Chapter # 4: Programmable and Steering Logic

Contemporary Logic Design

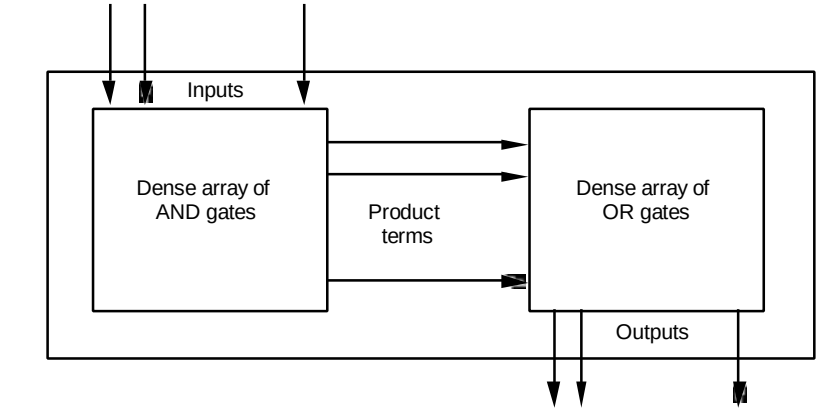
Randy H. Katz
University of California, Berkeley

June 1993

PALs and PLAs

Pre-fabricated building block of many AND/OR gates (or NOR, NAND) "Personalized" by making or breaking connections among the gates

Programmable Array Block Diagram for Sum of Products Form



Key to Success: Shared Product Terms

Example:

F0 = A + B' C'
F1 = A C' + A B
F2 = B' C' + A B
F3 = B' C + A

Personality Matrix

Product term	Inputs A	B	C	Outputs F_0	F_1	F_2	F_3
A B	1	1	-	0	(1)	(1)	0
$\overline{B}$ C	-	0	1	0	0	0	1
A $\overline{C}$	1	-	0	0	1	0	0
$\overline{B}$ $\overline{C}$	-	0	0	(1)	0	(1)	0
A	1	-	-	(1)	0	0	(1)

Reuse of terms

Input Side:

1 = asserted in term
0 = negated in term
- = does not participate

Output Side:

1 = term connected to output
0 = no connection to output

No. 4-3

Example Continued

All possible connections are available before programming

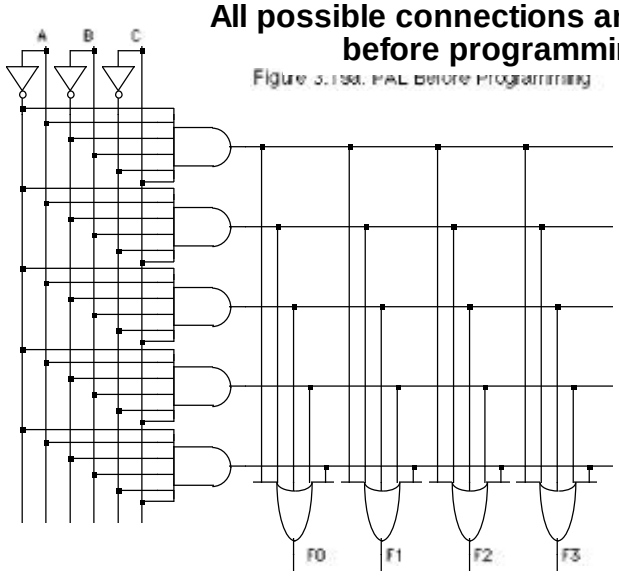
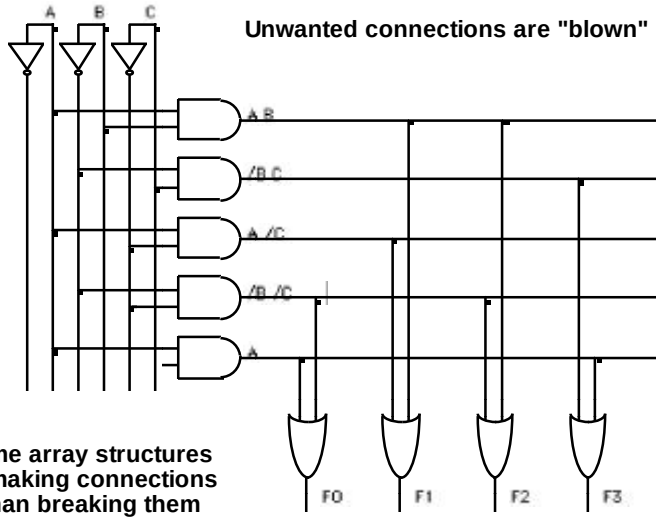


Figure 3.15a. PAL Before Programming

No. 4-4

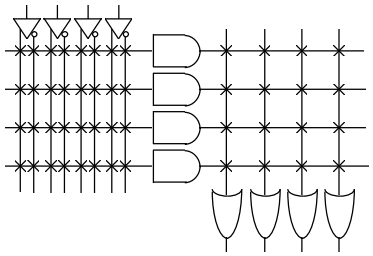
Example Continued



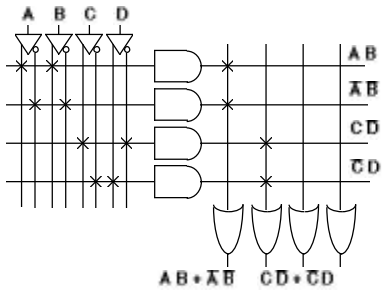
Unwanted connections are "blown"

Note: some array structures work by making connections rather than breaking them

Alternative representation for high fan-in structures



Short-hand notation so we don't have to draw all the wires!



Notation for implementing

F0 = A B + A' B'

F1 = C D' + C' D

Design Example

Multiple functions of A, B, C

F1 = A B C

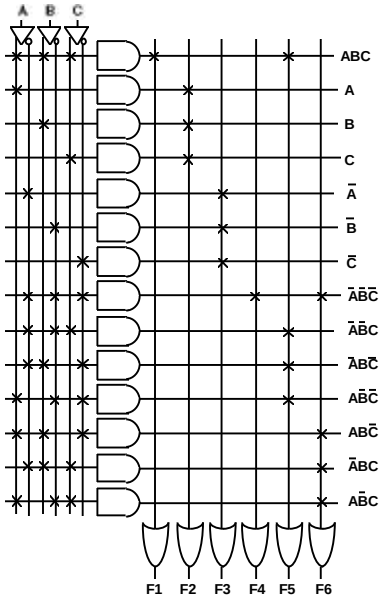
F2 = A + B + C

F3 = $\overline{A\ B\ C}$

F4 = $\overline{A + B + C}$

F5 = A xor B xor C

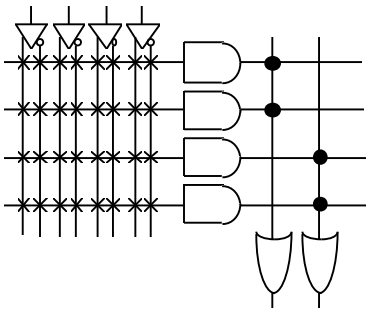
F6 = A xnor B xnor C



No. 4-7

What is difference between Programmable Array Logic (PAL) and Programmable Logic Array (PLA)?

