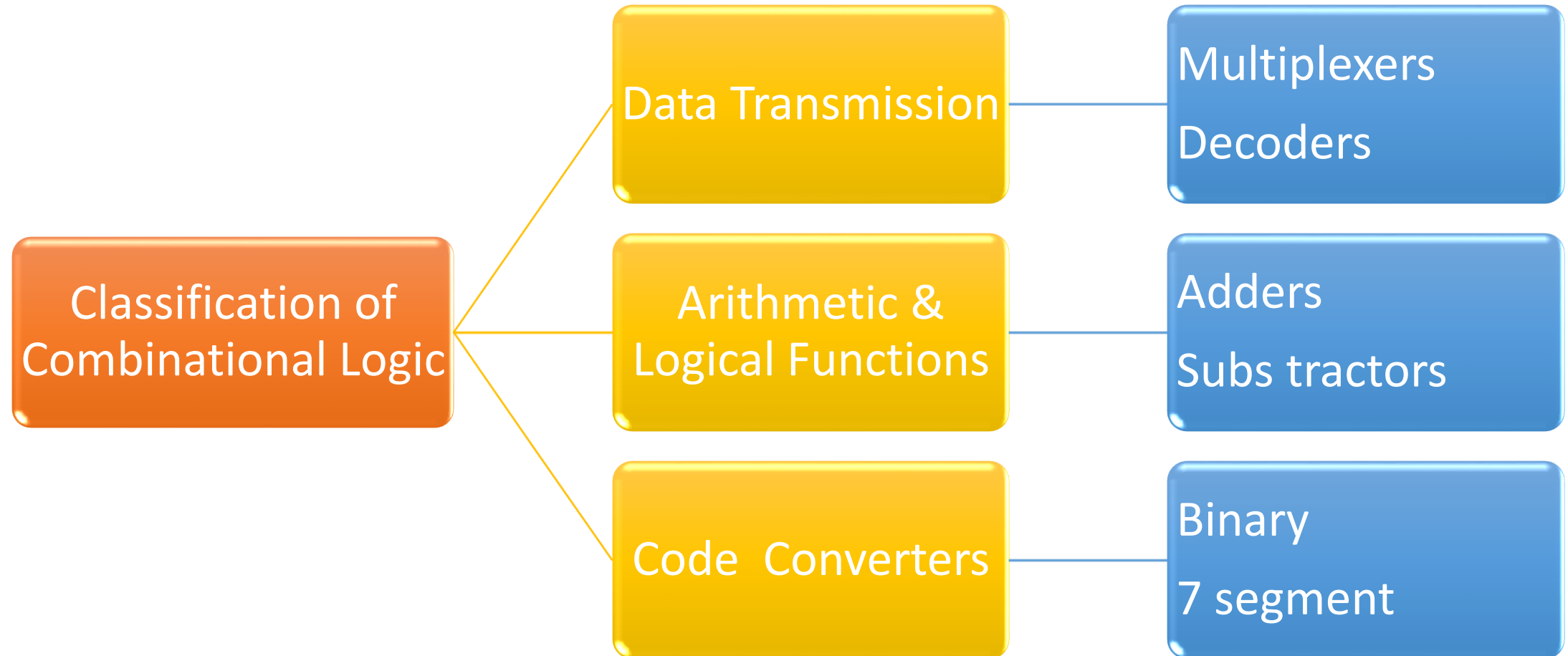
- Boolean algebra is a very natural way to represent digital information, making it an important concept if we want to understand how a computer makes decisions.
- Adders, decoders, multiplexers are just a few examples.
- Combinational Logic Circuits are memoryless digital logic circuits whose output at any instant in time depends only on the combination of its inputs
- These are made up from basic logic **NAND**, **NOR** or **NOT** gates that are “combined” or connected together to produce more complicated switching circuits.



Combinational Logic cont....

- The three main ways of specifying the function of a combinational logic circuit are:
 1. ***Boolean Algebra*** – This forms the algebraic expression showing the operation of the logic circuit for each input variable either True or False that results in a logic “1” output.
 2. ***Truth Table*** – A truth table defines the function of a logic gate by providing a concise list that shows all the output states in tabular form for each possible combination of input variable that the gate could encounter.
 3. ***Logic Diagram*** – This is a graphical representation of a logic circuit that shows the wiring and connections of each individual logic gate, represented by a specific graphical symbol, that implements the logic circuit.

Classification of Combinational Logic

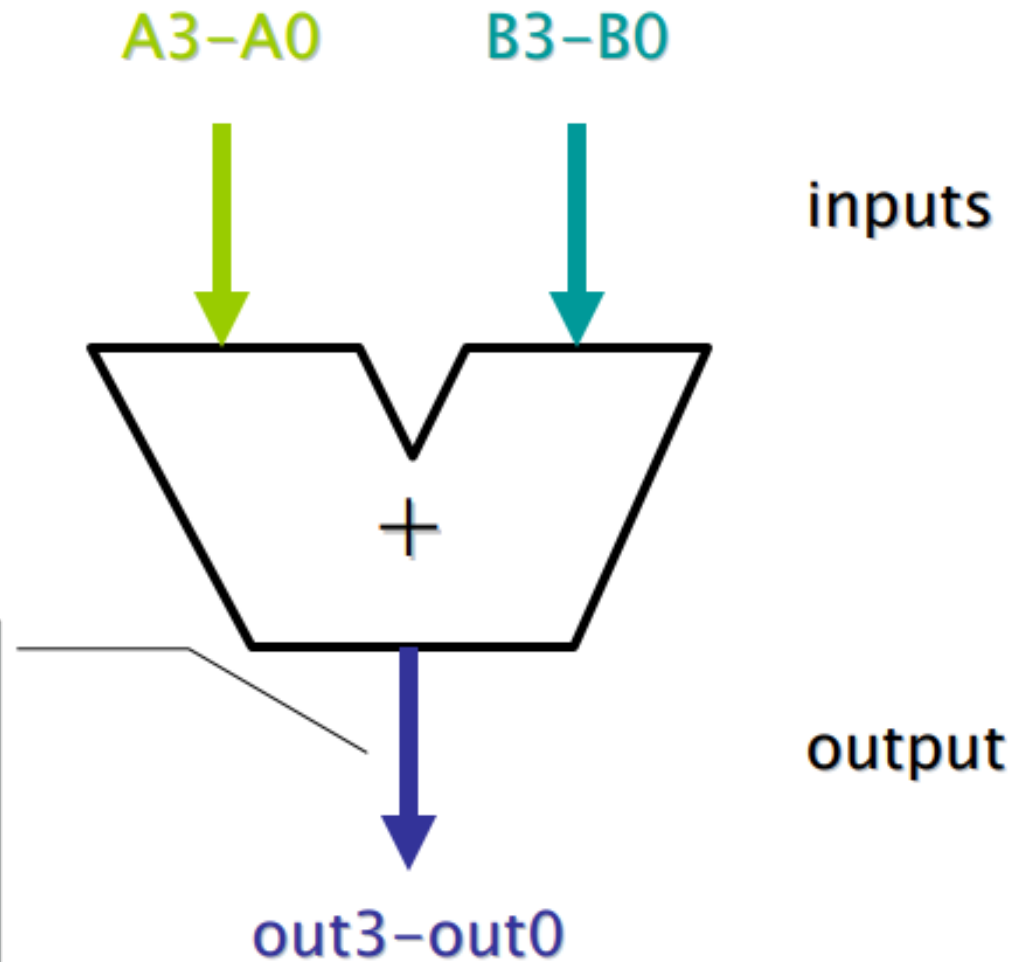


4.1 Adder

- An adder is a digital logic circuit in electronics that implements addition of numbers.
- In many computers and other kinds of processors, adders are used not only in the arithmetic logic units, but also in other parts of the processor, where they are used to calculate addresses, increment and decrement operators, and similar operations.
- Adders are classified into two types:
 - 1) half adder.
 - 2) full adder

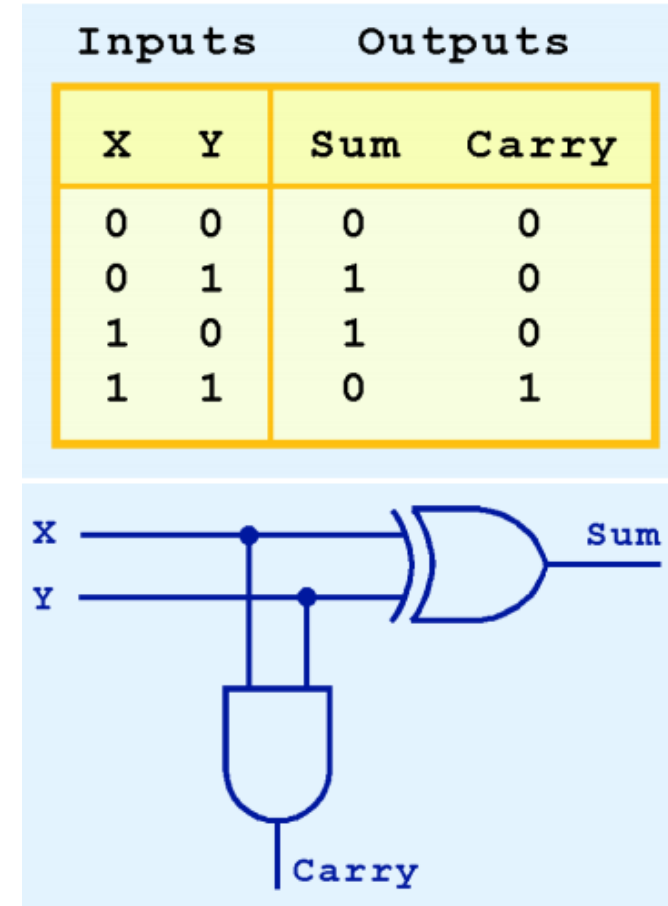
Adders (and other arithmetic circuits) are usually drawn like this in block diagrams

collections of parallel, related wires like this are known as buses; they carry multi-bit values between components



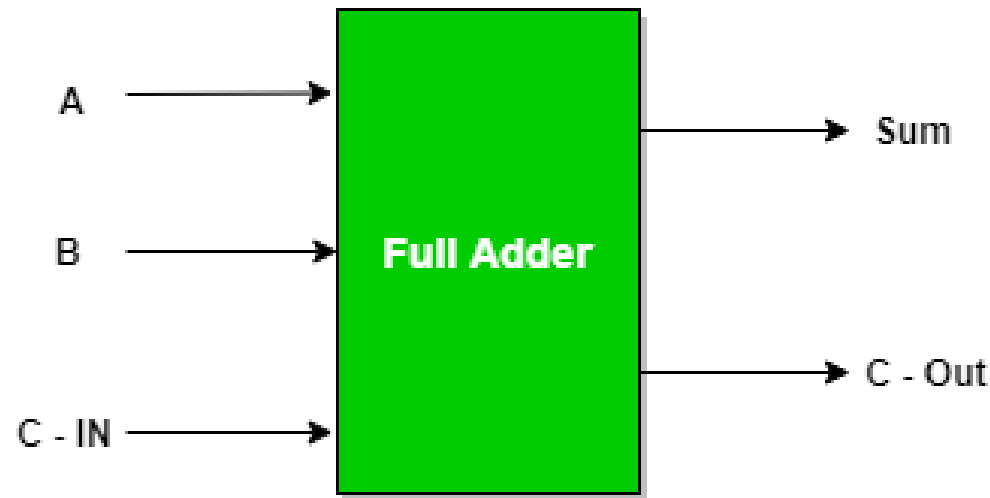
4.1.1 Half Adder

- Finds the sum of two bits
- The sum can be found using the XOR operation and the carry using the AND operation.



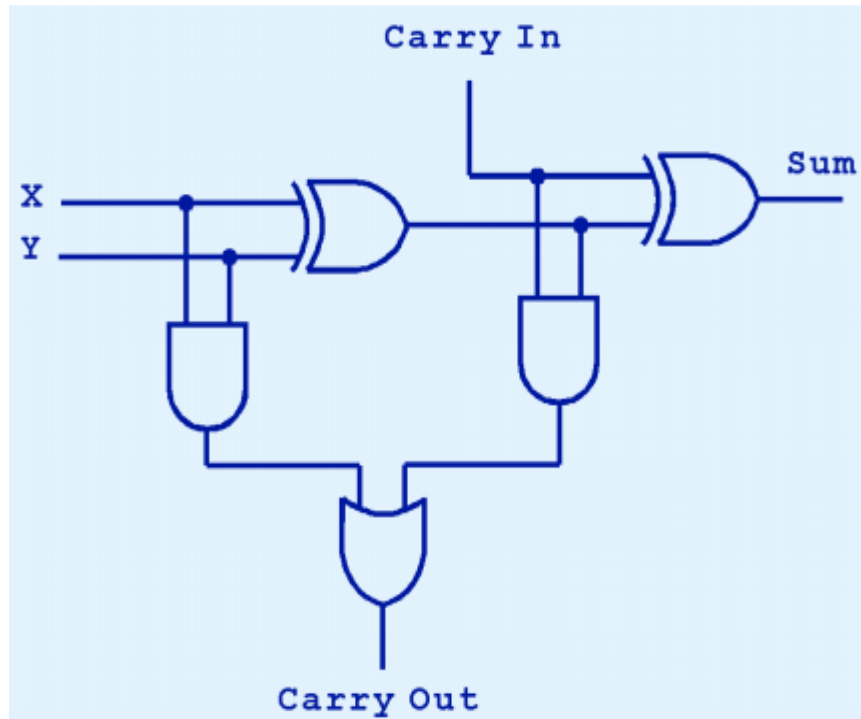
4.1.2 Full Adder

- *Full Adder* is the adder which adds three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as C-IN. The output carry is designated as C-OUT and the normal output is designated as S which is SUM.



