Santa Clara University

COEN 210

Chapter 4, Part 1 – Sequential Logic and Memory

Yan Cui

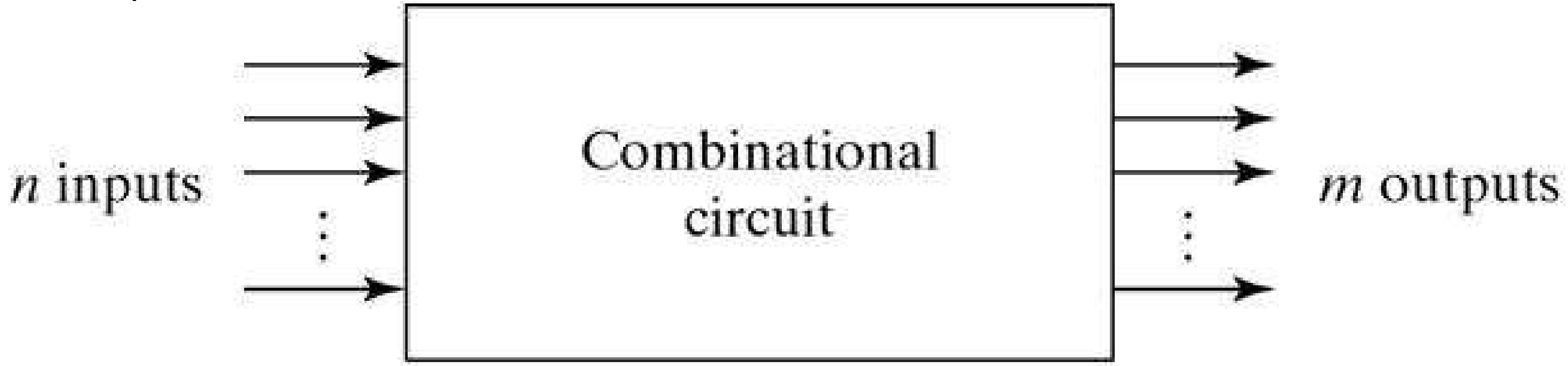
ycui4@scu.edu

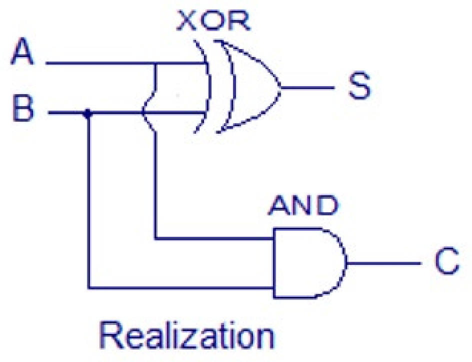
Sequential Logic and Memories

- Sequential logic can remember states for later logic operations
- Different memories use different sequential logics to achieve balance between speed and capacity
- They serve as components, roughly between circuits and micro-architecture

Recall Combinational Logic

- Consists of an acyclic network of logic gates
  - Continuously responds to changes in inputs
  - Outputs become (after a short delay) boolean functions of the inputs.
    - $F_J(i_1, i_2, \dots i_n), j = 1, 2, \dots m$





Single-Bit Half Adder

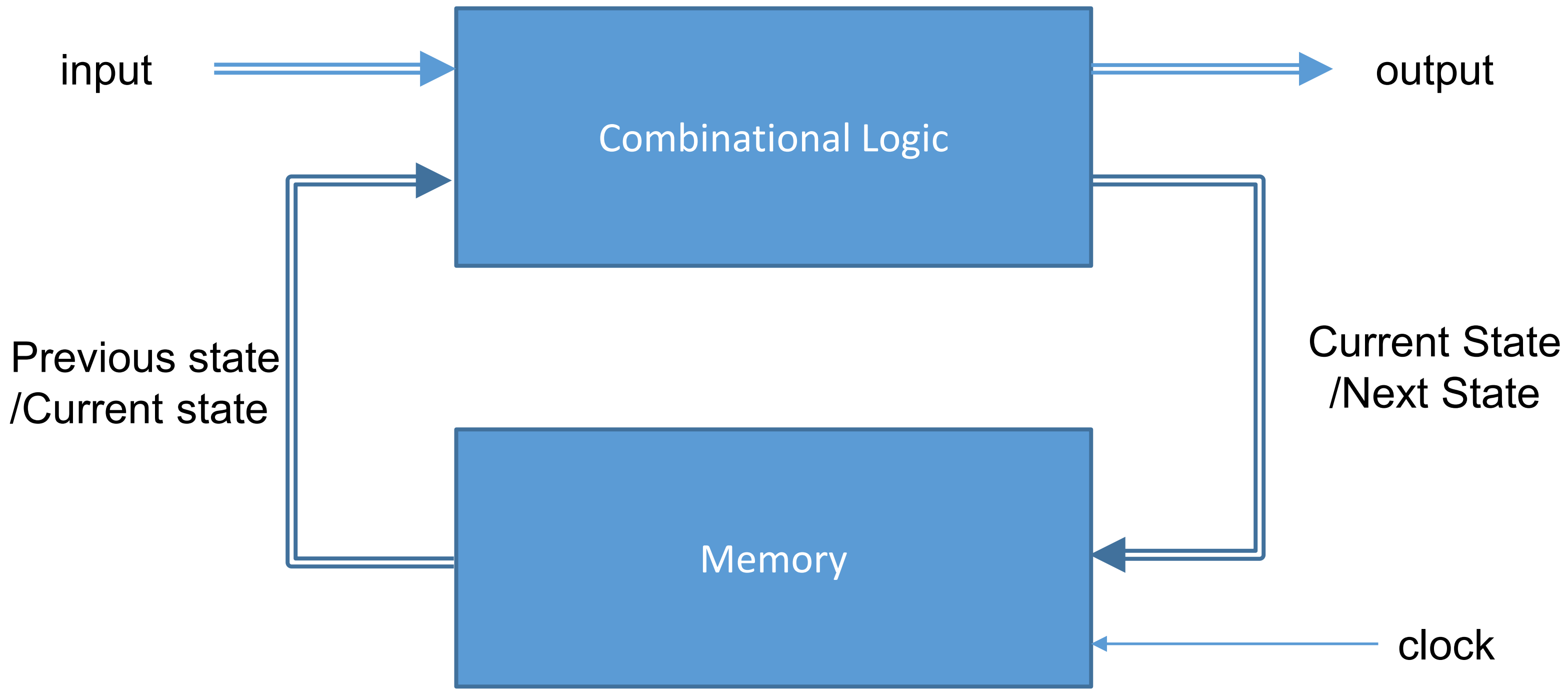
The Boolean functions:

