

Chapter-2



- Decoder
- Encoder
- MUX



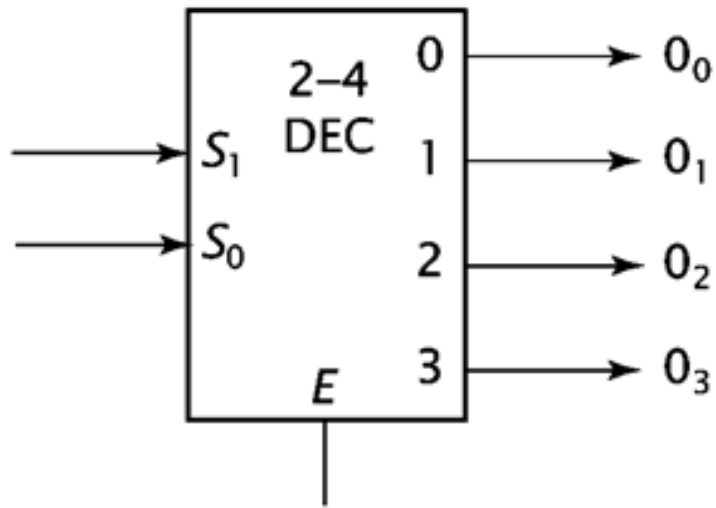
Decoder

- **Accepts a value and decodes it**
 - **Output corresponds to value of n inputs**
- **Consists of:**
 - Inputs (n)
 - Outputs (2^n , numbered from $0 \rightarrow 2^n - 1$)
 - Selectors / Enable (active high or active low)

The truth table of 2-to-4 Decoder

S_1	S_0	E	O_0	O_1	O_2	O_3
X	X	0	0	0	0	0
0	0	1	1	0	0	0
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	1