PAL concept — implemented by Monolithic Memories constrained topology of the OR Array



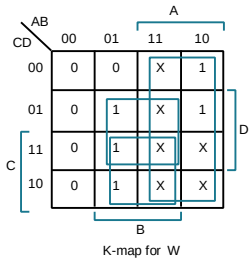
A given column of the OR array has access to only a subset of the possible product terms

PLA concept — generalized topologies in AND and OR planes

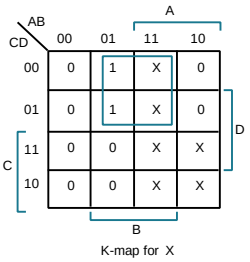
No. 4-8

Design Example: BCD to Gray Code Converter

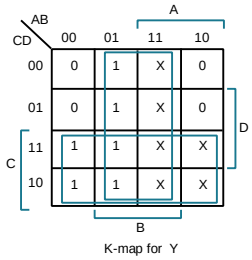
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	1	1	1	0
0	1	1	0	1	0	1	0
0	1	1	1	1	0	1	1
1	0	0	0	1	0	0	1
1	0	0	1	1	0	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X



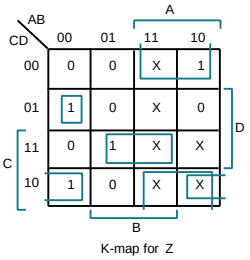
K-map for W



K-map for X



K-map for Y



K-map for Z

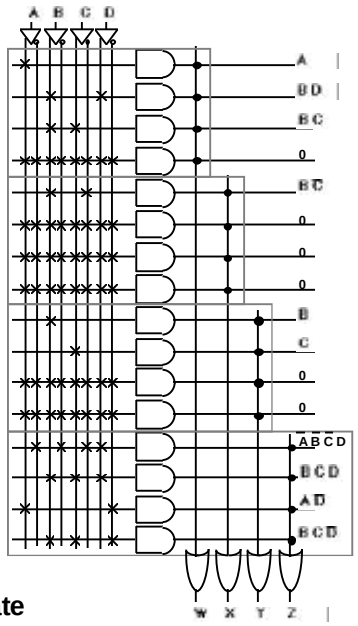
W = A + B D + B C

X = B C'

Y = B + C

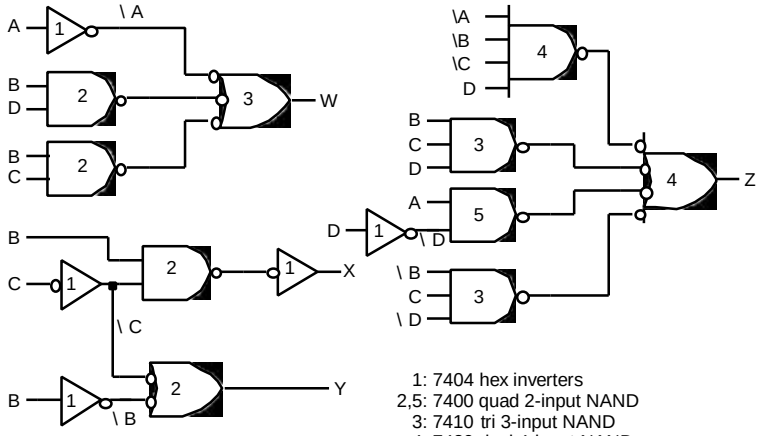
Z = A'B'C'D + B C D + A D' + B' C D'

Programmed PAL:



4 product terms per each OR gate

Code Converter Discrete Gate Implementation

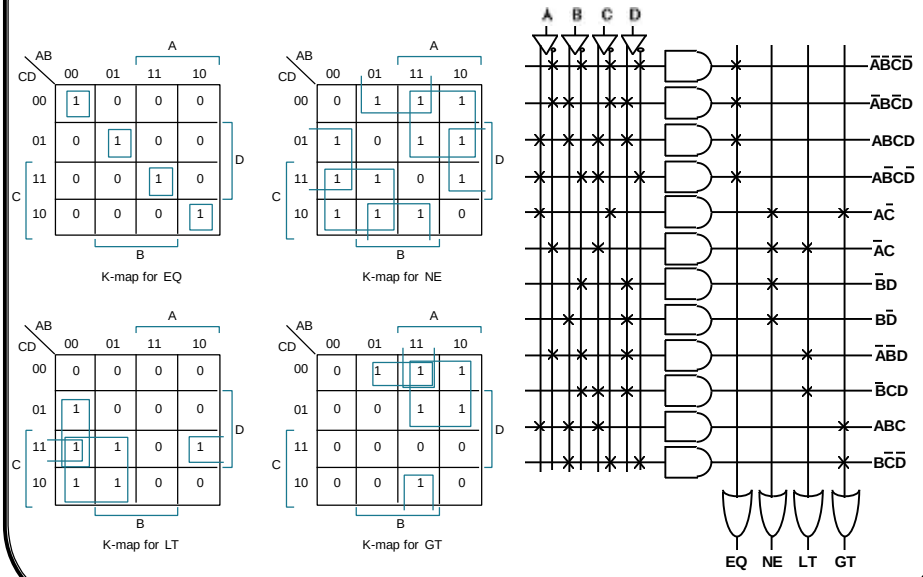


1: 7404 hex inverters
2,5: 7400 quad 2-input NAND
3: 7410 tri 3-input NAND
4: 7420 dual 4-input NAND

4 SSI Packages vs. 1 PLA/PAL Package!

No. 4-11

Another Example: Magnitude Comparator



No. 4-12

Non-Gate Logic

Introduction

AND-OR-Invert
PAL/PLA

Generalized Building Blocks
Beyond Simple Gates

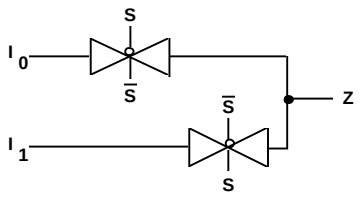
Kinds of "Non-gate logic":

- switching circuits built from CMOS transmission gates
- multiplexer/selecter functions
- decoders
- tri-state and open collector gates
- read-only memories

Selection Function/Demultiplexer Function with Transmission Gates

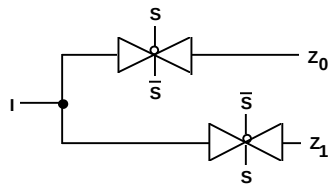
Selector:

Choose I_0 if S = 0
Choose I_1 if S = 1



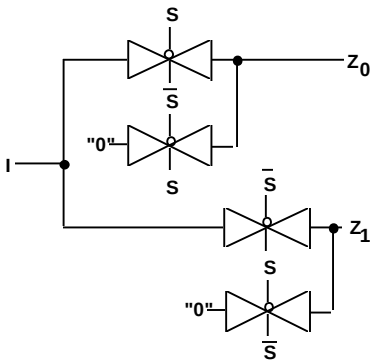
Demultiplexer:

I to Z_0 if S = 0
I to Z_1 if S = 1



Well-formed Switching Networks

Problem with the Demux implementation:
multiple outputs, but only one connected to the input!