Sum: $S(A,B) = A \oplus B$

Carry: $C(A,B) = A \cdot B$

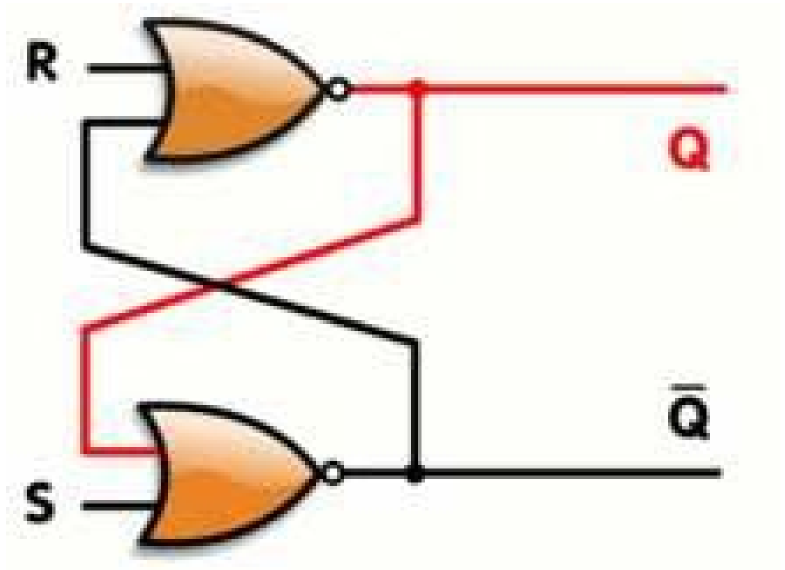
Sequential Logic

- A group of logic elements and memory elements
- output depends on the inputs and the current contents of the memory



S-R Latch– Store 1 Bit

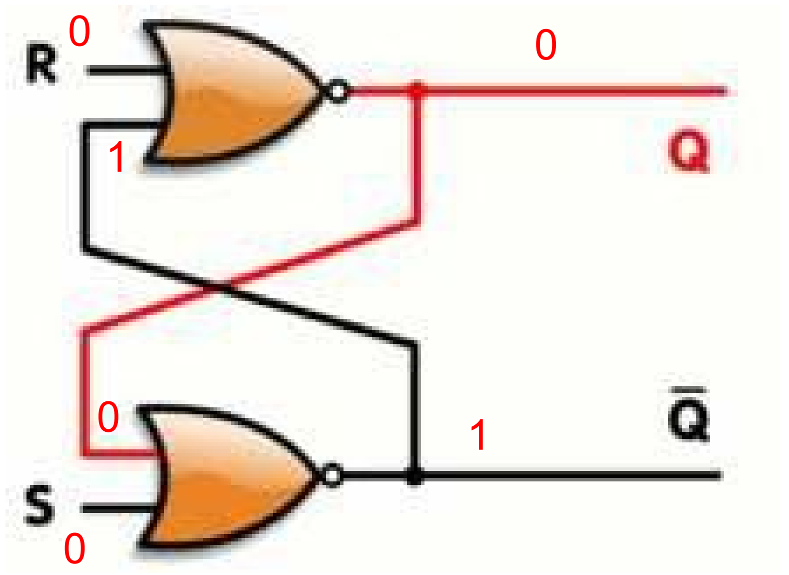
- Two input : R, S
- Two output : Q and $\overline{Q}$
  - Depending on R, S as well as themselves
  - Circular dependency



S-R Latch– Store 1 Bit

- Case 1: R = 0, S = 0
  - Assume Q=0,
    - Q = 0 ➔ $\overline{Q}$ = 1 ➔ Q_{next} = 0

Q_{next} is the same as Q, a stable state

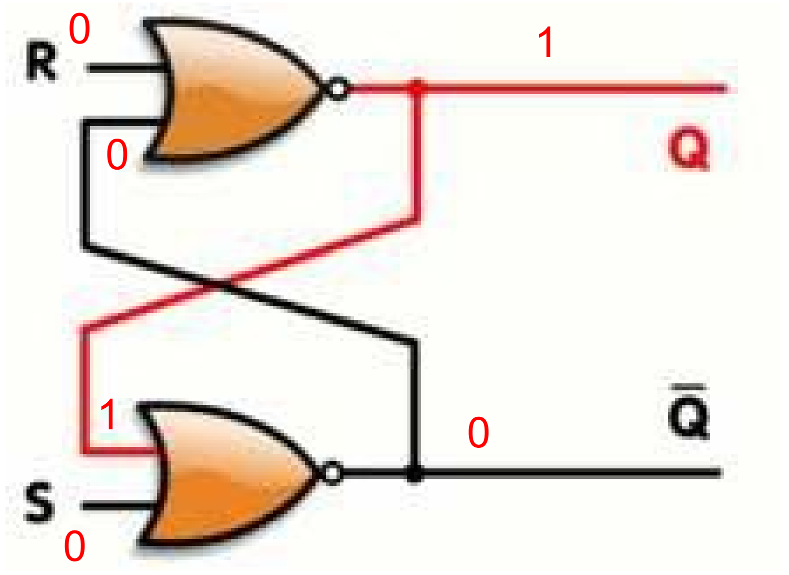


S-R Latch– Store 1 Bit

- Case 1: R = 0, S = 0
  - Assume Q=0
    - Q = 0 ➔ $\overline{Q}$ = 1 ➔ Q_{next} = 0
  - Assume Q=1
    - Q = 1 ➔ $\overline{Q}$ = 0 ➔ Q_{next} = 1

Q_{next} is the same as Q, a stable state

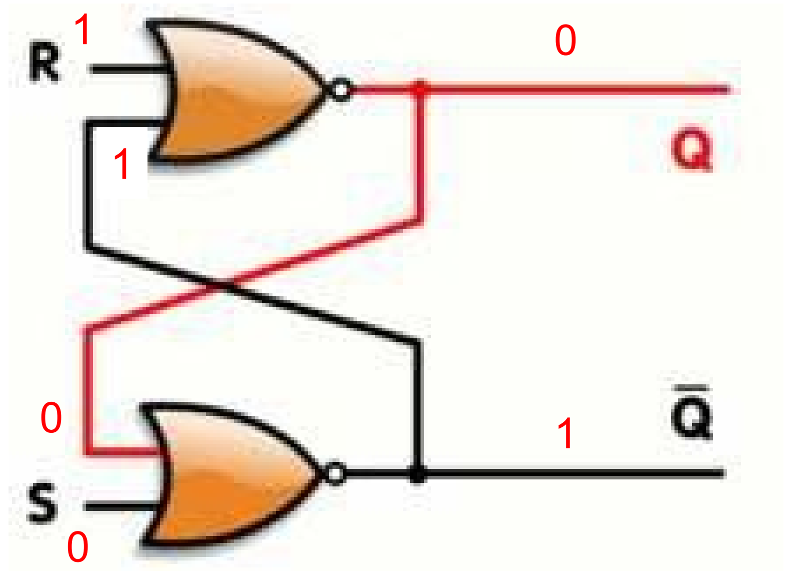
When R=0 and S=0, Q_{next} takes same value as before.



