CSE 331 / EEE 332 / ETE 332/EEE 453 Microprocessor and Interfacing

DR. SHAIKH ASIF MAHMOOD
PROFESSOR, EEE, BUET

Topics

1. Fundamentals of microprocessor and computer design, processor data path, architecture, microarchitecture, complexity, metrics, and benchmark; complexity, metrics, and benchmark, Instruction Set Architecture, introduction to CISC and RISC,

2. Assembly language programming of Arm Cortex M / RISC-V based embedded microprocessors (jump, call-return, stack, push and pop, shift, rotate, logic instructions, port operations, serial communication and interfacing), system clock, exceptions and interrupt handling.

3. Instruction-Level Parallelism, pipelining, pipelining hazards and data dependency, branch prediction, exceptions and limits, super-pipelined vs superscalar processing; Memory hierarchy and management, Direct Memory Access.

4. Introduction to embedded systems design, software concurrency and Realtime Operating Systems, Arm Cortex M / RISC-V microcontroller architecture, registers and I/O, memory map and instruction sets, endianness and image

Books

1. Sarah Harris, David Harris – "Digital Design and Computer Architecture", ARM Edition, Morgan Kaufmann (2015)
2. David A. Patterson and John L. Hennessy, "Computer Organization and Design– The Hardware / Software Interface ARM edition" Morgan Kaufmann
3. Yifeng Zhu "Embedded Systems with ARM Cortex-M Microcontrollers with Assembly Language and C"

Marks Distribution

Category	Marks	Percentage
Attendance (Class+Lab)	5+5	10%
Quiz (2 out of 3)	10 each	20%
Mid-term	20	20%
Lab	20	20%
Final	30	30%

Course Objective

Upon the completion of the course, the student should

1) Understand the architecture, instruction set, memory and input/output interface of a ARM Microprocessor and different principles of Embedded Systems
2) Construct simple microprocessor systems using state-of-the-art tools like Arm Assembly compiler and VerilogHDL understanding the limitations
3) Illustrate emerging technologies and trends in Microprocessor design to recognize the need to always learn the state-of-the art

Instructor-in-charge

Dr. Shaikh Asif Mahmood (SAMd)

Adjunct Professor, ECE, NSU

Office: SAC 1145

Office hours (tentative): RA 2:30 PM-3:30 PM

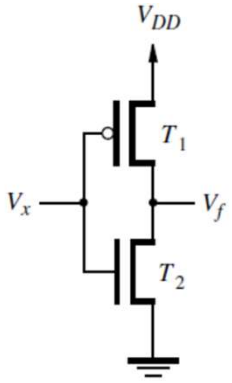
Email for appointment:
shaikh.mahmood@northsouth.edu

Number System (Review)

TABLE 1-4. EQUIVALENCES

Hexadecimal	Binary	Decimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

CMOS Logic Gates (Not Gate)

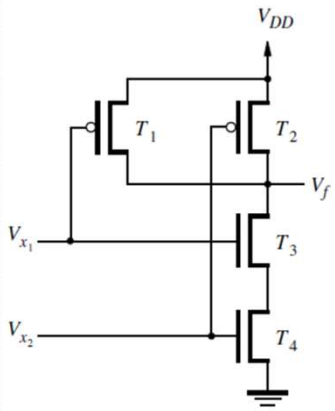


(a) Circuit

x	T_1	T_2	f
0	on	off	1
1	off	on	0

(b) Truth table and transistor states

CMOS Logic Gates (Nand Gate)

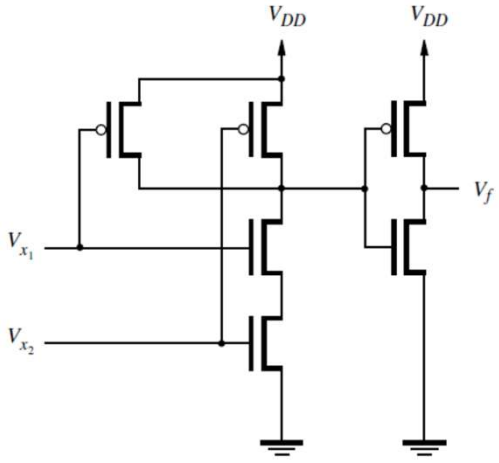


(a) Circuit

x_1	x_2	T_1	T_2	T_3	T_4	f
0	0	on	on	off	off	1
0	1	on	off	off	on	1
1	0	off	on	on	off	1
1	1	off	off	on	on	0

(b) Truth table and transistor states

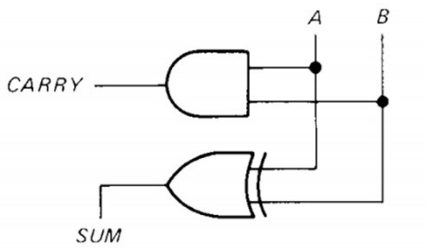
CMOS Logic Gates (And Gate)



