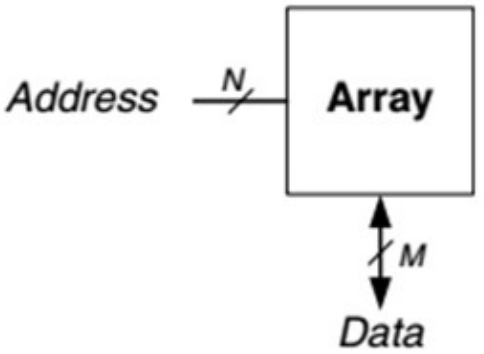
Lessen 3

Digital Electronics Review part 3

Mohd Saufy Rohmad
Technical Trainer and Consultant

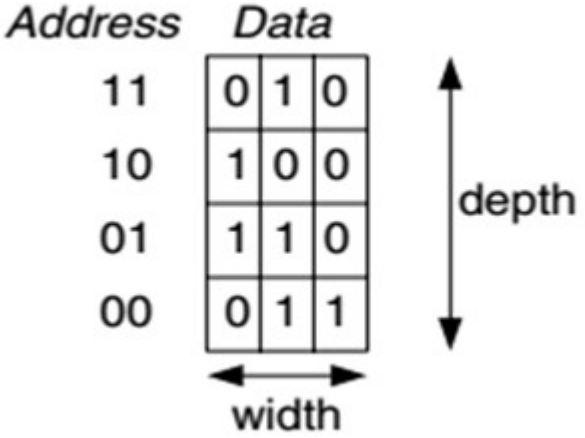
Memory Arrays

- In the digital system, memory is required for storing data which is generated by circuit.
- Resistor built from flip-flop is a memory that store small amount of data.
- If we observe the generic memory array symbol that consists of two dimensional array of memory cells.
- The memory reads or writes the contents of one of the rows of the array is specified by an Address and value read or written is called Data.
- On the other hand, an array with N-bit addresses and M-bit data has 2^N rows and M columns where each row of data is known as word.



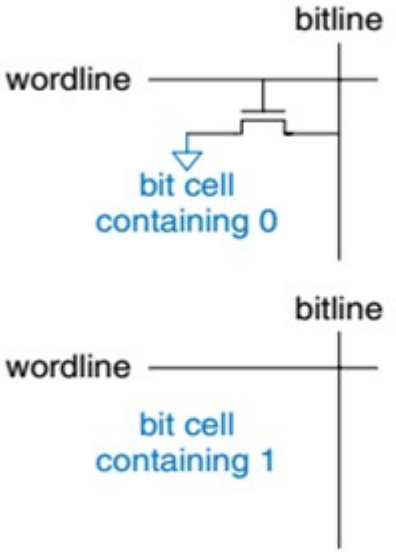
Memory Arrays

- In the below figure we noticed a memory array with two address bits and three data bits.
- The two address bits specify one of the four rows (data words) in the array.
- Each data word is three bits wide.
- The depth of an array is the number of rows and the width is the number of columns called word size.
- The size of an array is given as depth×width.



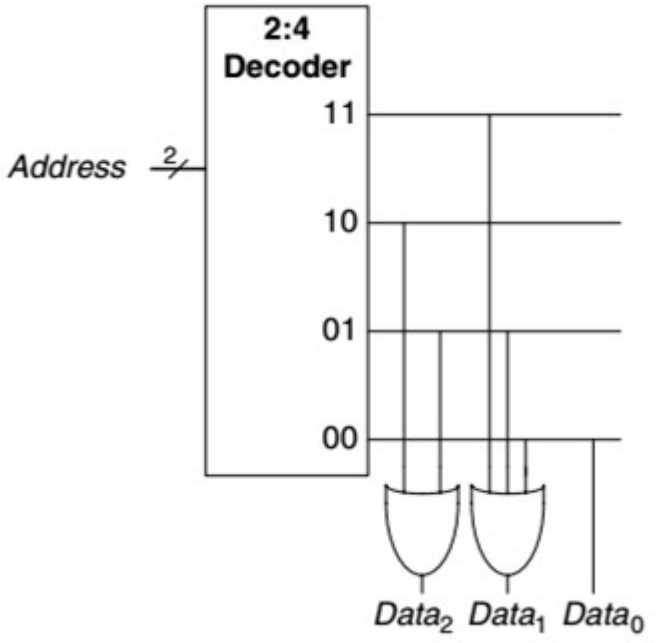
ROM

- ROM stores a bit as the presence or absence of transistor. ROM is nonvolatile, meaning that it retains its data indefinitely, even without a power source.
- In the simple ROM bit cell, the bitline pulled high to read the cell therefore the wordline is turned ON.
- After that, if the transistor presents it pulls the bitline low and if absents bitline remain high.
- Note that the ROM bit cell is a combinational circuit and has no state to “forget” if power is turned off.