S-R Latch– Store 1 Bit

- Case 2: R = 1, S = 0
  - Q = ? ➔ Q_{next} = 0 and $\overline{Q}$ = 1

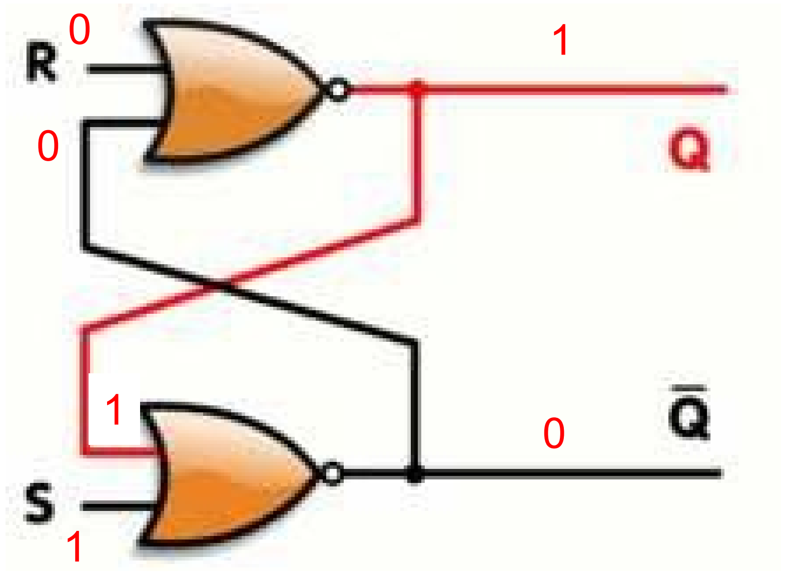
When R=1 and S=0, Q_{next}=0 and $\overline{Q}$=1.



S-R Latch– Store 1 Bit

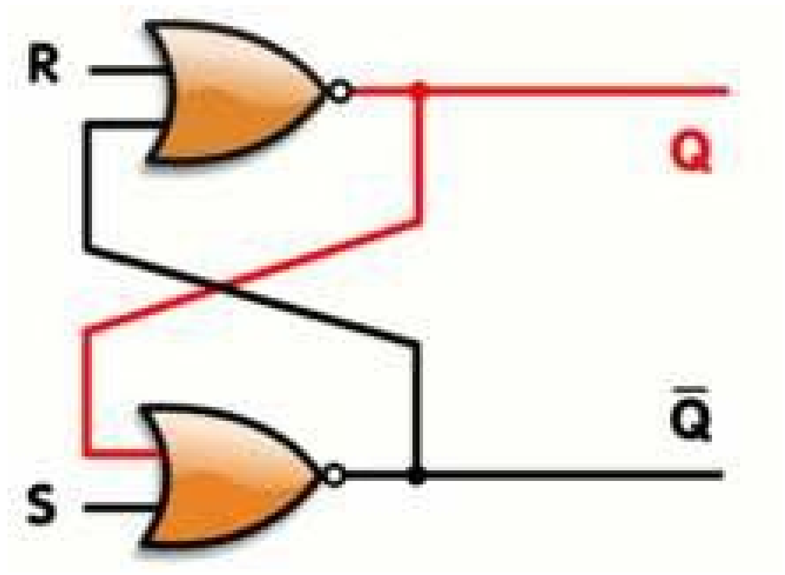
- Case 3: R = 0, S = 1
  - Q = ? ➔ Q_{next} = 1 and $\overline{Q}$ = 0

When R=0 and S=1, Q_{next}=1 and $\overline{Q}$=0.



S-R Latch– Store 1 Bit

- It works to store 1 bit
  - Pulse (temporarily raise) the R (reset) input to record a 0.
  - Pulse the S(set) input to record a 1.
  - Otherwise Q signal will remain the same
- But not very convenient to use in practice with 2 inputs

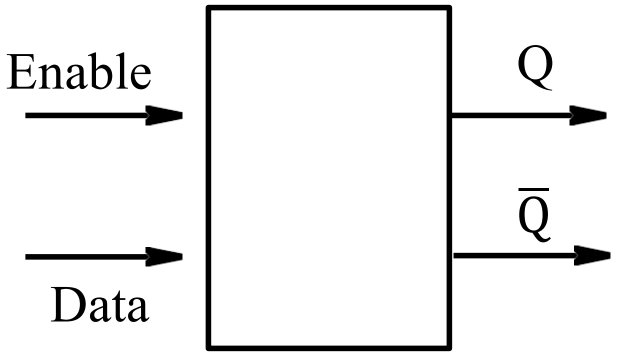


Characteristic table

S	R	Q_{next}	Action
0	0	Q	hold state
0	1	0	reset
1	0	1	set
1	1	?	not allowed

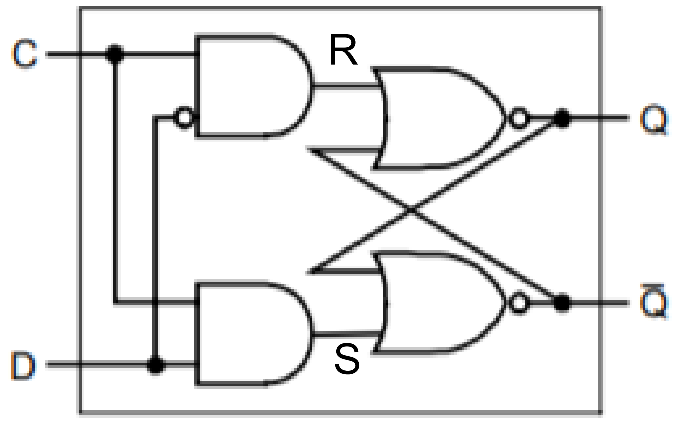
Ideal Memory to Store 1 Bit …

- Ideal device for storing a bit
  - Place the bit to store on Data line.
  - Raise Enable to high (1) temporarily
    - E.g., a clock signal
  - The value of data on line is stored (or latched) in the device.
  - Lower Enable to low (0).
  - Reading Q returns the stored bit until next store.



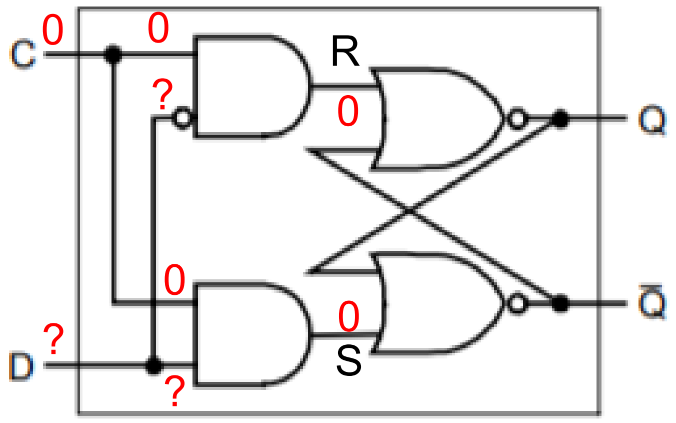
D Latch (or D Flip-Flop)

- Add 2 AND gates in front of flip flop
  - "C" stands for "clock" (or "Enabled" in previous page)
    - Also denoted as CP ("Clock Pulse")
  - "D" stands for "Data"



D Latch (or D Flip-Flop)

