1. Mux

2. Demultiplexers

3. ROMs

Slides of Adam Postula used

Unit 8

EE 510

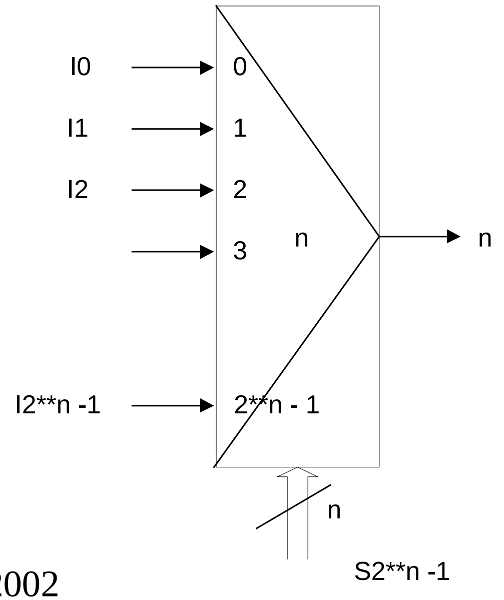
MSI Building Blocks

Topics to be Covered

- In this lecture we will cover the following set of basic combinational building blocks:
  - Multiplexers (MUXs) and demultiplexers (DMUXs).
    - As a result, define ACTIVE HIGH and ACTIVE LOW terminology.
  - Encoders
  - Decoders
  - ROMs

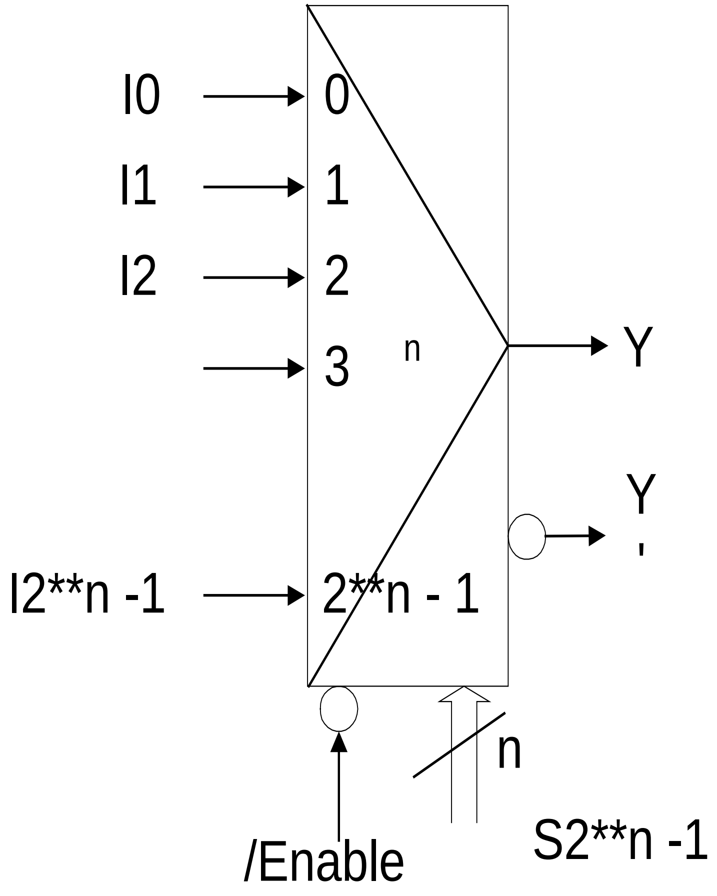
MUXs and DMUXs

- A multiplexer (MUX) or data selector is a combinational digital device that performs the function of selecting one of many data input lines for transmission to some destination.
- It uses *n* ***data select*** lines to control 2^n **data input** lines.
- It has the following block diagram:



MUXs and DMUXs

- The enable pin is active low, with the intention of deactivating the output Y (and Y').
- We need to define this terminology, and to define a set of standards to be followed.



Terminology

- The terms ACTIVE and INACTIVE are used throughout the digital area.
- They are supported by Institute of Electrical and Electronic Engineers (IEEE), the largest such professional group in the world.
- These terms are also frequently used in manufacturers data books.

Terminology

- A state is said to be ACTIVE if it is the condition for causing something to happen. And for every ACTIVE state there must exist one that is INACTIVE.
- In the binary system, these descriptors take the logic values:
  - ACTIVE = logic 1
  - INACTIVE = logic 0

Positive and Negative Logic

- In the positive logic system, we associate the physical voltage levels with the logic domain values as follows:

High Voltage (+5v)	$\longleftrightarrow$	Logic 1
Low Voltage (0v)	$\longleftrightarrow$	Logic 0

- In the negative logic system, we associate the physical voltage levels with the logic domain values as follows:

High Voltage (+5v)	$\longleftrightarrow$	Logic 0
Low Voltage (0v)	$\longleftrightarrow$	Logic 1

Positive and Negative Logic

- In the logic domain, logic 1 is always the ACTIVE state, while logic 0 is always the INACTIVE state.
- When we think and talk about logic circuits, we normally think in the positive logic system, when inputs and outputs are ACTIVE HIGH.
- Sometimes, we see signals referred to as ACTIVE LOW, e.g., the CLEAR pin on a register. This indicates that when the voltage drops to it low voltage level, the CLEAR action is activated.

Documentation Standards

Signal Names

- Each input and output in a logic circuit should have a distinctive alphanumeric label, the signal’s name.
- We have used single character signal names for many circuits in the past because the circuits didn’t do much.
- In a real system, well-chosen signal names convey information to someone reading the logic diagram the same way that variable names in a software program do.

Documentation Standards

Active Levels

- Each signal name should have an *active level* associated with it.
- A signal is *active high* if it performs the named action or denotes the named condition when it is HIGH or 1.
- A signal is *active low* if it performs the named action or denotes the named condition when it is LOW or 0..