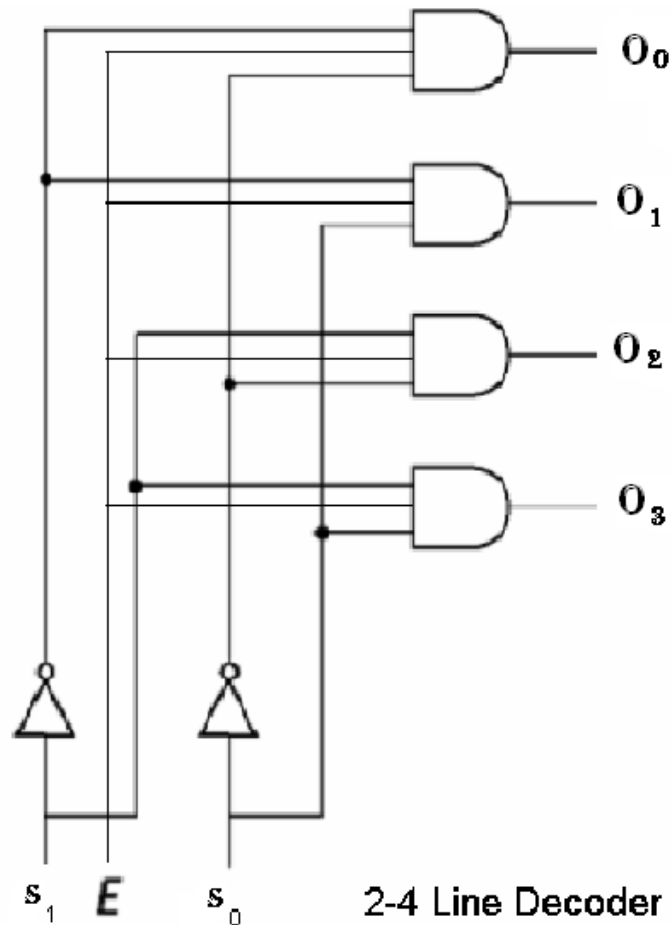
2-to-4 Decoder



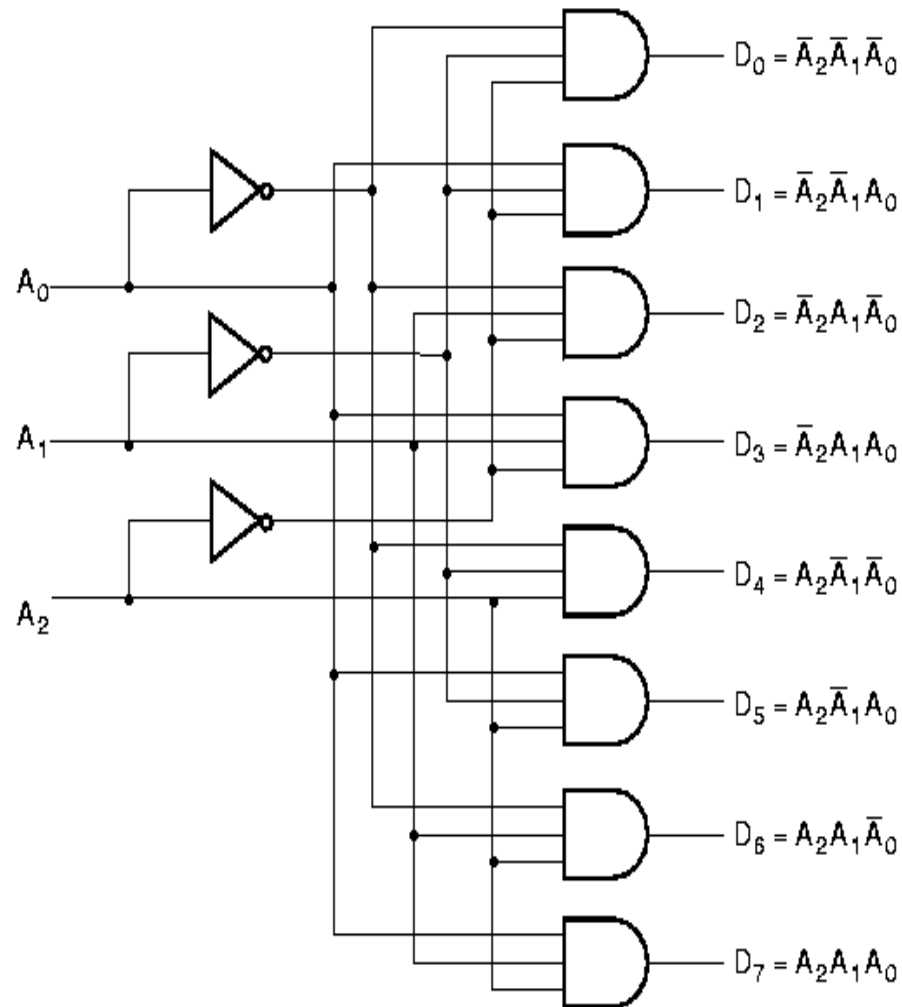
S_1	S_0	E	O_0	O_1	O_2	O_3
X	X	0	0	0	0	0
0	0	1	1	0	0	0
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	1

(b)