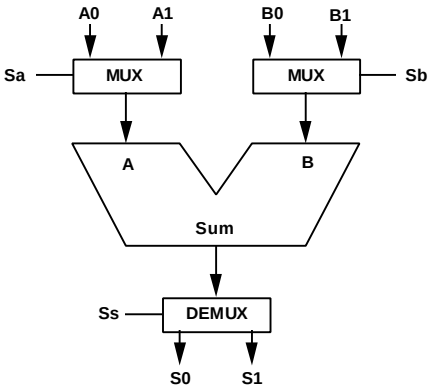


The fix: additional logic to drive every output to a known value

Never allow outputs to "float"

Multiplexers/Selectors

Multi-point connections



Multiple input sources

Multiple output destinations

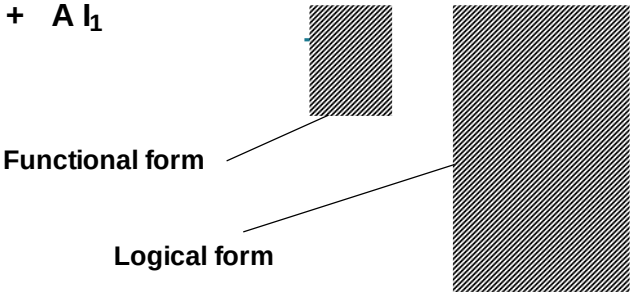
General Concept

2^n data inputs, n control inputs, 1 output

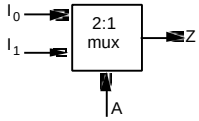
used to connect 2^n points to a single point

control signal pattern form binary index of input connected to output

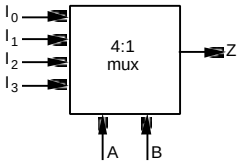
$$Z = A' I_0 + A I_1$$



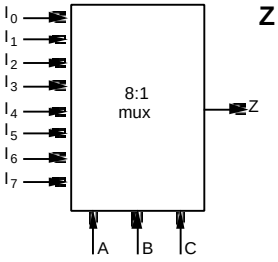
Two alternative forms for a 2:1 Mux Truth Table



$$Z = A' I_0 + A I_1$$



$$Z = A' B' I_0 + A' B I_1 + A B' I_2 + A B I_3$$

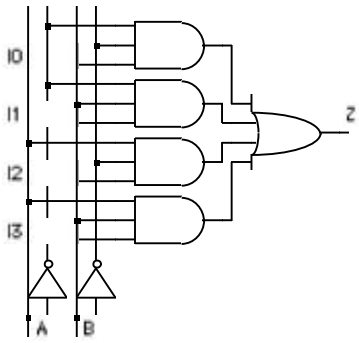


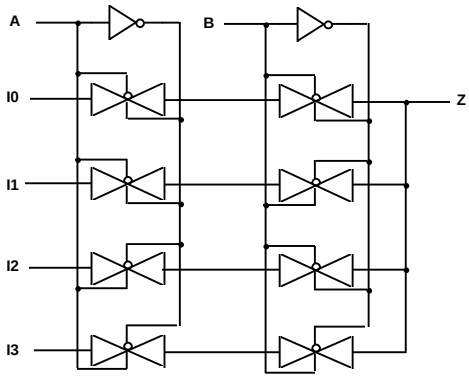
$$Z = A' B' C' I_0 + A' B' C I_1 + A' B C' I_2 + A' B C I_3 + A B' C' I_4 + A B' C I_5 + A B C' I_6 + A B C I_7$$

In general, $Z = S_{k=0}^{2^n-1} m_k I_k$

in minterm shorthand form for a 2^n :1 Mux

Alternative Implementations





Gate Level Implementation of 4:1 Mux

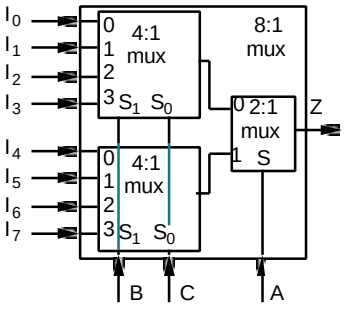
thirty six transistors

Transmission Gate Implementation of 4:1 Mux

twenty transistors

No. 4-19

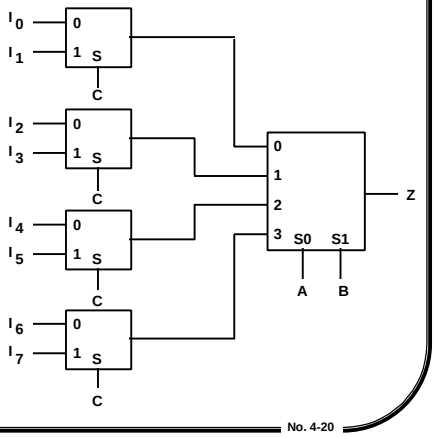
Large multiplexers can be implemented by cascaded smaller ones



Alternative 8:1 Mux Implementation

Control signals B and C simultaneously choose one of I_0-I_3 and I_4-I_7

Control signal A chooses which of the upper or lower MUX's output to gate to Z



No. 4-20

Multiplexers/selectors as a general purpose logic block

2^{n-1} :1 multiplexer can implement any function of n variables

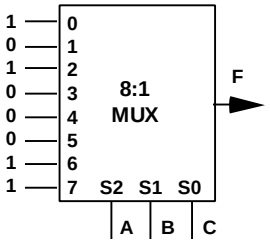
n-1 control variables; remaining variable is a data input to the mux

Example:

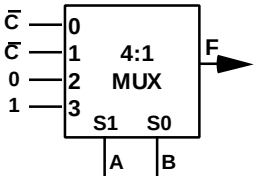
F(A,B,C) = m0 + m2 + m6 + m7

= A' B' C' + A' B C' + A B C' + A B C

= A' B' (C') + A' B (C') + A B' (0) + A B (1)

