COE 530
Quantum Computer And Architecture

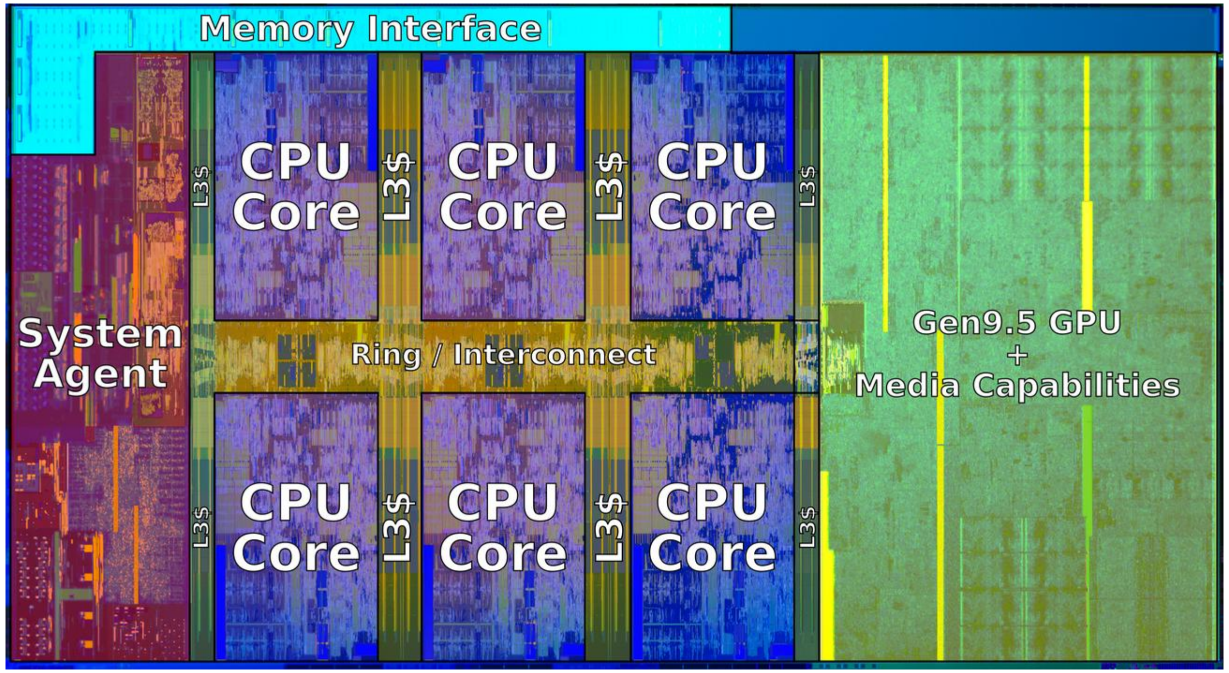
Lecture 2

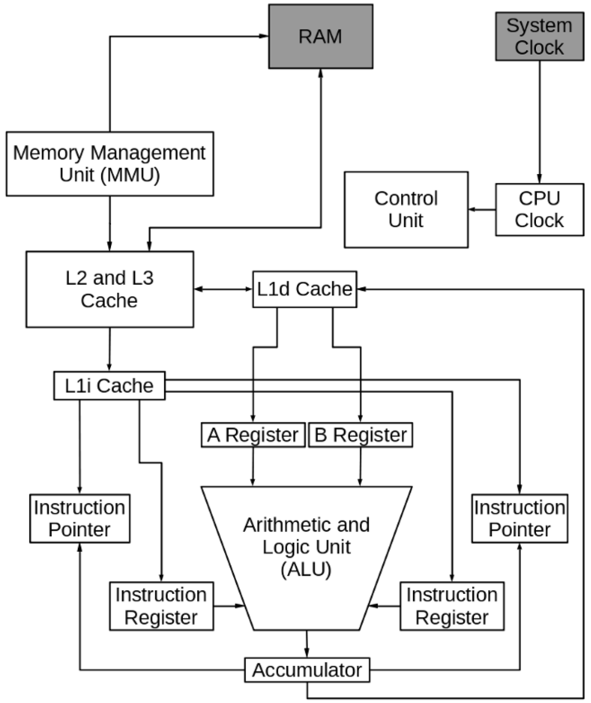
Classical Computer System III
Memory Units

References:

Quantum Random Access Memory For Dummies https://arxiv.org/pdf/2305.01178

State of the art CPU





https://www.redhat.com/sysadmin/cpu-components-functionality

This slide is from CMU course

Combinational vs Sequential Cricuit

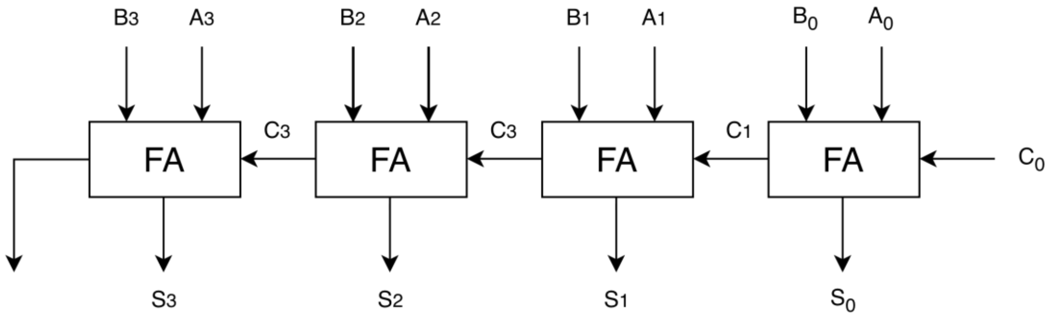
- FA is an example of combinational digital circuit

$$output = f(input)$$

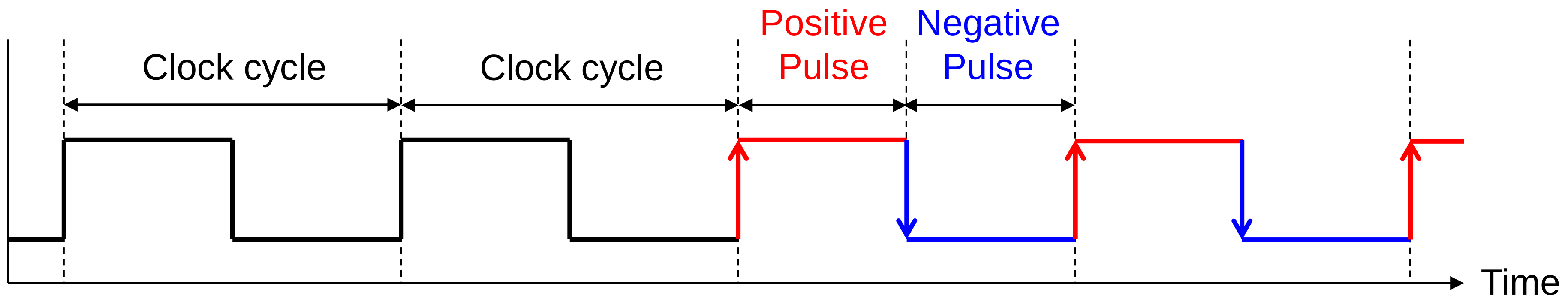
- That is, the output only depends on the input
- Sometimes, we want the output to be dependent on the input and the previous output

$$output = f(input, previous\ output)$$

- This type of circuits is called *Sequential Circuits*
- Sequential circuits are used to build memory elements

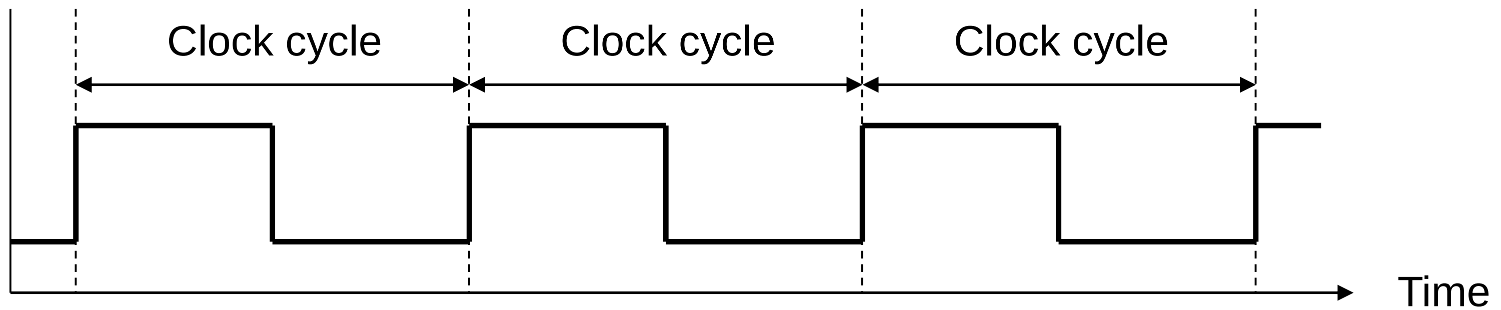


The Clock



- Clock is a periodic signal = Train of pulses (1's and 0's)
- The same clock cycle repeats indefinitely over time
- **Positive Pulse**: when the **level** of the clock is **1**
- **Negative Pulse**: when the **level** of the clock is **0**
- **Rising Edge**: when the clock goes **from 0 to 1**
- **Falling Edge**: when the clock goes **from 1 down to 0**