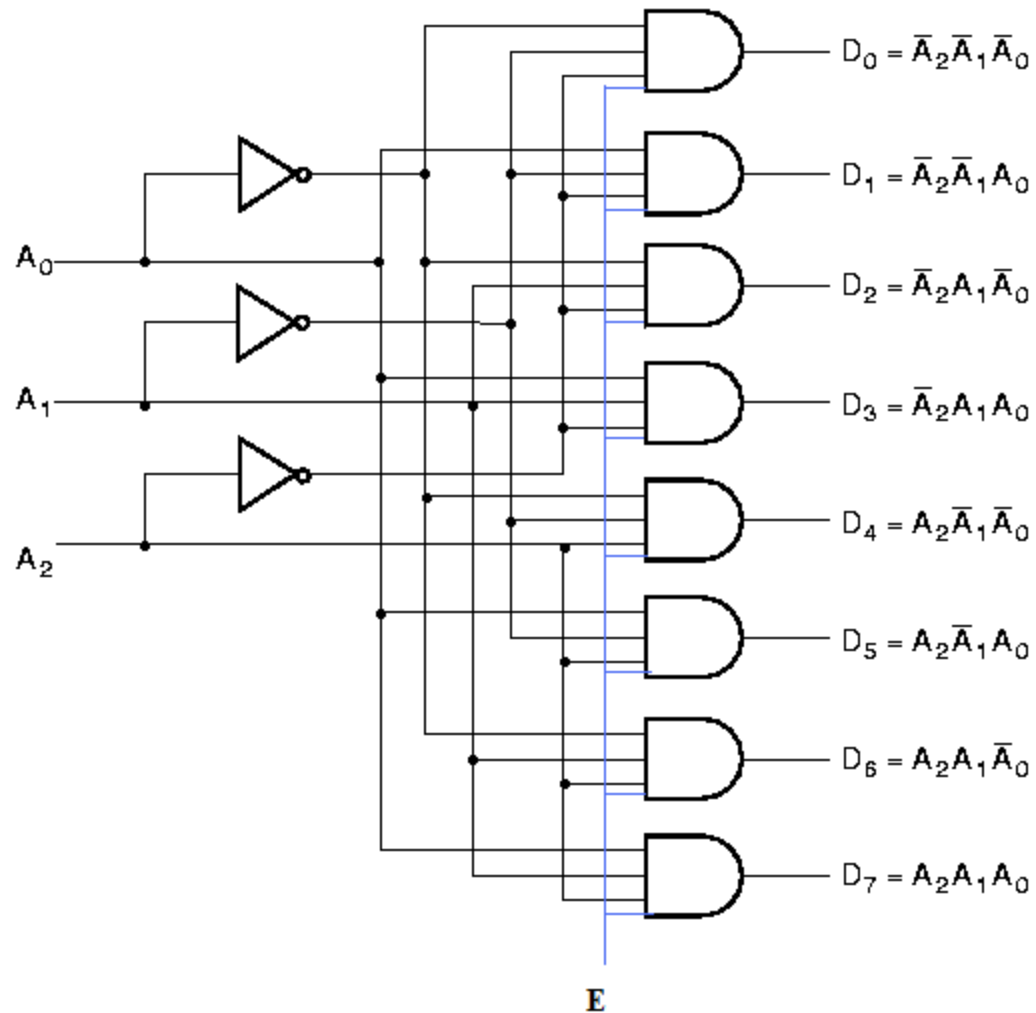
2-to-4 Decoder



3-to-8 Decoder



3-to-8 Decoder with Enable

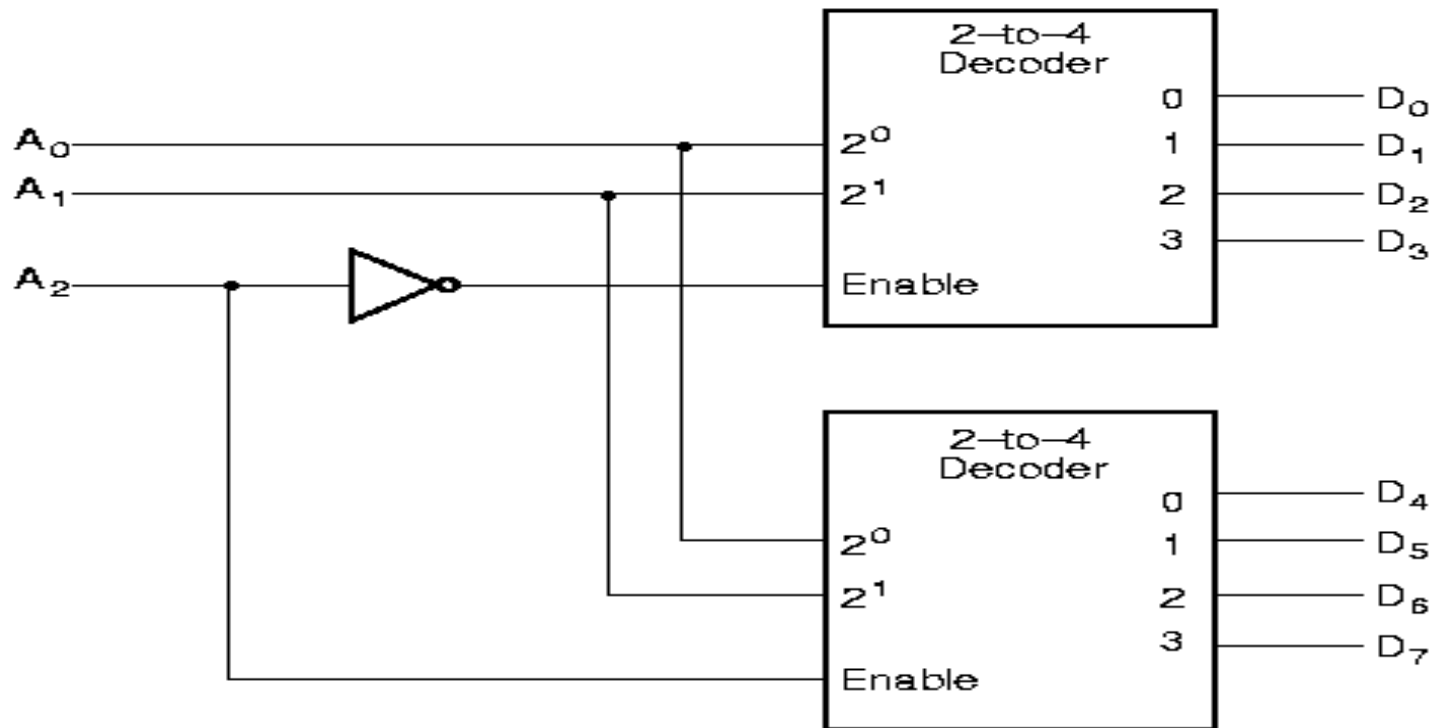




Decoder Expansion

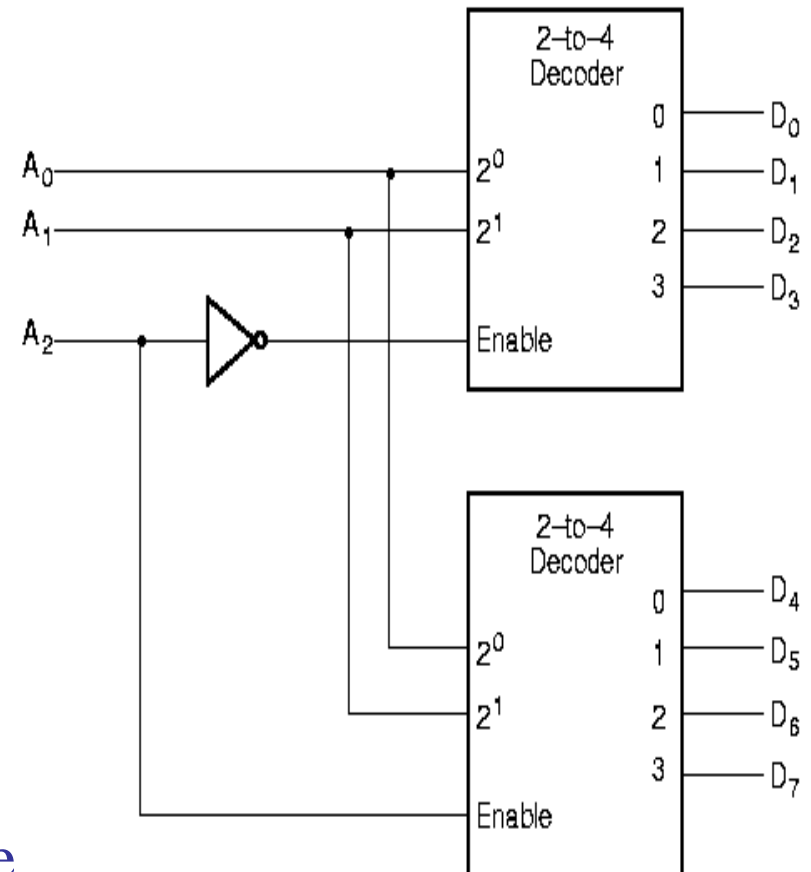
- Decoder expansion
 - Combine two or more small decoders with enable inputs to form a larger decoder
 - 3-to-8-line decoder constructed from two 2-to-4-line decoders
 - **The (Most Significant Bit) MSB is connected to the enable inputs**
 - **if $A_2=0$, upper is enabled; if $A_2=1$, lower is enabled.**

Decoder Expansion



Combining two 2-4 decoders to form one 3-8 decoder using enable switch