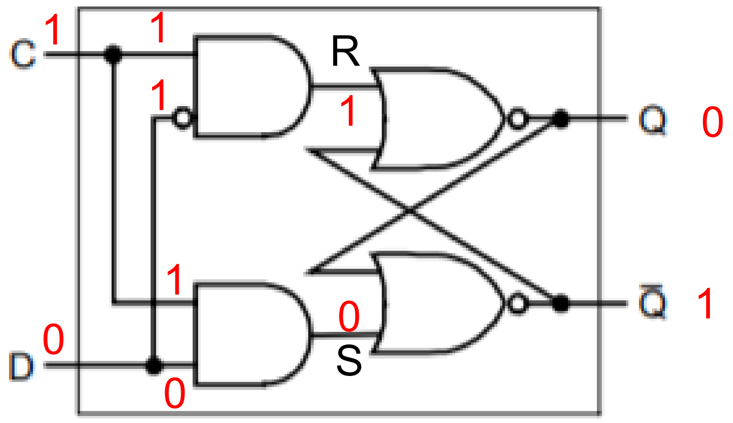
- Case 1 : C =0; D = ?
  - R=0, S=0 ➔ Q will hold its value in this state



C	D	R	S	Q_{next}	notes
0	?	0	0	Q	Hold state

D Latch (or D Flip-Flop)

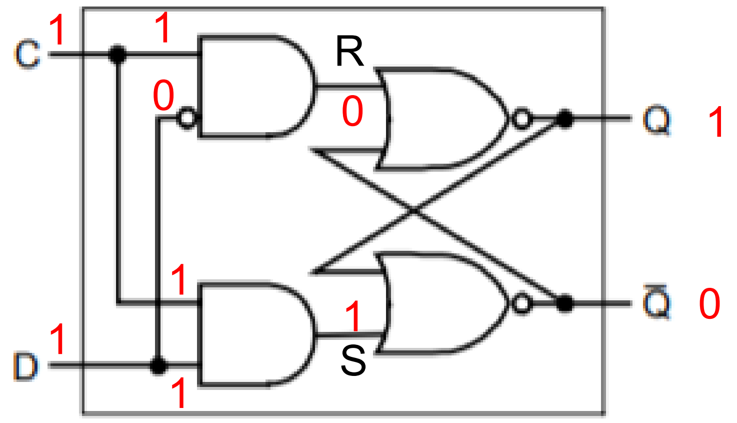
- Case 2 : C =1; D = 0
  - R=1 and S=0 ➔ Q will be set to 0



C	D	R	S	Q_{next}	notes
0	?	0	0	Q	Hold state
1	0	1	0	0	Reset (0)

D Latch (or D Flip-Flop)

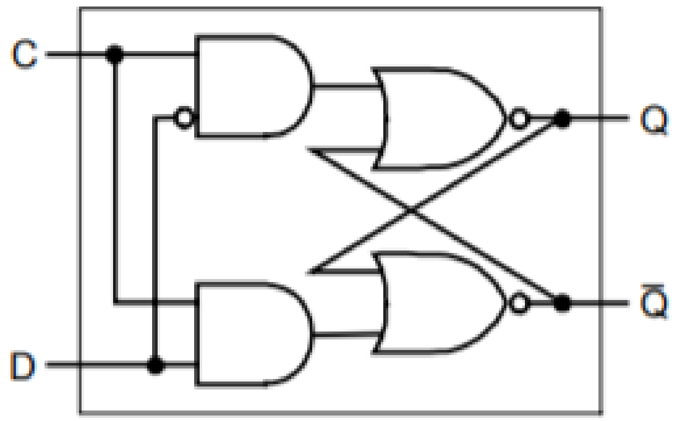
- Case 3 : C =1; D = 1
  - R=0 and S=1 ➔ Q will be set to 1

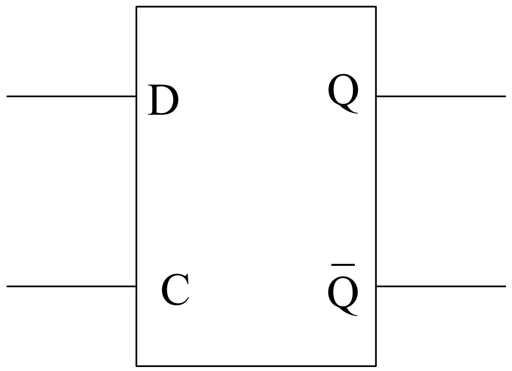


C	D	R	S	Q_{next}	notes
0	?	0	0	Q	Hold state
1	0	1	0	0	Reset (0)
1	1	0	1	1	Set (1)

D Latch (or D Flip-Flop)

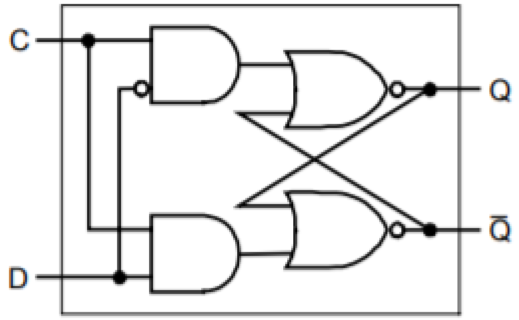
- Simplified representation for D latch

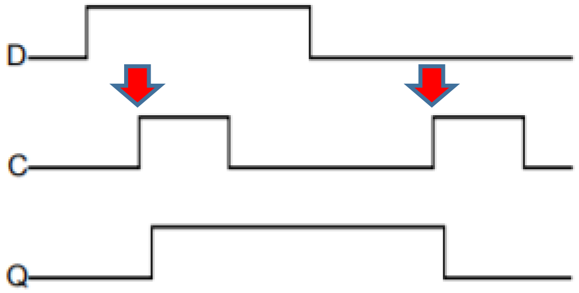




Clock Edge Triggering

- Q only changes its value on the rising edge of clock pulse

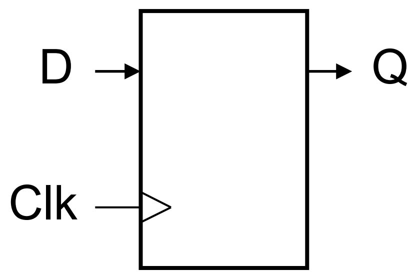


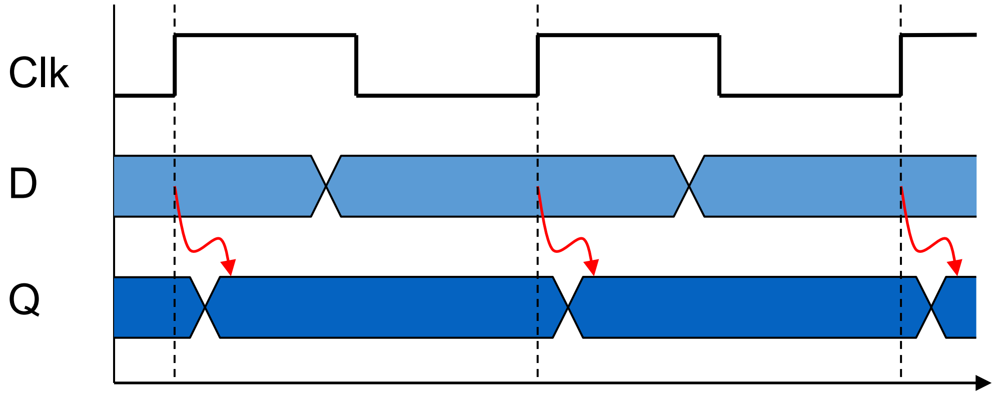


Registers

Register

- Store data in a circuit
- A small, fast, frequently accessed memory residing inside processor
  - Uses a clock signal to determine when to update the stored value
  - Edge-triggered: update when Clk changes from 0 to 1