Active Levels

- A signal is said to be *asserted* when it is at its active level.
- A signal is said to be *negated* or *deasserted* when it is not at its active level.
- The active level of each signal in a circuit is normally specified as part of its name, according to some convention.

Documentation Standards

Active Levels

Different Naming Conventions

Active Low	*Active High*
READY-	READY+
ERROR.L	ERROR.H
ADDR15(L)	ADDR15(H)
RESET*	RESET
ENABLE˜	ENABLE
/TRANSMIT	TRANSMIT

Naming convention must be compatible with the input requirements of and CAD tools.

Documentation Standards

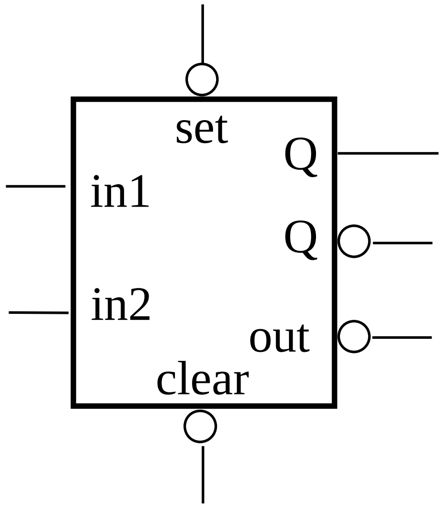
Active Levels

- In this case, we will take the convention of the last entry in the table, ie, /TRANSMIT for an active low signal. Tinder chose the third form, ie, ADDR15(L) for an active low signal.
- An active-low signal name has a prefix of /, and an active high signal has no prefix.
- The / may be read as “not”.
- The logic signal /GO treats the / as part of the name, and not as a separate symbol.

Documentation Standards

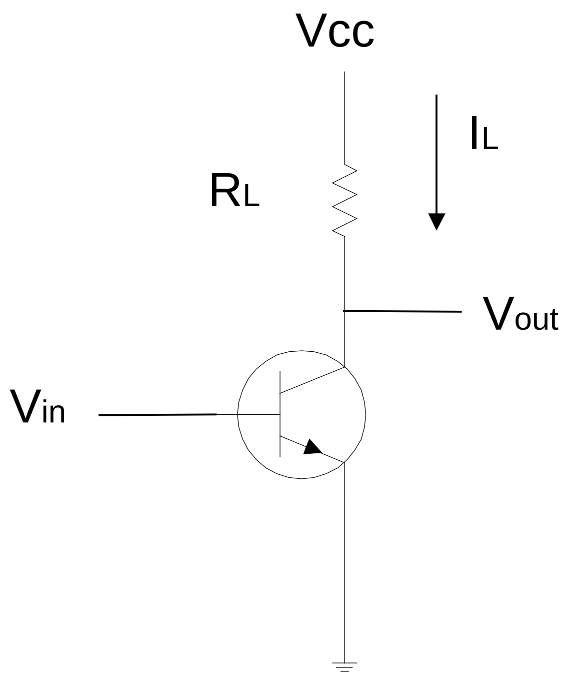
Active Levels for Pins

- Active levels may be associated with input and output pins of gates and larger-scale logic elements.
- We use an inversion bubble to indicate an active-low pin and the absence of a bubble to indicate an active-high pin.



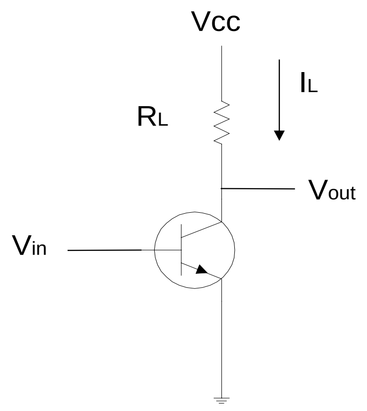
Why have ACTIVE LOW?

- By now, this question should have formed in your mind. It is tied up with power consumption, and requires that we return to a transistor level description of the output stage of (for example) a TTL gate:



Why ACTIVE LOW?

- When Vin turns the transistor ON, the collector voltage drops to ~0.7V, and current flows from Vcc into the ground, dissipating heat in R_L.
- The power dissipated is approximately Vcc^2/R_L.
- When Vin turns the transistor OFF, Vout rises to Vcc, the current through R_L is negligible, and no power is dissipated in R_L.

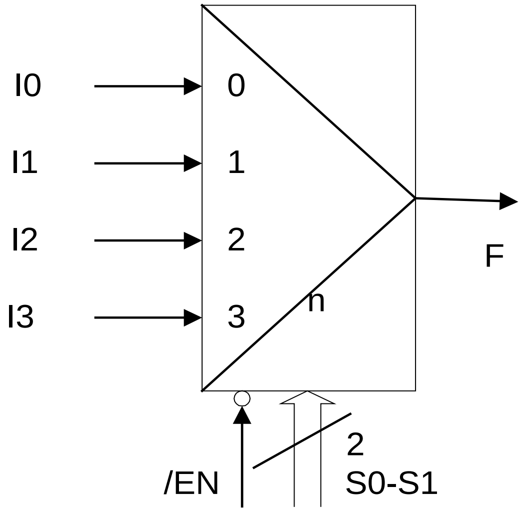


- **When a circuit is not to be used for long periods of time, leave it in the Vout high (logic 1) state, and assert the logic 0 state for the short time needed to activate it.**

Design of a MUX

- First, let us consider a small 4-to-1 MUX. This has 2 select lines. The truth table appears as follows:

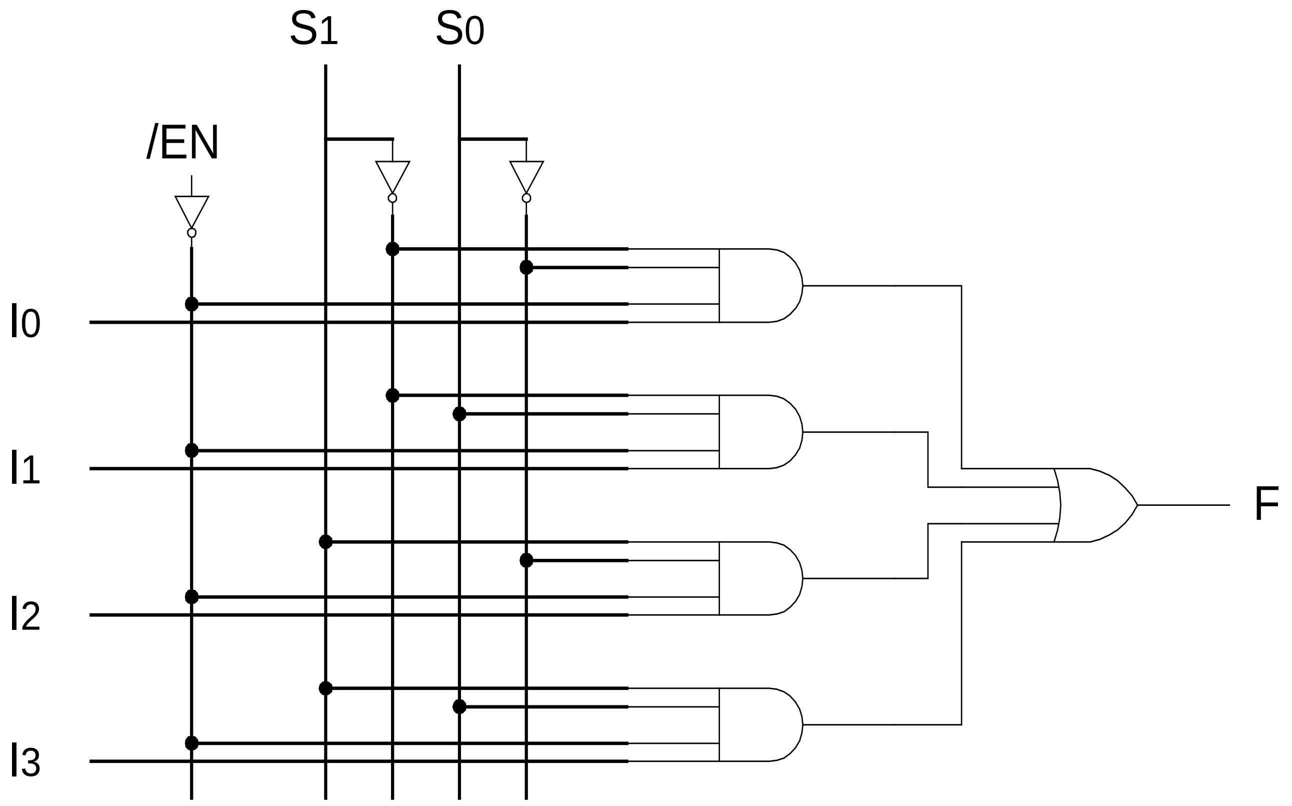
EN	S_1	S_0	F
0	x	x	0
1	0	0	I_0
1	0	1	I_1
1	1	0	I_2
1	1	1	I_3



$$F = (S_1'S_0'I_0 + S_1'S_0I_1 + S_1S_0'I_2 + S_1S_0I_3)EN'$$

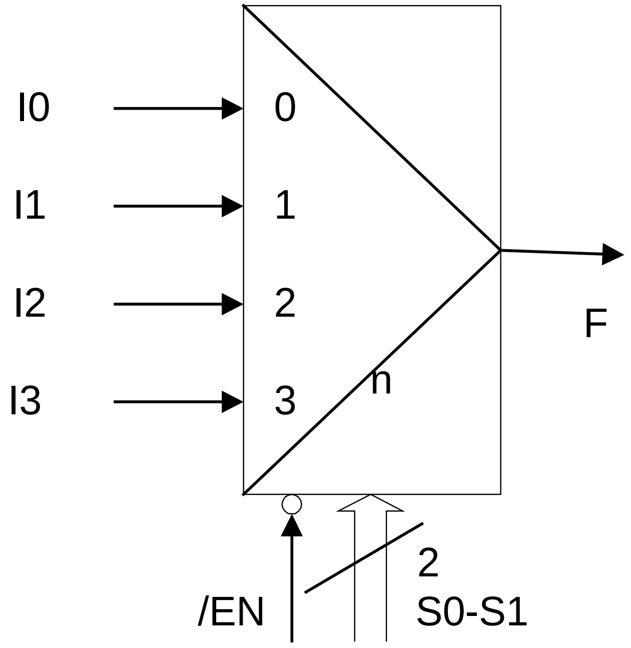
Design of a MUX

- Logic circuit:



Designing with MUXs

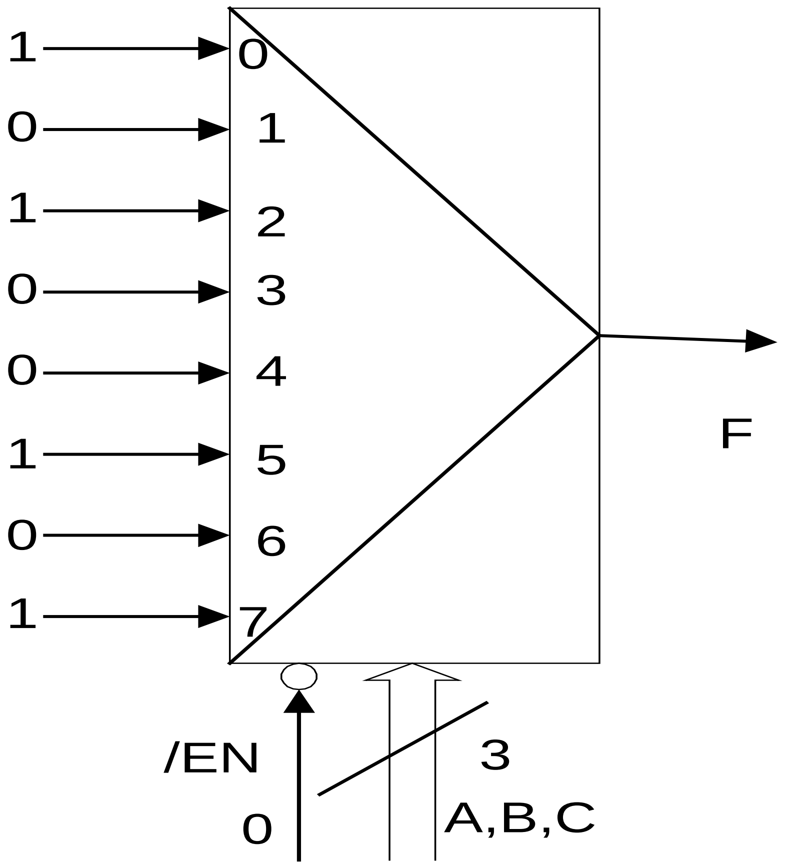
- Looked at in a slightly different way, a MUX is a 1-bit lookup lookup table:
  - S0 and S1 select which input bit I_i is to be looked up.
  - The input bit I_i is "programmed" by the designer.



Designing with a MUX

$$F(A,B,C) = \sum m(0,2,5,7)$$

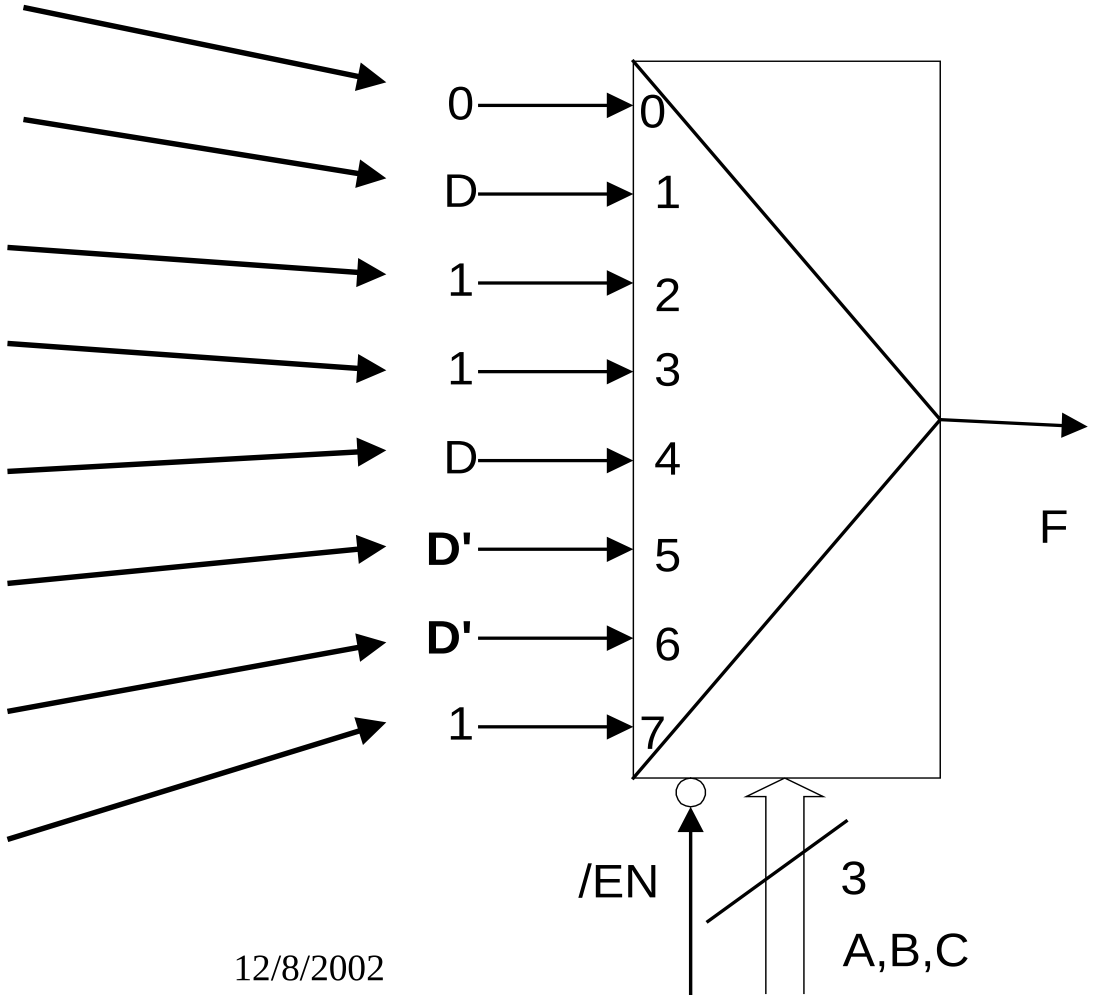
A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