Half-Adder

TABLE 6-1. HALF-ADDER

A	*B*	*CARRY*	*SUM*
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



$$SUM = A \oplus B$$
$$CARRY = AB$$

Full-Adder

$$SUM = A \oplus B \oplus C \quad (6\text{-}8)$$
$$CARRY = AB + AC + BC \quad (6\text{-}9)$$

TABLE 6-2. FULL ADDER

A	*B*	*C*	*CARRY*	*SUM*
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

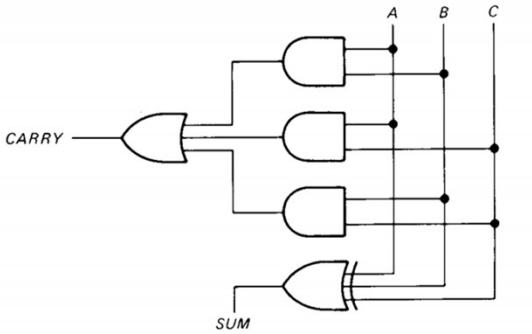
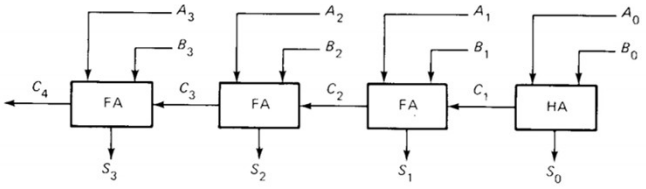


Fig. 6-2 Full adder.

Binary Adder



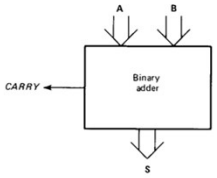


Fig. 6-5 Symbol for binary adder.

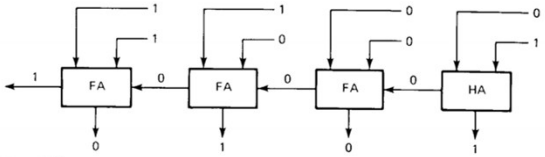


Fig. 6-4 Adding 12 and 9 to get 21.

Signed Numbers

- For computers, desirable to represent everything as *bits*.
- Three types of signed binary number representations: **signed magnitude, 1's complement, 2's complement.**
- In each case: left-most bit indicates sign: positive (0) or negative (1).

Consider ***signed magnitude***:

$00001100_2 = 12_{10}$

Sign bit — Magnitude

$10001100_2 = -12_{10}$

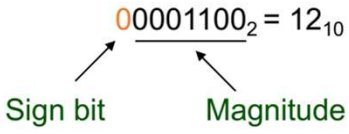
Sign bit — Magnitude

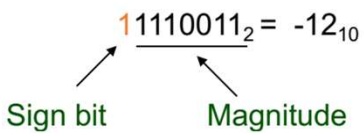
One's Complement

The one's complement of a binary number involves inverting all bits.

- 1's comp of 00110011 is 11001100
- 1's comp of 10101010 is 01010101

To find negative of 1's complement number take the 1's complement.



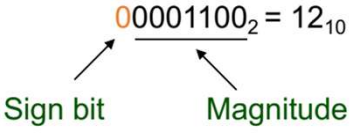


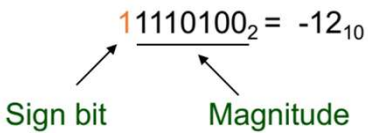
Two's Complement

The two's complement of a binary number involves inverting all bits and adding 1.

- 2's comp of 00110011 is 11001101
- 2's comp of 10101010 is 01010110

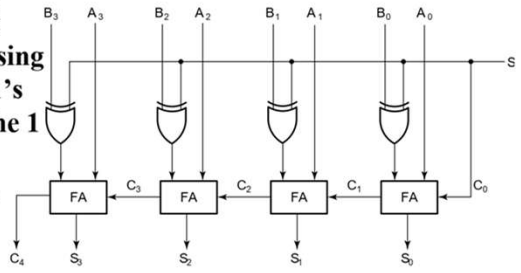
To find negative of 2's complement number take the 2's complement.





2's Complement Adder/Subtractor

- Subtraction can be done by addition of the 2's Complement.
  1. Complement each bit (1's Complement.)
  2. Add 1 to the result.
- The circuit shown computes A + B and A – B:
- For S = 1, subtract, the 2's complement of B is formed by using XORs to form the 1's comp and adding the 1 applied to C_0.
- For S = 0, add, B is passed through unchanged



Flip-Flops, Registers, and Counters

(REVIEW)

Combinational vs Sequential Logic

- In **combinational circuits,** the value of each output depends solely on the current values of signals applied to the inputs.
- In **sequential circuits**, the values of the outputs depend not only on the present values of the inputs but also on the past behavior of the circuit.
- Such circuits include **storage elements** that store the values of logic signals. The contents of the storage elements are said to represent the **state** of the circuit.
- Over time, the network changes through **a sequence of states** as a result of changes in the inputs.