Clock Cycle versus Clock Frequency



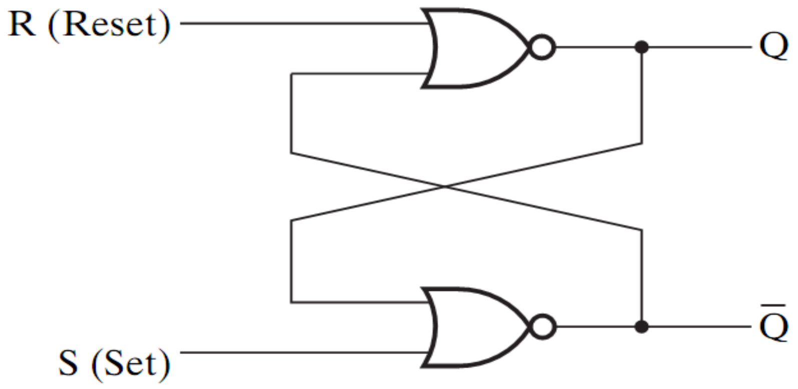
- Clock cycle (or period) is a time duration
  - Measured in seconds, milli-, micro-, nano-, or pico-seconds
  - 1 ms = 10^{-3} sec, 1 µs = 10^{-6} sec, 1 ns = 10^{-9} sec, 1 ps = 10^{-12} sec
- Clock frequency = number of cycles per second (Hertz)
  - 1 Hz = 1 cycle/sec, 1 KHz = 10^3 Hz, 1 MHz = 10^6 Hz, 1 GHz = 10^9 Hz
- Clock frequency = 1 / Clock Cycle
  - Example: Given the clock cycle = 0.5 ns = 0.5×10^{-9} sec
  - Then, the clock frequency = $1/(0.5\times10^{-9}) = 2\times10^9$ Hz = 2 GHz

Memory Elements

- Memory can store and maintain binary state (0's or 1's)
  - Until directed by an input signal to change state
- Main difference between memory elements
  - Number of inputs they have
  - How the inputs affect the binary state
- Examples of memory elements
  - Latch
  - Flip-flop
  - Registers

SR Latch

- A latch is a memory element that can store 0 or 1
- An SR Latch can be built using two cross-coupled NOR gates
- Two inputs: S (Set) and R (Reset)
- Two outputs: Q and $\overline{Q}$



(a) Logic diagram

S	R	Q	$\overline{Q}$	
1	0	1	0	Set state
0	0	1	0	
0	1	0	1	Reset state
0	0	0	1	
1	1	0	0	Undefined