4.1.2 Full Adder cont....

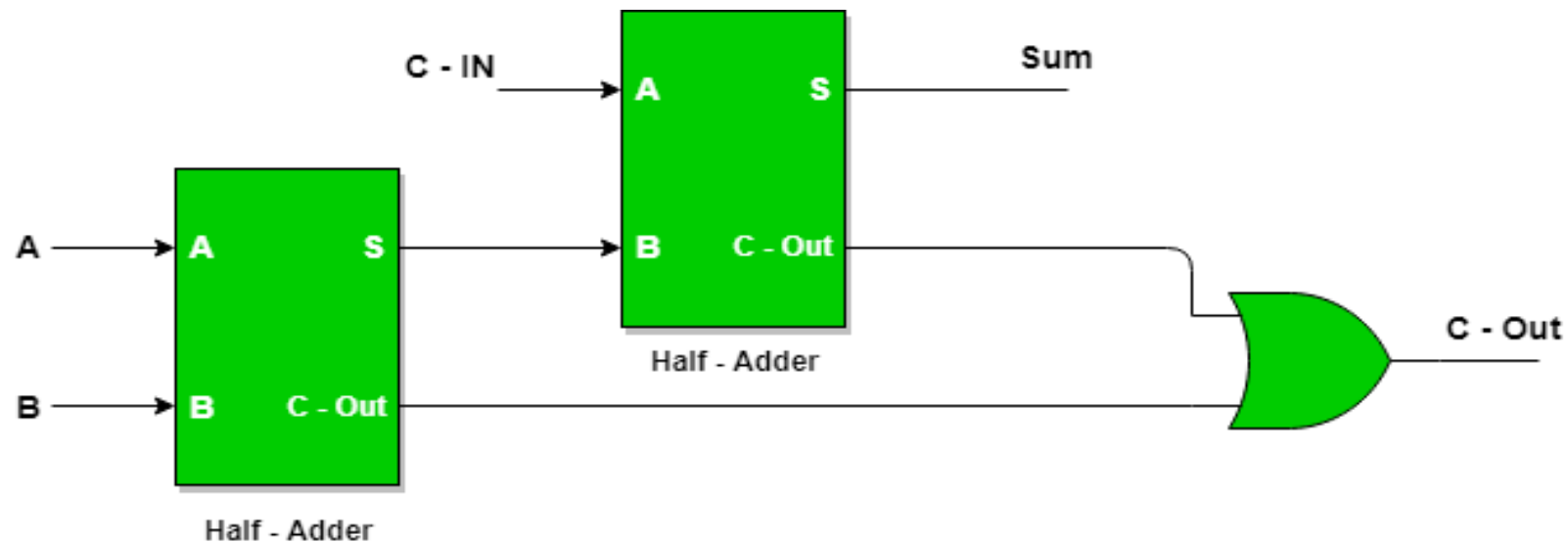
- The truth table for a full adder is:



Inputs			Outputs	
X	Y	Carry In	Sum	Carry Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

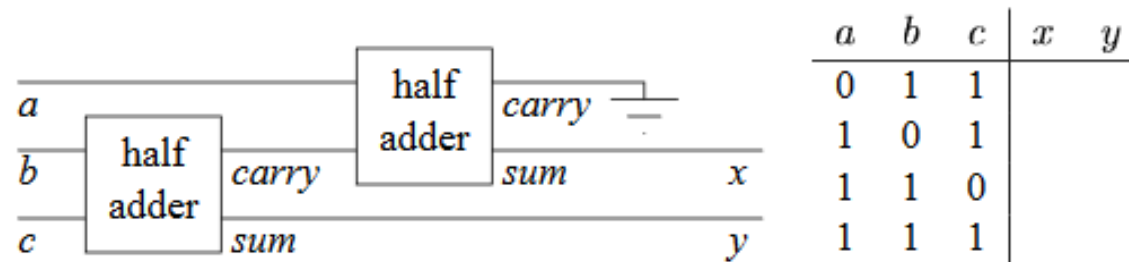
4.1.3 Converting a Half Adder into a Full Adder

- We can change our half adder into to a full adder by including gates for processing the carry bit.
- 2 Half Adders and a OR gate is required to implement a Full Adder.



Activity 01

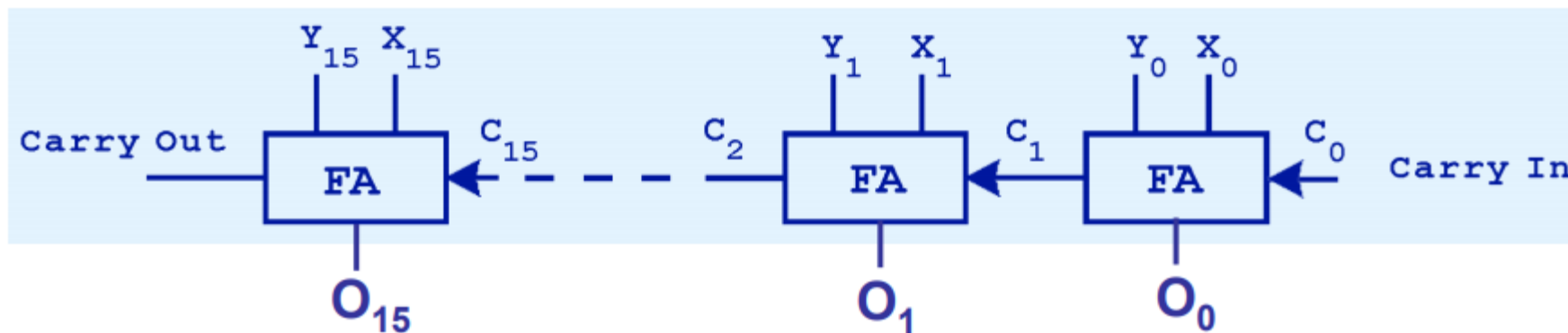
1. Fill in the truth table at right for the following circuit. Ignore rows not included in the table.



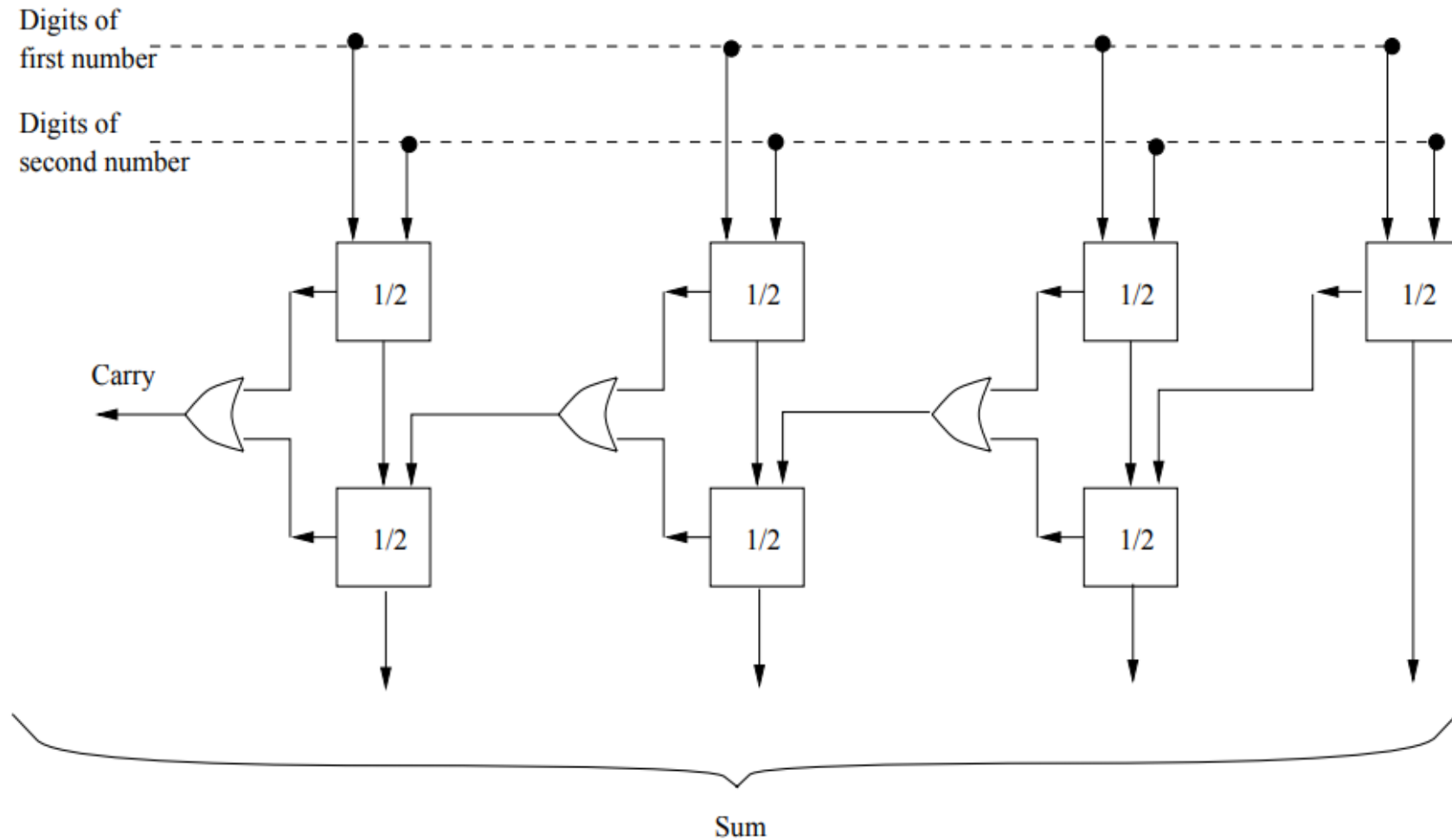
2. What distinguishes the meanings of a half adder's inputs and outputs from a full-adder's?
3. Construct the truth table, logical expressions, and circuit (using AND, OR, and NOT gates) for the half-adder
4. Using only four-bit adders, construct an eight-bit adder. Each four-bit adder has two four-bit inputs and one five-bit output. Your eight-bit adder should have two eight-bit inputs and a one eight-bit output(don't worry about the ninth output bit).

4.1.4 Ripple-carry Adder (I)

- Just as we combined half adders to make a full adder, full adders can be connected in series.
- The carry bit “ripples” from one adder to the next; hence, this configuration is called a ripple-carry adder.



4.1.4 Ripple-carry Adder (II)...

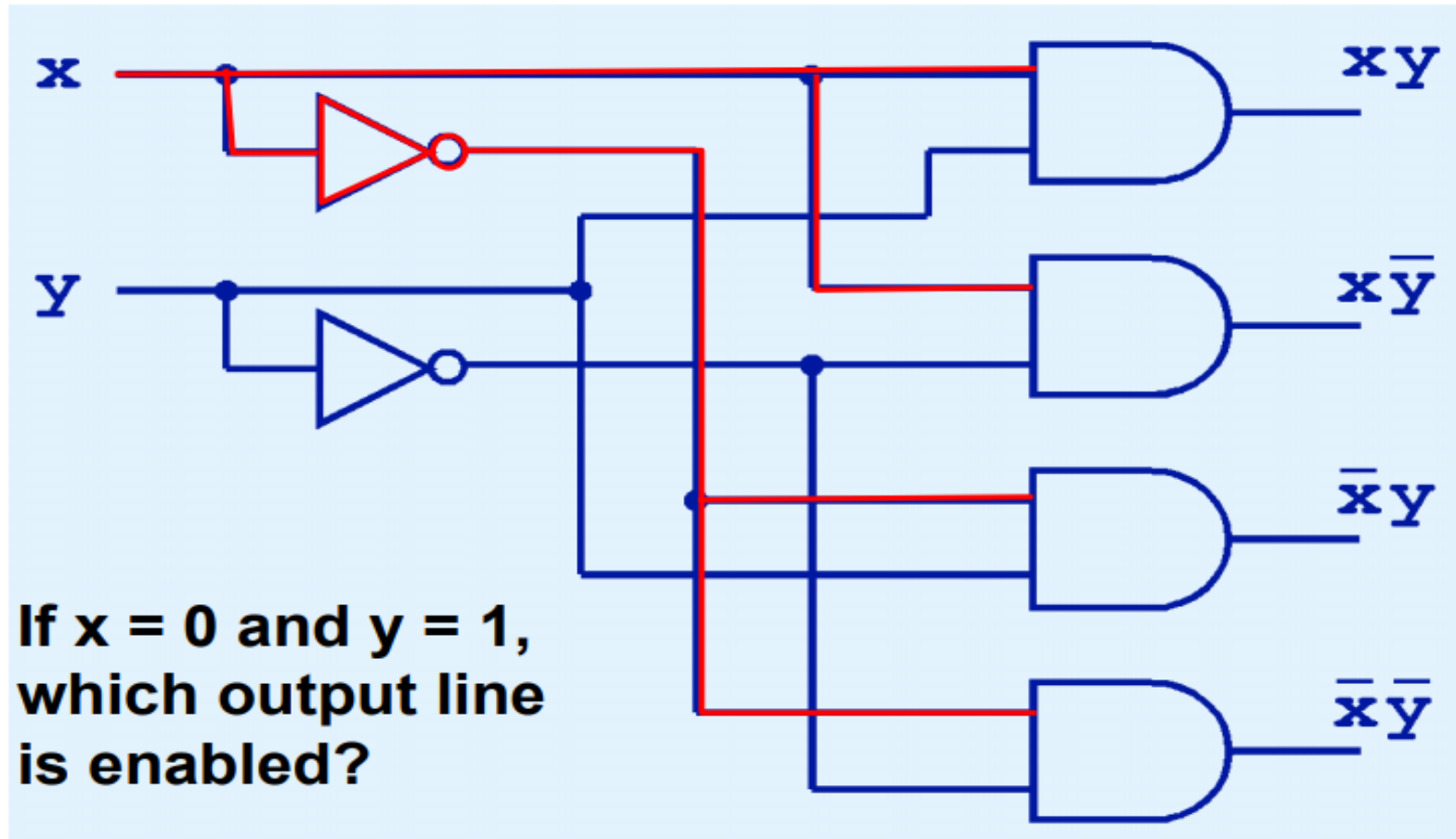


4.2 Decoder

- Binary code of N digits can be used to store 2^N distinct elements of coded information. This is what encoders and decoders are used for. **Encoders** convert 2^N lines of input into a code of N bits(inputs) and **Decoders** decode the N bits into 2^N lines.
- Selects a memory location according a binary value placed on the address lines of a memory bus.