A Memory Element

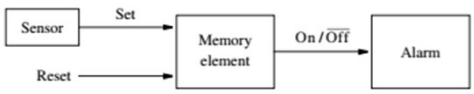


Figure 5.1 Control of an alarm system.

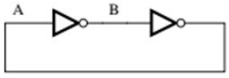
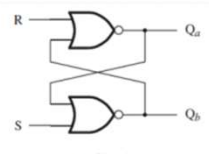


Figure 5.2 A simple memory element.

If we assume that $A = 0$, then $B = 1$. The circuit will maintain these values indefinitely because of the feedback loop.

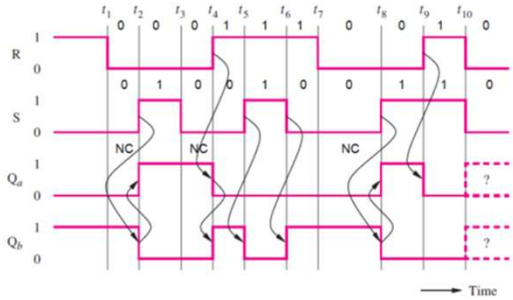
Basic Latch



(a) Circuit

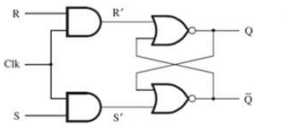
S	R	Q_a	Q_b	
0	0	0/1	1/0	(No change)
0	1	0	1	
1	0	1	0	
1	1	0	0	

(b) Truth table



- 2 NOR gates, cross-coupled
- Qa, Qb indicate state of the circuit
- R(Reset) and S(Set) control the circuit state
- S=0,R=0 (states are maintained)
  No change in Qa, Qb
- S=1,R=0 (Set)
  Qa=1, Qb=0
- S=0,R=1 (Reset)
  Qa=0, Qb=1
- S=1,R=1 (Invalid Control Inputs)
  Qa=0, Qb=0
- For the timing diagram, no delay is assumed
- Avoid transition from S=1,R=1 to S=0,R=0 as it causes Qa,Qb to oscillate between 0,0 and 1,1

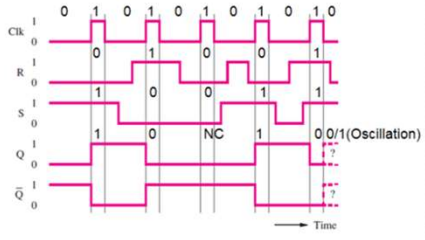
Gated SR Latch



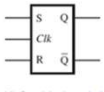
(a) Circuit

Clk	S	R	$Q(t+1)$
0	x	x	$Q(t)$ (No change)
1	0	0	$Q(t)$ (No change)
1	0	1	0
1	1	0	1
1	1	1	x

(b) Truth table



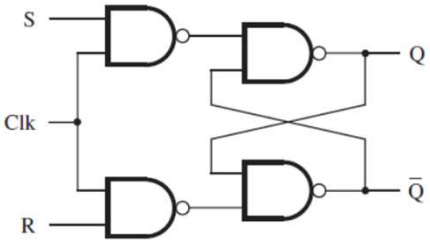
(c) Timing diagram



(d) Graphical symbol

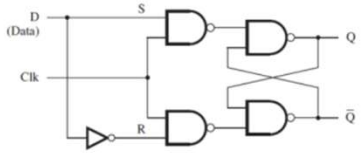
- Use 2 AND gates with 2 NOR gates, add Clk with R and S
- Clk works like an enable signal (it changes at well defined intervals)
- If Clk =0, S and R don't matter, No change in Q
- If Clk =1, it works like a basic latch
- R(Reset) and S(Set) control the circuit state
- Transition from S=1,R=1 to S=0,R=0 causes Qa,Qb to oscillate between 0,0 and 1,1
- So avoid the S=1,R=1 mode

Gated SR Latch with NAND Gates



- Use 4 NAND gates instead of 2 AND and 2 NOR gates
- Its truth table is similar to the previous gated SR latch
- It requires fewer transistors than the previous latch

Gated D Latch

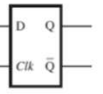


(a) Circuit

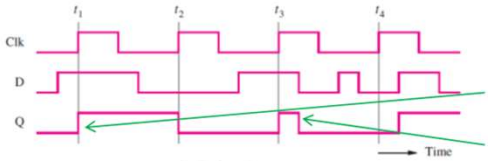
Clk	D	Q(t + 1)
0	x	Q(t)
1	0	0
1	1	1

(No change)
(Reset)
(Set)