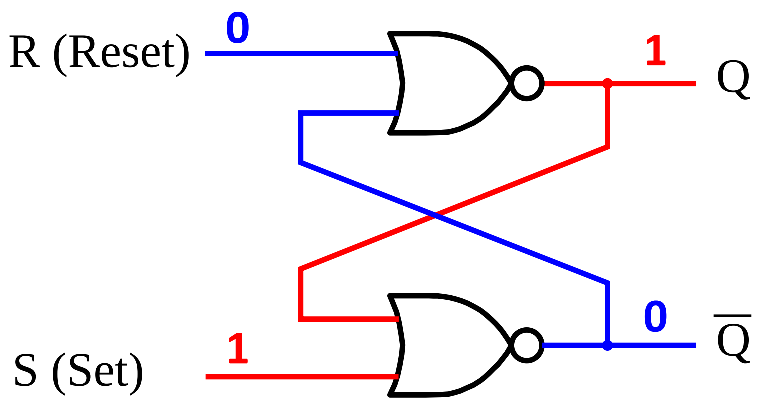
(b) Function table

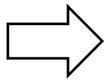
NOR

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

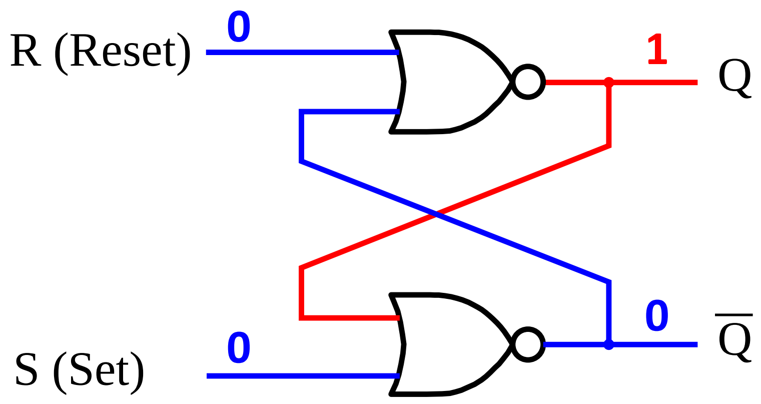
SR Latch Operation

Set Operation

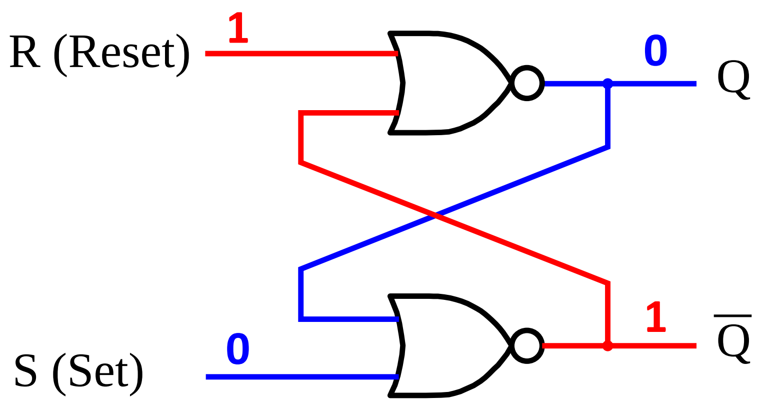


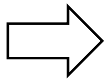


Store Operation

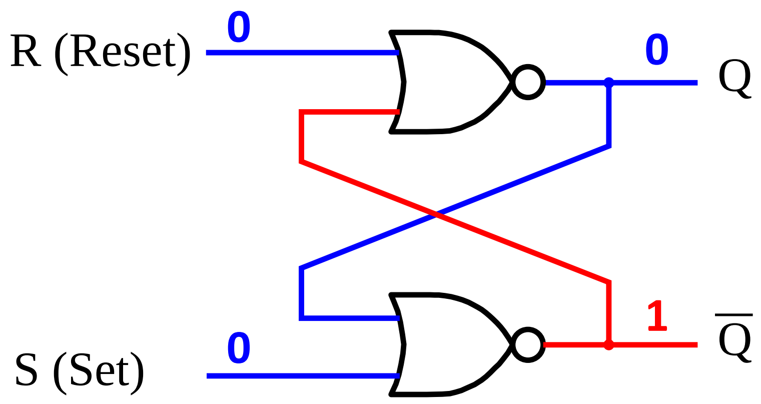


Reset Operation



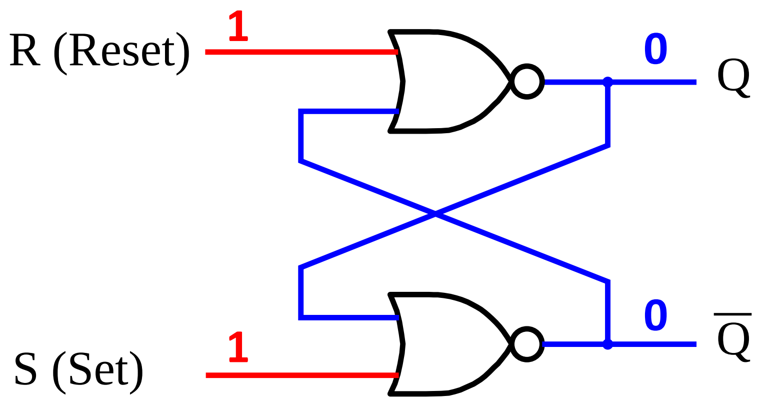


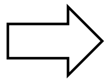
Store Operation



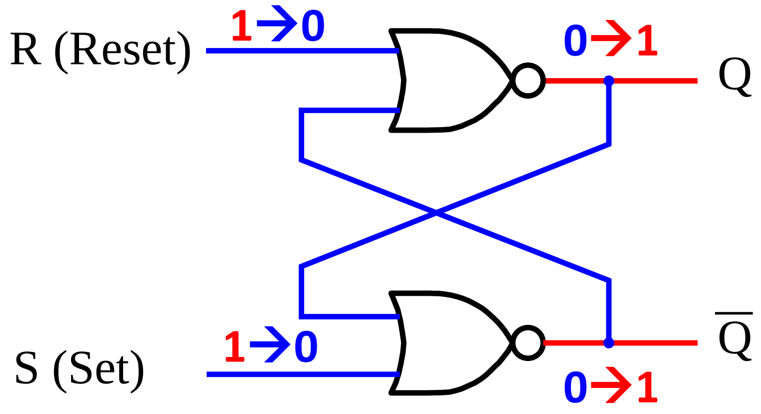
SR Latch Invalid Operation

Invalid Operation





Race Condition

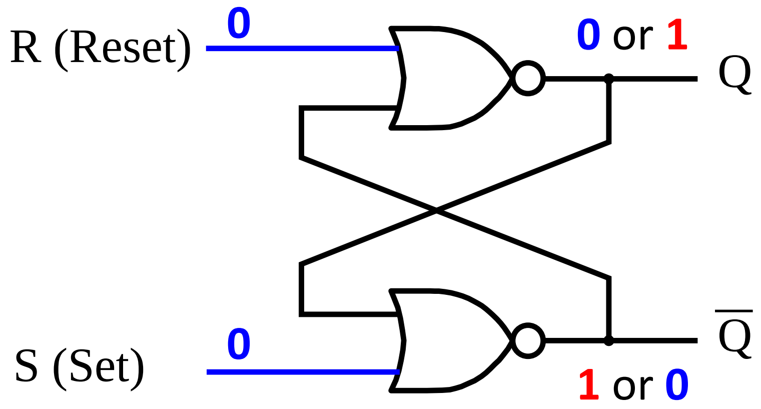


S = R = 1 should never be used

If S and R change from 1 → 0 simultaneously then race condition (oscillation) occurs

Final Q and $\overline{Q}$ are unknown

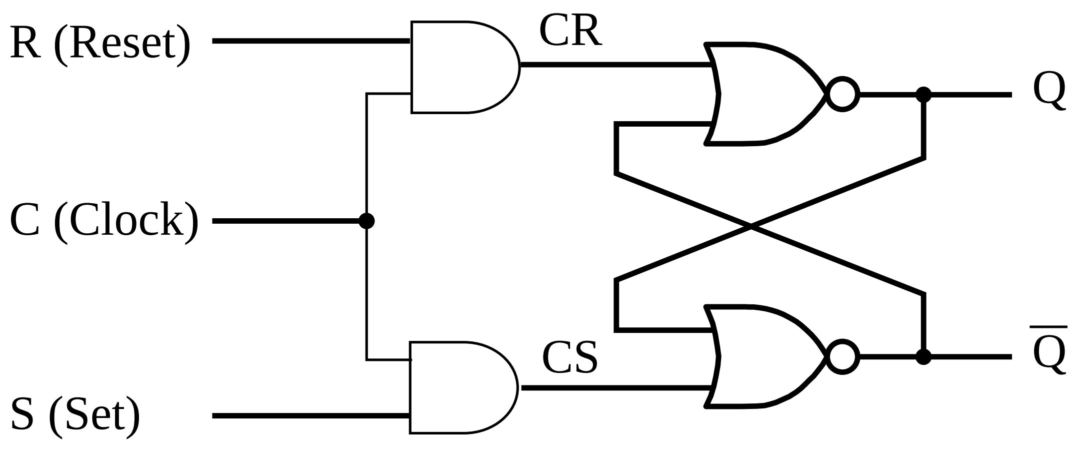
Unknown State



Slides are from Dr. Mohammad Mudawar – COE233

SR Latch with Clock Input

- An additional Clock (enable) input signal **C** is used
- Clock controls **when** the state of the latch can be changed
- When **C=0**, the S and R inputs have no effect on the latch
  - The latch will remain in the same state, regardless of S and R
- When **C=1**, then normal SR latch operation



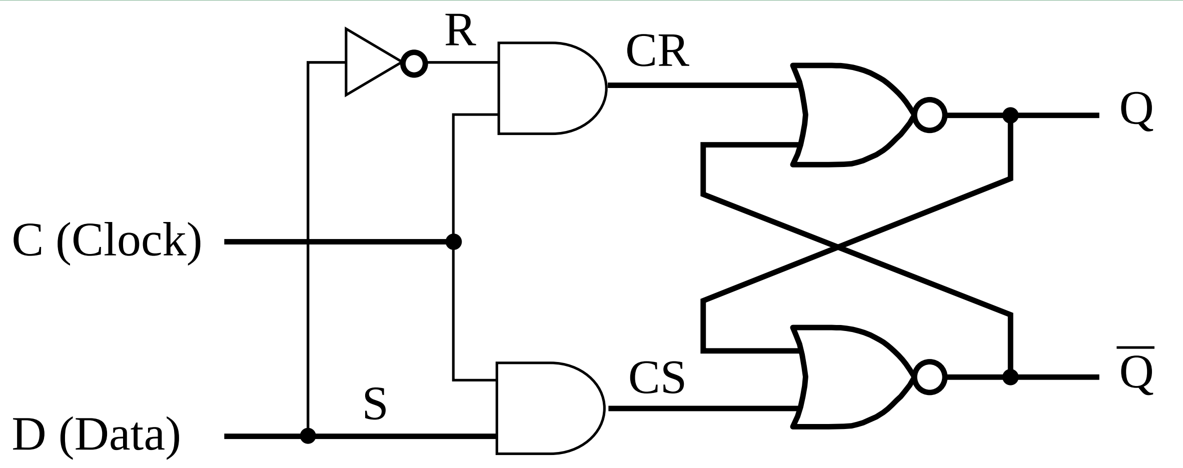
C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; Set state
1	1	1	Undefined

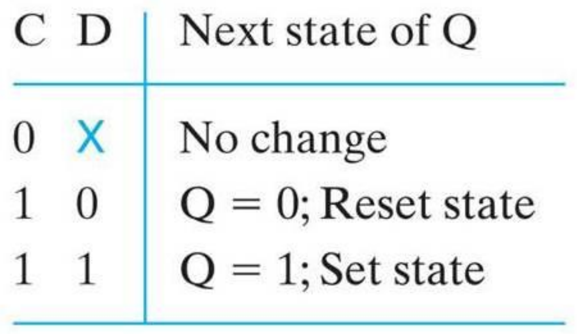
D Latch with Clock Input

- One data input D, $S = D$ and $R = \overline{D}$, No

  undefined state
- Clock controls **when** the state of the latch can be changed
- When **C=0**, the *D* input has no effect on the latch

  The latch will remain in the same state, regardless of *D*
- When **C=1**, latch is enabled, reset if *D* is 0 and set if *D* is 1