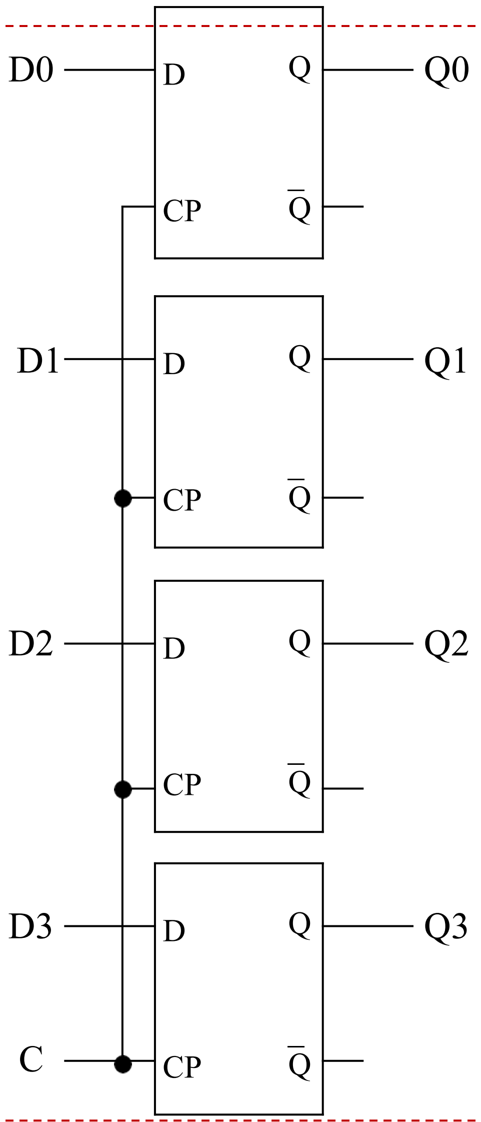


- A register is made up of several flip flops or D latches , each of which provides storage and access for an individual bit.

Example: Implement a 4-bit Register

- Use 4 D latches
  - Share the same C input (or CP input) so that the values change in synchronization
    - C input can be a global clock signal!
  - D0–D3 are the data input
  - Q0–Q3 are the output

- How to implement a 32bit RISC-V register?



Register File

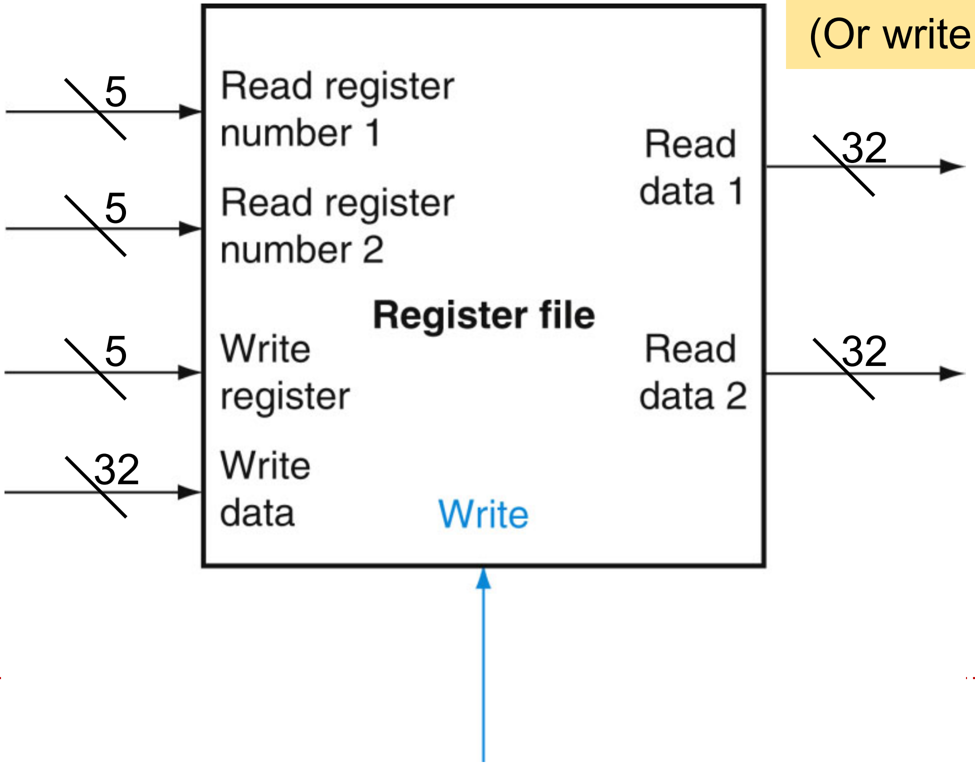
- Implemented by
  - A decoder for each read or write port
  - An array of processor registers built from D flip-flops
    - Read or written by a supplying a register number
- made up of several registers and a control logic
  - For example,
  - The RISC-V processor has a RISC-V register file includes 32 32-bit general purpose registers.
    - use by integer and logic instructions.
  - The RISC-V processor has a separate register file, which contains another 32 32-bit registers.
    - for floating point instructions

Example: A RISC-V Register File

- Input
  - two 5-bit read register numbers
  - one 5-bit write register
  - 32-bit data to write
  - One write signal (control signal)
- Output
  - Two 32-bit read data

Why do we have 2 registers for reading?

Can read 2 register at a time;
(Or write to one register at a time)