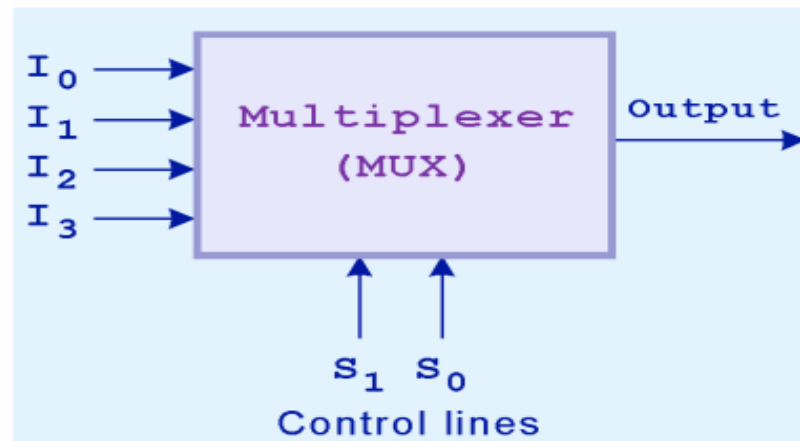


2-to-4 Decoder

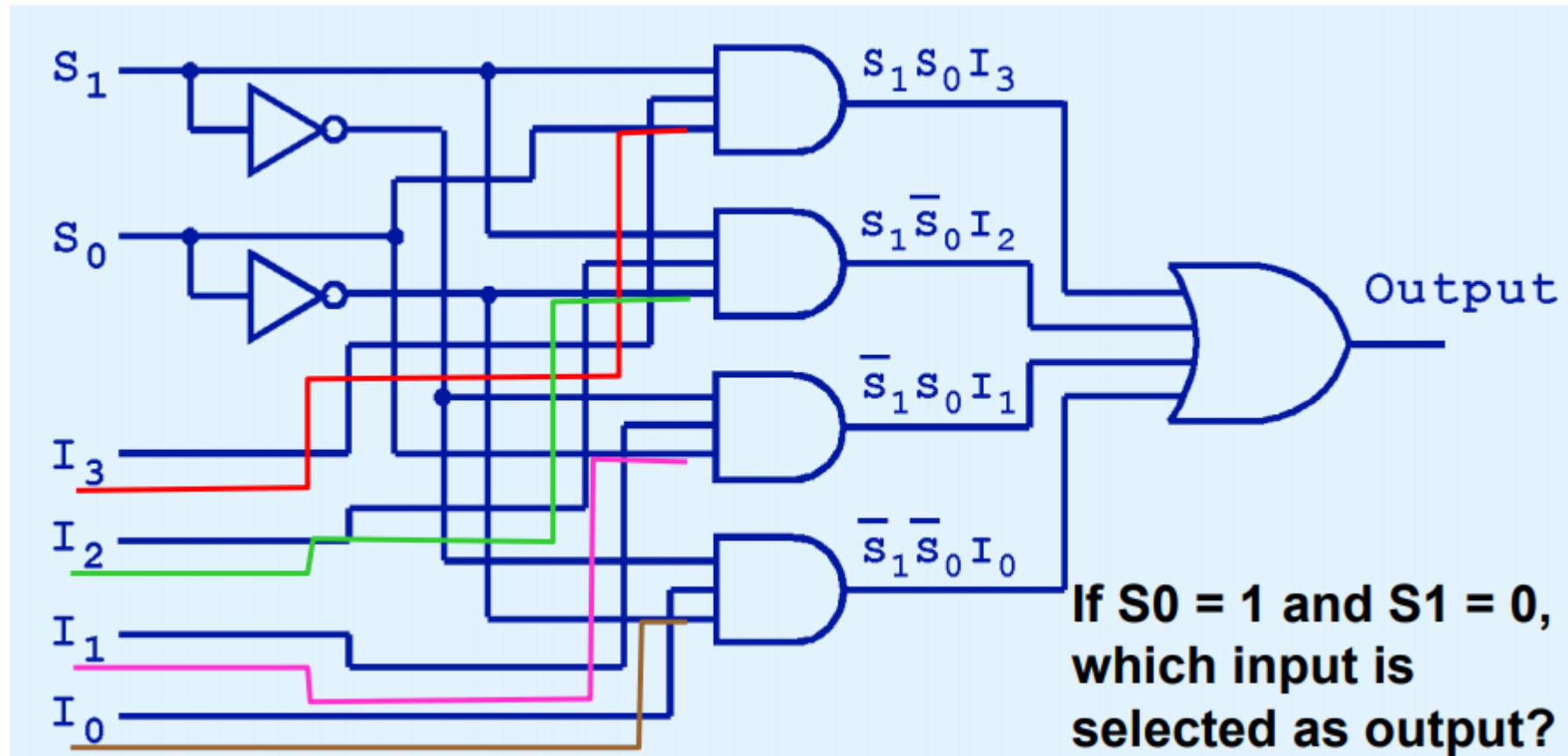


4.3 Multiplexer

- It is a combinational circuit which have many data inputs and single output depending on control or select inputs.
 - For N input lines, $\log n$ (base2) selection lines, or we can say that for 2^n input lines, n selection lines are required.
 - Multiplexers are also known as **“Data n selector, parallel to serial convertor, many to one circuit, universal logic circuit”**.
 - Multiplexers are mainly used to increase amount of the data that can be sent over the network within certain amount of time and bandwidth.



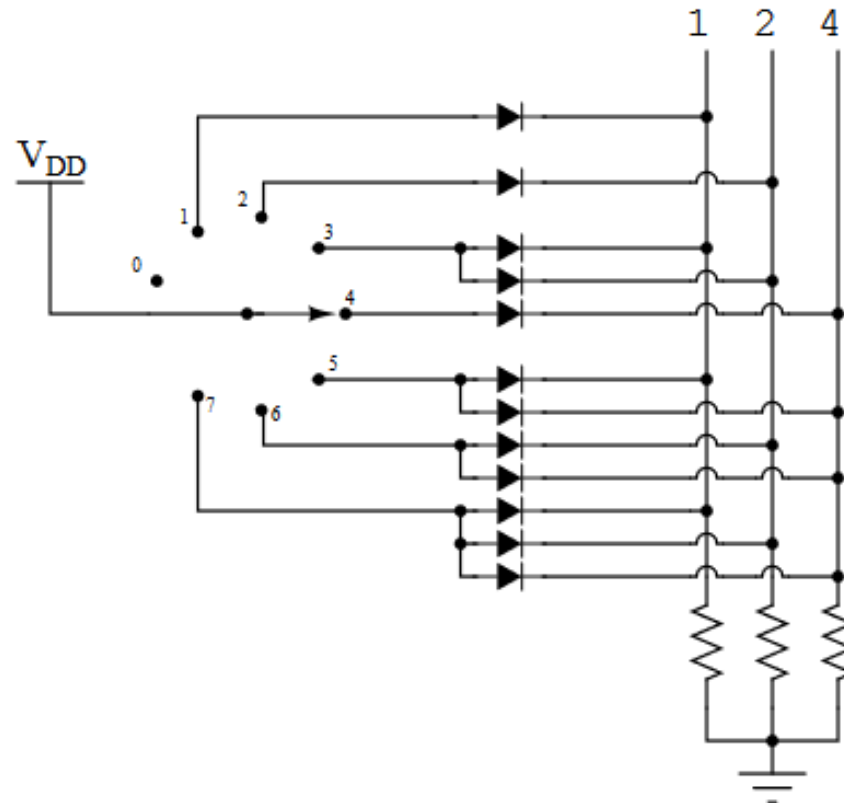
Activity 02 : 4-to-1 Multiplexer



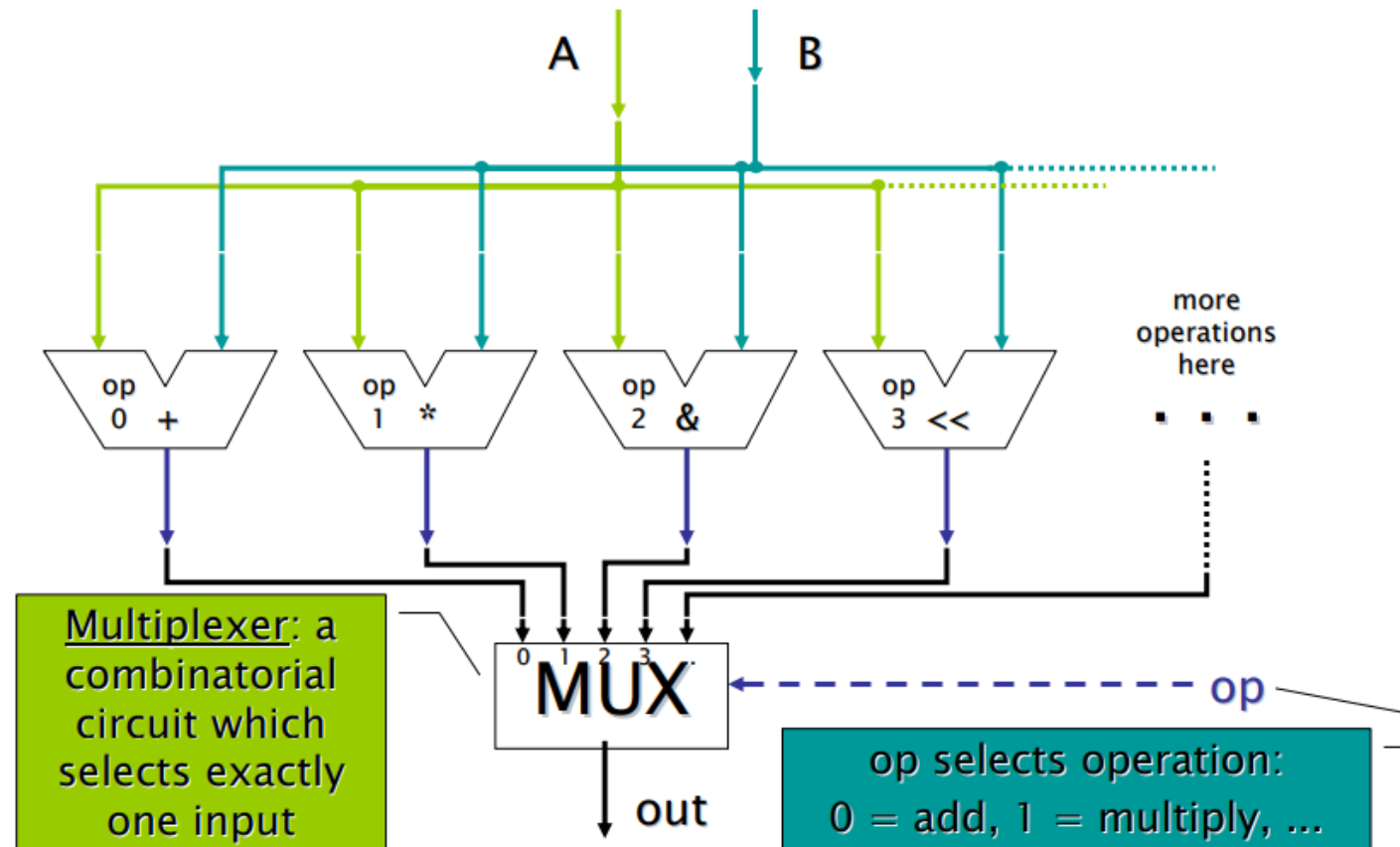
Activity 02

- What does it mean, in general terms, to *encode* something? Conversely, what does it mean to *decode* something?
- Implement 4:1 Multiplexers using truth table for following gates.
 - i. NOT gate
 - ii. AND gate
 - iii. NOR gate
 - iv. NAND gate
 - v. XOR gate
 - vi. XNOR gate

3. The simple switch-and-diode circuit shown here is an example of a digital encoder. Explain what this circuit does, as the switch is moved from position to position:

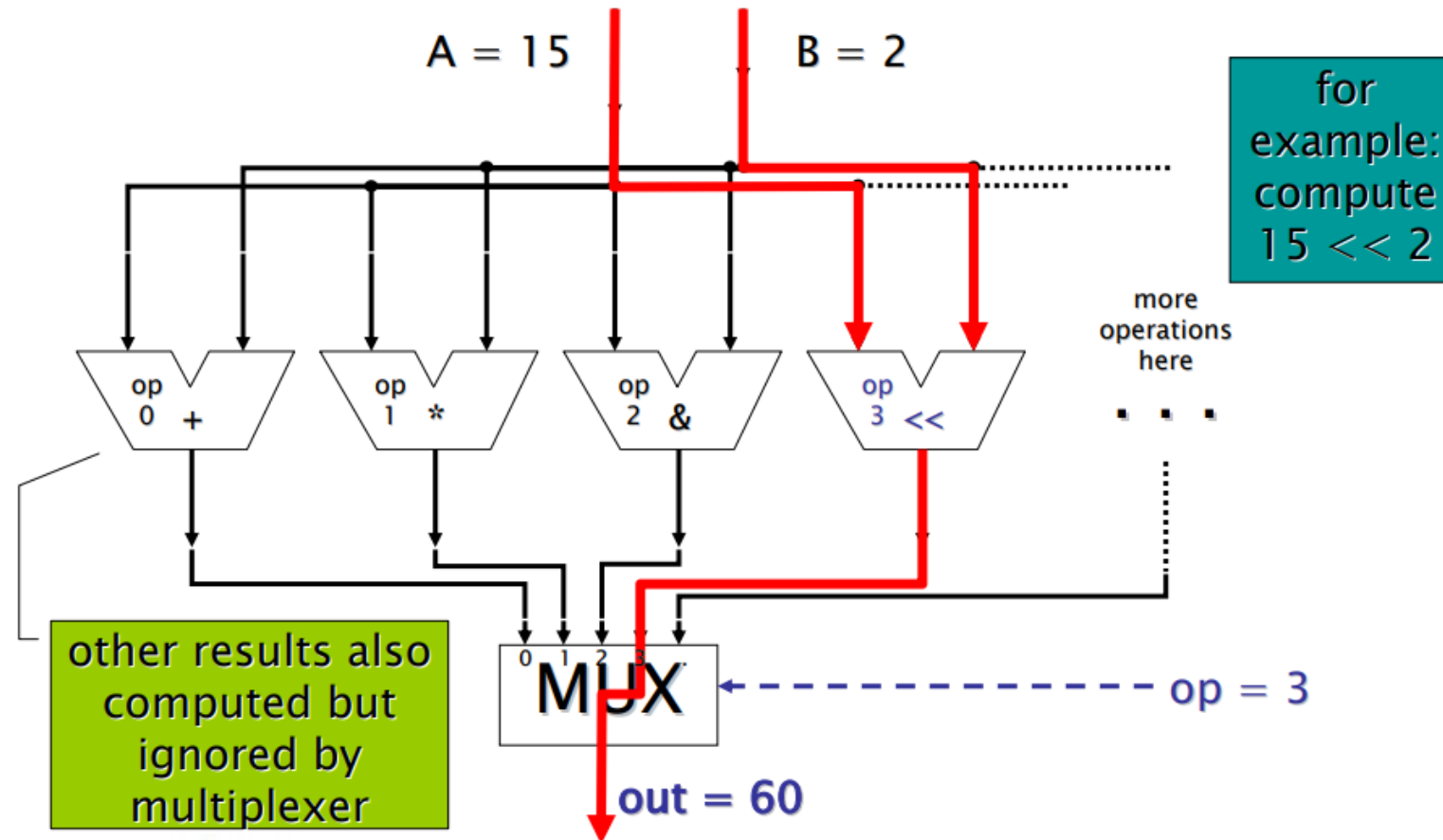


4.4 Arithmetic Logic Unit (ALU)



- Computers need to do more than just addition
 - arithmetic : + − * / %
 - logic : & | ~ << >>
- Need a circuit that can select operation to perform
- ALUs routinely perform the following operations:
 - **Logical Operations:** These include AND, OR, NOT, XOR, NOR, NAND, etc.
 - **Bit-Shifting Operations:** This pertains to shifting the positions of the bits by a certain number of places to the right or left, which is considered a multiplication operation.
 - **Arithmetic Operations:** This refers to bit addition and subtraction. Although multiplication and division are sometimes used, these operations are more expensive to make. Addition can be used to substitute for multiplication and subtraction for division.

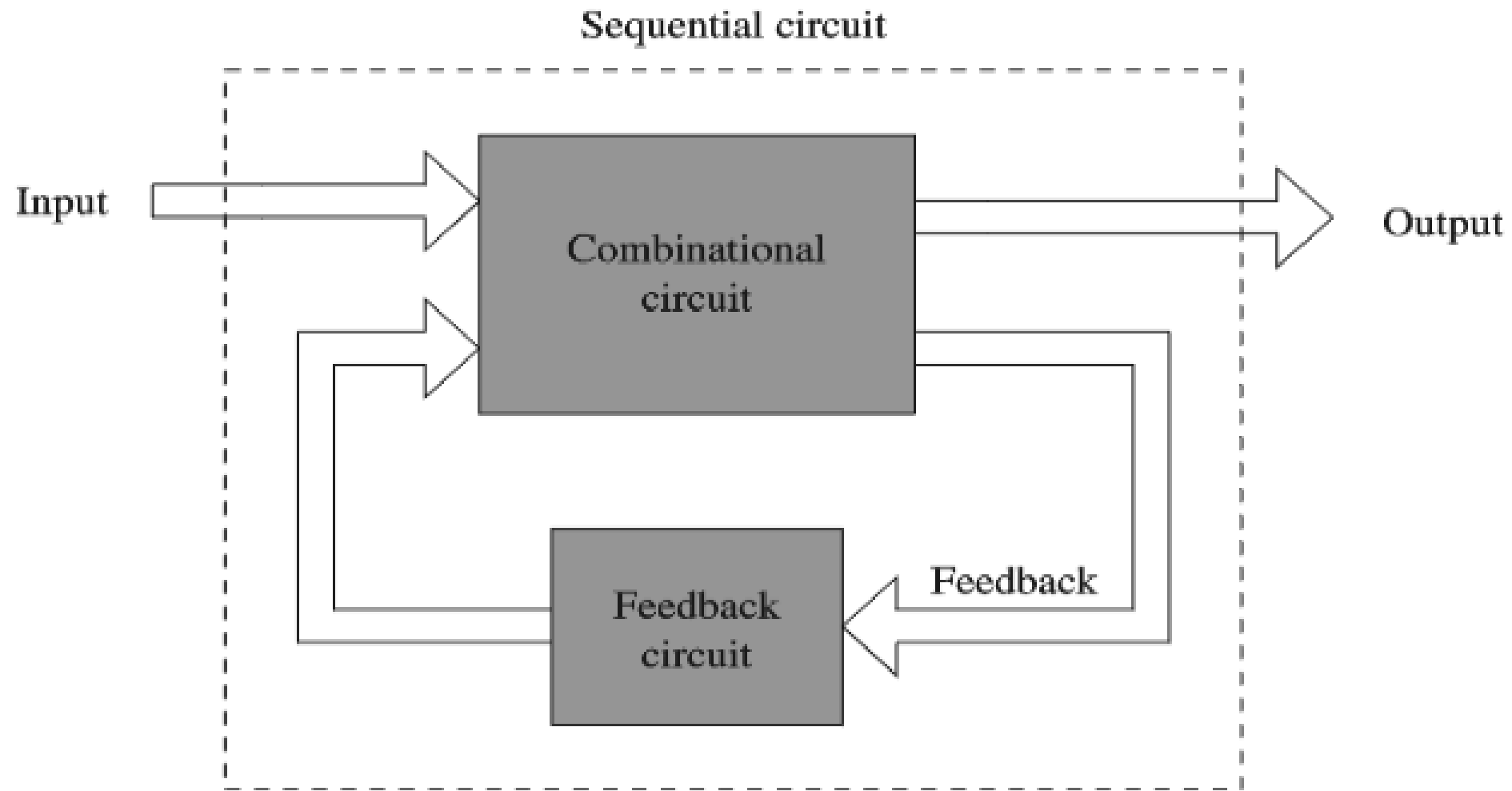
Arithmetic Logic Unit (ALU) - Example



Sequential Logic Circuits

- **Combinational logic** circuits are perfect for situations which require the immediate application of a Boolean function to a set of inputs
- But, here are times when we need a circuit to change its value with consideration to its current state as well as its inputs
 - These circuits have to “remember” their current state
- Sequential Logic Circuits use flip-flops as memory elements and in which their output is dependent on the input state

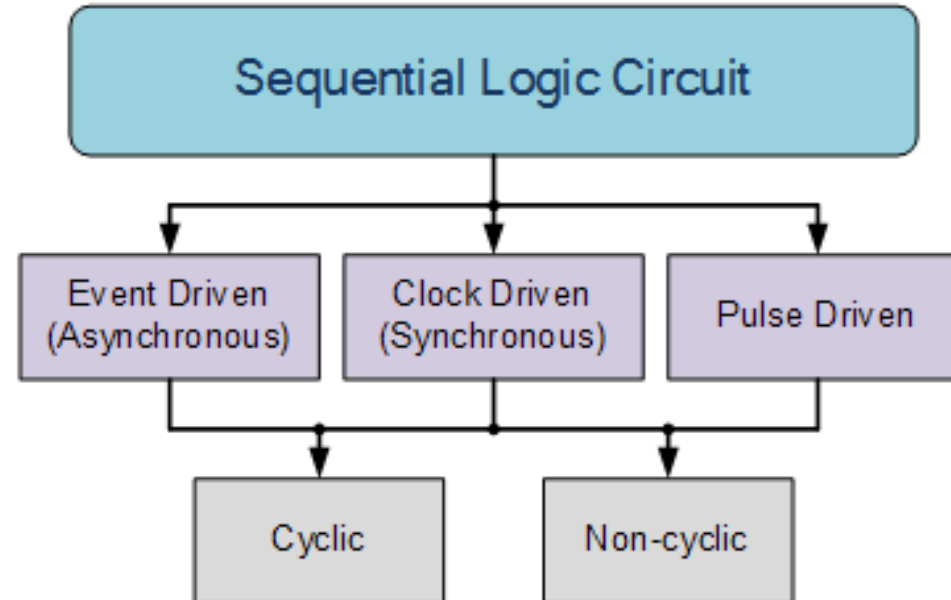
Main components of a sequential circuit



Sequential Logic Circuits cont....

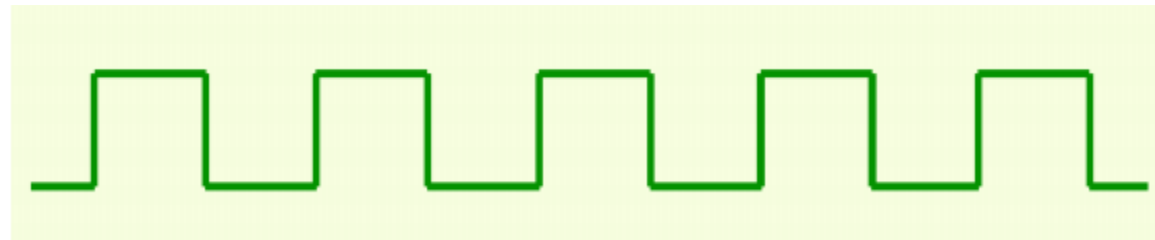
sequential logic circuits can be divided into the following three main categories:

1. **Event Driven** – asynchronous circuits that change state immediately when enabled.
2. **Clock Driven** – synchronous circuits that are synchronized to a specific clock signal.
3. **Pulse Driven** – which is a combination of the two that responds to triggering pulses.



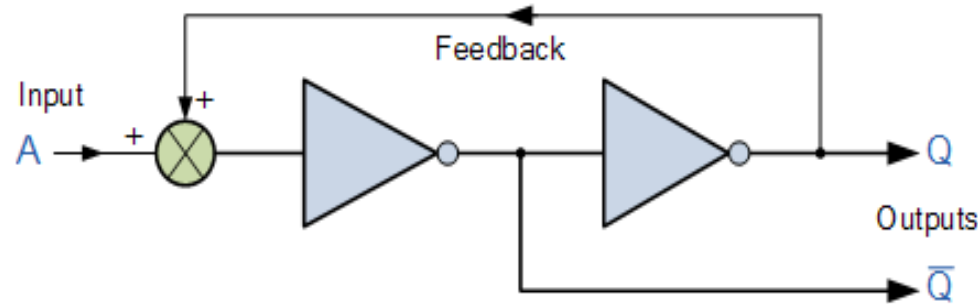
Sequencing Events

- Sequential logic circuits require a means by which events can be sequenced
- State changes are controlled by clocks
 - A "clock" is a special circuit that sends electrical pulses through a circuit ^{3/4}
- Clocks produce electrical waveforms such as this one



Feedback in Sequential Logic Circuits

- Sequential circuits rely on feedback to retain their state values
- Feedback in digital circuits occurs when an output is looped back to the input
 - Example,



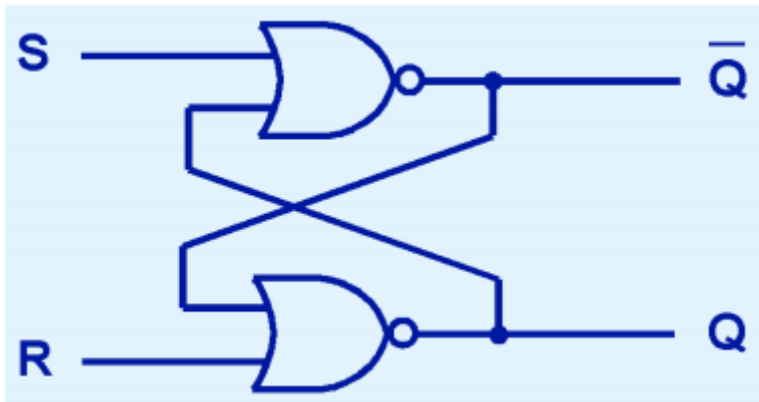
- If Q is 0 it will always be 0, if it is 1, it will always be 1

4.5 Flip Flop

- Flip flop is a sequential circuit which generally samples its inputs and changes its outputs only at particular instants of time and not continuously.
- Flip flop is said to be edge sensitive or edge triggered rather than being level triggered like latches.
- The most common types of flip flops are:
 - **SR flip-flop:** Is similar to an SR latch. Besides the CLOCK input, an SR flip-flop has two inputs, labeled SET and RESET.
 - **D flip-flop:** Has just one input in addition to the CLOCK input. This input is called the DATA input.
 - **JK flip-flop:** A JK flip-flop has two inputs, labeled *J* and *K*. The *J* input corresponds to the SET input in an SR flip-flop, and the *K* input corresponds to the RESET input.