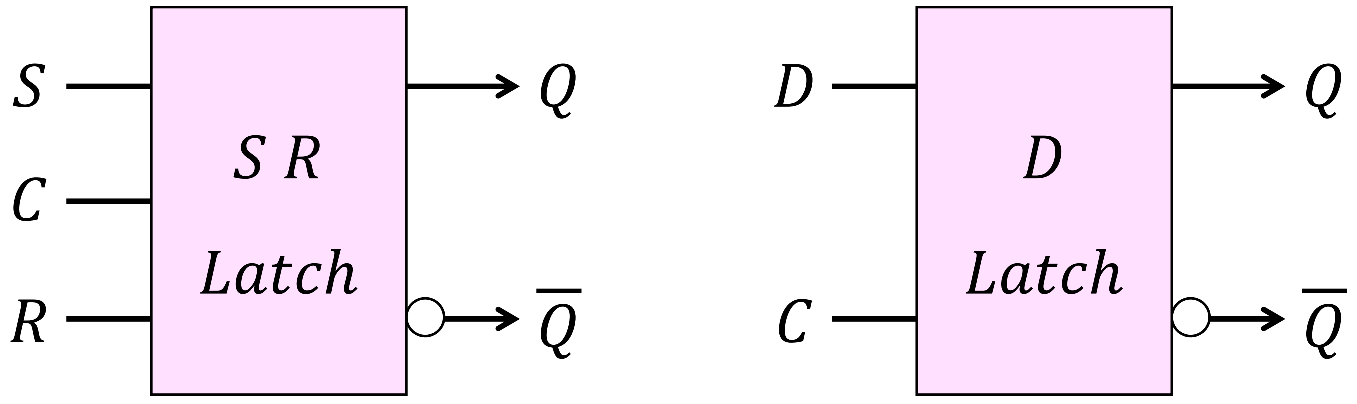




C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

(b) Function table

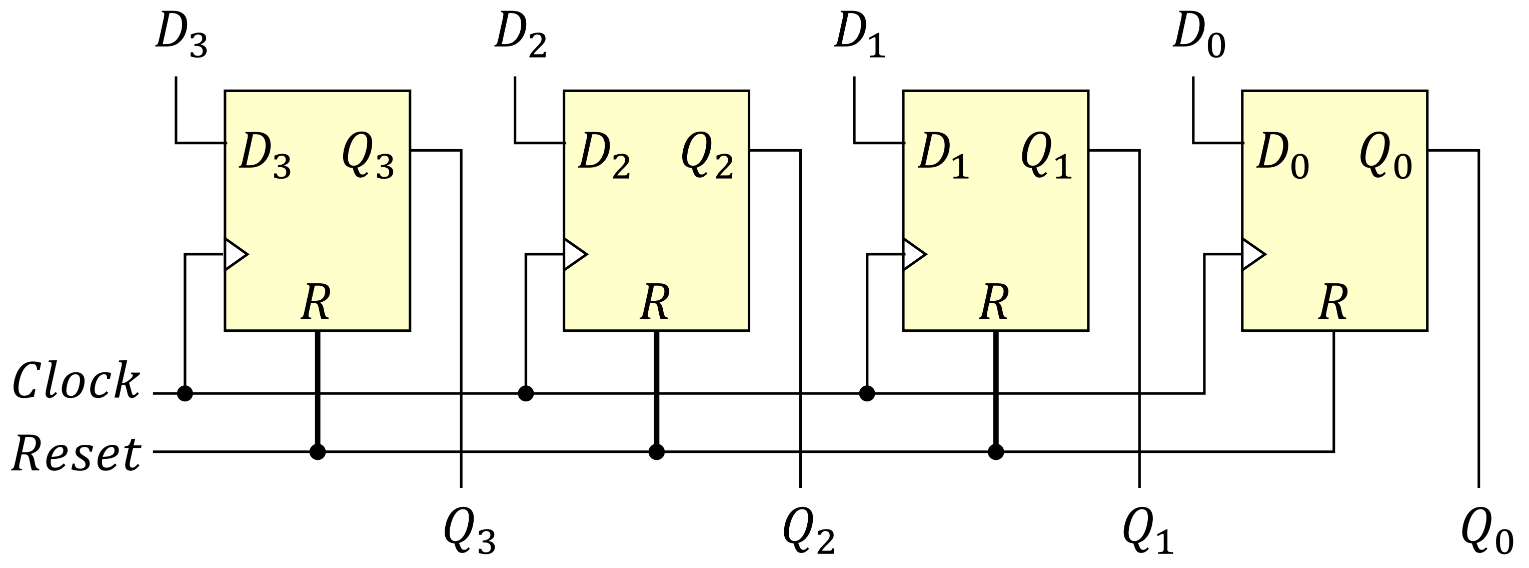
Graphic Symbols for Latches

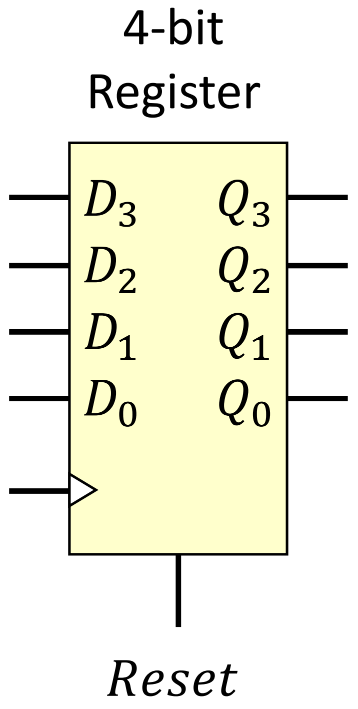


- A bubble appears at the complemented output $\overline{Q}$
- Indicates that $\overline{Q}$ is the complement of Q

Register

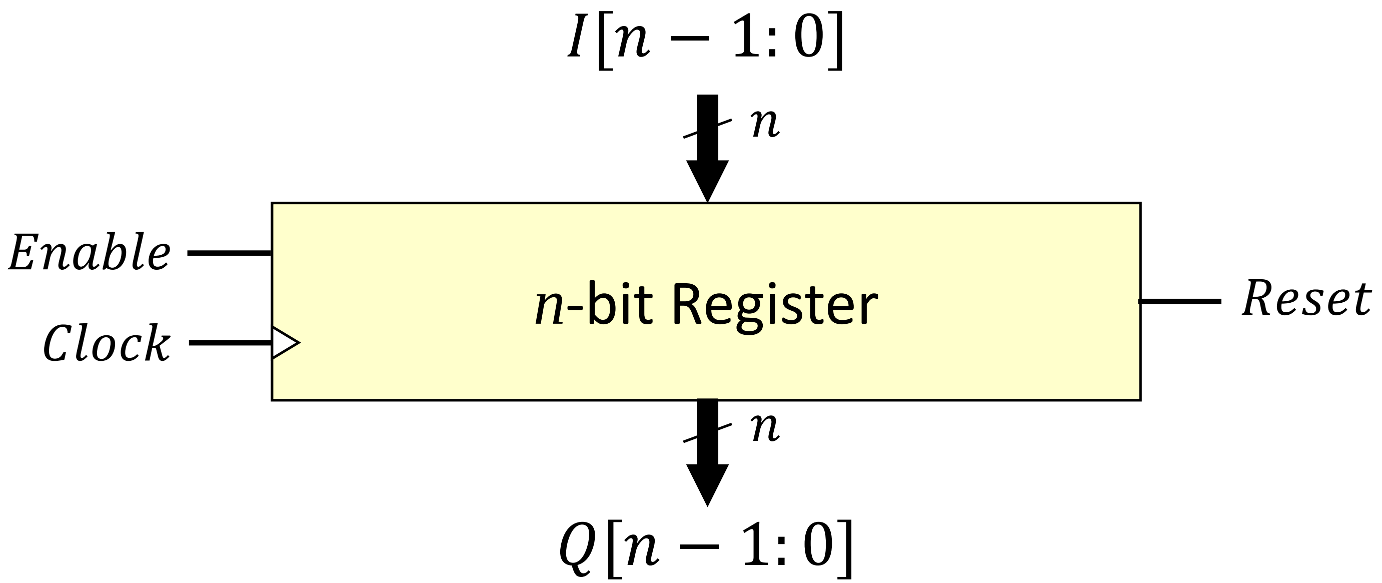
- A register is a circuit capable of storing data
- An n-bit register consists of n latches and stores n bits of data
- Common clock: data is loaded in parallel at the same clock
- Common reset: All latches are reset in parallel