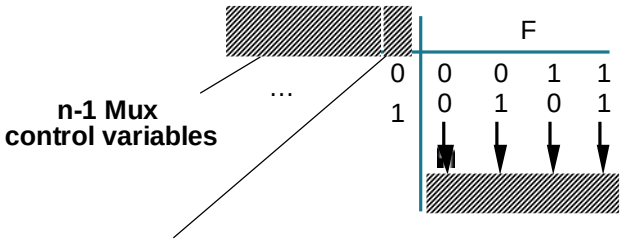


A	B	C	F	
0	0	0	1	$\bar{C}$
0	0	1	0	
0	1	0	1	$\bar{C}$
0	1	1	0	
1	0	0	0	0
1	0	1	0	
1	1	0	1	1
1	1	1	1	



"Lookup Table"

Generalization

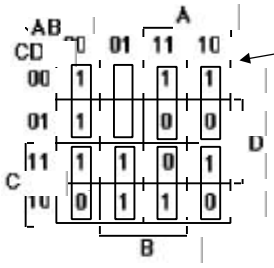


Four possible configurations of the truth table rows

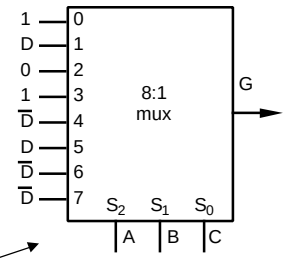
Can be expressed as a function of In, 0, 1

Example:

G(A,B,C,D) can be implemented by an 8:1 MUX:



K-map
Choose A,B,C
as control variables



Multiplexer
Implementation

TTL package efficient
May be gate inefficient

Decoders/Demultiplexers

Decoder*:* single data input, **n** control inputs, $\mathbf{2^n}$ outputs

- control inputs (called **select S**) represent Binary index of output to which the input is connected
- data input usually called **"enable" (G)**

1:2 Decoder:

$O0 = G \cdot \overline{S}$; $O1 = G \cdot S$

2:4 Decoder:

$O0 = G \cdot \overline{S0} \cdot \overline{S1}$

$O1 = G \cdot \overline{S0} \cdot S1$

$O2 = G \cdot S0 \cdot \overline{S1}$

$O3 = G \cdot S0 \cdot S1$

3:8 Decoder:

$O0 = G \cdot \overline{S0} \cdot \overline{S1} \cdot \overline{S2}$

$O1 = G \cdot \overline{S0} \cdot \overline{S1} \cdot S2$

$O2 = G \cdot \overline{S0} \cdot S1 \cdot \overline{S2}$

$O3 = G \cdot \overline{S0} \cdot S1 \cdot S2$

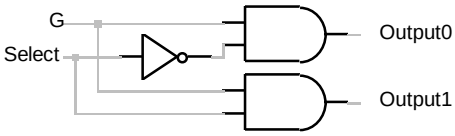
$O4 = G \cdot S0 \cdot \overline{S1} \cdot \overline{S2}$

$O5 = G \cdot S0 \cdot \overline{S1} \cdot S2$

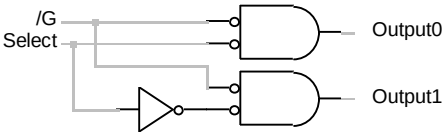
$O6 = G \cdot S0 \cdot S1 \cdot \overline{S2}$