Using a 3-to-8 MUX with 4 input variables

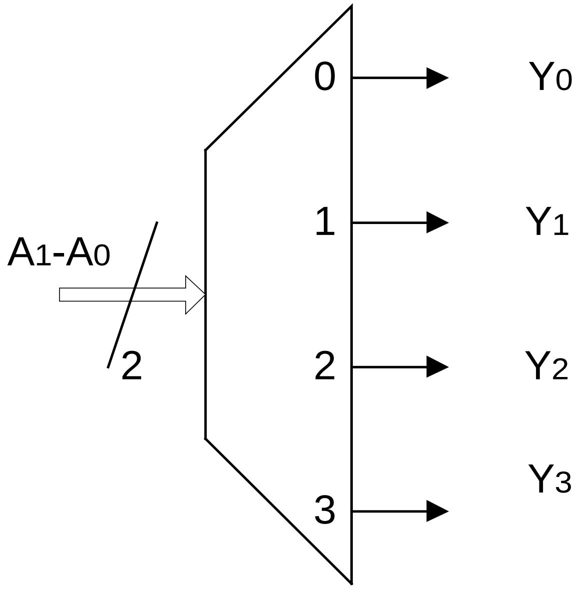
$$F(A,B,C,D) = \sum m(3,4,5,6,7,9,10,12,14,15)$$

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1



DECODERS

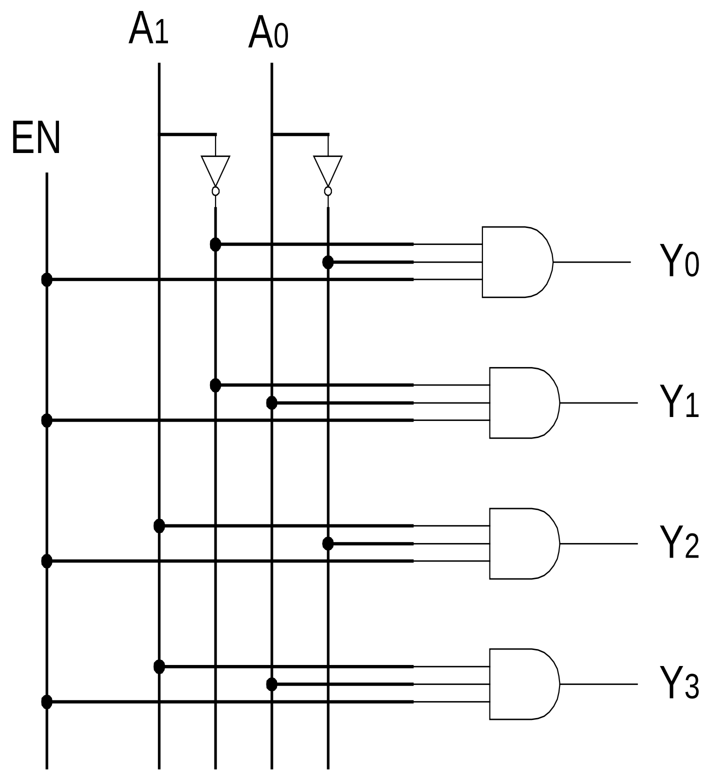
- A DECODER is an *n*-input/2^n-output combinational logic device which has the function of activating one of its 2^n outputs for every unique input pattern (or WORD) of *n* bits.
- Each output is identified by the MINTERM CODE, m_i, of the input WORD pattern it represents.



Design of a Decoder

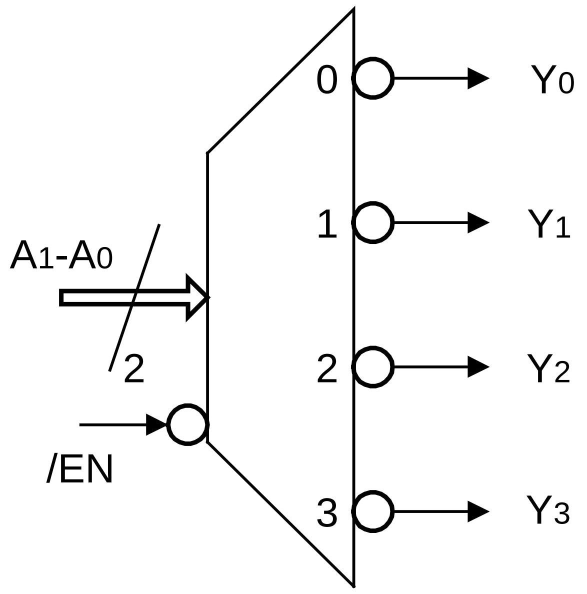
- Truth table of 2-to-4 decoder with ACTIVE HIGH OUTPUTS and ACTIVE HIGH ENABLE.

EN	A1	A0	Y3	Y2	Y1	Y0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0



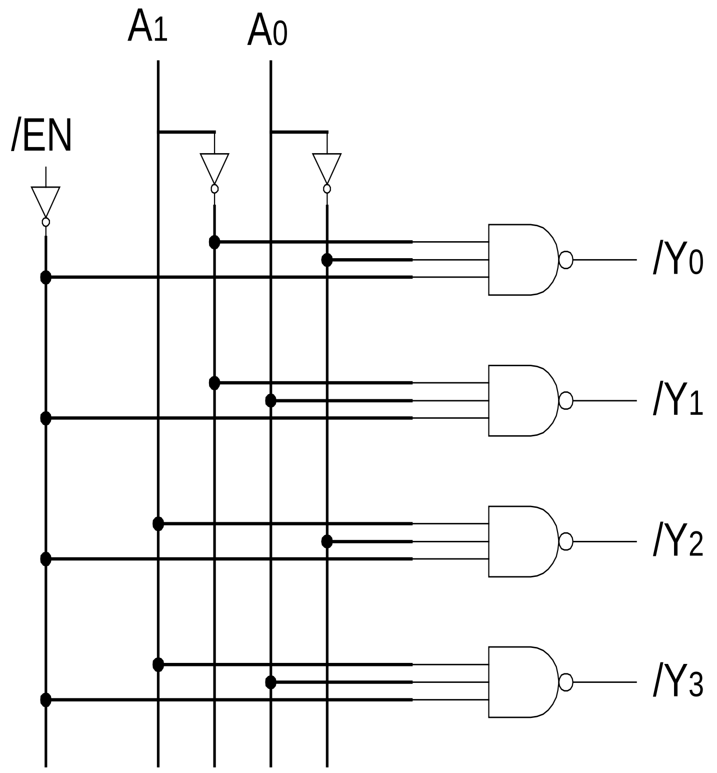
Commercial Decoders

- This is a representation of a commercially available 2-to-4 decoder, with an ACTIVE LOW ENABLE and ACTIVE LOW OUTPUTS.
- The ACTIVE LOW ENABLE is provided so that decoders can be stacked.