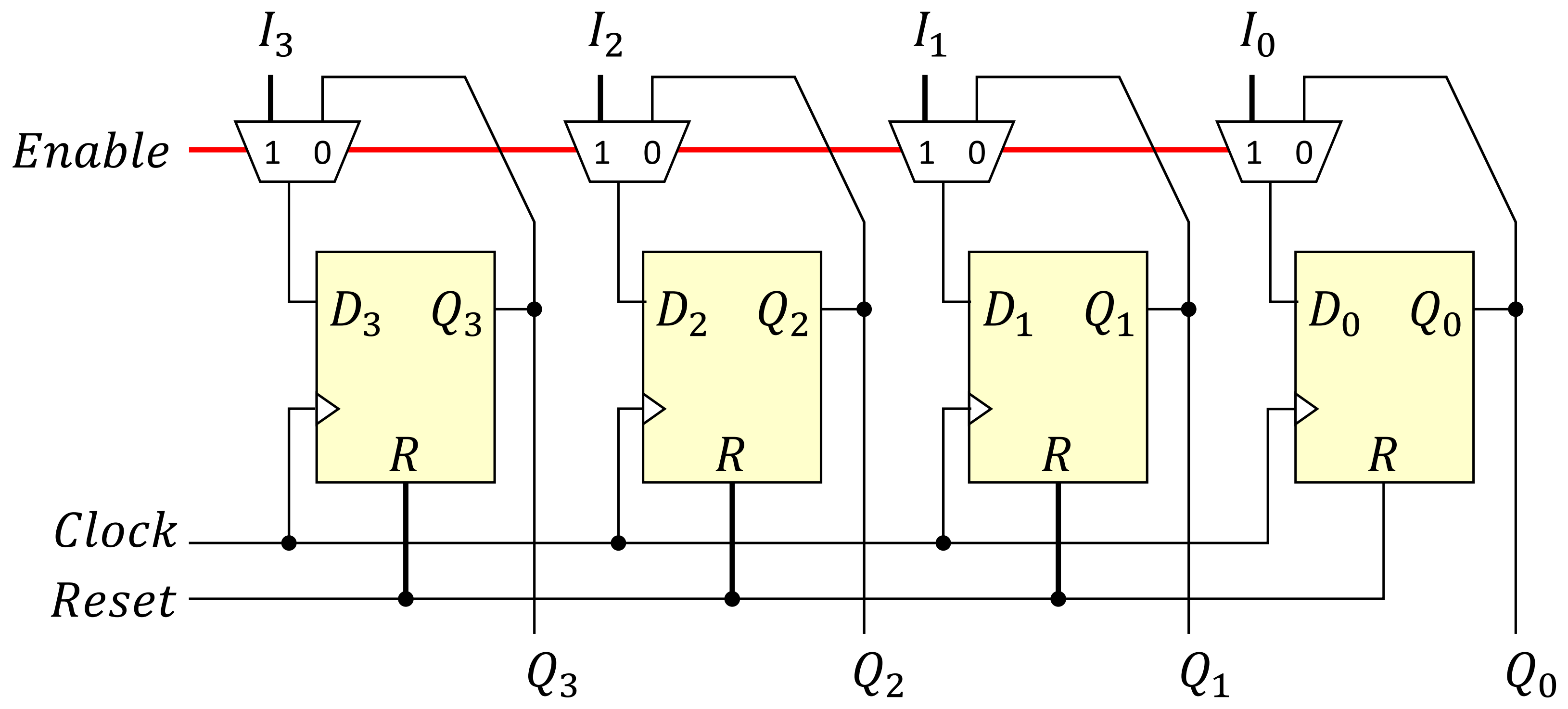
Register Enable

- **Question:** How to control the loading of data into a register?
- **Solution:** Introduce an Enable control signal
- If the register is enabled, load the data into the register
- Otherwise, do not change the value of the register
- Question: How to implement register enable?



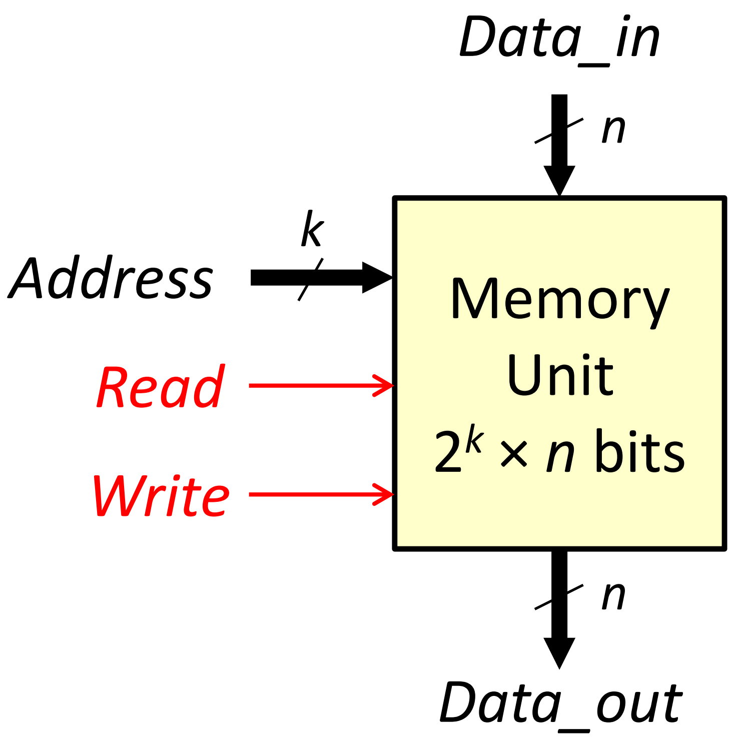
Implementing Register Enable

- **Solution:** Add a mux (multiplexer) at the D input of the register
- $D_i = Enable \cdot I_i + \overline{Enable} \cdot Q_i$
- If $Enable$ is **1** then $D_i = I_i$ If $Enable$ is **0** then $D_i = Q_i$



Random-Access Memory

- Large array of storage cells, capable of storing many 0's and 1's
- **Random Access:** bits can be accessed randomly
- Memory is addressable
- Memory address consists of k bits
- Can address 2^k words in memory
- Each word consists of n bits
- Memory capacity = $2^k \times n$ bits
- Two control functions: Read and Write
- Read: Data_out ← Memory [Address]
- Write: Memory [Address] ← Data_in