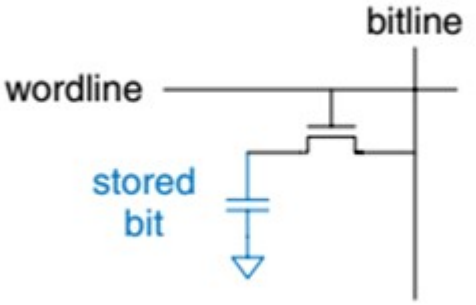
ROM

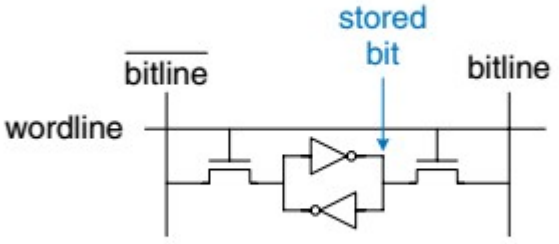
- ROMs can be built using two level logic with a group of AND gates followed by a group of OR gates.
- The AND gates produce all possible minterms in order to form a decoder.



RAM

- RAM allows data items to be read or write in same amount of time.
- However, RAM is volatile means it loses data whenever power is turn off.
- The two major types of RAMs based on how they store bits in the bit cell i.e., dynamic RAM (DRAM) and static RAM (SRAM).
- Dynamic RAM stores data as a charge on a capacitor, whereas static RAM stores data using a pair of cross-coupled inverters.

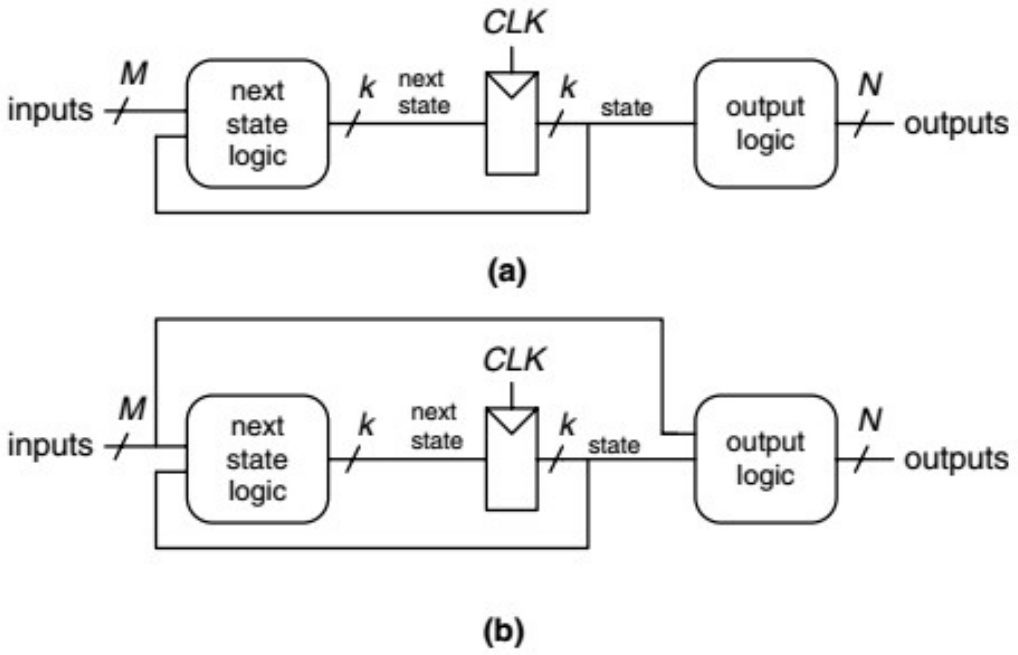




Finite State Machine(FSM)