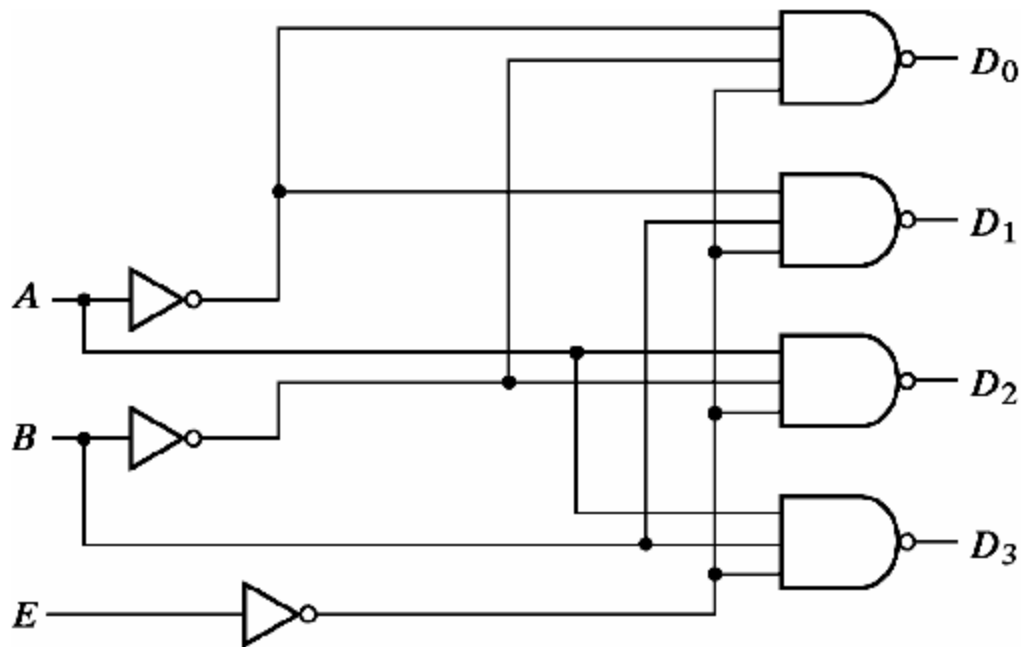
A_2	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0



The highest bit is used for the enable

2-to-4 Decoder: NAND implementation

Decoder is enabled when $E=0$ and an output is active if it is 0



(a) Logic diagram

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

(b) Truth table



Outline

- Decoder
- Encoder
- Mux



Encoders

- Perform the inverse operation of a decoder
 - 2^n (or less) input lines and n output lines



Encoders

- Perform the inverse operation of a decoder
 - 2^n (or less) input lines and n output lines