Design of a Commercial Decoder

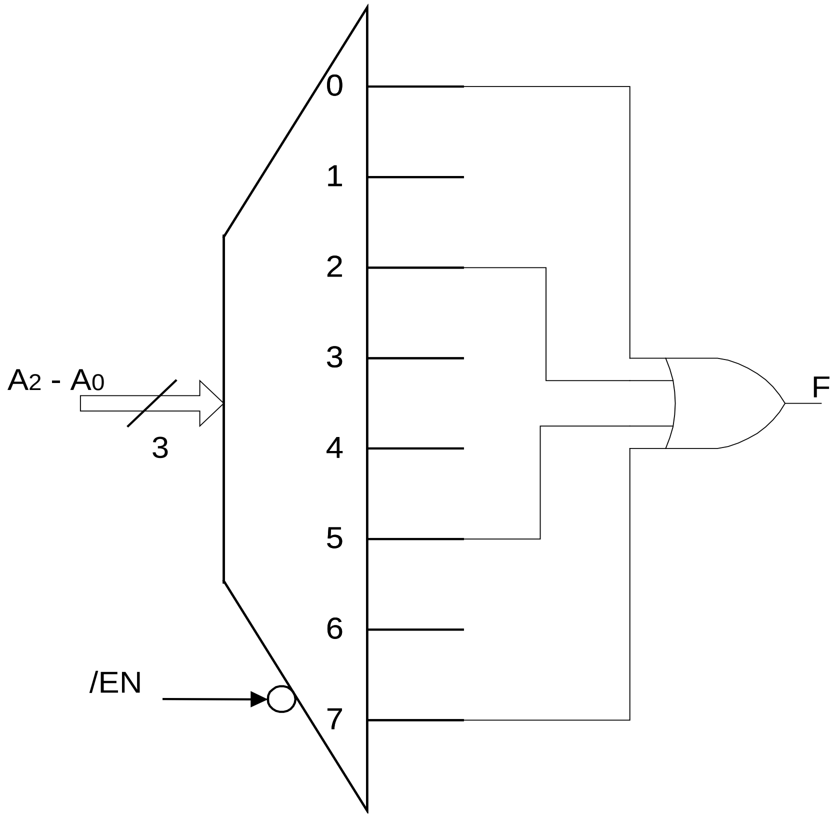
- Implementation of a 2-to-4 decoder. This has ACTIVE LOW ENABLE and ACTIVE LOW OUTPUTs.



Designing with Decoders

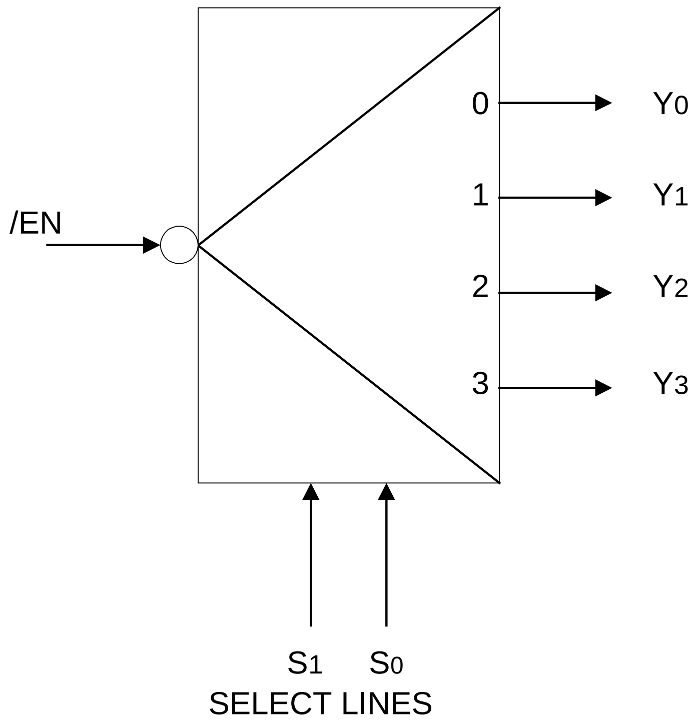
- Since each output is identified by the MINTERM CODE, combinational circuits can be implemented by ORing the appropriate outputs together.

$$F = \sum m(0,2,5,7)$$



DEMULTIPLEXERs (DMUXs)

- A DEMULTIPLEXER is a device with
  - 1 data input
  - *n* data select lines
  - 2^n output lines
  - which can route data from the single input source to one of several destinations.



DMUXs

- DMUXs are *not* commercially available, because DECODERs can be adapted easily to this particular use.
- In fact, manufacturers catalogues describe this device as a DECODER/DEMULTIPLEXER. The name chosen depends upon the use to which the device is put.
- To operate a decoder as a DMUX, use one of the inputs (usually the ENABLE) as the DATA INPUT line with the remaining inputs used as DATA SELECT lines.