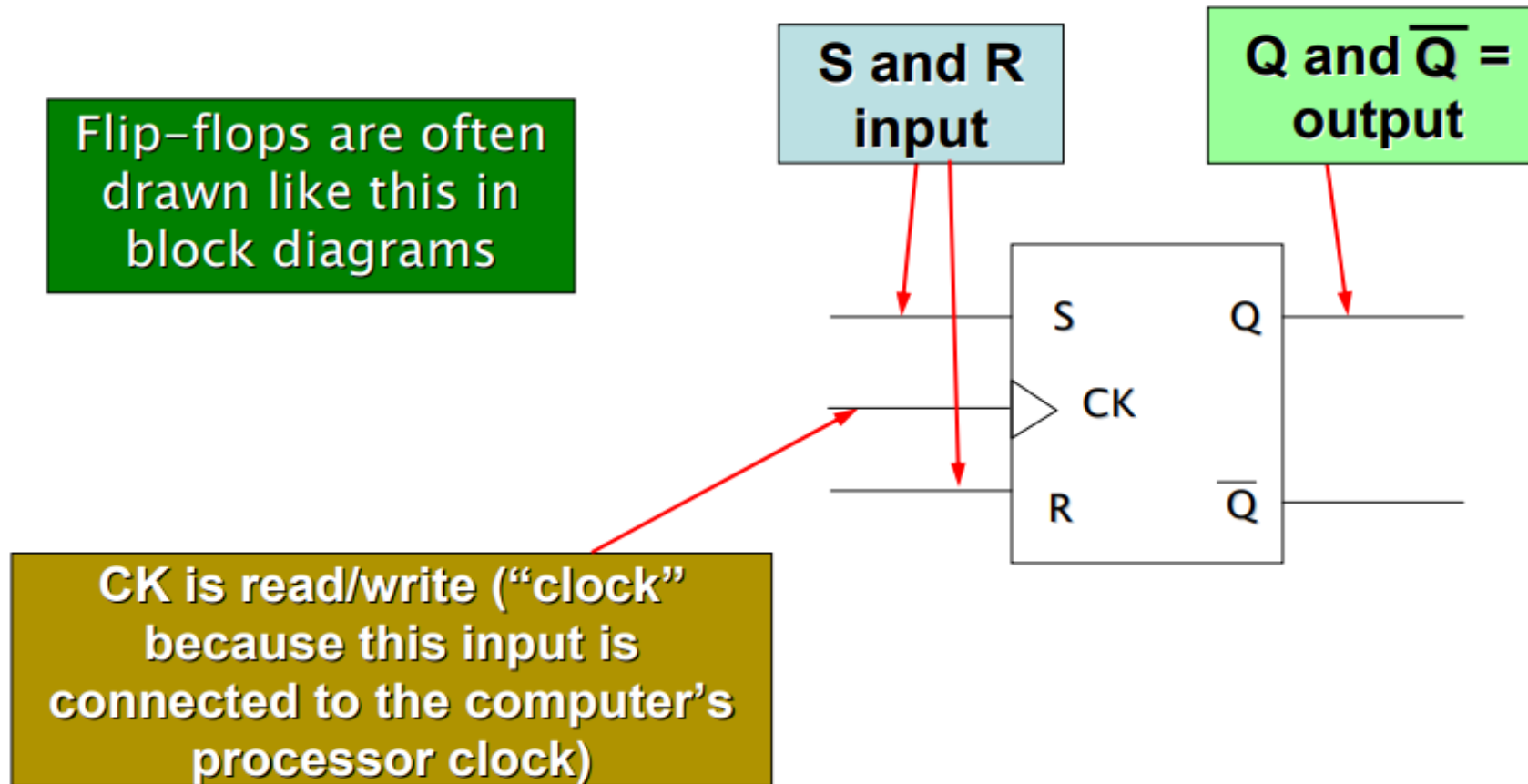
SR Flip-flop (Set-Reset)

- The **SR flip-flop**, also known as a *SR Latch*.
- This has two inputs, one which will "**SET**" the device (meaning the output = "1"), and is labelled **S** and one which will "**RESET**" the device (meaning the output = "0"), labelled **R**.
- The behavior of an SR flip flop is described by a characteristic table
 - $Q(t)$ output at time t
 - $Q(t+1)$ output after the next clock pulse

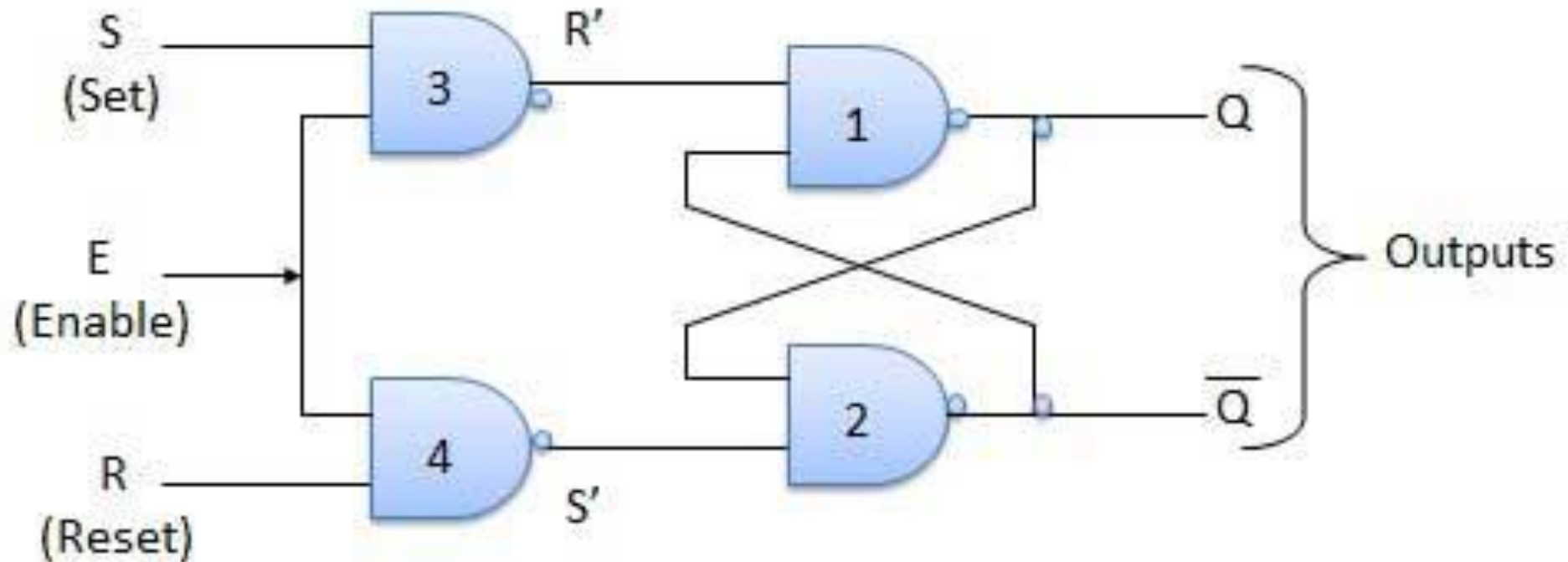


S	R	$Q(t+1)$
0	0	$Q(t)$ (no change)
0	1	0 (reset to 0)
1	0	1 (set to 1)
1	1	undefined

SR Flip-flop: Block Diagram



SR Flip-flop: Circuit Diagram

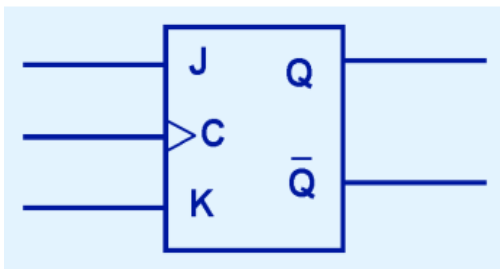


SR Flip-flop: Truth Table

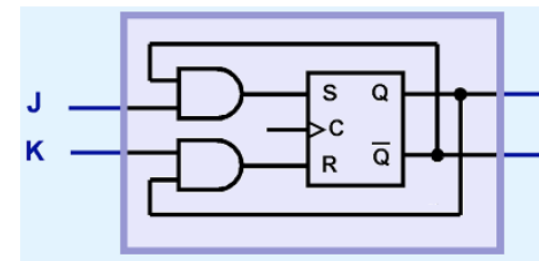
Inputs			Outputs		Comments
E	S	R	Q_{n+1}	\overline{Q}_{n+1}	
1	0	0	Q_n	\overline{Q}_n	No change
1	0	1	0	1	Rset
1	1	0	1	0	Set
1	1	1	x	x	Indeterminate

JK Flip-flop (Jack Kilby)

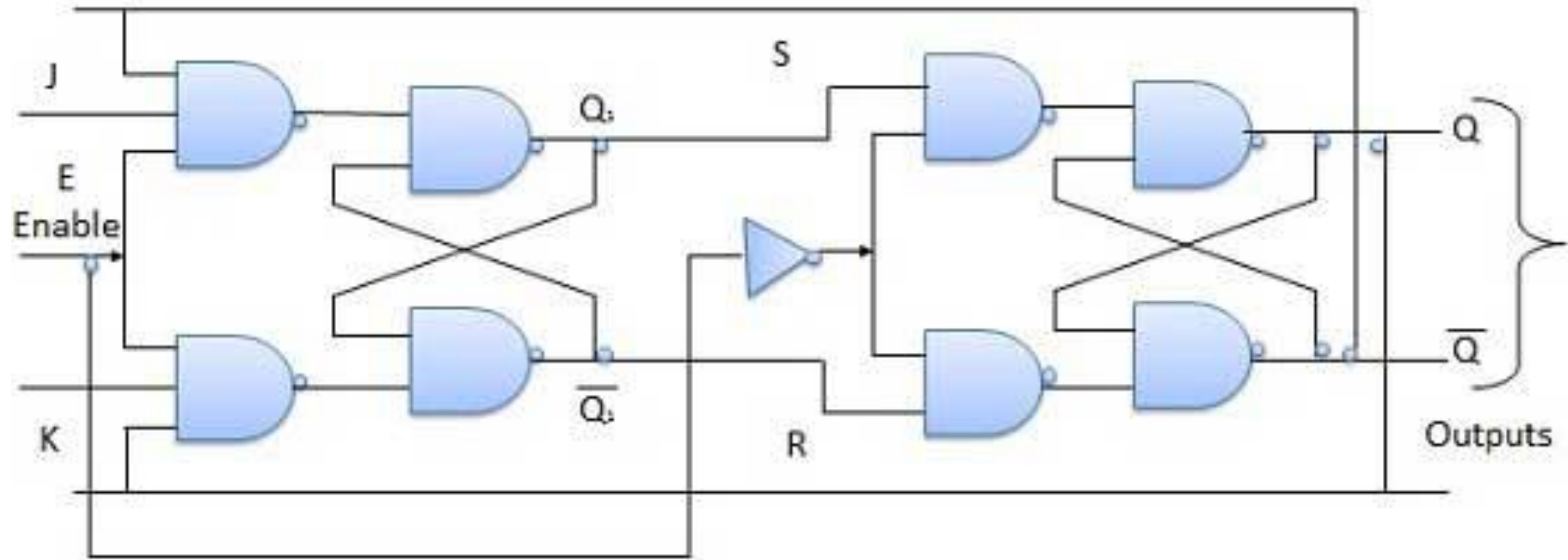
- Modified version of the SR flip-flop to provide a stable state when both inputs are 1.
- Master Slave JK FF** is a cascade of two SR FF with feedback from the output of second to input of first. Master is a positive level triggered. But due to the presence of the inverter in the clock line, the slave will respond to the negative level. Hence when the clock = 1 (positive level) the master is active and the slave is inactive. Whereas when clock = 0 (low level) the slave is active and master is inactive



J	K	Q (t+1)
0	0	Q (t) (no change)
0	1	0 (reset to 0)
1	0	1 (set to 1)
1	1	Q-bar (t)