Table: Truth Table for Octal-to-Binary

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

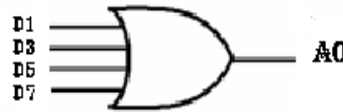
Encoders

Encoders can be implemented with OR gates whose inputs are determined directly from the truth table. Output $A_0=1$ if the input octal digit is 1 or 3 or 5 or 7. Similar conditions apply for the other two outputs. These can be expressed by the following Boolean functions with 3 OR gates

$$A_0 = D_1 + D_3 + D_5 + D_7;$$

$$A_1 = D_2 + D_3 + D_6 + D_7;$$

$$A_2 = D_4 + D_5 + D_6 + D_7;$$





Outline

- Decoder
- Encoder
- Mux

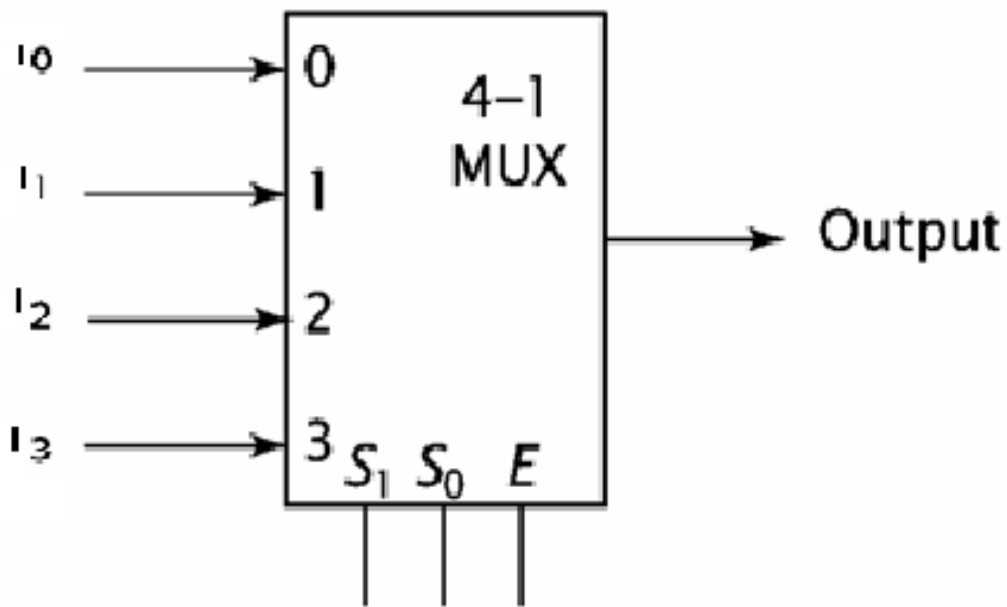
Multiplexer (MUX)



A multiplexer can use addressing bits to select one of several input bits to be the output.

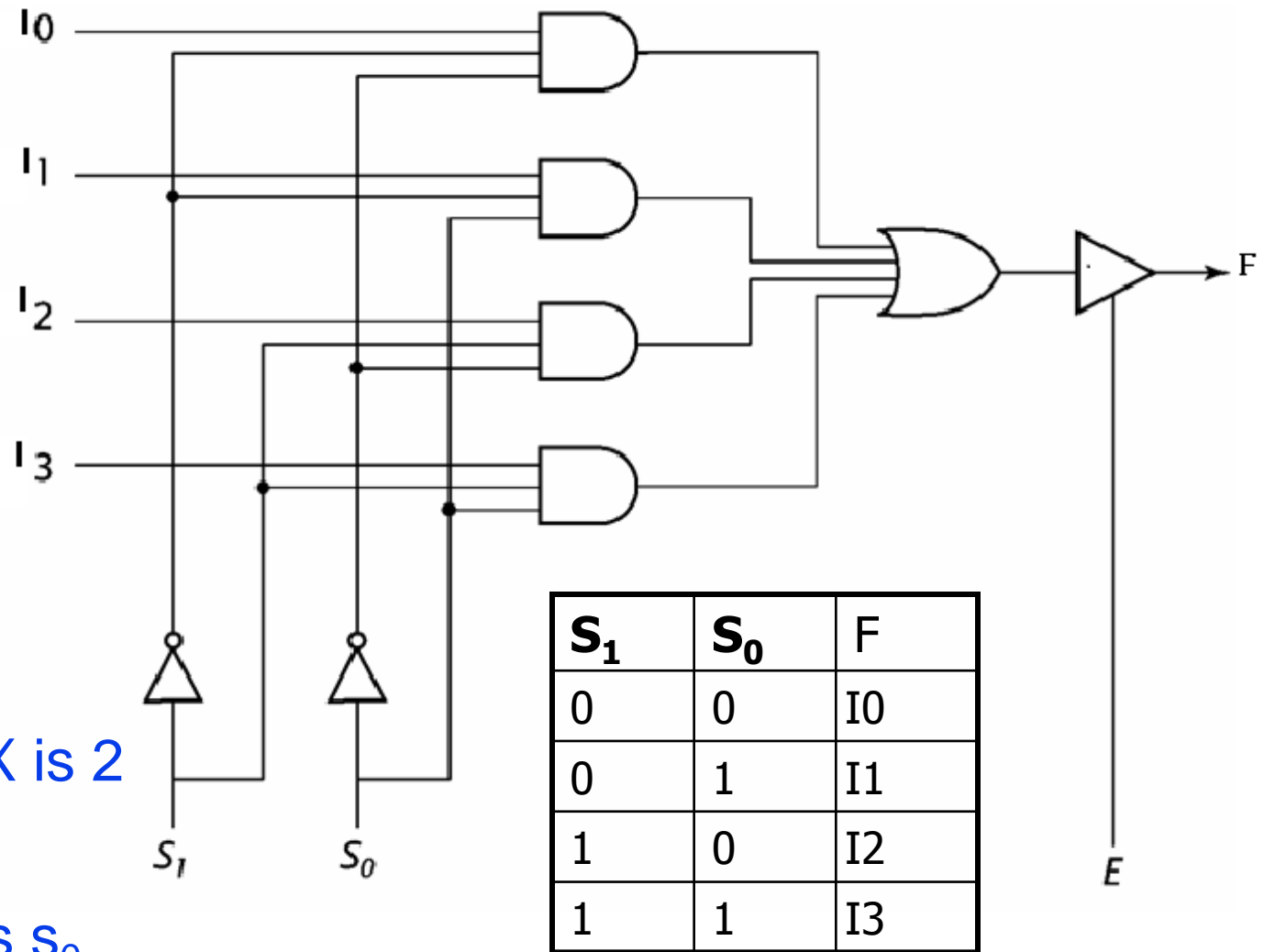
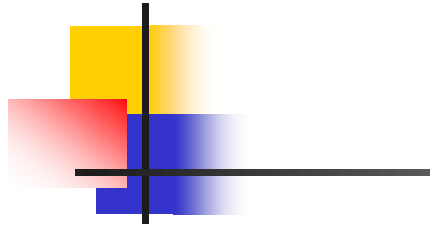
- A selector chooses a single data input and passes it to the MUX output
- It has one output selected at a time.