Truth Table for DMUX

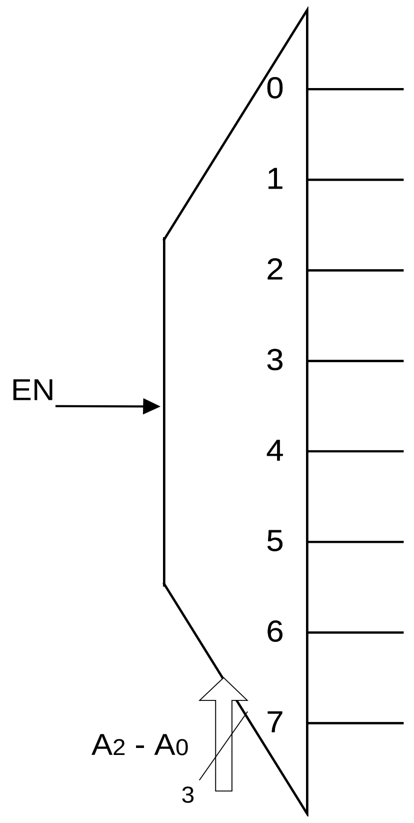
- This is the truth table for a decoder. However, it is also the truth table for a DMUX. Consider the case of DATA SELECT lines having the value 00. This selects output Y_0.

EN	A1	A0	Y3	Y2	Y1	Y0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

- From the first line, if EN is 0 the output Y_0 is 0.
- From the second line, output Y_0 is 1 if EN is 0.
- Thus, output Y_0 follows input EN; all other outputs remain at 0.

DMUX from a DECODER

- Shown here is the alternative view of a DECODER as a DMUX.
- It is left to you to determine what further gates are needed (if any) when utilizing a commercial DECODER/DMUX chip. remember that these usually have an ACTIVE LOW ENABLE and ACTIVE LOW OUTPUTs.

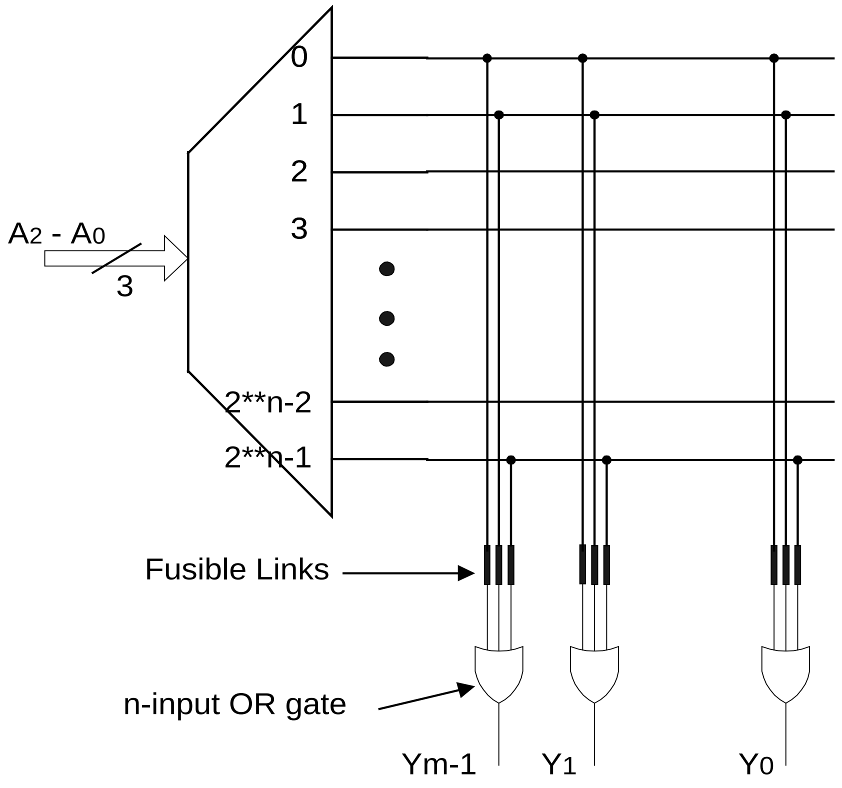


Read-Only Memory (ROM)

- ROMs belong to an important class of general-purpose IC logic array devices which find extensive use in combinational logic design.
- We met the use of ROMs in combinational logic design in Lecture 1. Here we shall examine the logic structure of a ROM.

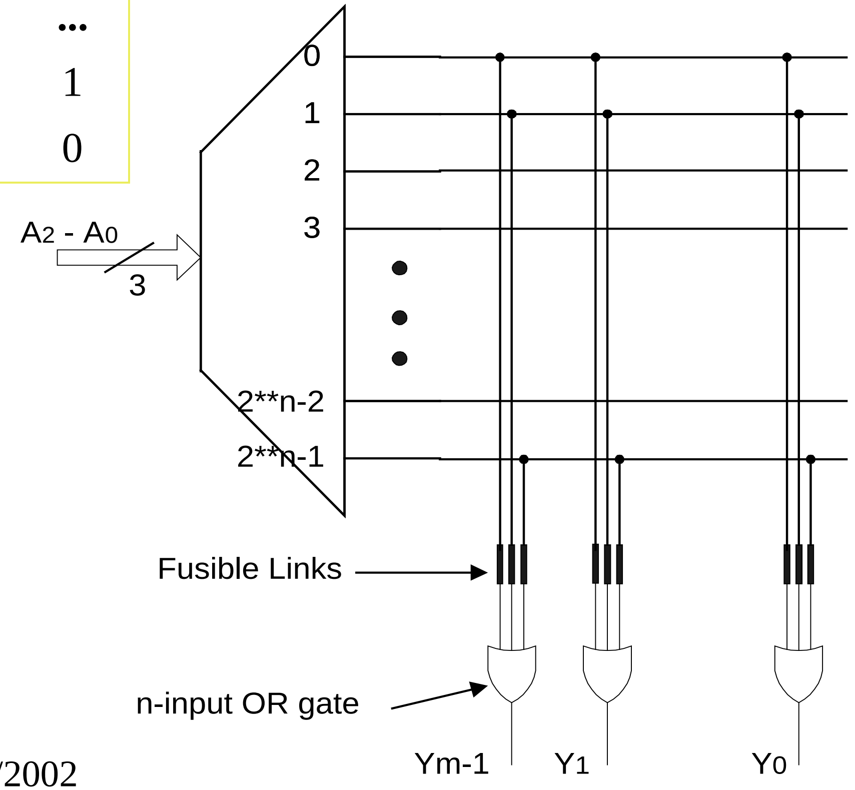
ROM Logic Structure

- A ROM is an *n*-input/*m*-output device, composed of a DECODER stage and a "memory" array.
- Bit combinations of *n* input variables are called *addresses*.
- There are 2^n possible addresses each representing a coded MINTERM on the output side of the DECODER stage.
- To program, the fusible links are "blown: to remove the connection.