(b) Truth table



(c) Graphical symbol



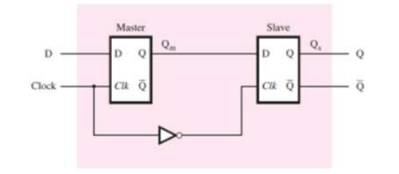
(d) Timing diagram

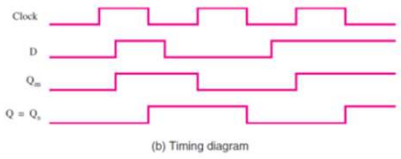
- Single data input D (S and R combined into D)
- If D =1, S=1 R=0, Set
- If D =0, S=0 R=1, Reset
- State of circuit (Q,~Q) is changed only when Clk=1
- Impossible to have S=1 R=1 (invalid combination is not achieved)
- If Clk =1, Q follows the state of D
- If Clk =0, Q does not change its state
- The latch is level sensitive (as opposed to being edge-triggered)
- Q may change at edge of Clk pulse (depending on D)
- Q may change in middle of Clk pulse (depending on D)

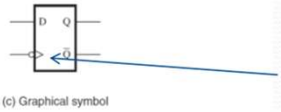
Flip-Flops (FF)

- In the **level-sensitive latches**, the state of the latch keeps changing according to the values of input signals during the period when the clock signal is active.
- So Q may change **more than once** during a Clk cycle for a latch.
- **Flip-flops** are storage elements that can change their states **no more than once** during one clock cycle.
- Type of Flip-flops:
- **Edge-triggered flip-flop** is affected only by the input values present when the active edge of the clock occurs. It can be **positive edge-triggered** or **negative edge-triggered.**
- **Master-slave flip-flop** is built with two gated latches (one master and one slave).

Master-Slave D Flip-Flop (Negative Edge-triggered)

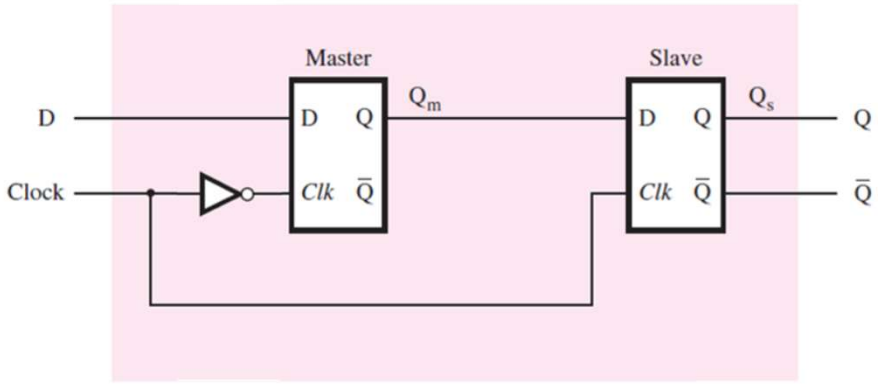




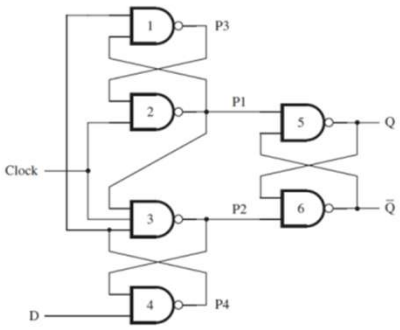


- Two gated D latches (One master, One Slave)
- It is typically built with 8 NAND gates (4 for each latch)
- Master changes state when Clk=1
- Slave changes state when Clk=0
- Qm follows any change in D while Clk=1
- Qm does not change while Clk=0
- Qm is output of Master and input of Slave
- So, Qs changes at most once during a clock cycle (When Clk goes from 1 to 0)
- As Q=Qs, from outside, it seems that Q follows D only at negative edge of Clk
- So this D flip-flop is **negative edge-triggered**

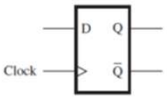
Master-Slave D Flip-Flop (Positive Edge-triggered)



Edge-Triggered Flip-Flop (Positive Edge-triggered)