Function table with enable



S_1	S_0	E	Output
X	X	0	X
0	0	1	I_0
0	1	1	I_1
1	0	1	I_2
1	1	1	I_3

4 to 1 line multiplexer



4 to 1 line
multiplexer

2^n MUX to 1

n for this MUX is 2

This means 2
selection lines s_0
and s_1



Multiplexer (MUX)

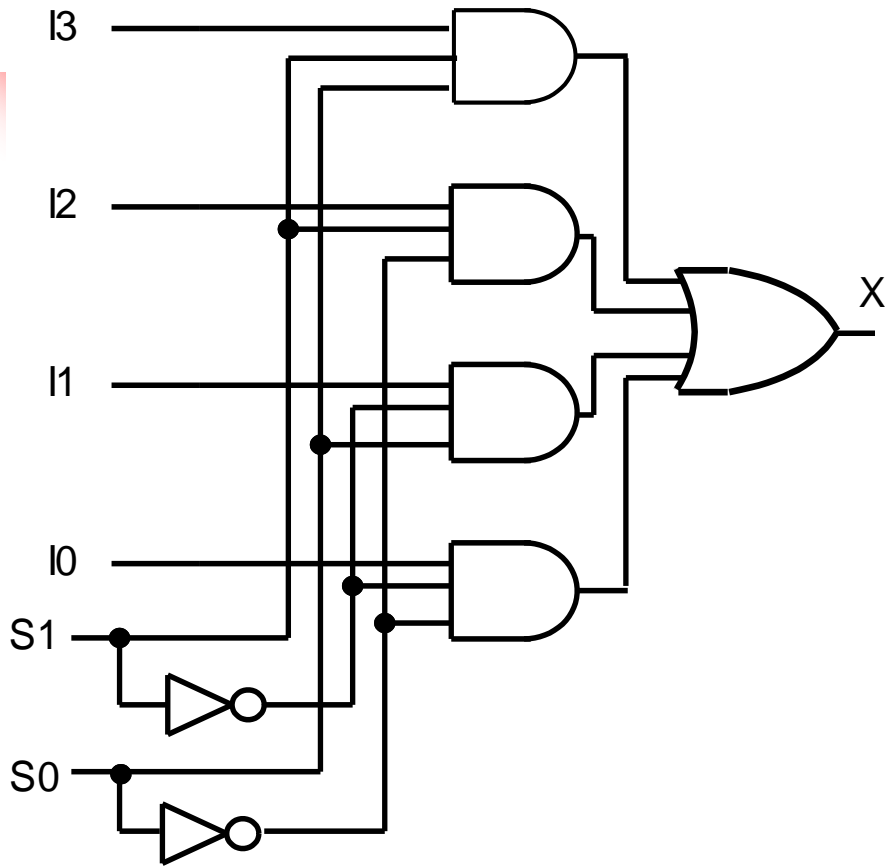
- Consists of:
 - Inputs (multiple) = 2^n
 - Output (single)
 - Selectors (# depends on # of inputs) = n
 - Enable (active high or active low)