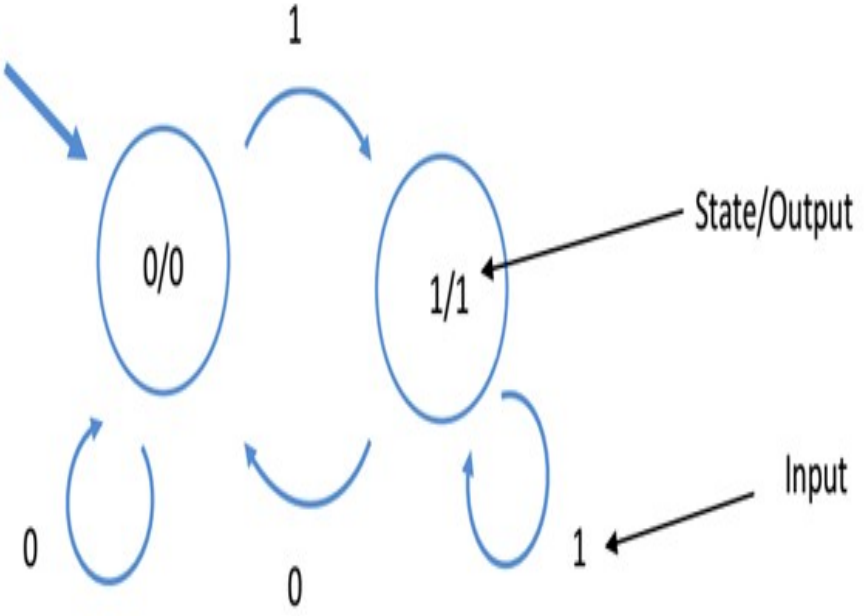
- An abstract machine that consist of finite number of state at any given time known as FSM.
- Normally, we consider a circuit with K registers that can be in one of a finite number (2^k) of unique states.
- In the figure we show that an FSM has M inputs, N outputs and k bits of state (determine by possible values in sequential storage elements).
- It also receives a clock that controls when state can change by controlling storage elements.
- FSM consists of two block of combinational logic next state logic and output logic.
- There are two general classes of finite state machines, characterized by their functional specifications.
- In Moore machines, the outputs depend only on the current state of the machine.
- In Mealy machines, the outputs depend on both the current state and the current inputs.

Finite State Machine (FSM)



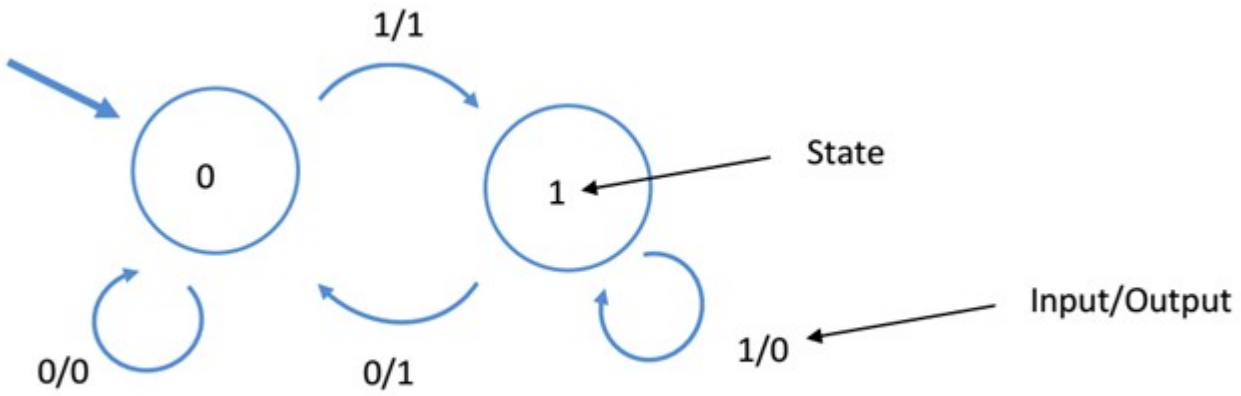
Finite State Machine (FSM)

- Moore machines: The output values are determined by its current state.
- In this case, it’s just a coincidence that the output and state values match.



Finite State Machine (FSM)

- Mealy machine: The outputs depend on both the current state and the current inputs.