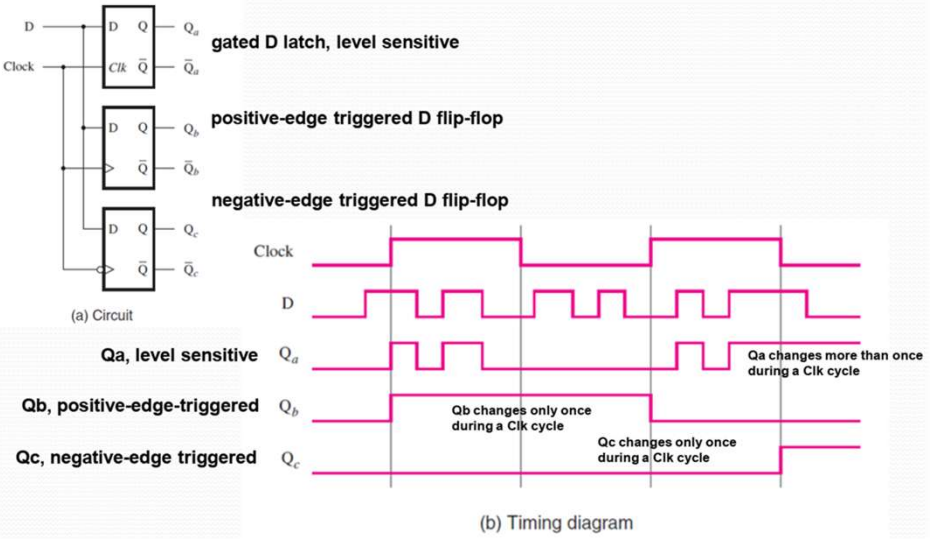
(a) Circuit



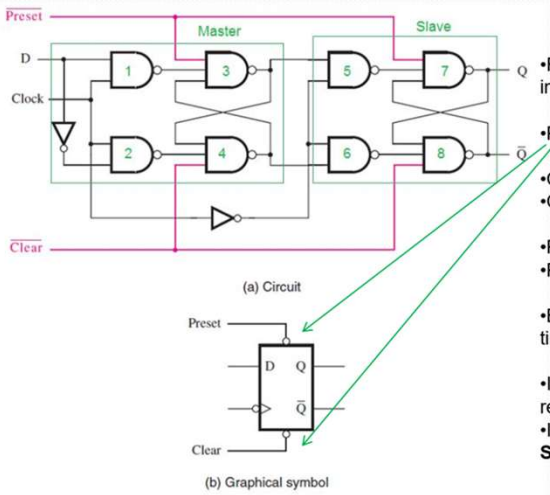
(b) Graphical symbol

- It is typically built with 6 NAND gates (instead of 8)
- When Clk=0, P1=1 P2=1,
  Q ~Q maintained in present state
- As P2=1, P4=~D, as P1=1, P3=D
  When Clk=1,
  as P3=D P1=~D, as P4=~D P2=D
  It sets Q=D ~Q=~D
- Output changes only once at positive edge of Clk
- So this D flip-flop is **positive edge-triggered**

Latch vs Flip-Flop

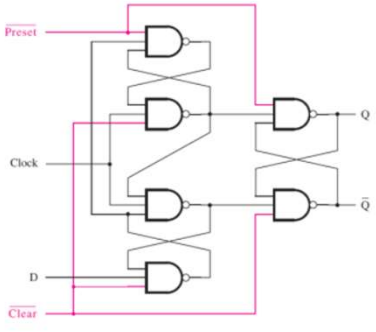


Master-slave Flip-flop with Clear and Preset

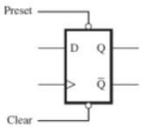


- For Prs (Preset) and Clr (Clear), one extra input is added to cross-coupled NAND gates
- Prs and Clr are active-low
- Clr=0, Q forced to 0
- Clr=1, no effect
- Prs=0, Q forced to 1
- Prs=1, no effect
- Both Prs and Clr should not be 0 at the same time
- If Prs=0 or Clr=0, effect is immediate regardless of Clk, so **Asynchronous Clear**
- If Prs and Clr depend on Clk, it would be **Synchronous Clear**

Edge-triggered Flip-flop with Clear and Preset



(a) Circuit



(b) Graphical symbol

Truth Table
(Positive-edge-triggered D flip-flop)

Prs	Clr	Clk	D	Q
0	0	x	x	* (invalid condition)
0	1	x	x	1 (Set)
1	0	x	x	0 (Reset)
1	1	0	x	No Change
1	1	1	x	No Change
1	1	↓	x	No Change
1	1	↑	0	0
1	1	↑	1	1

Synchronous vs Asynchronous Reset

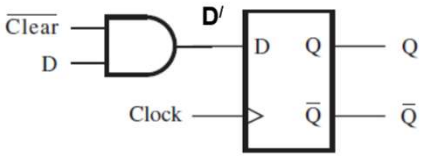
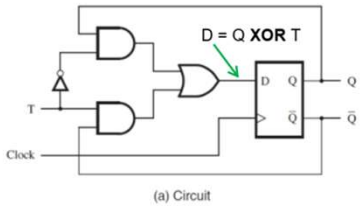


Figure 7.15 Synchronous reset for a D flip-flop.