$O7 = G \cdot S0 \cdot S1 \cdot S2$

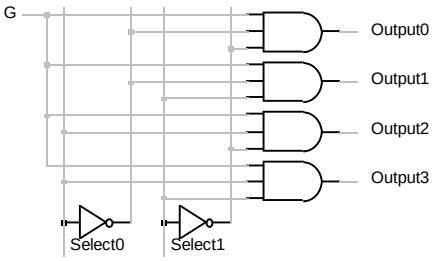
Alternative Implementations



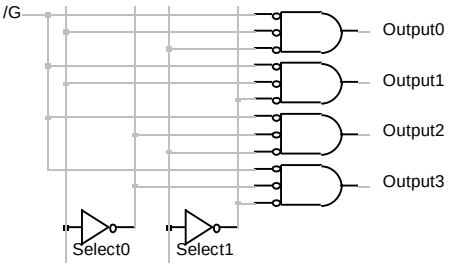
1:2 Decoder, Active High Enable



1:2 Decoder, Active Low Enable



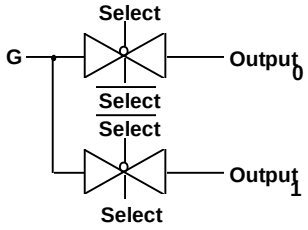
2:4 Decoder, Active High Enable



2:4 Decoder, Active Low Enable

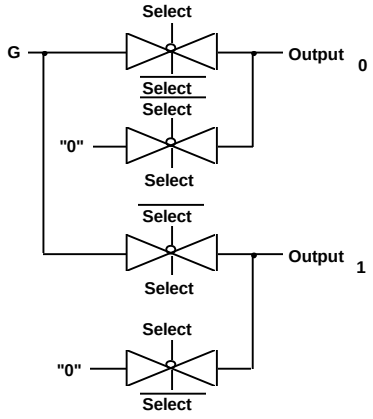
No. 4-25

Switch Logic Implementations



Naive, Incorrect Implementation

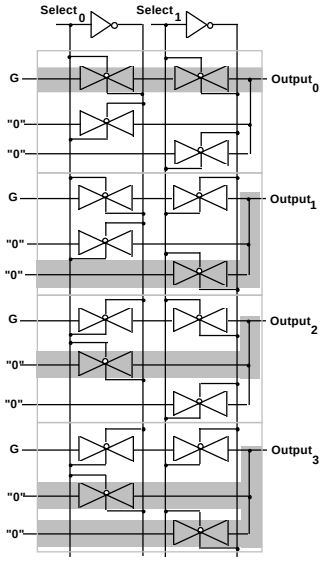
All outputs not driven at all times



Correct 1:2 Decoder Implementation

No. 4-26

Switch Implementation of 2:4 Decoder



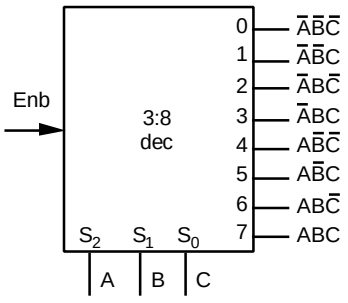
Operation of 2:4 Decoder

S0 = 0, S1 = 0

one straight thru path

three diagonal paths

Decoder as a Logic Building Block



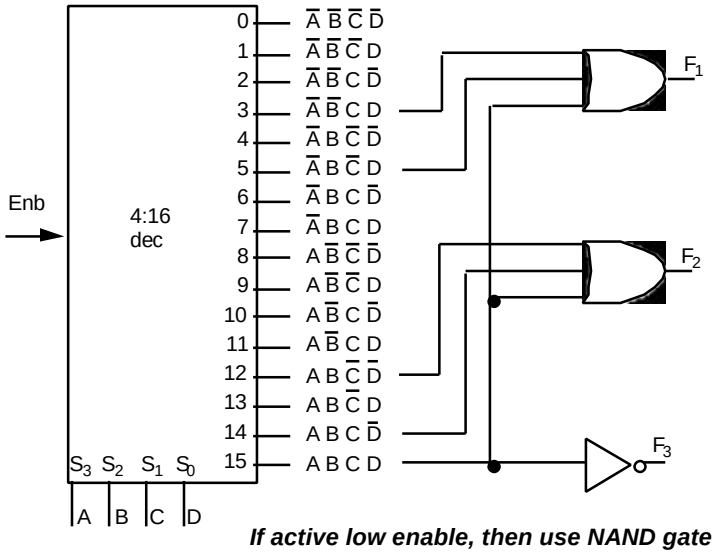
Decoder Generates Appropriate Minterm based on Control Signals

Example Function:

F1 = A' B C' D + A' B' C D + A B C D
F2 = A B C' D' + A B C
F3 = (A' + B' + C' + D')

Decoder/Demultiplexer

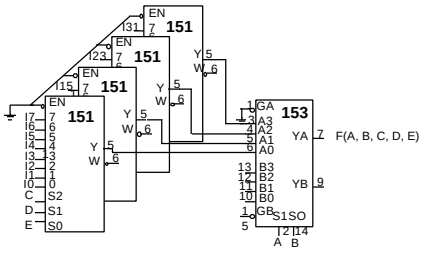
Decoder as a Logic Building Block



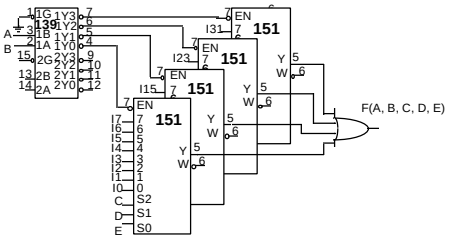
If active low enable, then use NAND gates!

Multiplexers/Decoders

Alternative Implementations of 32:1 Mux



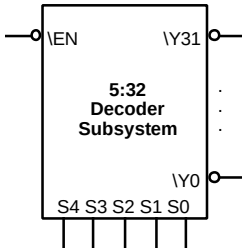
Multiplexer Only

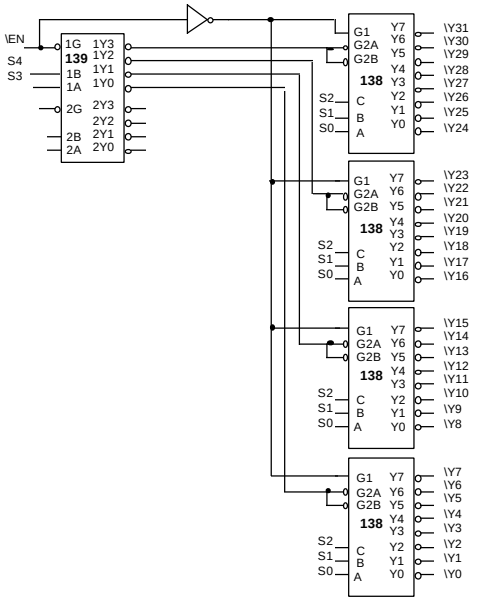


Multiplexer + Decoder

Multiplexers/Decoders

5:32 Decoder





Tri-State and Open-Collector

The Third State

Logic States: "0", "1"
Don't Care/Don't Know State: "X" (must be some value in real circuit!)

Third State: "Z" — high impedance — infinite resistance, no connection

Tri-state gates: **output values are "0", "1", and "Z"**
additional input: output enable (OE)

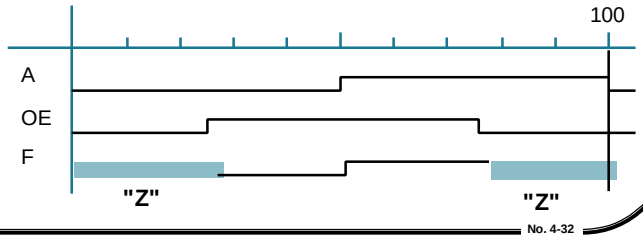
A	OE	F
X	0	Z
0	1	0
1	1	1

When OE is high, this gate is a non-inverting "buffer"

When $\overline{OE}$ is low, it is as though the gate was *disconnected* from the output!

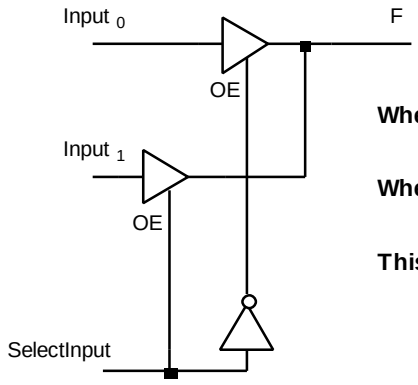
This allows more than one gate to be connected to the same output wire, as long as only one has its output enabled *at the same time*

Non-inverting buffer's timing waveform



Tri-state and Open Collector

Using tri-state gates to implement an economical multiplexer:



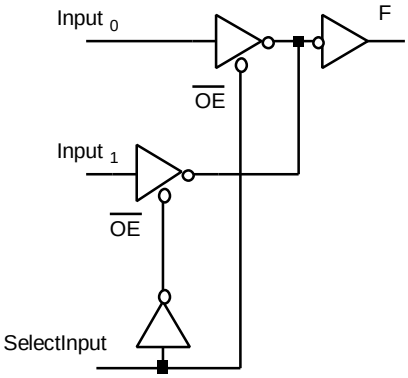
When SelectInput is asserted high
Input1 is connected to F

When SelectInput is driven low
Input0 is connected to F

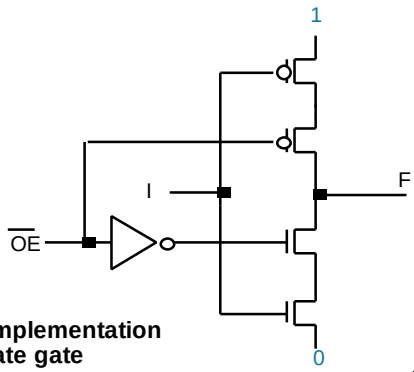
This is essentially a 2:1 Mux

Tri-state and Open Collector

Alternative Tri-state Fragment



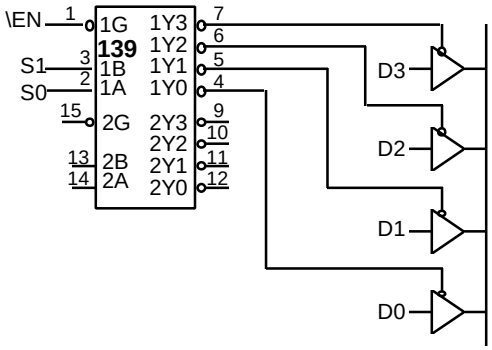
Active low tri-state enables
plus inverting tri-state buffers