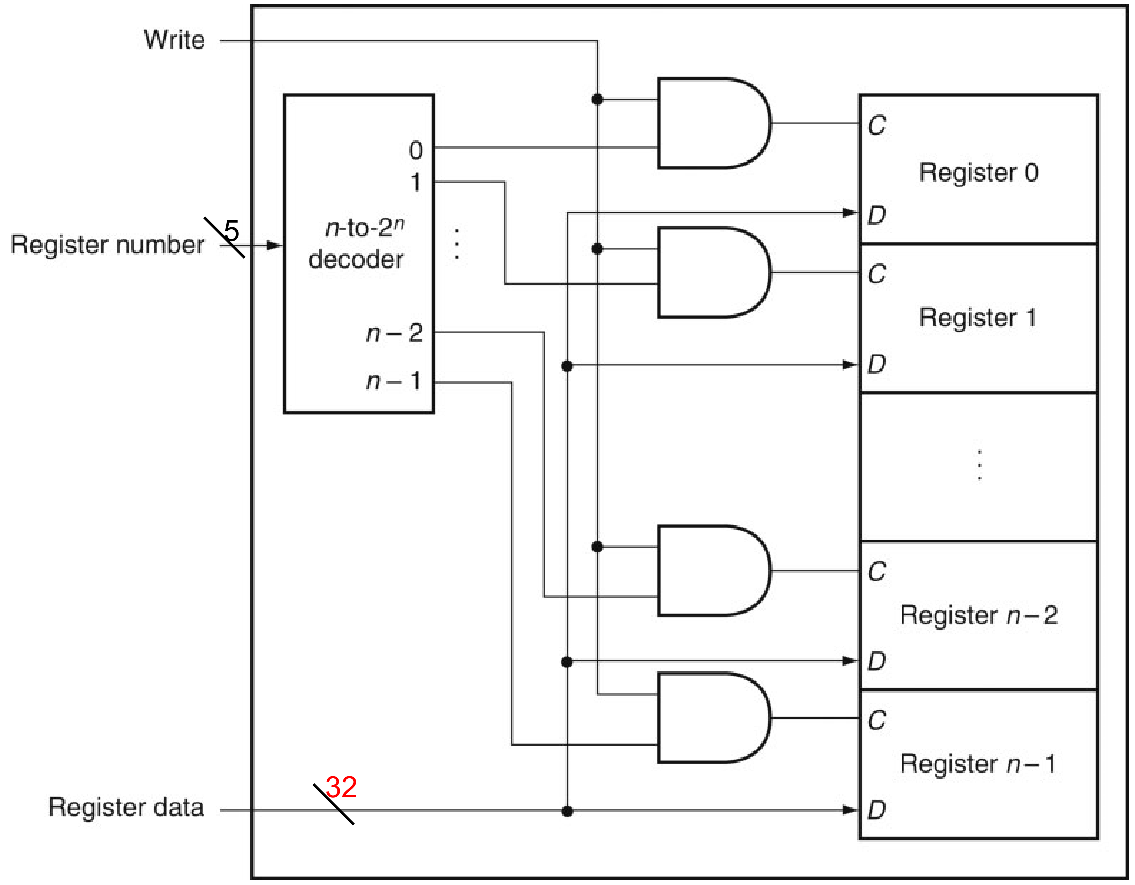


Register File functioning

- A register file functions as follows:
  - any value provided on 5-line Read register number 1 port results in the content of the corresponding register being provided on the 32-line Read data 1 port
  - any value provided on 5-line Read register number 2 port results in the content of the corresponding register being provided on the 32-line Read data 2 port
  - on the falling edge of write line, values that appear on 32-bit Write data port are written into the register with the number specified on the 5-line Write register port.

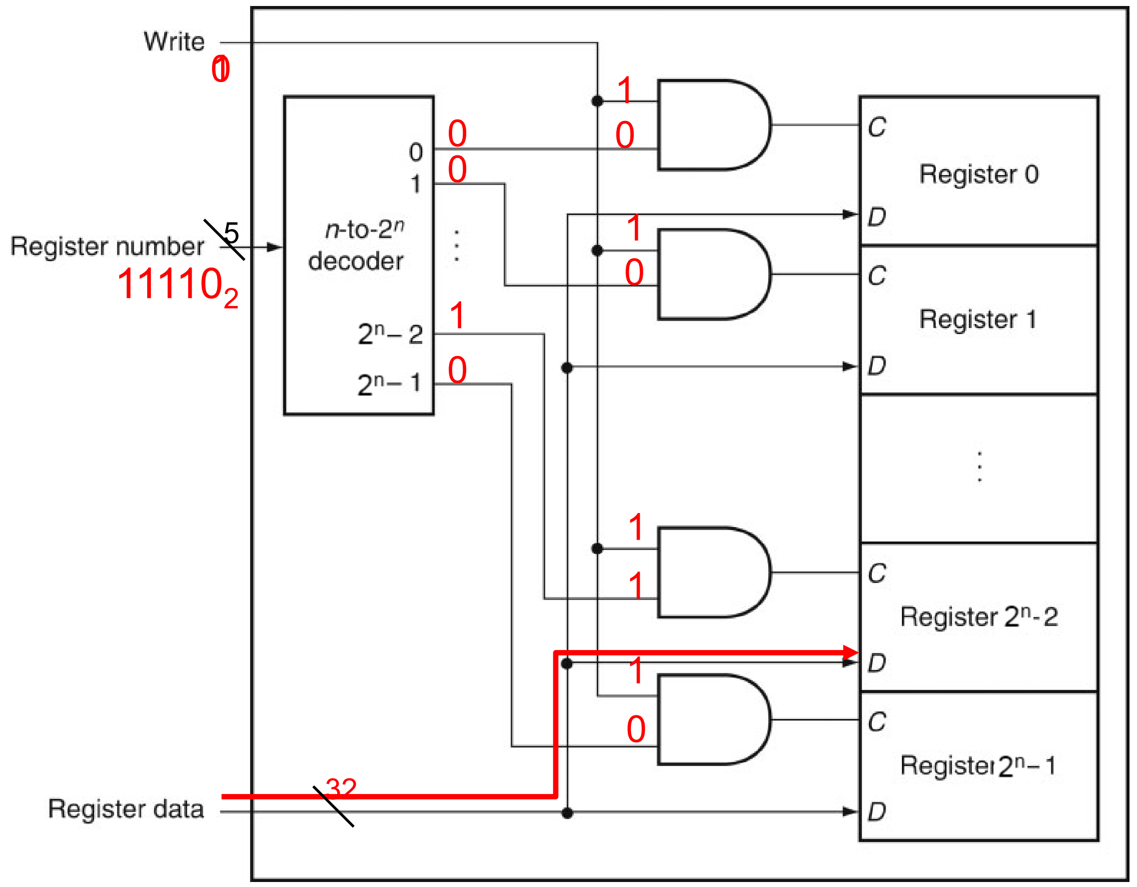
Write to A Register

- Set up "register number" and "register data"
- Send "Write" pulse to perform write



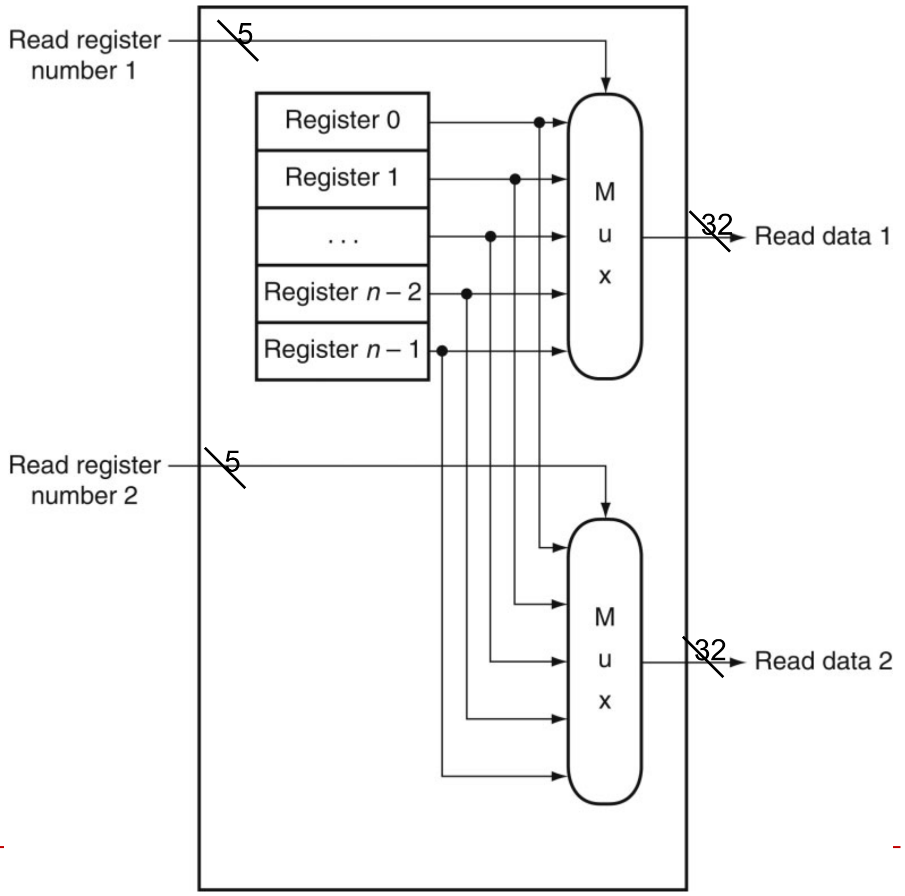
Example: Write data on a register for 32bit RISC-V