JK Flip-flop: Circuit Diagram



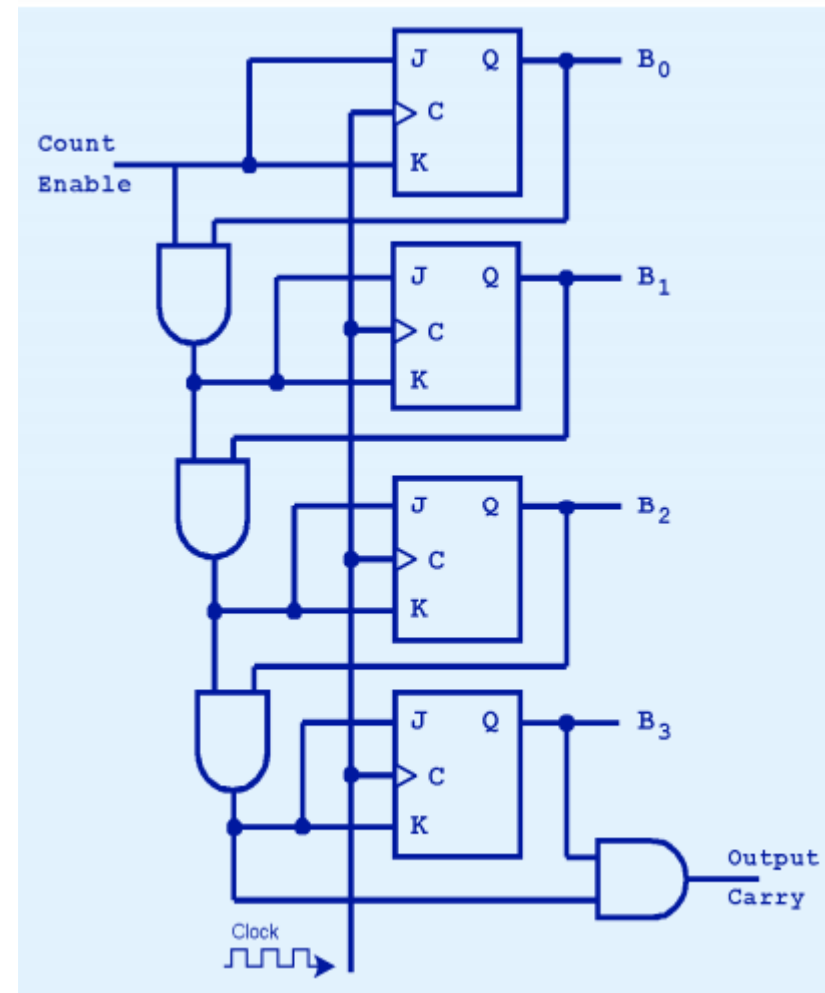
JK Flip-flop: Truth Table

Inputs			Outputs		Comments
E	J	K	Q_{n+1}	\overline{Q}_{n+1}	
1	0	0	Q_n	\overline{Q}_n	No change
1	0	1	0	1	Rset
1	1	0	1	0	Set
1	1	1	\overline{Q}_n	Q_n	Toggle

JK Flip-flop: Binary Counter

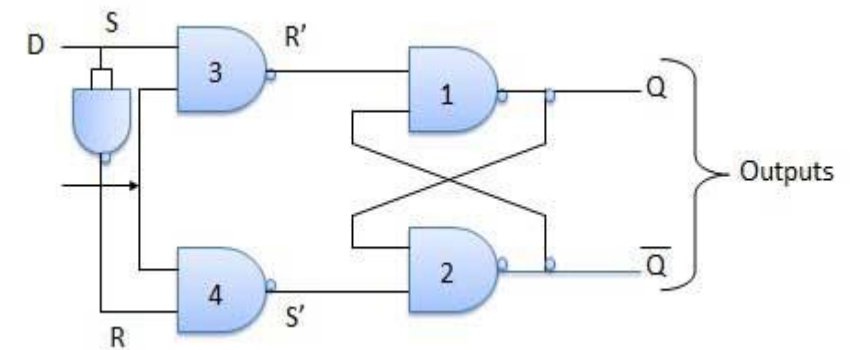
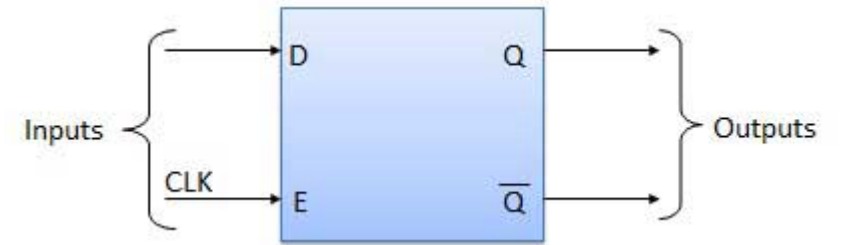
- The low-order bit is complemented at each clock pulse
- Whenever it changes from 1 to 0, the next bit is complemented, and so on through the other flip-flops

J	K	$Q(t+1)$
0	0	$Q(t)$ (no change)
0	1	0 (reset to 0)
1	0	1 (set to 1)
1	1	$\bar{Q}(t)$



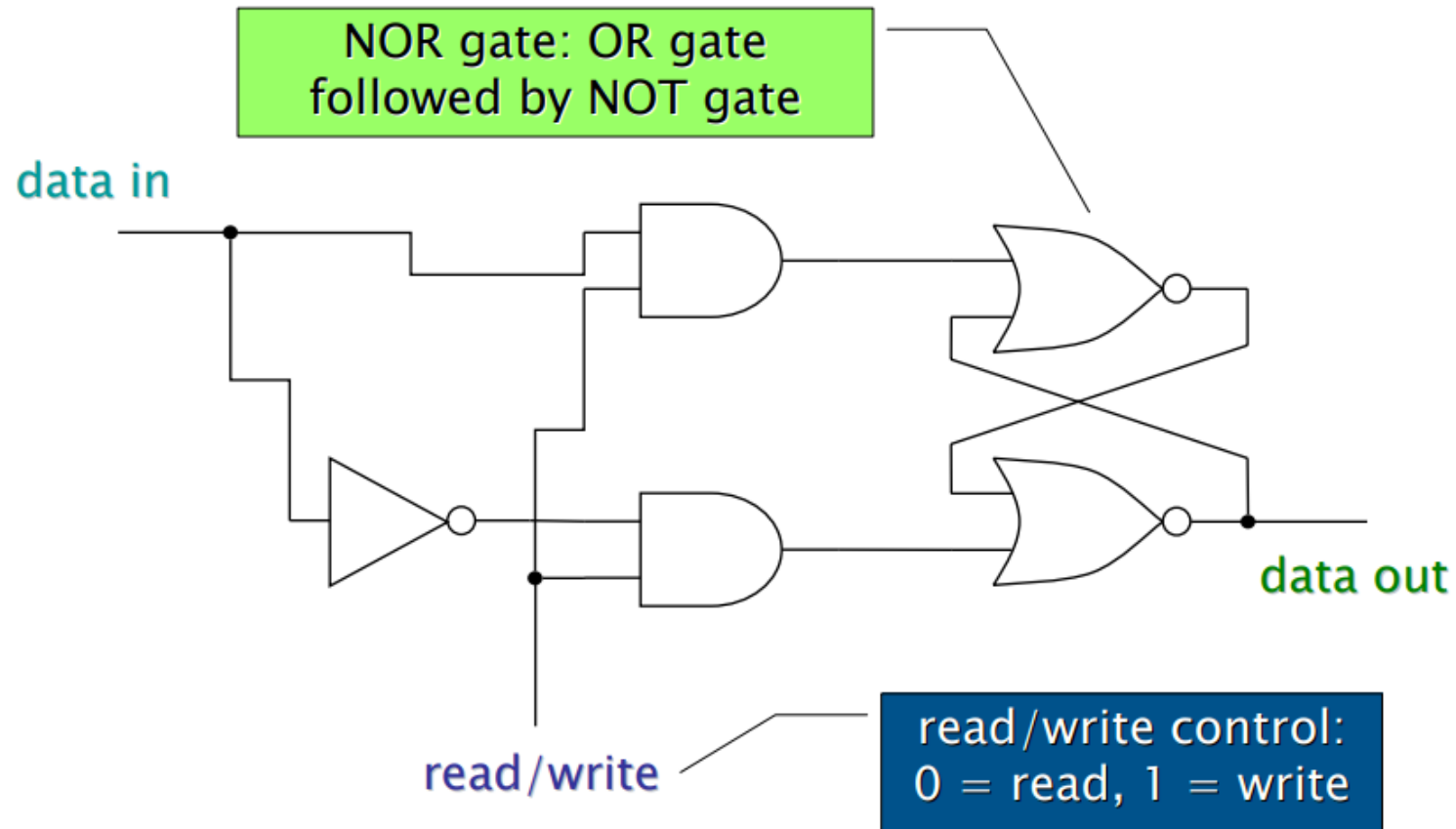
D Flip-flop (Data)

- Fundamental circuit of computer memory.
- Used to store 1 bit.
- Can be implemented with gates.
- Also known as *Data Latch*, *Delay flip flop*, *D-type Bistable*, *D-type Flip Flop* or just simply a **D Flip Flop**
- Not combinatorial logic
 - because current output may depend on previous state
- Example of sequential logic
 - current output depends on inputs and prior output

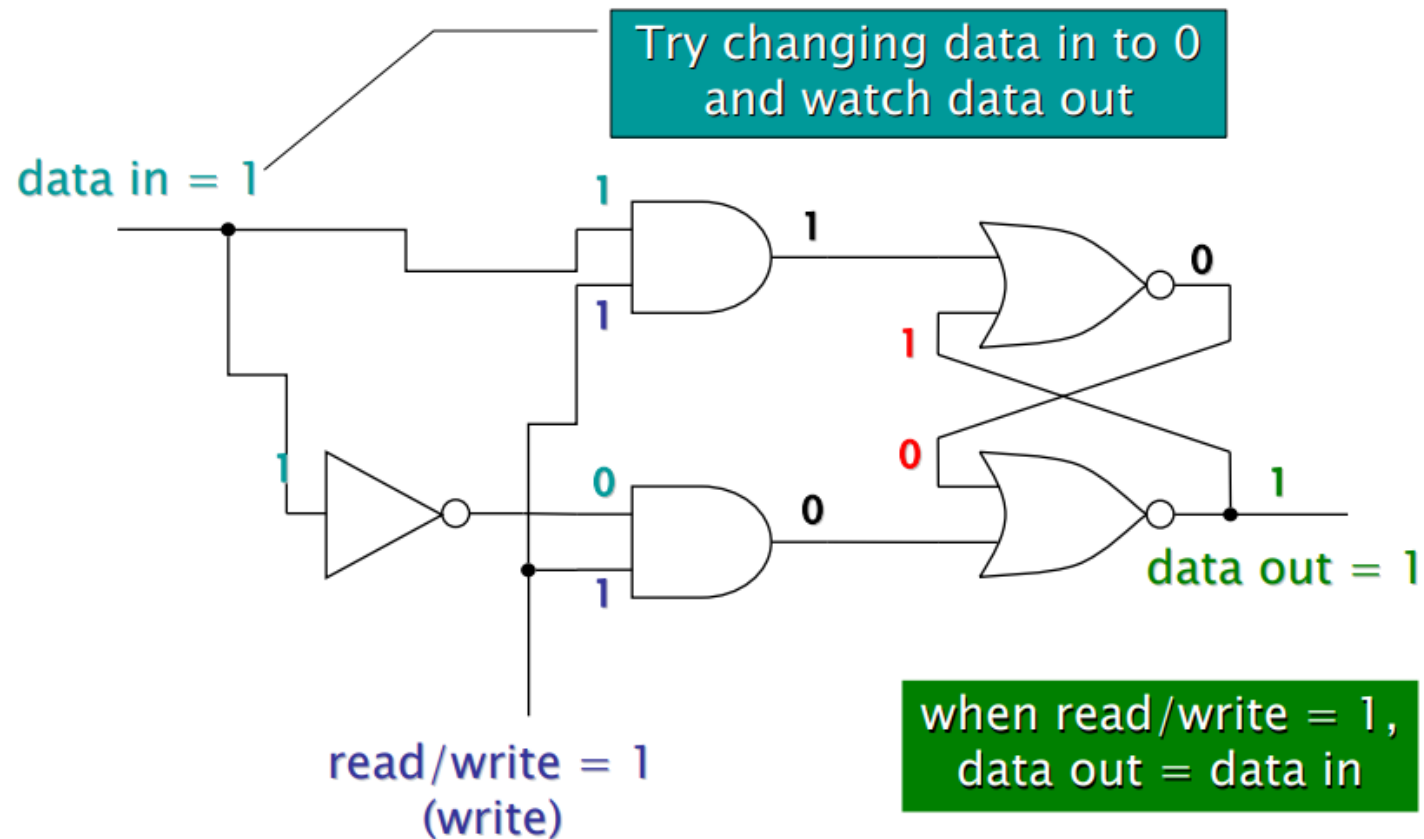


Inputs		Outputs		Comments
E	D	Q_{n+1}	\overline{Q}_{n+1}	
1	0	0	1	Rset
1	1	1	0	Set