Switch Level Implementation
of tri-state gate

Tri-State and Open Collector

4:1 Multiplexer, Revisited



Decoder + 4 tri-state Gates

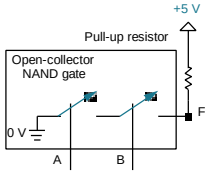
Tri-State and Open Collector

Open Collector

another way to connect multiple gates to the same output wire

gate only has the ability to pull its output low; it cannot actively drive the wire high

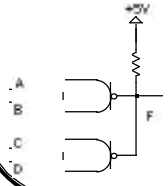
this is done by pulling the wire up to a logic 1 voltage through a resistor

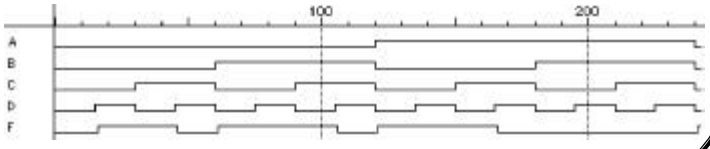


OC NAND gates

Wired AND:

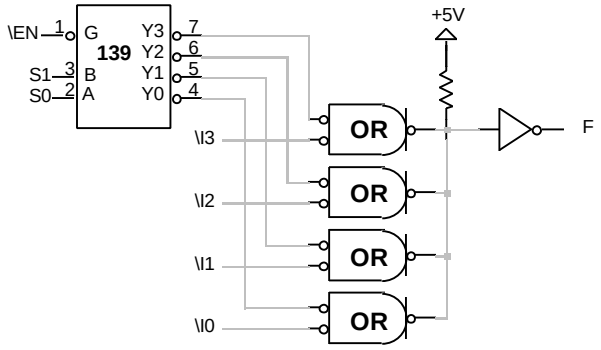
If A and B are "1", output is actively pulled low
if C and D are "1", output is actively pulled low
if one gate is low, the other high, then low wins
if both gates are "1", the output floats, pulled high by resistor
Hence, the two NAND functions are AND'd together!





Tri-State and Open Collector

4:1 Multiplexer



Decoder + 4 Open Collector Gates

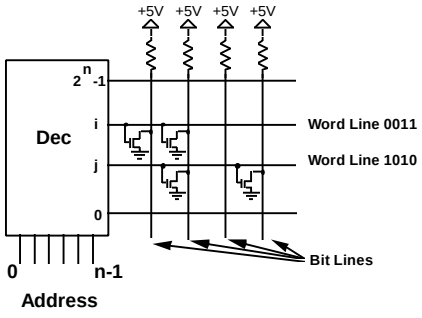
Read-Only Memories

ROM: Two dimensional array of 1's and 0's

Row is called a "word"; index is called an "address"

Width of row is called *bit-width* or *wordsize*

Address is input, selected word is output



Internal Organization

Read-Only Memories

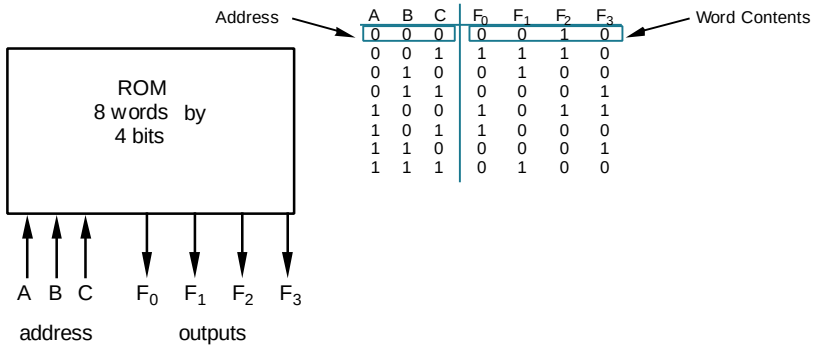
Example: Combination Logic Implementation

F0 = A' B' C + A B' C' + A B' C

F1 = A' B' C + A' B C' + A B C

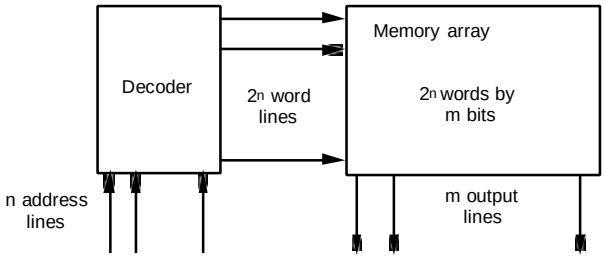
F2 = A' B' C' + A' B' C + A B' C'

F3 = A' B C + A B' C' + A B C'



Read-Only Memories

Not unlike a PLA structure with a fully decoded AND array!



ROM vs. PLA:

ROM approach advantageous when

(1) design time is short (no need to minimize output functions)
(2) most input combinations are needed (e.g., code converters)
(3) little sharing of product terms among output functions

ROM problem: size doubles for each additional input, can't use don't cares

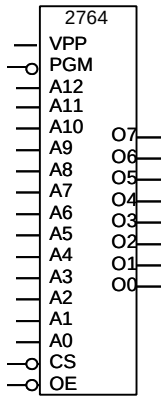
PLA approach advantangeous when

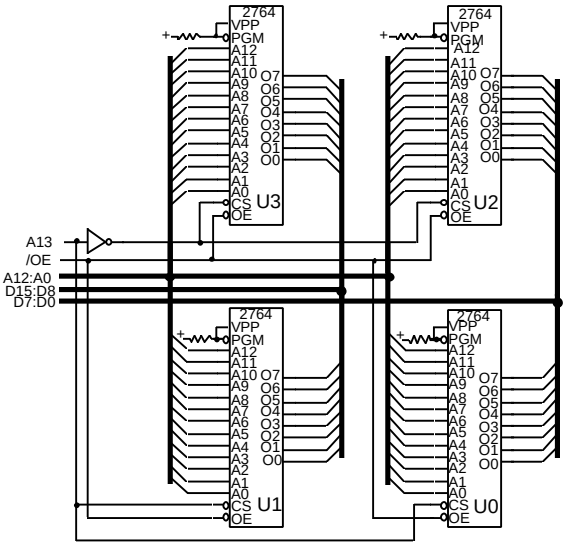
(1) design tool like espresso is available
(2) there are relatively few unique minterm combinations
(3) many minterms are shared among the output functions