Memory Address and Content

- Example of a RAM

  Address = 10 bits

  2^{10} addresses

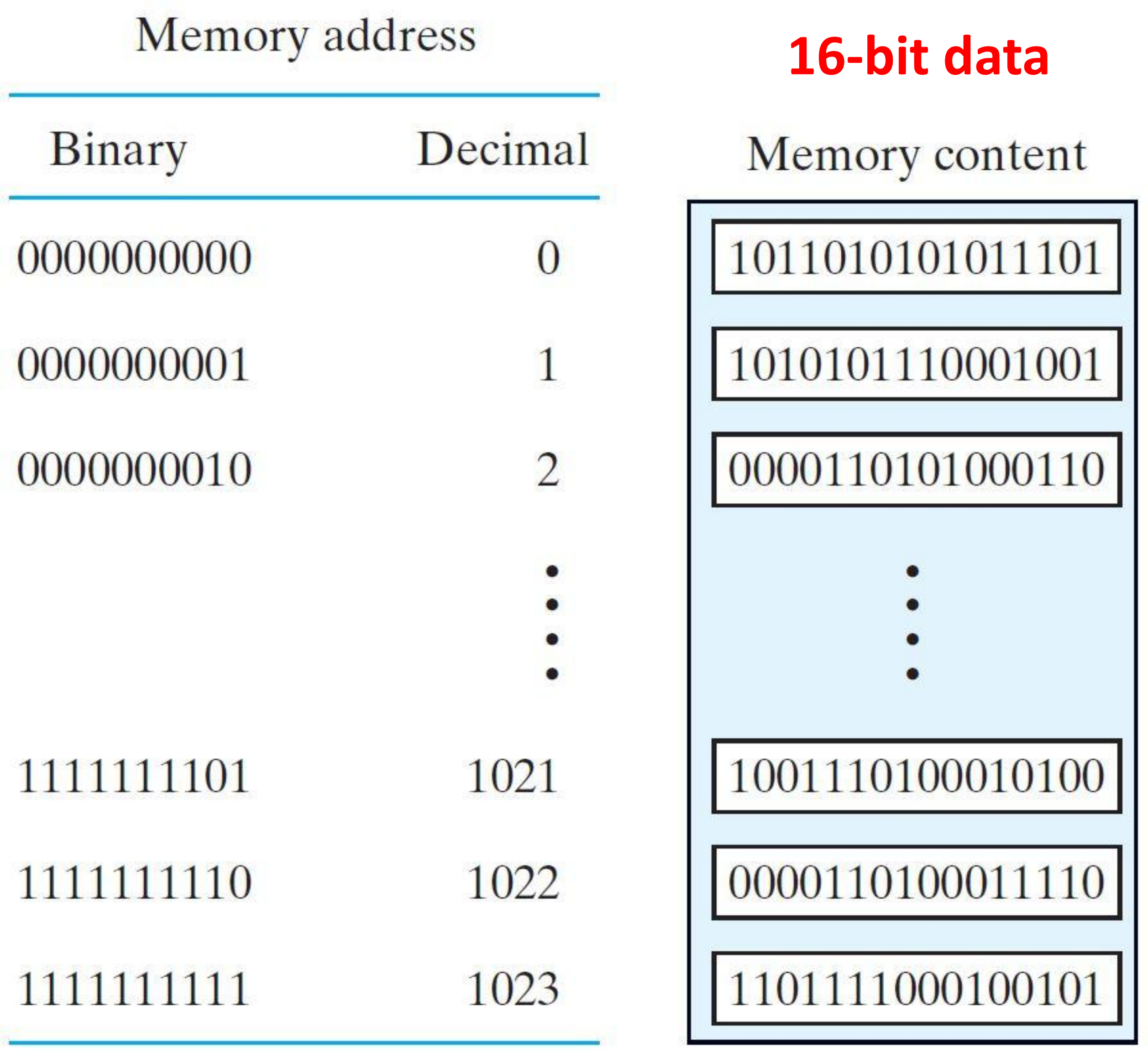
  From 0 to 1023

  Data = 16 bits
- Memory capacity =

  $2^{10} \times 16$ bits =

  16 Kbits = 2 KBytes
- Memory can be addressed randomly
- Memory can be read and written

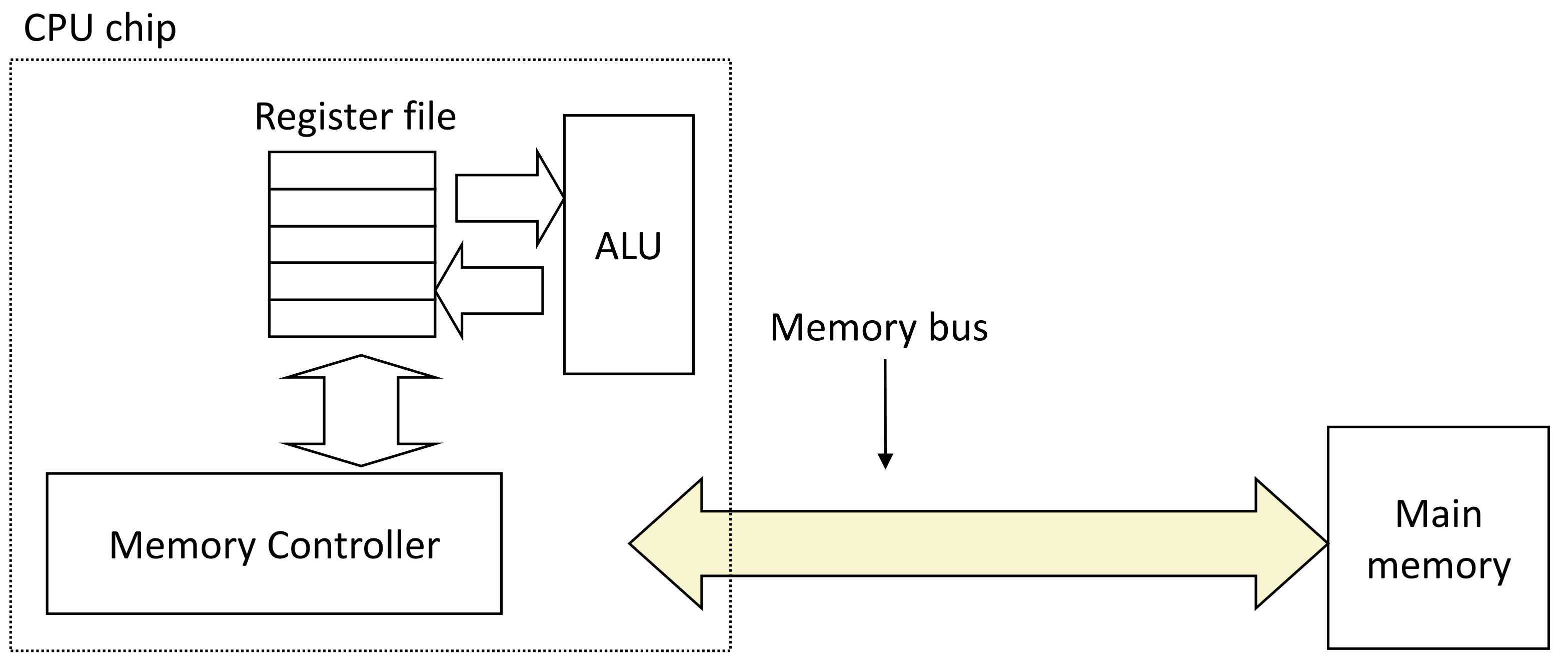
16-bit data



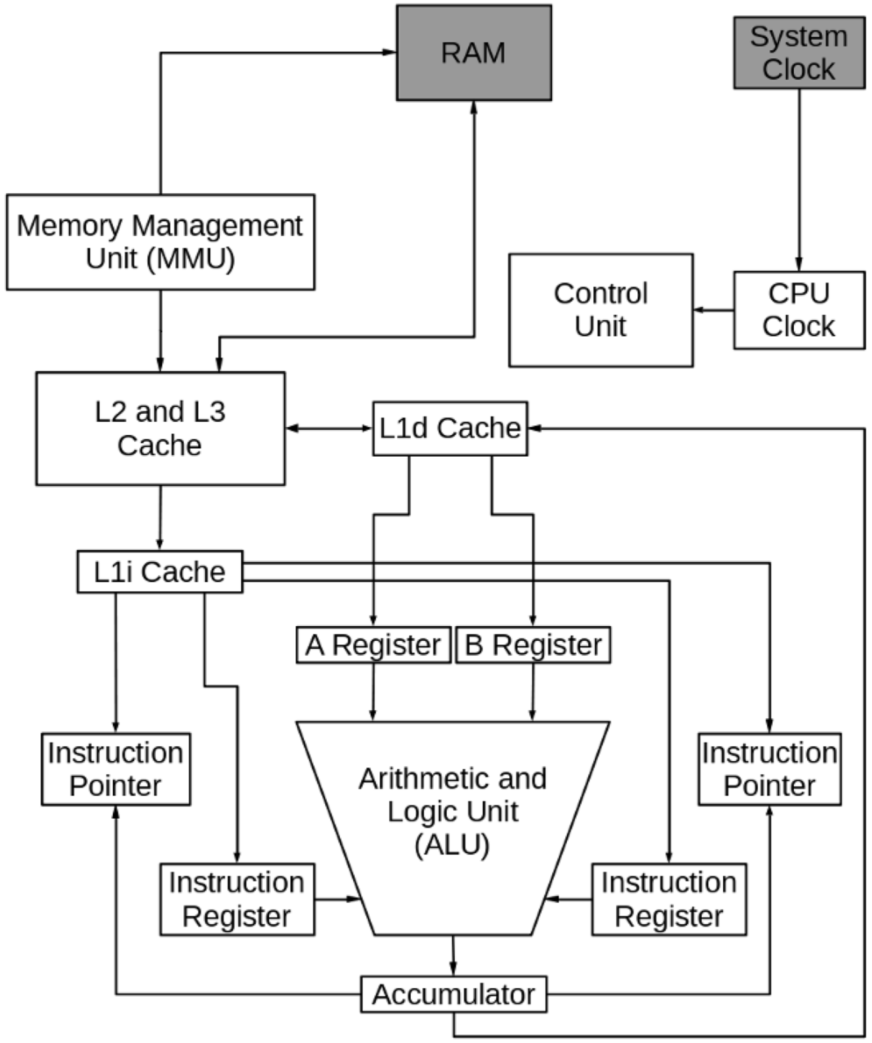
Memory address		Memory content
Binary	Decimal	
0000000000	0	1011010101011101
0000000001	1	1010101110001001
0000000010	2	0000110101000110
	⋮	⋮
1111111101	1021	1001110100010100
1111111110	1022	0000110100011110
1111111111	1023	1101111000100101

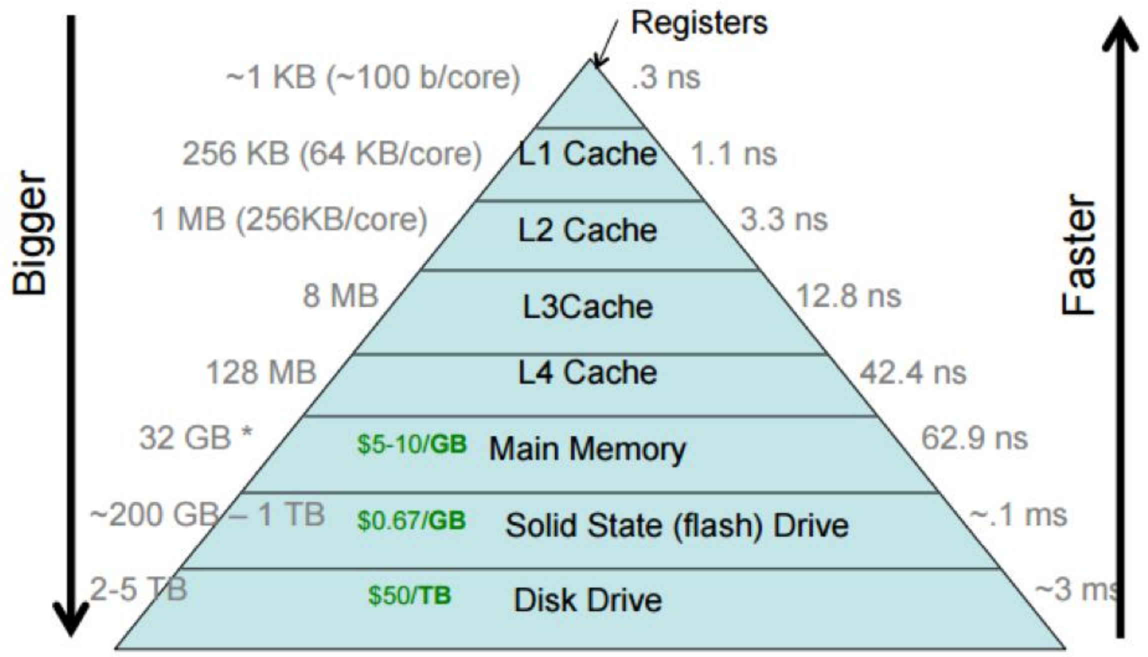
Modern Connection between CPU and Memory

- A bus is a collection of parallel wires that carry address, data, and control signals.
- Buses are typically shared by multiple devices.