- The previous two flip-flops used **Asynchronous Clear**
- Add an **AND** gate to combine the Clr signal with the D input to achieve **Synchronous Clear**
- **If Clr=1, D/ = D, usual flip-flop operation**
- **If Clr=0, D/ = 0, flip-flop output (Q) will be zero on next Clk edge**
- As Clr depend on Clk, it is an example of **Synchronous Clear**

T Flip-Flop

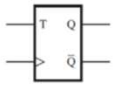


(a) Circuit

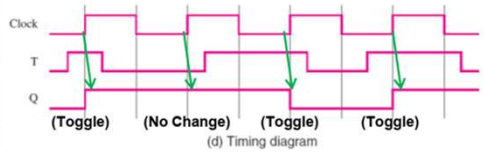
T	Q(t + 1)
0	Q(t)
1	Q̄(t)

(No Change)
(Toggle)

(b) Truth table



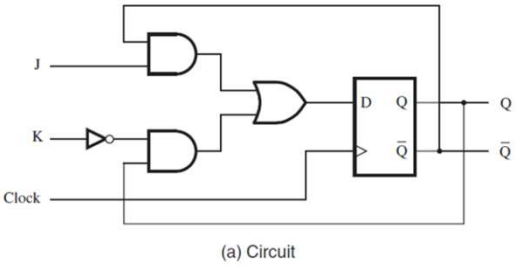
(c) Graphical symbol



(d) Timing diagram

- T stands for **toggle**
- An **XOR** gate with an **extra input T** are added at the input of a D flip-flop so that
  D = Q **XOR** T
- If T=0, D=Q, Q(t+1)=Q(t), Q retains its state
- If T=1, D=~Q, Q(t+1)=~Q(t), Q toggles its state
- Useful for counters

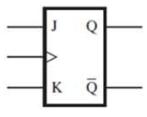
JK Flip-Flop



(a) Circuit

J	K	Q(t + 1)	
0	0	Q(t)	(No Change)
0	1	0	(Reset)
1	0	1	(Set)
1	1	Q̄(t)	(Toggle)

(b) Truth table



(c) Graphical symbol

- It has two inputs (**J and K**)
  D = J.~Q + ~K.Q
- If **J and K are tied together**, it works as a T flip-flop
- It combines behavior of SR flip-flop and T flip-flop
- For JK=00,01,10, it behaves likes a SR flip-flop
- For JK=11, it behaves likes a T flip-flop

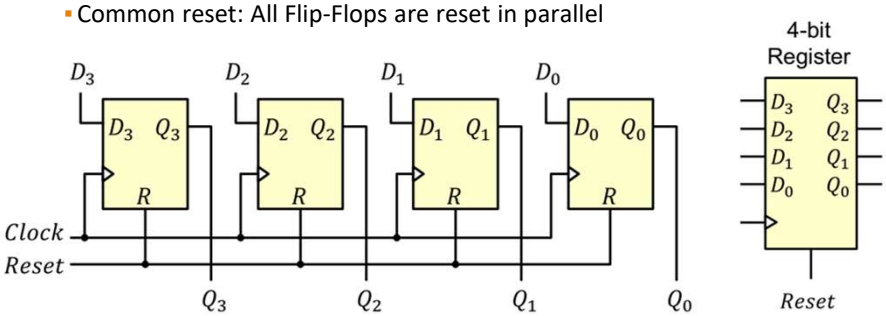
Clk	J	K	Q
0	x	x	No Change
1	x	x	No Change
↓	x	x	No Change
↑	0	0	No Change
↑	0	1	0 (Reset)
↑	1	0	1 (Set)
↑	1	1	Toggle

Registers

- A **flip-flop** stores one bit of information.
- When a set of n flip-flops is used to store n bits of information, such as an n-bit number, we refer to these flip-flops as a **register**.
- A **common clock** is used for each flip-flop in a register

4-bit Register

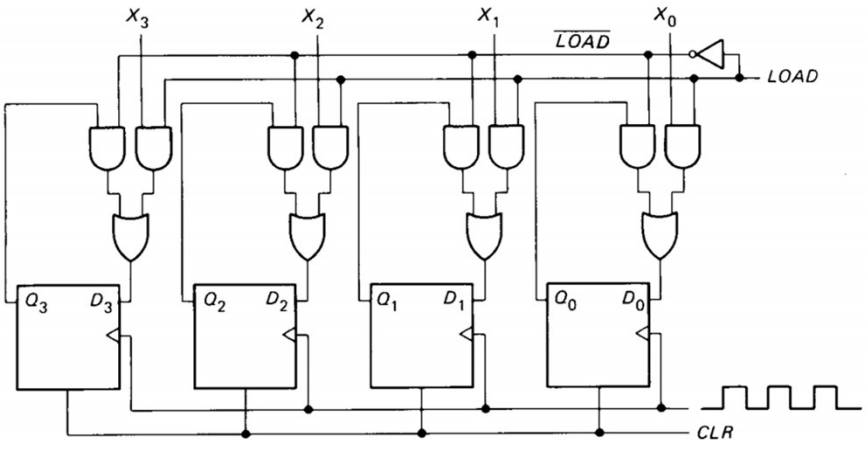
- A register is a circuit capable of storing data
- An n-bit register consists of n Flip-Flops and stores n bits
- Common clock: data is loaded in parallel at the same clock edge
- Common reset: All Flip-Flops are reset in parallel



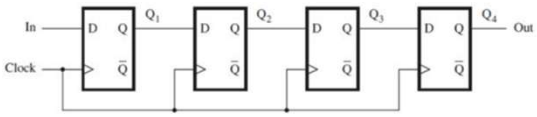
Register Load (or Enable)

- Question: How to control the loading of data into a register?
- Solution: Introduce a register Load (or Enable) 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 Load?