PAL problem: constrained fan-ins on OR planes

Read-Only Memories

2764 EPROM
8K x 8





16K x 16
Subsystem

No. 4-41

Combinational Logic Word Problems

General Design Procedure

1. **Understand the Problem**
   - **what is the circuit supposed to do?**
   - **write down inputs (data, control) and outputs**
   - **draw block diagram or other picture**

2. **Formulate the Problem in terms of a truth table or other suitable design representation**
   - **truth table or waveform diagram**

3. **Choose Implementation Target**
   - **ROM, PAL, PLA, Mux, Decoder + OR, Discrete Gates**

4. **Follow Implementation Procedure**
   - **K-maps, espresso, misII**

No. 4-42

Combinational Logic Word Problems

Process Line Control Problem

Statement of the Problem

Rods of varying length (+/-10%) travel on conveyor belt
Mechanical arm pushes rods within spec (+/-5%) to one side
Second arm pushes rods too long to other side
Rods too short stay on belt

3 light barriers (light source + photocell) as sensors

Design combinational logic to activate the arms

Understanding the Problem

Inputs are three sensors, outputs are two arm control signals

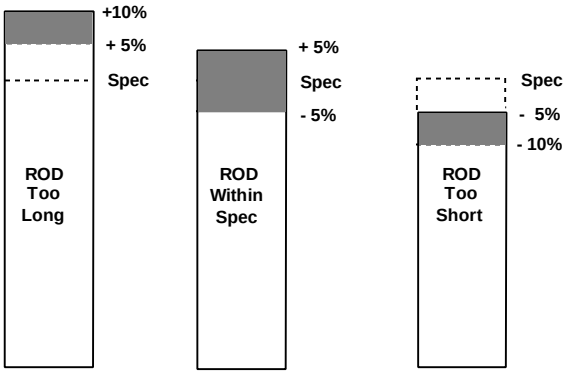
Assume sensor reads "1" when tripped, "0" otherwise

Call sensors A, B, C

Draw a picture!

Combinational Logic Word Problems

Process Control Problem

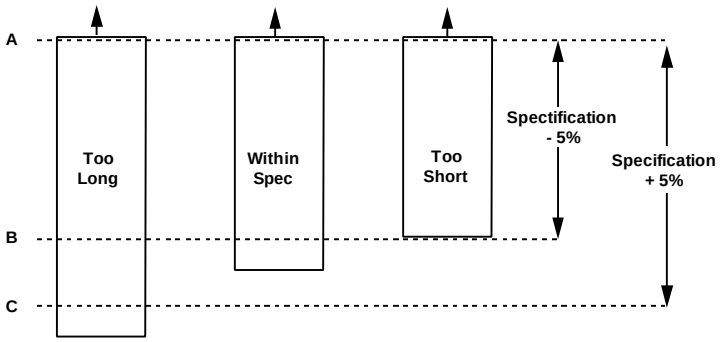


Where to place the light sensors A, B, and C to distinguish among the three cases?

Assume that A detects the leading edge of the rod on the conveyor

Combinational Logic Word Problems

Process Control Problem



A to B distance place apart at specification - 5%

A to C distance placed apart at specification +5%

Combinational Logic Word Problems

Process Control Problem

A	B	C	Function
0	0	0	X
0	0	1	X
0	1	0	X
0	1	1	X
1	0	0	too short
1	0	1	X
1	1	0	in spec
1	1	1	too long

Truth table and logic implementation now straightforward

"too long" = A B C
(all three sensors tripped)

"in spec" = A B C'
(first two sensors tripped)

Combinational Logic Word Problems

BCD to 7 Segment Display Controller

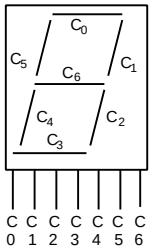
Understanding the problem:

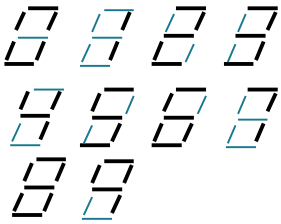
input is a 4 bit bcd digit

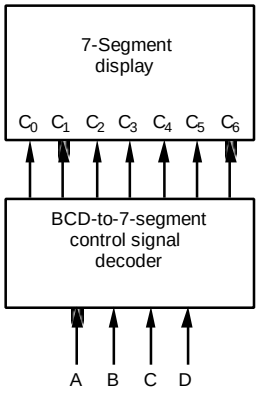
output is the control signals for the display

4 inputs A, B, C, D

7 outputs C0 — C6







Block Diagram

Combinational Logic Word Problems

BCD to 7 Segment Display Controller

A	B	C	D	C0	C1	C2	C3	C4	C5	C6
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1	1
1	0	1	0	X	X	X	X	X	X	X
1	0	1	1	X	X	X	X	X	X	X
1	1	0	0	X	X	X	X	X	X	X
1	1	0	1	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X

Formulate the problem in terms of a truth table

Choose implementation target:

if ROM, we are done

don't cares imply PAL/PLA may be attractive

Follow implementation procedure:

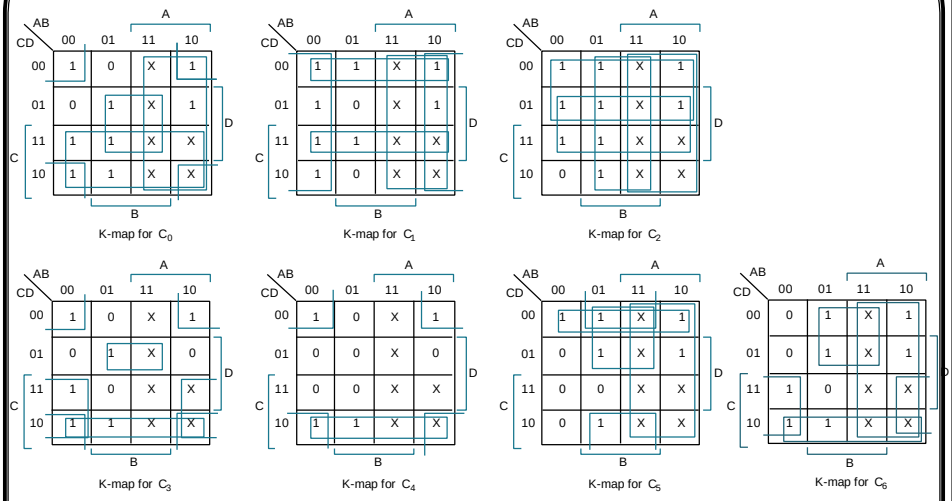
hand reduced K-maps

vs.