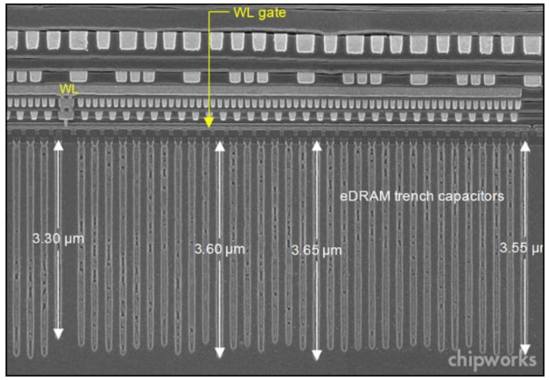
Memery Hierarchy





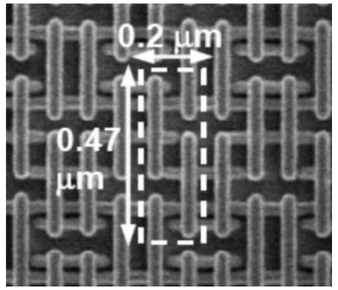
RAM Technologies

- DRAM



- 1 Transistor + 1 capacitor / bit
  - Capacitor oriented vertically
- Must refresh state periodically

- SRAM



- 6 transistors / bit
- Holds state indefinitely

Quatnum RAM

- A QRAM is a memory element analogous to RAM that is able to store data in quantum format
- The input and output registers are composed of qubits instead of bits, while the memory

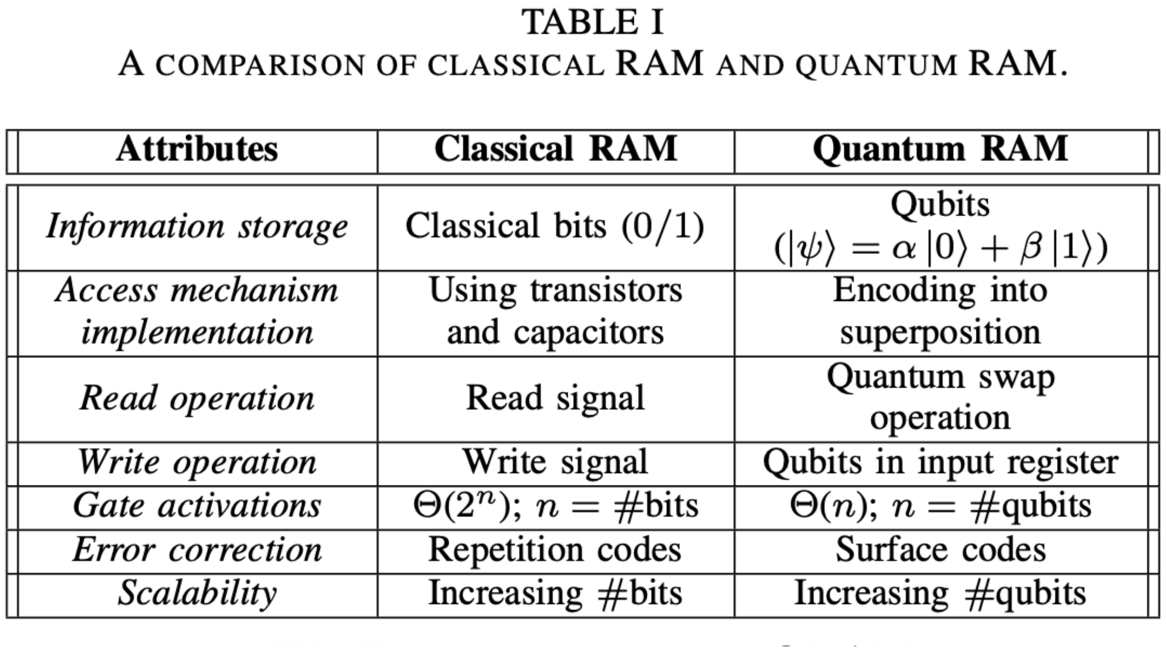


TABLE I
A COMPARISON OF CLASSICAL RAM AND QUANTUM RAM.

Attributes	**Classical RAM**	**Quantum RAM**
Information storage	Classical bits (0/1)	Qubits ($\|\psi\rangle = \alpha\|0\rangle + \beta\|1\rangle$)
Access mechanism implementation	Using transistors and capacitors	Encoding into superposition
Read operation	Read signal	Quantum swap operation
Write operation	Write signal	Qubits in input register
Gate activations	$\Theta(2^n)$; $n = \#$bits	$\Theta(n)$; $n = \#$qubits
Error correction	Repetition codes	Surface codes
Scalability	Increasing #bits	Increasing #qubits

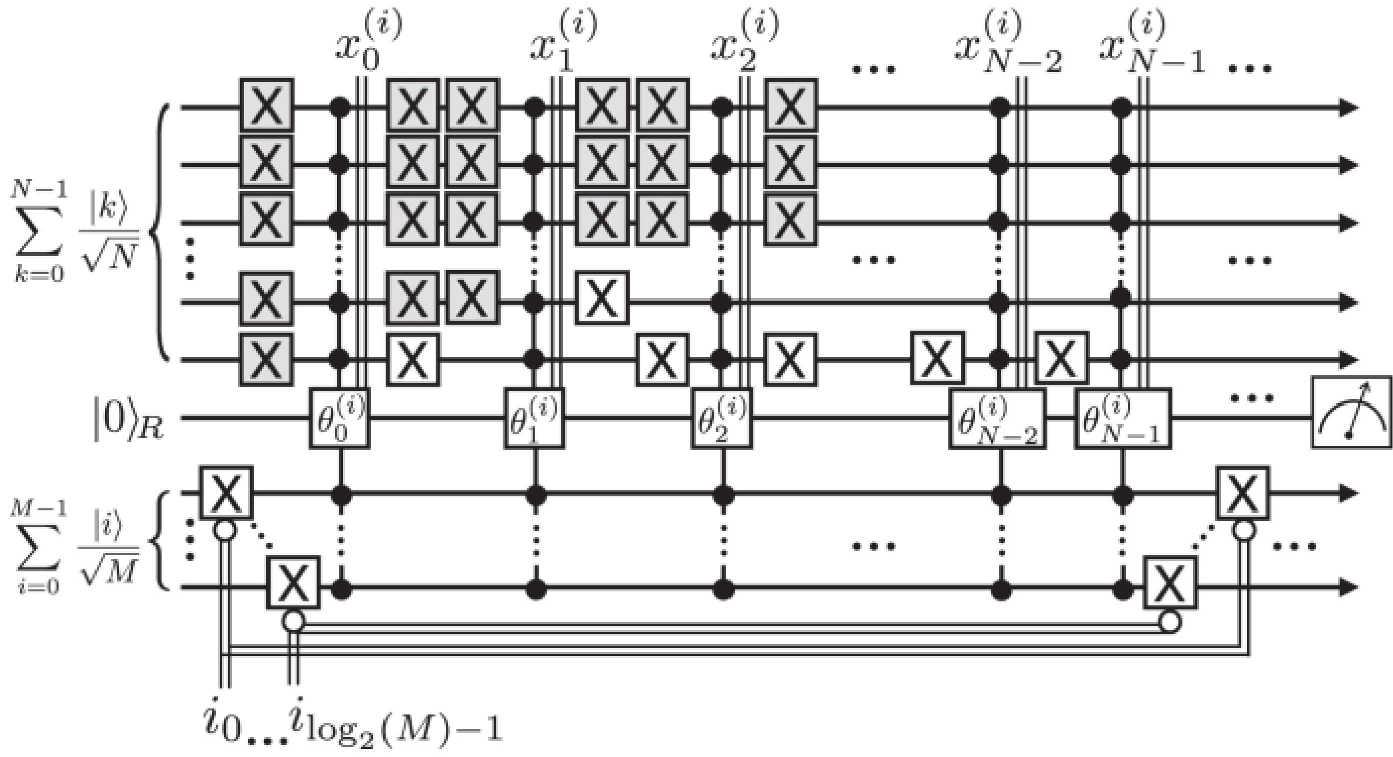
QRAM Mechanism

- Unlike RAM, QRAM can access multiple mumory locations simultanously using superposition
- Assume n qubits (i.e. $N = 2^n$ address lines)
- All addresses are presented using basis states $|0\rangle \ldots |N-1\rangle$ and are stored in register r
- Each register $|i\rangle$ will have amplitude α_i, so the superposition of all address states is $\sum_{i=0}^{N-1} \alpha_i |i\rangle_r$
- This superposition is sent to r and outputs another superposition states $\sum_{i=0}^{N-1} \alpha_i |i\rangle_r |X_i\rangle_o$
- QRAM can be considered as a quantum gate that performs the following

$$\sum_{i=0}^{N-1} \alpha_i \, |i\rangle_r \xrightarrow{QRAM} \sum_{i=0}^{N-1} \alpha_i \, |i\rangle_r |X_i\rangle_o$$