Useful Terminology for Digital Design

- Propagation Delays
  - A logic signal always experiences a delay in going through a circuit.
  - The two propagation delay times are defined as follows:
    - T_plh : Delay time in going from logical 0 to logical 1 state (LOW to HIGH)
    - T_phl: Delay time in going from logical 1 to logical 0 state (HIGH to LOW)
  - Figure 8-2 illustrates these propagation delays for an INVERTER.
  - Note that T_phl is the delay in the output's response as it goes from HIGH to LOW.
  - It is measured between the 50 percent points on the input and output transitions.
  - The T_plh value is the delay in the output's response as it goes from LOW to HIGH.
  - In some logic circuits, T_phl and T_plh are not the same value, and both will vary depending on capacitive loading conditions.
  - The values of propagation times are used as a measure of the relative speed of logic circuits.
  - For example, a logic circuit with values of 10 ns is a faster logic circuit than one with values of 20 ns under specified load conditions.

Useful Terminology for Digital Design

- Fan-Out
  - In general, a logic-circuit output is required to drive several logic inputs.
  - Sometimes all ICs in the digital system are from the same logic family, but many systems have a mix of various logic families.
  - The fan-out (also called loading factor) is defined as the maximum number of logic inputs that an output can drive reliably.
  - For example, a logic gate that is specified to have a fan-out of 10 can drive 10 logic inputs.
  - If this number is exceeded, the output logic-level voltages cannot be guaranteed.
  - Obviously, fan-out depends on the nature of the input devices that are connected to an output.
  - Unless a different logic family is specified as the load device, fan-out is assumed to refer to load devices of the same family as the driving output.

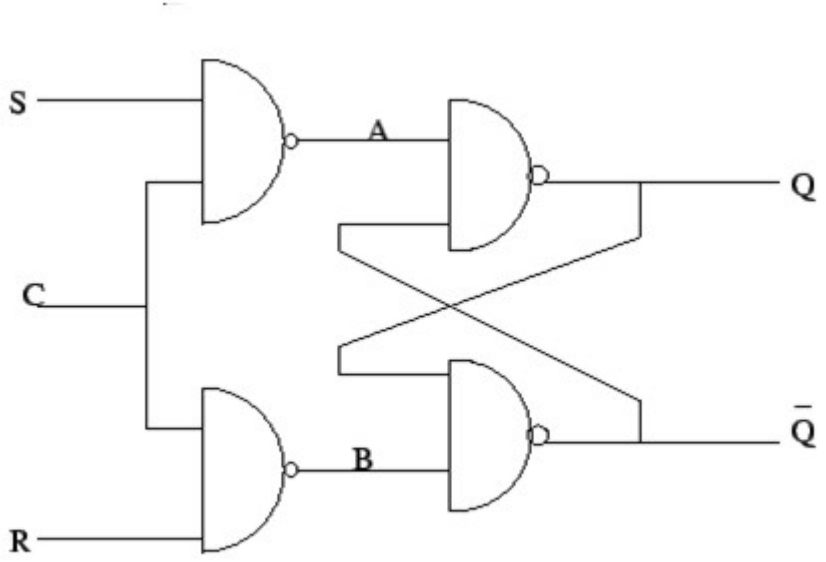
Useful Terminology for Digital Design

- Glitch
  - In the digital design small unwanted pulses appears in the output called glitch.

Useful Terminology for Digital Design

- Race Condition
  - In digital circuits Race conditions occur when the final state of the output depends on how the inputs arrive.
  - Digital circuits have inherent delays.
  - So it is possible that one of the inputs arrive a little earlier or later than others i.e. the inputs which were meant to be present at the same time actually arrive at different times due to different delays along their path.
  - As a result of this, the output changes unpredictably.
  - In other words, there is a race among the inputs as to which one will affect the output.
  - Generally this takes the form of spikes, which can be both high or low.

Useful Terminology for Digital Design



Useful Terminology for Digital Design

- Consider what will happen if both S and R are high.
- Assume that Q = 0 and Q' = 1 initially.
- If A arrives before B, Q will change to High which will momentarily set Q' low which in turn should have ideally kept Q high and so on.
- Now after a moment, B arrives (very short duration).
- This will turn Q' High which in turn will set Q High.
- You can check to see what happens when B arrives before A.
- Now actually, there are two things happening here:
  - 1) The output depends momentarily on which input arrives first. This in essence is the race condition.
  - 2) The final state is Q = 1 and Q' = 1 . This is NOT race condition. This is merely an invalid state. Ideally, Q and Q' must be opposite, which is not the case here.

Thank You