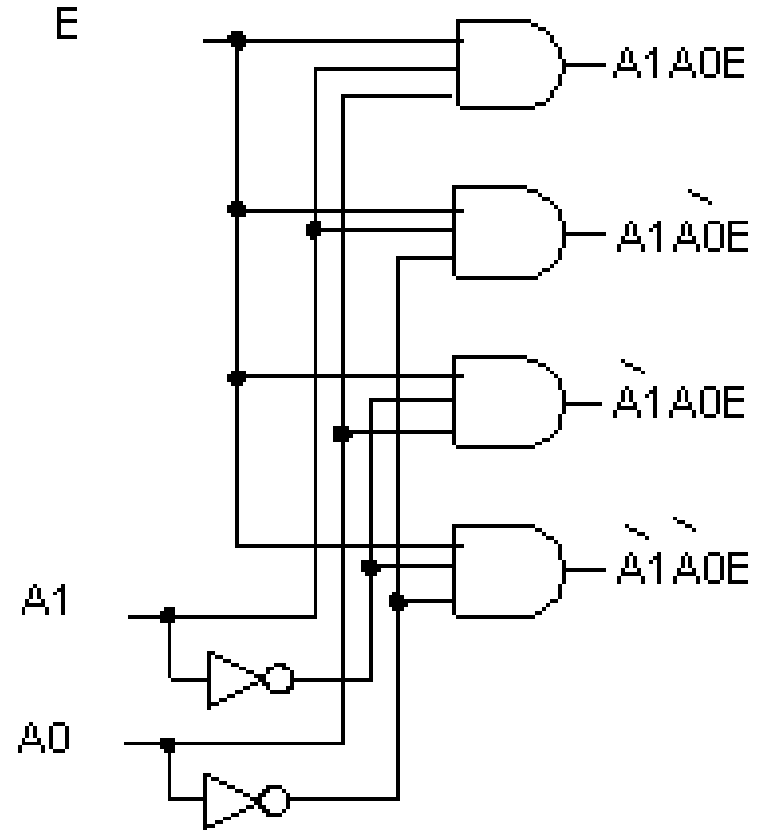
Multiplexers versus decoders

- A Multiplexer uses n binary select bits to choose from a maximum of 2^n unique input lines.
- Decoders have 2^n number of output lines while multiplexers have only one output line.
- The output of the multiplexer is the data input whose index is specified by the n bit code.

Multiplexer Versus Decoder



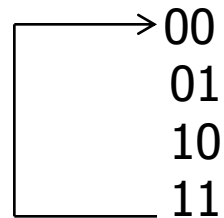
4-to-1 Multiplexer



2-to-4 Decoder

Note that the multiplexer has an extra OR gate. A_1 and A_0 are the two inputs in decoder. There are four inputs plus two selects in multiplexer.

Binary Counter (example of sequential circuit): A register that goes through a predetermined sequence of states upon the application of input pulses is called a counter. They may occur at uniform intervals of time. They are used for counting no. of occurrences of an event A counter that follows the binary number sequence is called a binary counter. There are two types of counters: Binary up counter and Binary down counter. 2-bit binary up counter generates values iteratively in ascending order e.g.



while 2-bit binary down counter generates values in descending order e.g. 11, 10, 01, 00

1. Registers

A register is a **storage location** located inside the processor. A modern processor has many registers.

Registers are used to hold :

- **data** which is being processed
- **instructions** which are being executed
- **addresses** which are about to be accessed.

Register

Register

Register

Register

Registers

- A common sequential device: Registers
 - They're a good example of sequential analysis and design
 - They are also frequently used in building larger sequential circuits
- **Registers** hold larger quantities of data than individual flip-flops
 - Registers are central to the design of modern processors
 - There are many different kinds of registers
 - We'll show some applications of these special registers