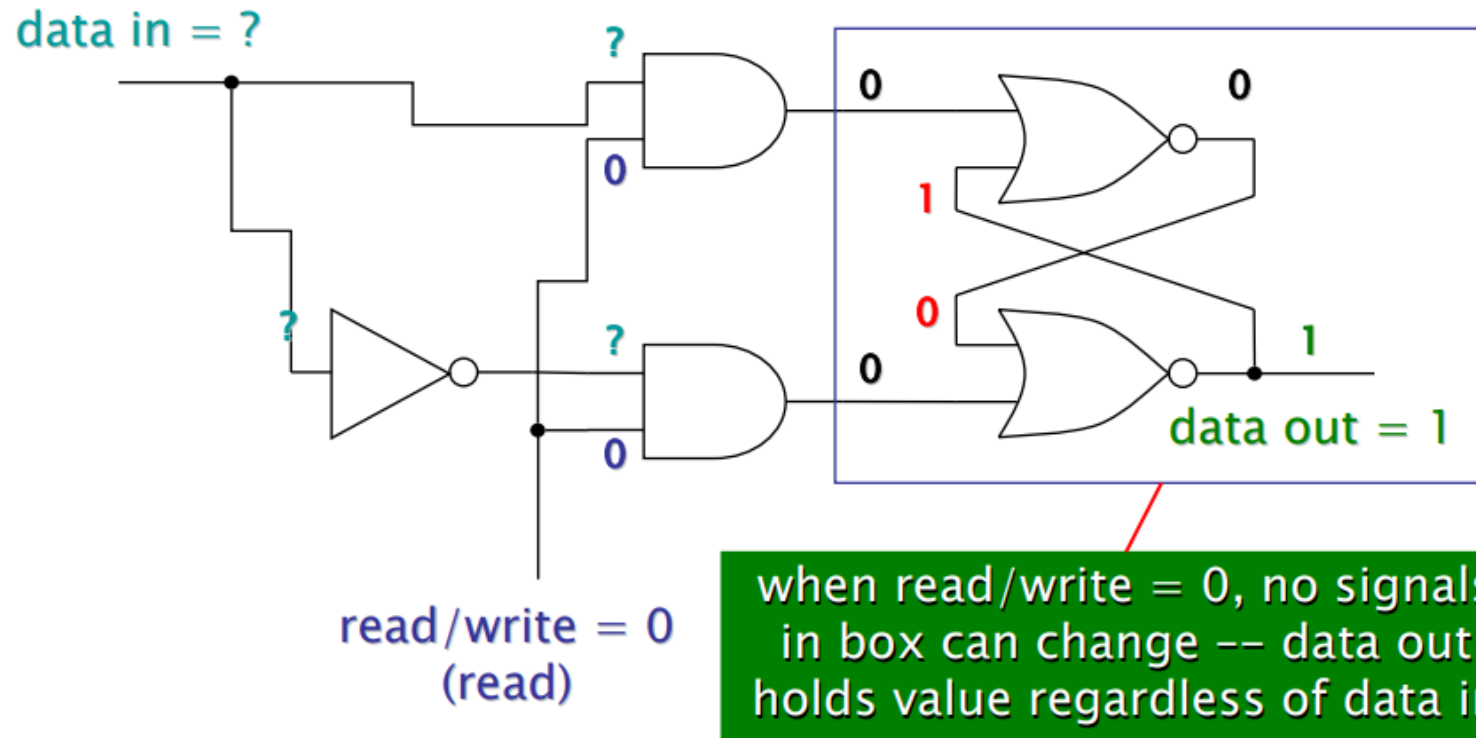
D Flip-flop



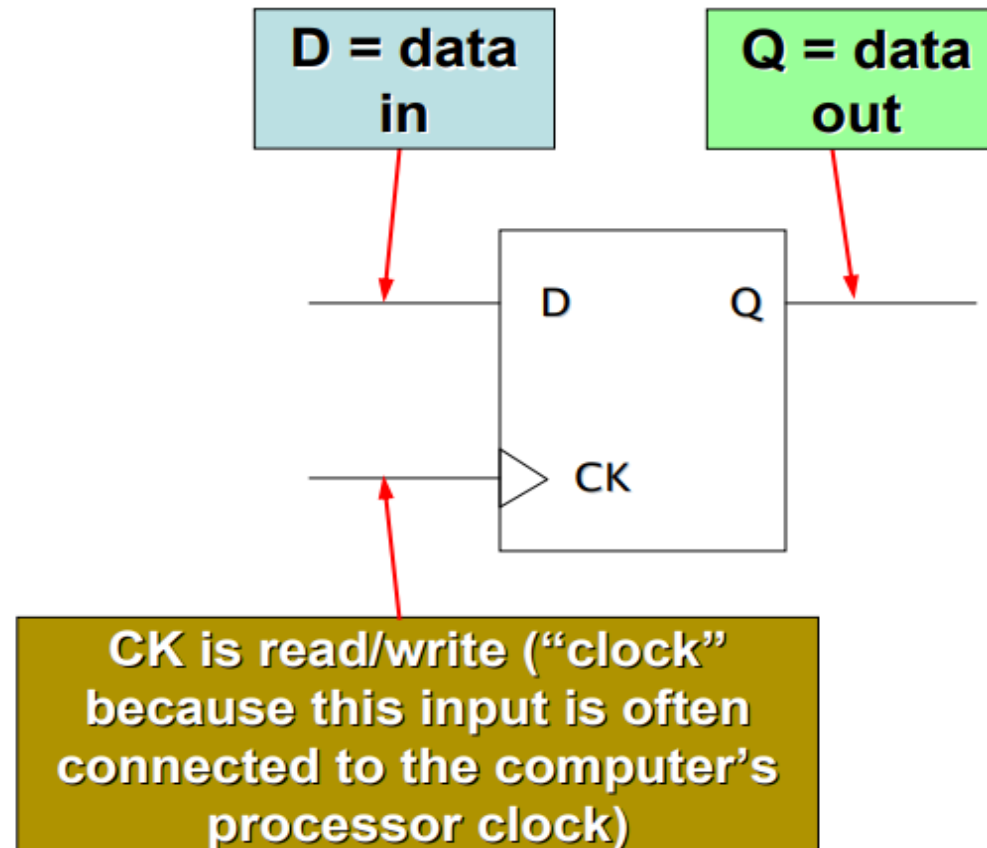
D Flip-flop: Writing



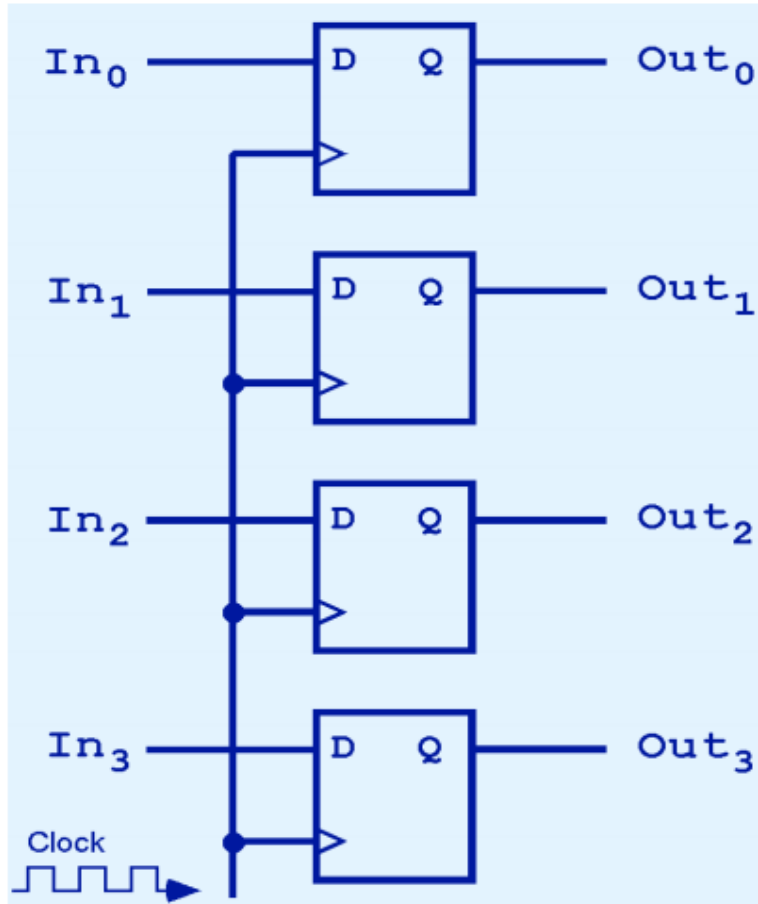
D Flip-flop: Reading



D Flip-flop: Block Diagram



D Flip-flop: 4-bit Register



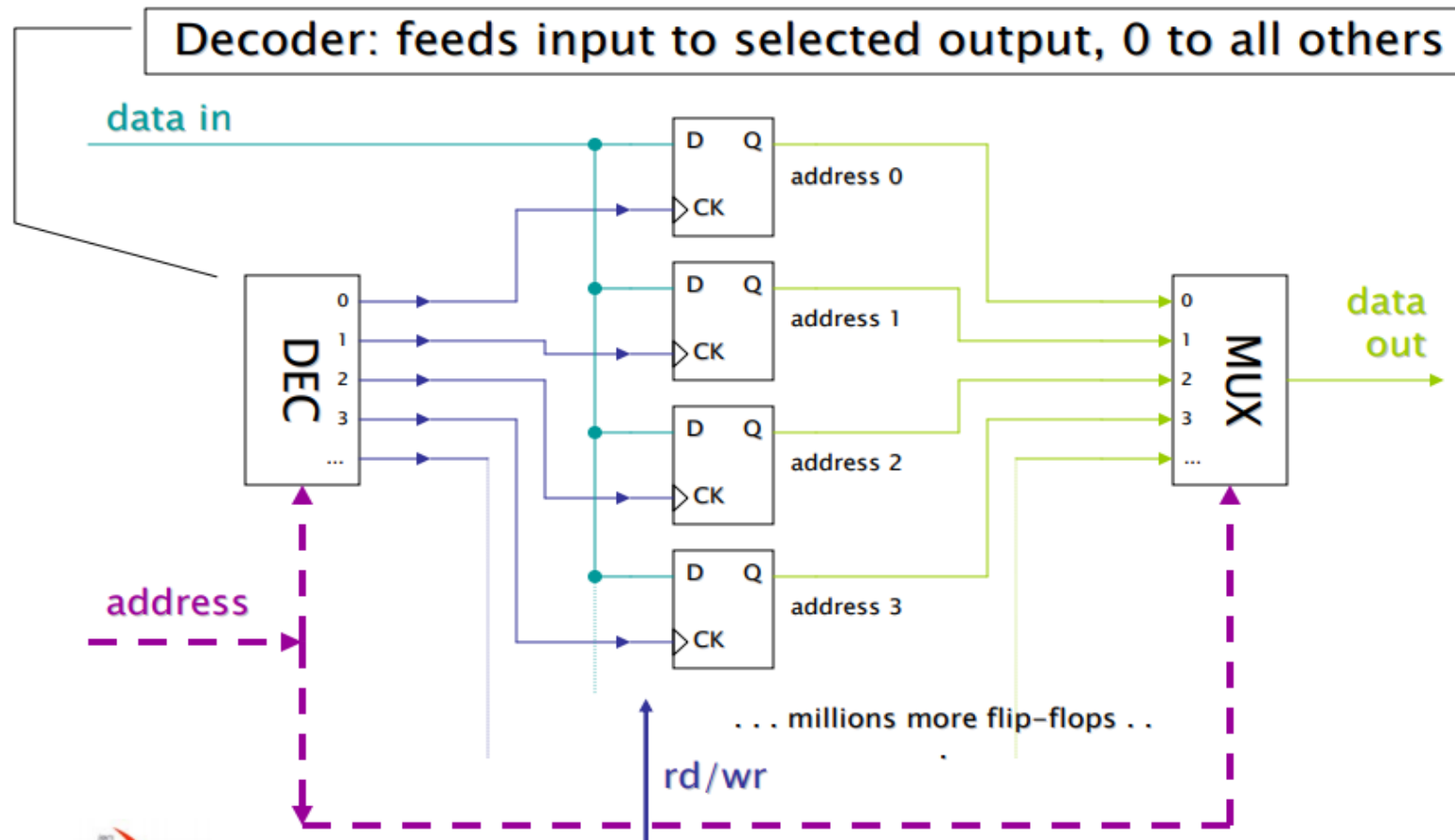
A register stores data
inside the CPU



Memory

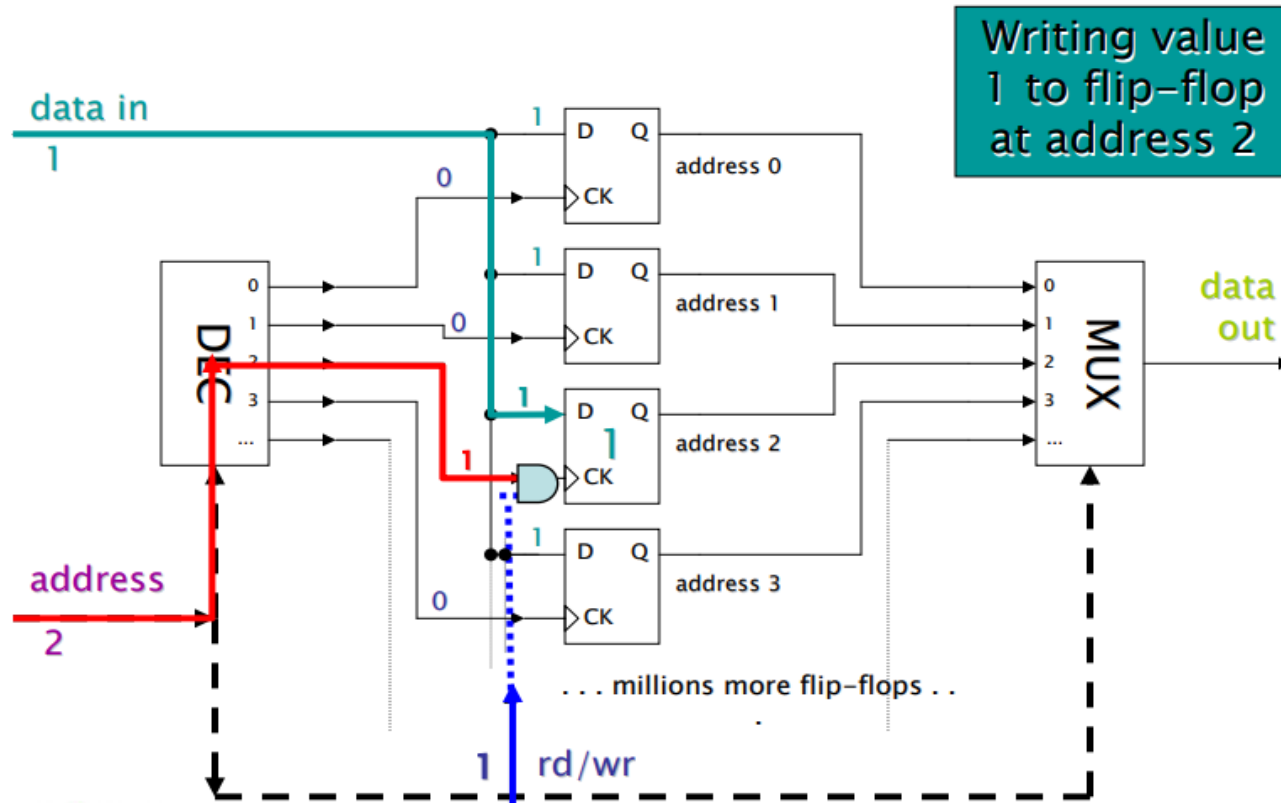
- Memory can store many bits independently
 - register banks contain many flip-flops
- Need to identify which bit (flip-flop) to read or write
- Give each flip-flop a unique number (address)

Memory: Circuit Diagram



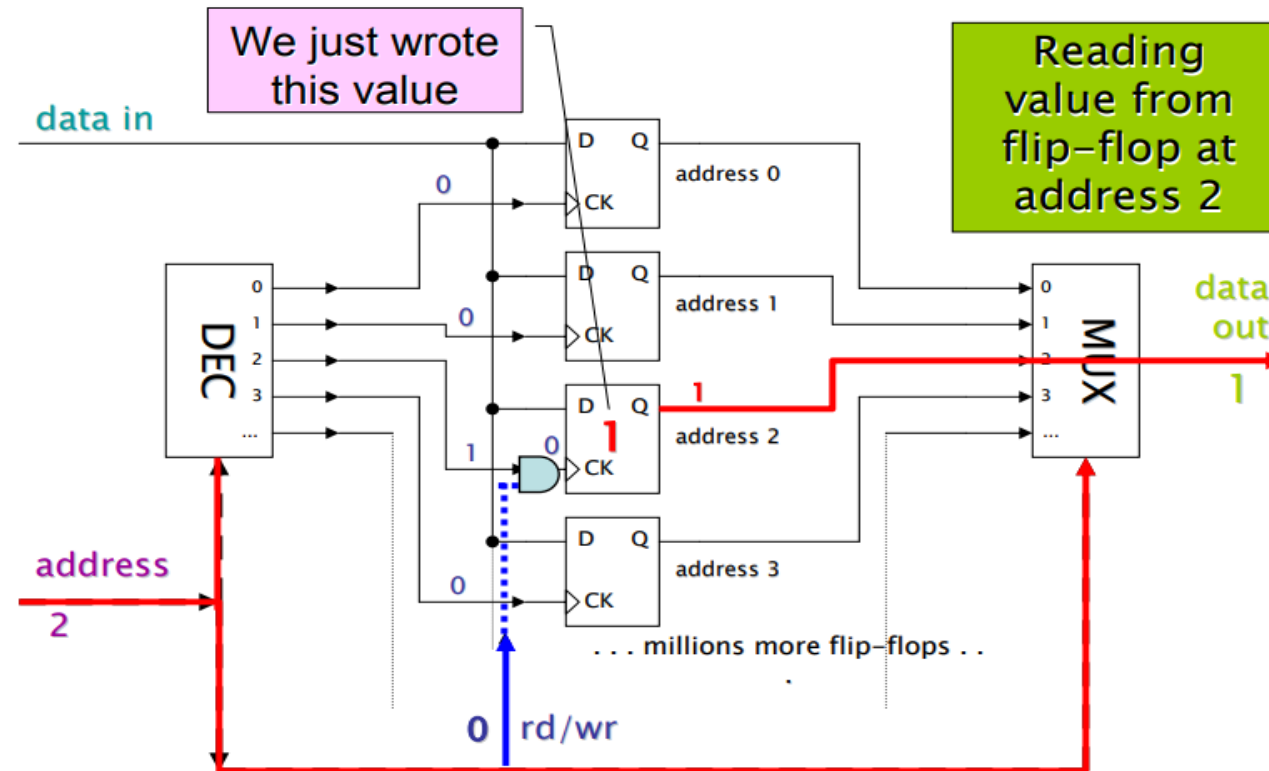
Memory: Writing

Memory: Writing

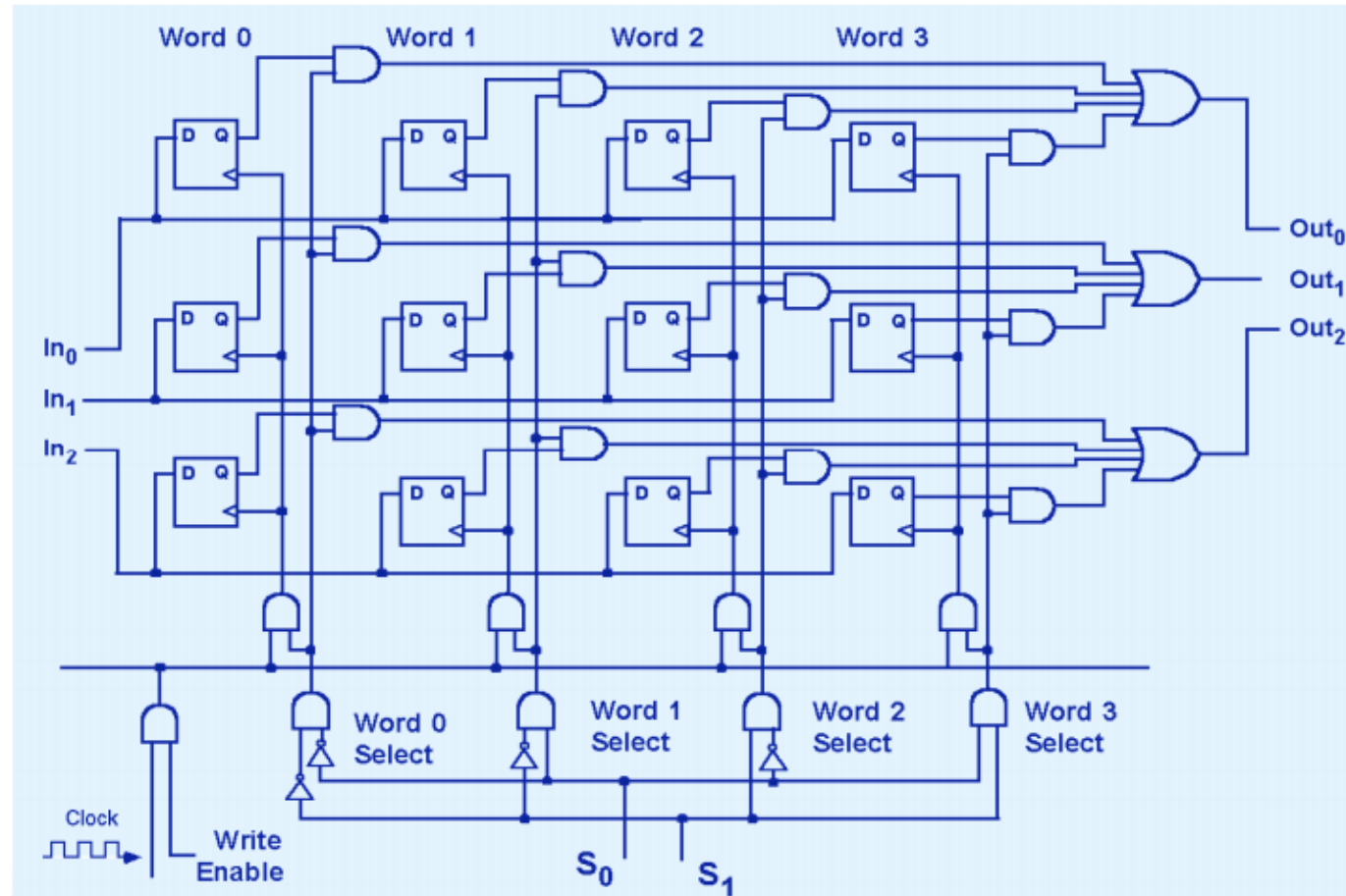


Memory: Reading

Memory: Reading

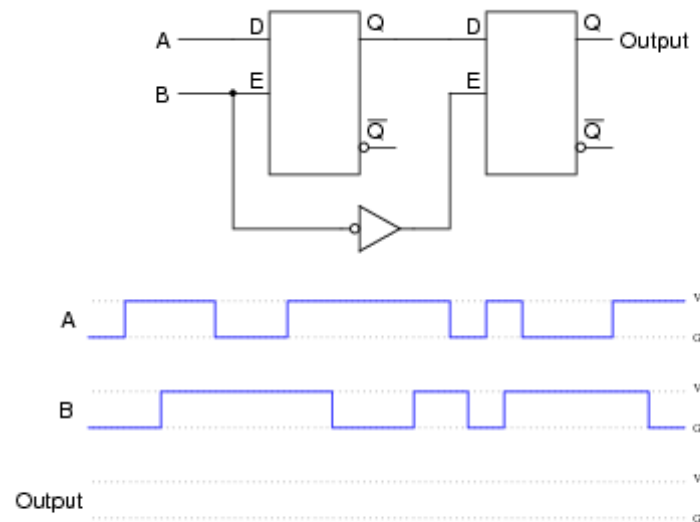


Memory: 4-words, 3 bits/word



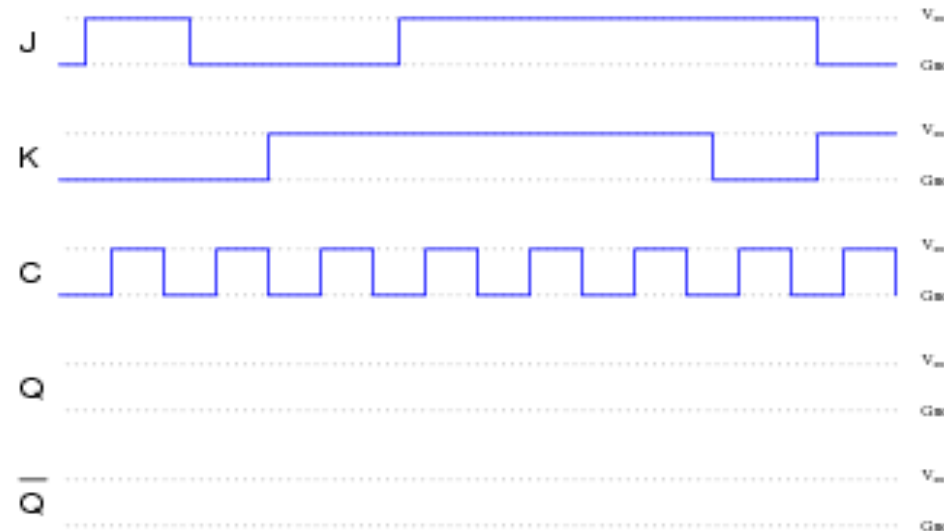
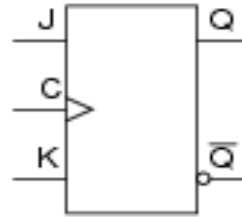
Activity - 03

- Determine the final output states over time for the following circuit, built from D-type gated latches:



At what specific times in the pulse diagram does the final output assume the input's state? How does this behavior differ from the normal response of a D-type latch?

2. Determine the output states for this J-K flip-flop, given the pulse inputs shown:



Pre- requisites

- Computers are implementations of Boolean logic.
- Boolean functions are completely described by Truth Tables.
- Logic gates are small circuits that implement Boolean operators.
- The basic gates are AND, OR and NOT.
- The “universal gates” are NOR and NAND.

Summary

- Computer circuits consist of combinational logic circuits and sequential logic circuits
- Combinational circuits produce outputs (almost) immediately when their inputs change
- Sequential circuits have internal states as well as combinations of input and output logic
 - The outputs may also depend on the states left behind by previous inputs
 - Sequential circuits require clocks to control their changes of state
 - The basic sequential circuit unit is the flip-flop