Register with Parallel Load



Shift Register



(a) Circuit

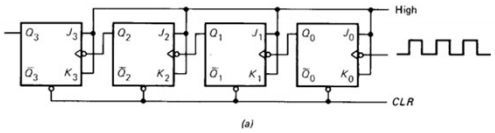
	In	Q_1	Q_2	Q_3	Q_4 = Out
t_0	1	0	0	0	0
t_1	0	1	0	0	0
t_2	1	0	1	0	0
t_3	1	1	0	1	0
t_4	1	1	1	0	1
t_5	0	1	1	1	0
t_6	0	0	1	1	1
t_7	0	0	0	1	1

(b) A sample sequence

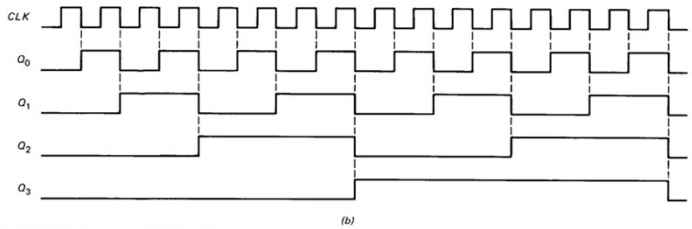
- 4 bit shift register (4 flip-flops, all have common Clk)
- It processes 4 bit data (Q1 Q2 Q3 Q4)
- Shift 1 bit position to the right at each Clk edge
- Data bit loaded serially at In (In Fig. (b) 8 bits loaded with 8 Clk cycles)
- All flip-flop initial state 0 (at t0, Q1=Q2=Q3=Q4=0)
- Need edge triggered storage elements (level sensitive elements are not suitable)

Counter

- A *counter is a register capable of counting the number of* clock pulses that have arrived at its clock input.



(a)

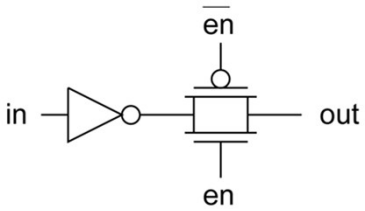


(b)

Fig. 8-9 (*a*) Ripple counter; (*b*) timing diagram.

Count	Q_3 Q_2 Q_1 Q_0
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1
12	1 1 0 0
13	1 1 0 1
14	1 1 1 0
15	1 1 1 1

Tri-state Output



en	in	out
0	0	Z (high impedance)
0	1	Z(high impedance)
1	0	1

The main application of three-state switches is to convert the two-state output of a register to a three-state output. For instance, Fig. 8-24 shows a three-state buffer register, so called because of the three-state switches on the output lines. When *ENABLE* is low, the *Y* outputs float. But when *ENABLE* is high, the *Y* outputs equal the *Q* outputs; therefore,

$$Y = Q$$

Tri-state Buffer Register

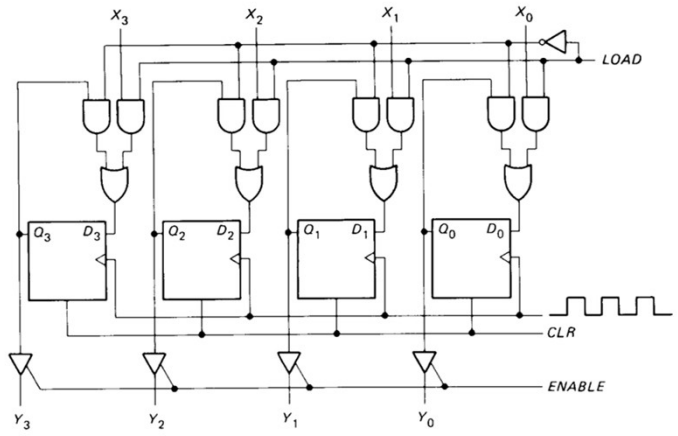
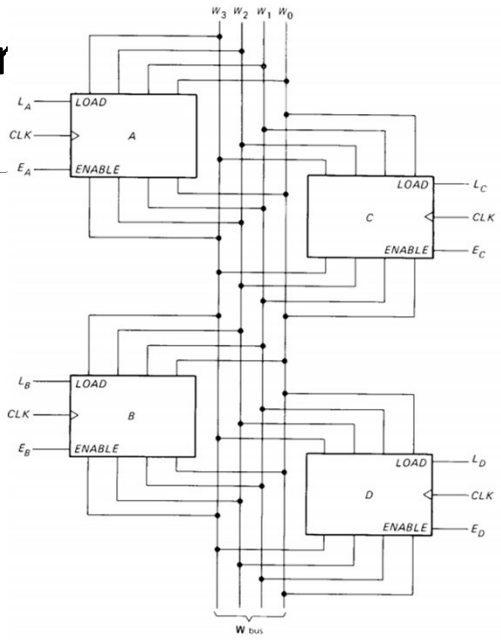