QRAM Implementation

- QRAM has three components:
  - the input (or address) register,
  - the output (or data) register,
  - and the memory arrays.
- One challenge in QRAM is the *no-cloning* therem, i.e.,
  - You can’t copy the qubit
  - Qubits are transferred using entanglement operations
- Another challenge is the qubit decoherence time
  - Qubits need to be encoded in a special way



Bucket-Brigade QRAM

Works in two modes:
1- single address case
2- superposition case

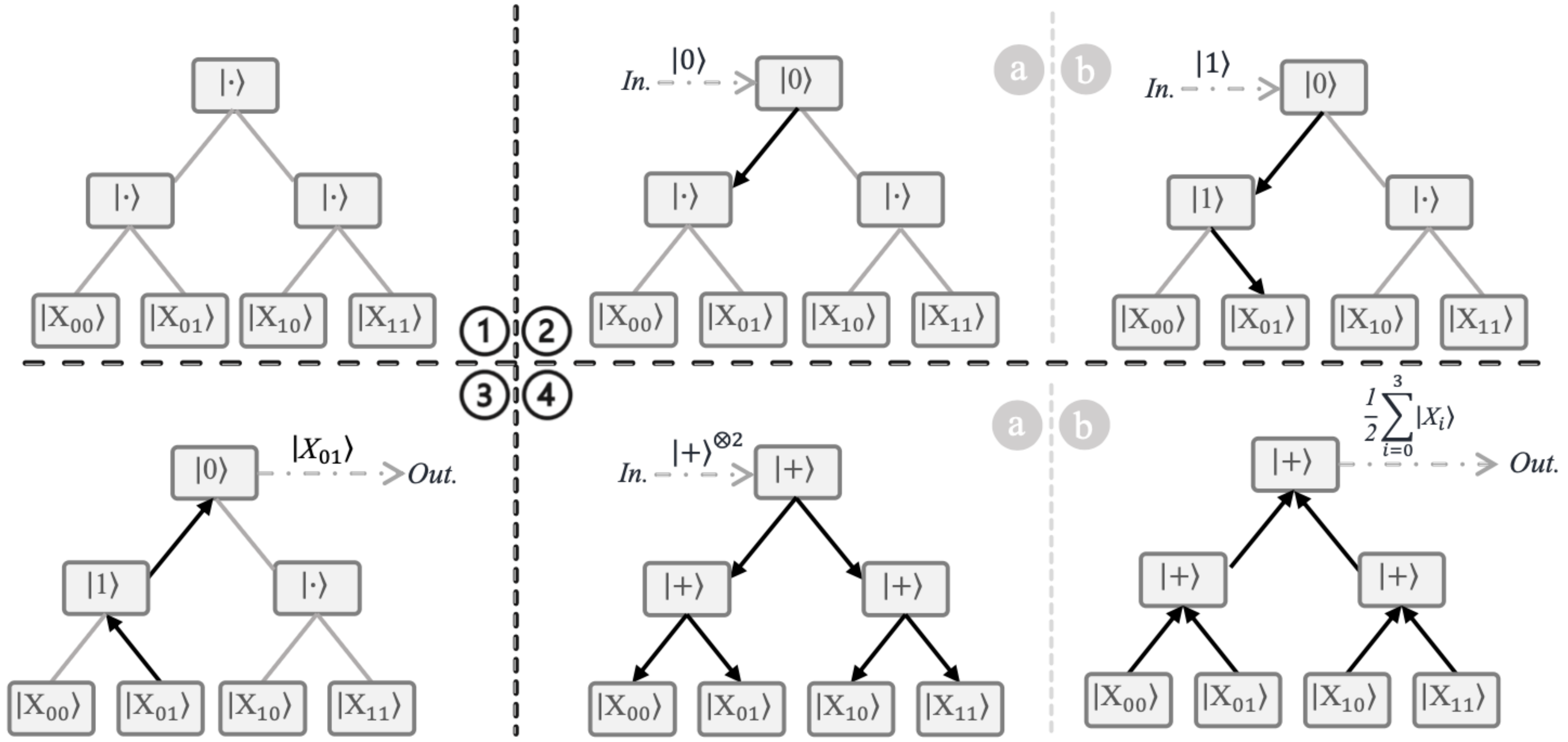


Fig. 3. Working of a bucket-brigade QRAM with 2 address lines and 4 memory cells. ① Initial state of the QRAM, all quantum switches are initialized to $|\cdot\rangle$ state which is a waiting state where the quantum switch waits for incoming qubit states of the memory address to be accessed. ② Input register activates switches for output register to access data in address $|01\rangle$ ($|X_{01}\rangle$). The address qubits are sent in a sequential top-down fashion starting from the Most Significant Bit (MSB) all the way to the Least Significant Bit (LSB). In the example shown, first MSB qubit $|0\rangle$ is sent that changes the state of the root quantum switch ⓐ, followed by the LSB qubit $|1\rangle$ that routes the switch to the direction of the memory cell $|X_{01}\rangle$ ⓑ. ③ Output register reads data $|X_{01}\rangle$ via route of activated quantum switches. ④ Superposition of all addresses turning on all quantum switches ⓐ to read superposition of all the data ⓑ. Note that $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle) + |1\rangle)$., *In.*: Input register, *Out.*: Output register.

Quantum Circuit Implementation of Bucket-Brigade QRAM

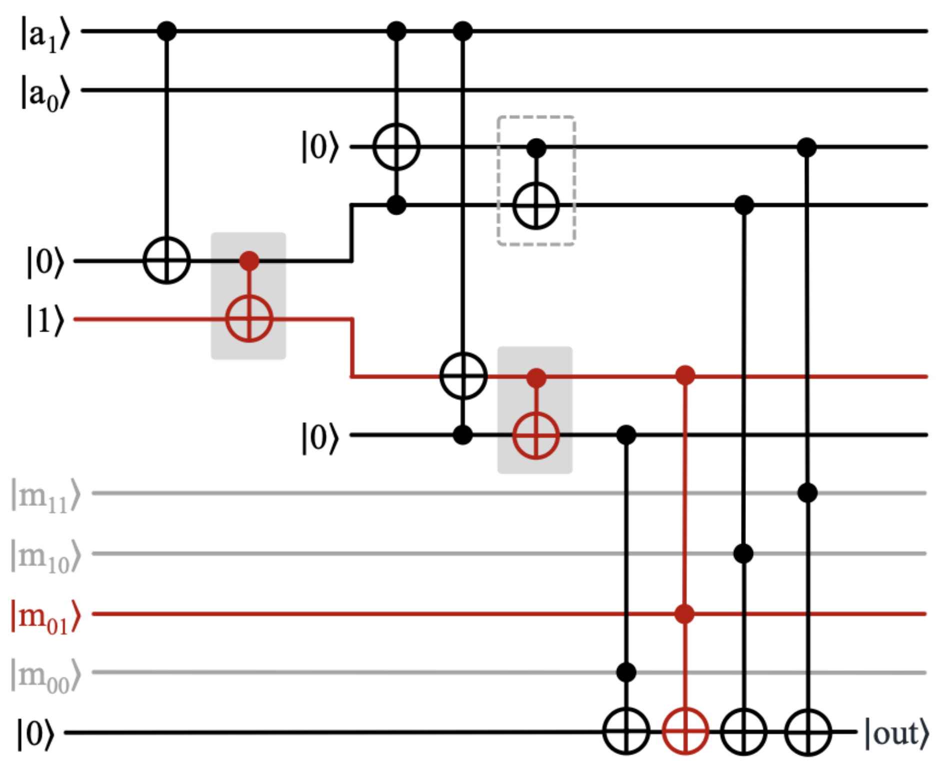


Fig. 4. Circuit-based implementation of a bucket-brigade QRAM. Data in memory cell m_{01} with address $|01\rangle$ is being accessed via a series of CNOT and Toffoli gates performing intermediate computation on ancilla qubits. Note that the CNOT gates highlighted in red are the ones getting activated and the red path represents the active route of the QRAM.

More on QRAMs later in the course

- Other Implementations of QRAM
  - Fanout QRAM
  - Flip-Flop QRAM
  - Entangling Quantum Generative Adversarial Network (EQGAN) QRAM
  - Qudits-based memory
  - Approximate PQC-based QRAM
- Applications of QRAMs
  - Database search: auxiliary circuit to Grover’s algorithm
  - Element distinctness
  - Collision detection
  - Quantum forking
  - Storage of classical data