ROM Logic Structure

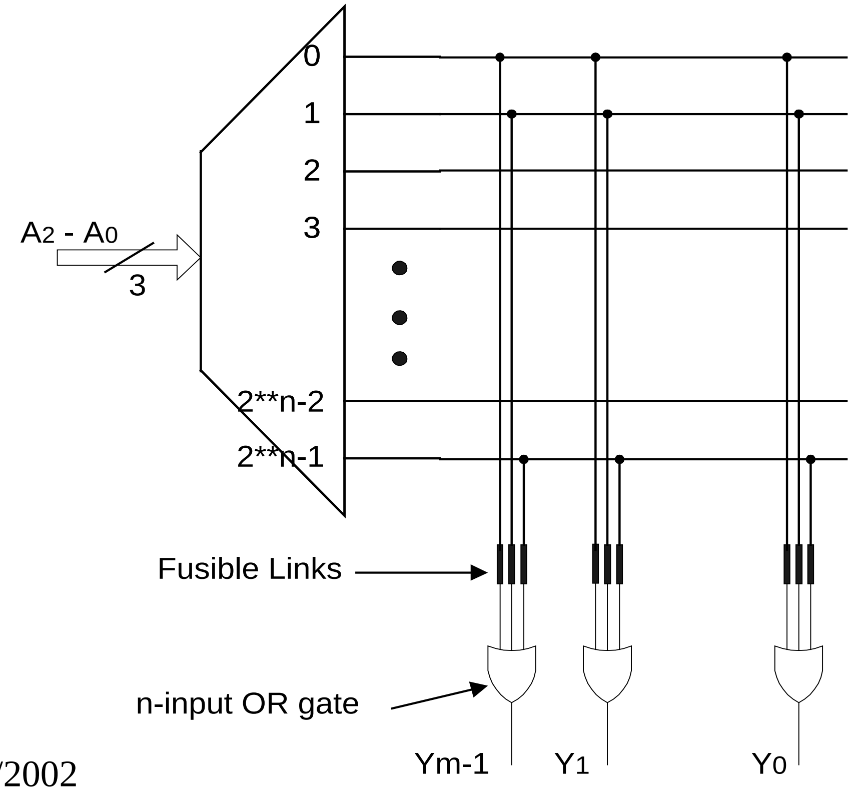
	n Input Variables				m Output Variables			
	I_{n-1}	...	I_1	I_0	Y_{m-1}	...	Y_1	Y_0
m_0	0	...	0	0	1	...	0	1
m_1	0	...	0	1	0	...	1	1
...	...	...	...	...	...	...	...	...
$m_{2^{n-2}}$	1	...	1	0	0	...	0	1
$m_{2^{n-1}}$	1	...	1	1	1	...	1	0



ROM Logic Structure

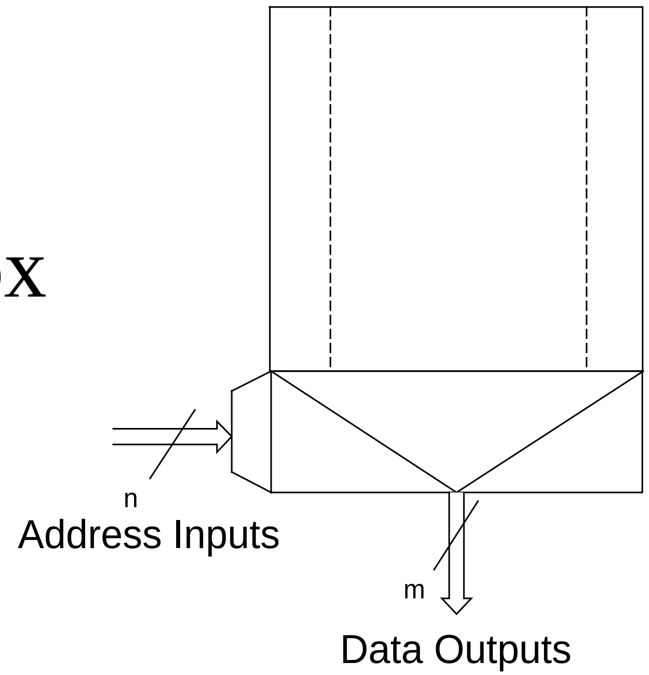
- In mask programmable ROMs the fusible links are manufactured in the silicon process, and cannot be field programmable.

- In Erasable Programmable ROMs (EPROMs) the fusible links are replaced by special nonvolatile charge-storage mechanisms.
- Fusible link based ROMs are called Programmable ROMs (PROMs).



Structural Diagram of a ROM

- Use this symbol whenever you need to draw a ROM in your logic diagram. It is far more expressive than a rectangular box with the name ROM (or DECODER or MUX) plastered somewhere near the box.



What have we covered in this lecture?

- We have looked at a small selection of MSI combinational logic devices.
- We examined MUXs, what they do, how they are constructed, and how they are used.
- We examined DECODERs, what they do, how they are constructed, and how they are used.
- We saw how to use decoders as DMUXs.

What have we covered in this lecture?

- We revisited ROMs, to see how they are constructed. We saw in Lecture 2 how to use ROMs in combinational design.
- Read the textbooks and my book for additional information.
- Think about your preferred grading option and project
- Write me about your project ideas