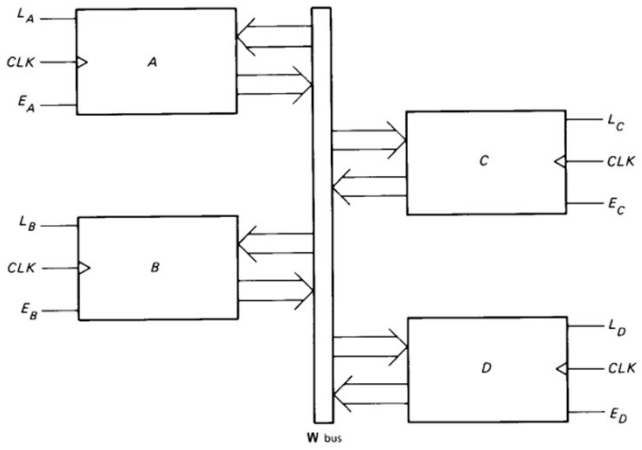
Fig. 8-24 Three-state buffer register.

Bus - Organizatior

- A bus is a group of wires that transmit a binary word. In the Figure, vertical wires W_3, W_2, W_1, and W_0 are a bus.
- These wires are a common transmission path between the three state registers.



Simplified Drawing



Memory

- Read-only memory (ROM) is a type of non-volatile memory that permanently stores data and programs needed for a device to function.

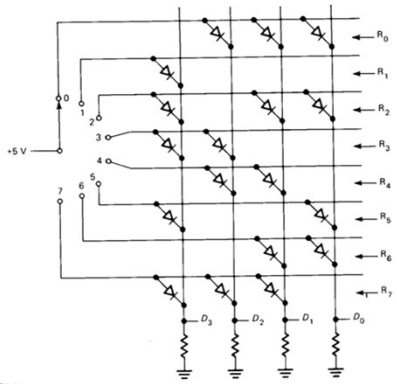
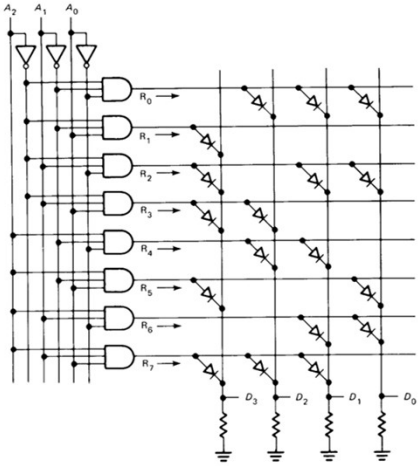


Fig. 9-1 Simple diode ROM.

Register	Address	Word
R_0	0	0111
R_1	1	1000
R_2	2	1011
R_3	3	1100
R_4	4	0110
R_5	5	1001
R_6	6	0011
R_7	7	1110

ROM with On-Chip Decoding



1-of-8 decoder produces a high output to one of the registers. For instance, if

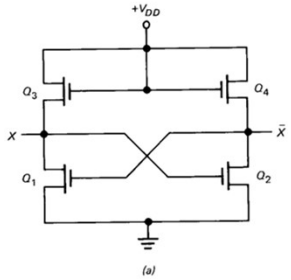
ADDRESS = $A_2A_1A_0$ = 100

the 1-of-8 decoder applies a high voltage to the R_4 register, and the ROM output is

D = 0110

RAM – Random Access Memory

Memory	ROM	RAM
Type	Non-volatile	Volatile
Read-write	Slow	Fast
Cost	Low	High



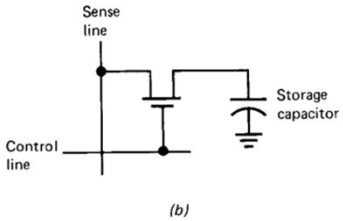


Fig. 9-4 (*a*) Static cell; (*b*) dynamic cell.

Semiconductor RAMs

Semiconductor RAMs may be *static* or *dynamic*. The static RAM uses bipolar or MOS flip-flops; data is retained indefinitely as long as power is applied to the flip-flops. On the other hand, a dynamic RAM uses MOSFETs and capacitors that store data. Because the capacitor charge leaks off, the stored data must be *refreshed* (recharged) every few milliseconds. In either case, the RAMs are volatile; turn off the power and you lose the stored data.