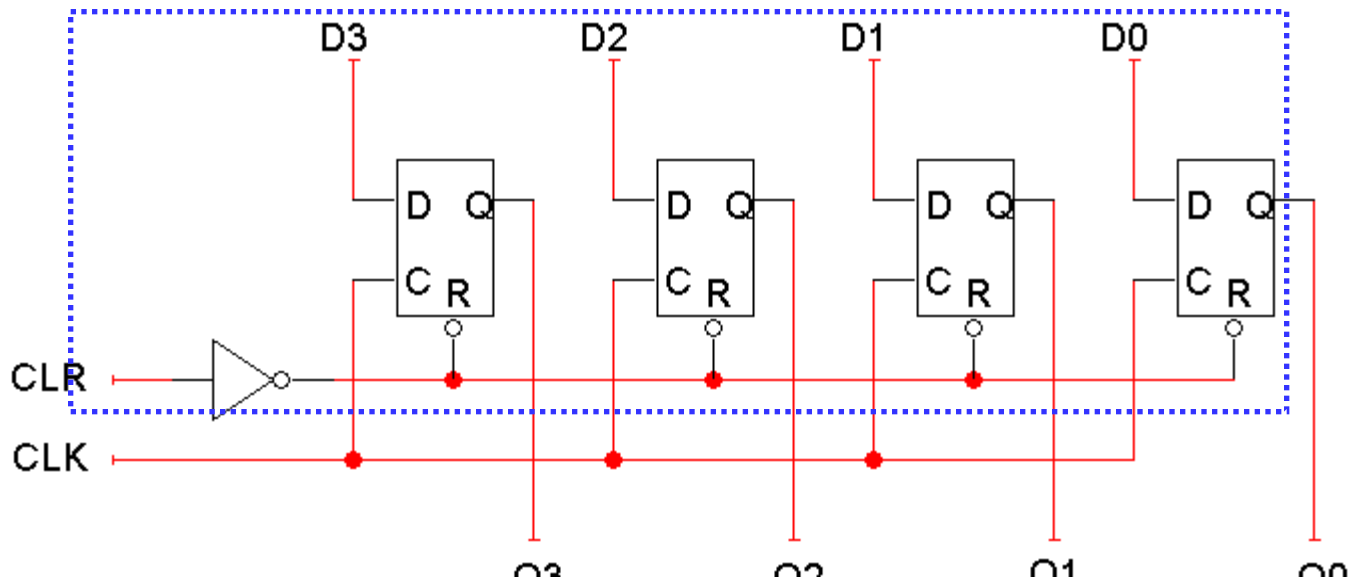


What good are registers?

- Flip-flops are limited because they can store only one bit
- A **register** is an extension of a flip-flop that can store multiple bits
- Registers are commonly used as temporary storage in a processor
 - They are faster and more convenient than main memory
 - More registers can help speed up complex calculations

A basic register

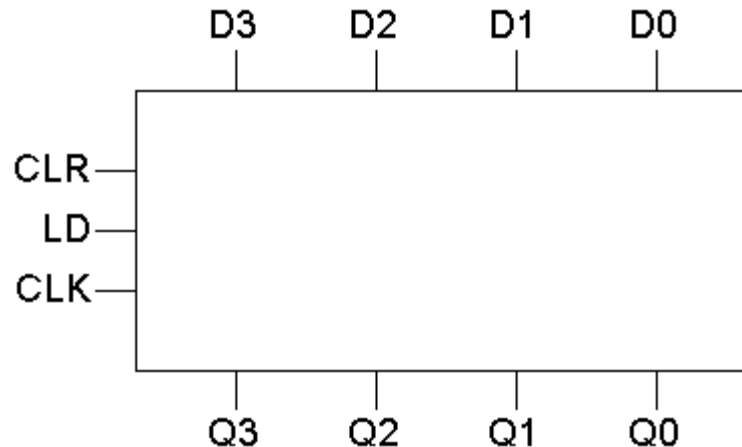
- Basic registers are easy to build. We can store multiple bits just by putting a bunch of flip-flops together!
- A 4-bit register is given below
 - This register uses D flip-flops, so it's easy to store data without worrying about flip-flop input equations
 - All the flip-flops share a common CLK and CLR signal
 - -With each rising edge of clock pulse data is inputted as D₀, D₁, D₂ and D₃ and as D-flip-flop is used so finally same values are retained in flip-flops so as to be generated as final output values of Q₀, Q₁, Q₂ and Q₃
 - Clear input is used for clearing all values of register 0's before clock operation



Adding a parallel load operation

- Transfer of new information into a register is referred to as loading a register
- The input D_3-D_0 is copied to the output Q_3-Q_0 on every clock cycle
- How can we store the current value for more than one cycle?
- Let's add a load input signal LD to the register
 - If $LD = 0$, the register keeps its current contents
 - If $LD = 1$, the register stores a new value, taken from inputs D_3-D_0

LD	$Q(t+1)$
0	$Q(t)$
1	D_3-D_0



RAM

Random Access Memory. This is where information is stored temporarily while the computer is on although this is lost when the computer is switched off!



ROM

Read Only Memory. This is where data and programs are stored permanently.




MEMORY CLASSIFICATION

In general the memory is classified in two types based on their mode of access of a memory system.

1. Random access memory
2. Sequential access memory

- ▶ **Random Access Memory**: The world of data reading or writing from or to the memory requires same time. We can access the data randomly.

Example: hard disk.



Types of ROM

- **PROM - Programmable Read Only Memory:**

Creating ROM chips totally from scratch is time-consuming and very expensive in small quantities. For this reason, developers created a type of ROM known as programmable read-only memory (PROM). Blank PROM chips can be bought inexpensively and coded by the user with a programmer. PROM chips have a grid of columns and rows just as ordinary ROMs do. It can be programmed using electrical fuses

- **EPROM - Erasable Programmable Read Only Memory**

Working with ROMs and PROMs can be a wasteful business. Even though they are inexpensive per chip, the cost can add up over time. Erasable programmable read-only memory (EPROM) addresses this issue. EPROM chips can be rewritten many times. Erasing an EPROM requires a special tool that emits a certain frequency of ultraviolet (UV) light.

- **EEPROM - Electrically Erasable Programmable Read Only Memory**

- Though EPROMs are a big step up from PROMs in terms of reusability, they still require dedicated equipment and a labor-intensive process to remove and reinstall them each time a change is necessary. Also, changes cannot be made incrementally to an EPROM; the whole chip must be erased. Electrically erasable programmable read-only memory (EEPROM) chips remove the biggest drawbacks of EPROMs.

In EEPROMs:

- The chip does not have to be removed to be rewritten.
- The entire chip does not have to be completely erased to change a specific portion of it.
- Changing the contents does not require additional dedicated equipment.