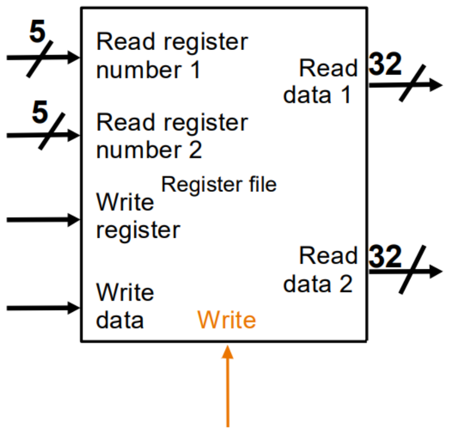
- n=5
- suppose register number is 30 (11110_2)



Read from Two Registers

- This is combinational logic
  - Shortly after register numbers are setup, proper register data would appear on output





n = 32 for RISC-V

How many bits in each input?

How many inputs to MUX for 32RISC-V?

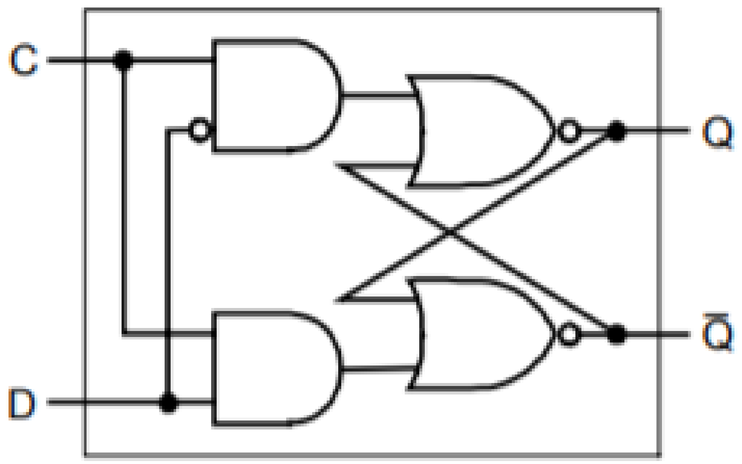
- What happens if the same register is read and written during a clock cycle?
  - Because the write of the register file occurs on the clock edge, the register will be valid during the time it is read. The value returned will be the value written in an earlier clock cycle.
- If you want a read to return the updated value, additional logic will be needed in the register file or outside of it.

RAM
(Random Access Memory)

How Many Transistors in D Latch?

- Each AND/OR gate needs 2 transistors
- Each NOT gate needs 1 transistor
- We will have 11 transistors in total!

D latches are expensive for storing many bits!



Introduction to memory design

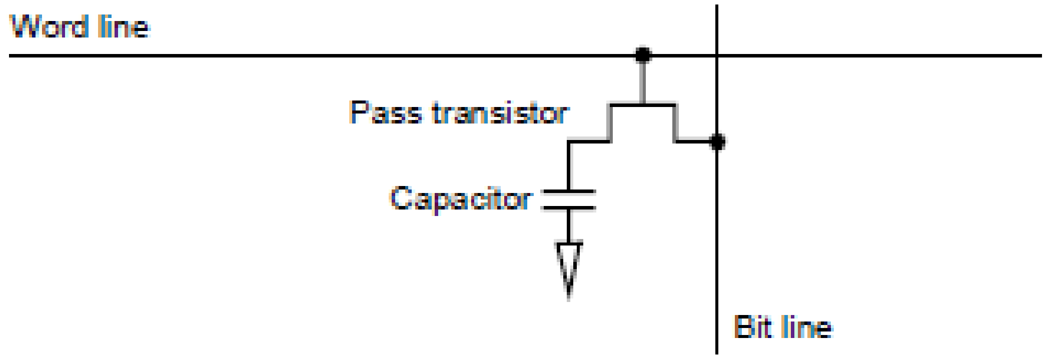
- Main memory is built in one of two technologies:
  - SRAM - Static Random Access Memory
  - DRAM - Dynamic Random Access Memory
- Both memories are volatile.
- A memory is normally built using a number of memory chips.
- Memory chips have specific configurations given as a product
- of two numbers, e.g.
  - 128M*1 – 128M addressable locations with 1 bit in each location, i.e. width of read/write operations is 1 bit
  - 16M*8 – 16M addressable locations with 8 bits in each location, i.e. width of read/write operations is 8 bits

Notice that two chips above accommodate identical number of bits (128M bits).

DRAM – Dynamic Random Access Memory

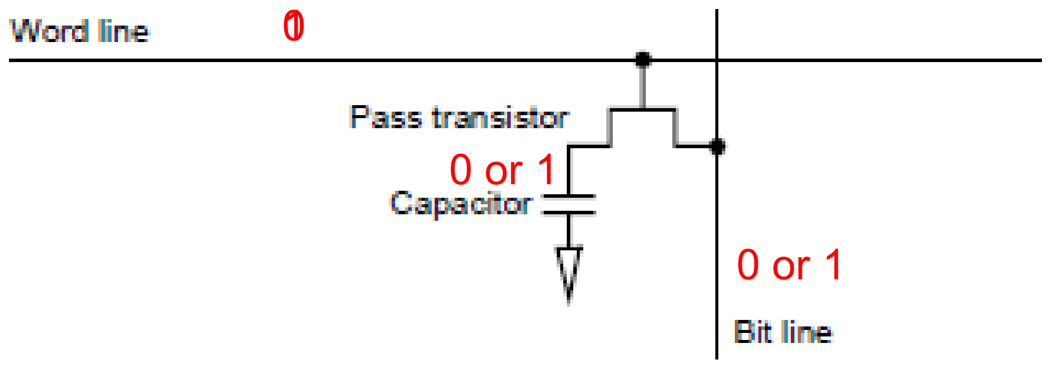
- DRAM cell for storing 1 bit
  - A pass transistor controls on/off between capacitor and bit line
  - Value (0 or 1) is stored in capacitor (high/low voltage)
  - "bit line" is used to read/write

- Only need 1 transistor per bit!



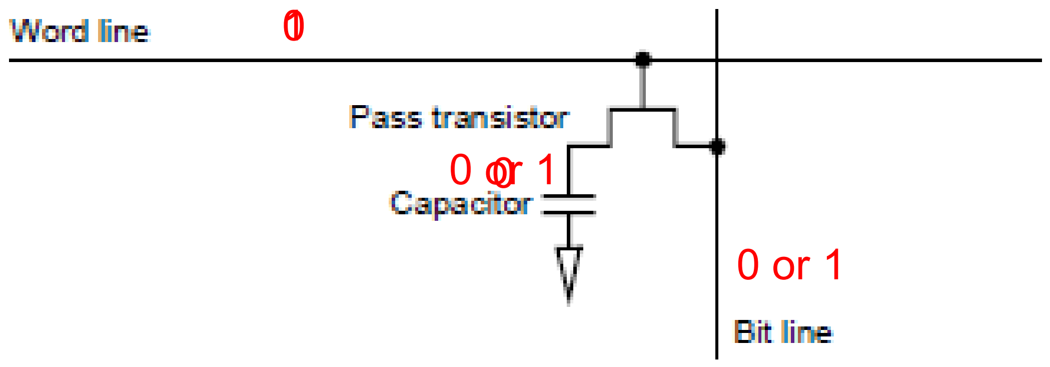
DRAM Cell - Write

- Apply data to write (0 or 1) on bit line
- Apply high voltage (1) to word line
- Restore low voltage (0) to word line
- Capacitor will hold the value from bit line



DRAM Cell - Read

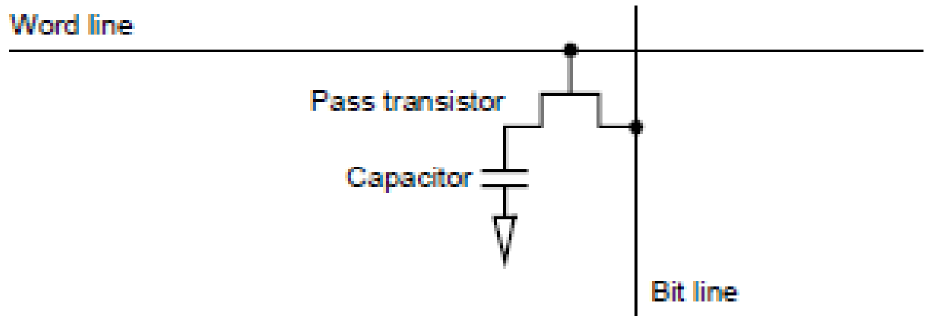
- Apply high voltage (1) to word line
- Sense voltage on "bit line" (0 or 1)
- Restore low voltage (0) to word line
- But capacitor will drop to low voltage (0) – why?
  - Solution is to write back the read value
  - This is called destructive read of DRAM cell



DRAM Cell – Refresh

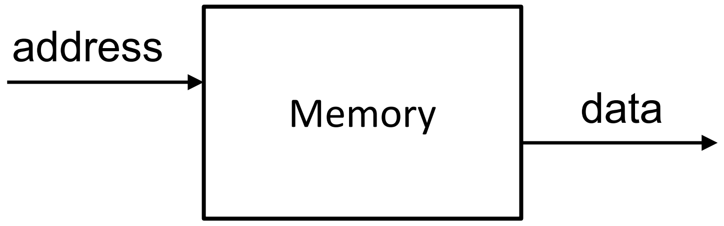
- In fact, even without any READ operations, capacitor will lose its value.

- Periodic refreshing (read and write back the read value) is needed
  - Typically <64ms per refresh (i.e., > 16 times/sec)
  - Hence the reason it is called "dynamic" RAM ("dynamically refreshing")



DRAM Summary

- Conceptually, a large array where each row (1 byte) is uniquely addressable.



- Characteristics
  - Dynamic RAM (DRAM) is large, inexpensive, but relatively slow.
  - 1 transistor and 1 capacitor per bit.
  - Reads are destructive. Requires periodic refresh.
  - Access time takes hundreds of CPU cycles.
  - Used for system main memory

Static Random Access Memory (SRAM)

- Static (SRAM) is faster but more expensive.
  - Similar to D latch
  - 6 transistors per bit.
  - Typically used for cache memory

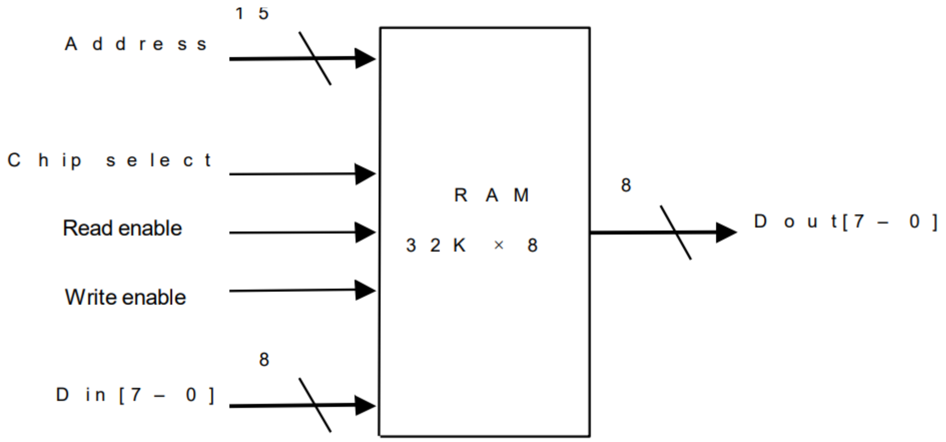
DRAM & SRAM Characteristics

- Current DRAM chip capacity has reached 16GB on a single memory module with an access time in the range 40-60 nsec and a cycle time of about 80 nsec.
- For SRAM technology access time and cycle time are identical.
- In comparison, SRAM cycle time is about 8 to 16 times faster than DRAM, e.g. currently 0.5-5 nsec
- Access time & cycle time are two measures of memory latency:
  - access time – the time between a read is requested and when the desired content arrives,
  - cycle time – the minimum time between two memory requests.

- Also, SRAM is more expensive, e.g. 1GB in 2004 $4,000 – $10,000 for SRAM and $100 – $200 for DRAM.
- In addition, SRAM chips have higher power consumption and power dissipation than DRAM chips.
- Thus, SRAM designs are concerned with speed, while in DRAM designs the emphasis is on cost per bit and capacity

Memory Chip Functioning

- Example: 32K×8 chip
  - read and write operations are 8 bits wide
  - 32K addressable locations



- Functioning of memory chip:
  - CS (Chip select) has to be set for either reading or writing
  - R (Read enable)=0 & W (Write enable)=0 -> chip is not being accessed
  - R=0 and W=1 ->write values at Din lines into the chip address at Address lines
  - R=1 and W=0 -> read into Dout lines values from the chip address at Address lines
  - R=1 and W=1 ->not allowed

Reference Readings

- Patterson, "Computer Organization and Design RISC-V Edition"
  - Appendix A.8, A.9
- Tanenbaum, "Structured Computer Organization"
  - Sec 3.3