espresso

Combinational Logic Word Problems

BCD to 7 Segment Display Controller



C0 = A + B D + C + B' D'
C1 = A + C' D' + C D + B'
C2 = A + B + C' + D

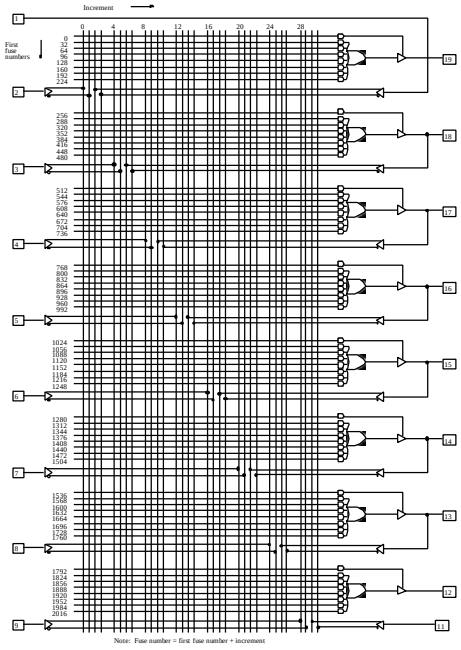
14 Unique Product Terms

C3 = B' D' + C D' + B C' D + B' C
C4 = B' D' + C D
C5 = A + C' D' + B D' + B C'
C6 = A + C D' + B C' + B' C

Combinational Logic Word Problems

BCD to 7 Segment Display Controller

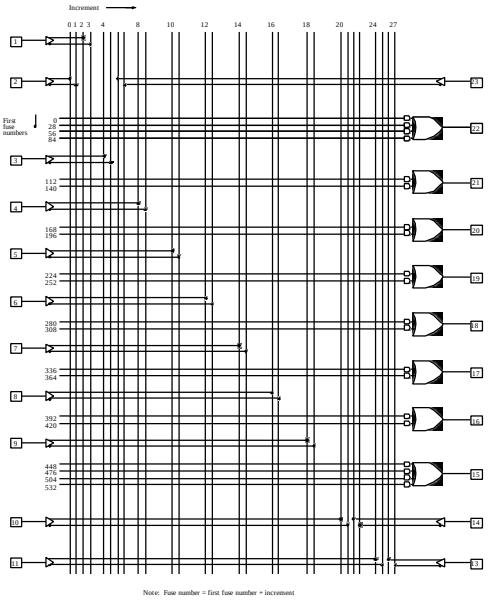
16H8PAL
Can Implement
the function



Combinational Logic Word Problems

BCD to 7 Segment Display Controller

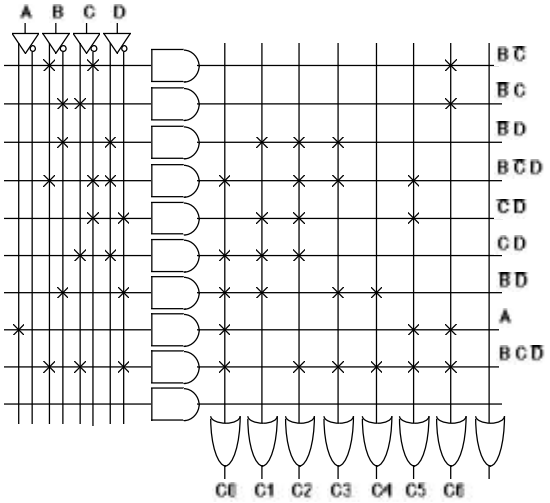
14H8PAL
Cannot Implement the function



Combinational Logic Word Problems

BCD to 7 Segment Display Controller

PLA Implementation



Combinational Logic Word Problems

BCD to7 Segment Display Controller

Multilevel Implementation

X = C' + D'

Y = B' C'

C0 = C3 + A' B X' + A D Y

C1 = Y + A' C5' + C' D' C6

C2 = C5 + A' B' D + A' C D

C3 = C4 + B D C5 + A' B' X'

C4 = D' Y + A' C D'

C5 = C' C4 + A Y + A' B X

C6 = A C4 + C C5 + C4' C5 + A' B' C

52 literals

33 gates

Ineffective use of don't cares

Combinational Logic Word Problems

Logical Function Unit

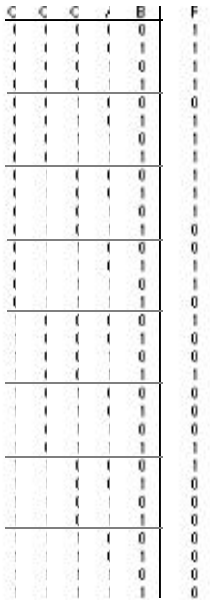
Statement of the Problem:

C	C	C2	F	Comments
0	0	0	1	always 1
0	0	1	A + B	logical or
0	1	0	$\overline{A \cdot B}$	logical nand
0	1	1	A xor B	logical xor
1	0	0	A xnorB	logical xnor
1	0	1	A·B	logical and
1	1	0	$\overline{A + B}$	logical nor
1	1	1	0	always 0

3 control inputs: C0, C1, C2
2 data inputs: A, B
1 output: F

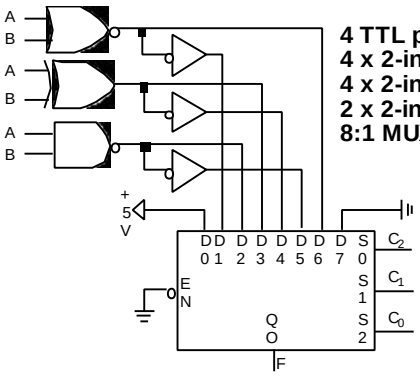
Combinational Logic Word Problems

Logical Function Unit



Formulate as a truth table

Choose implementation technology
- **5-variable K-map**
- **espresso**
- **multiplexor implementation**

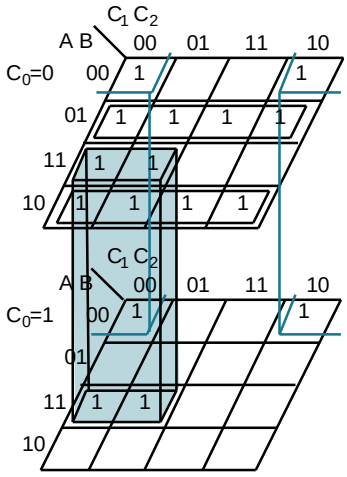


4 TTL packages:
4 x 2-input NAND
4 x 2-input NOR
2 x 2-input XOR
8:1 MUX

Combinational Logic Word Problems

Logical Function Unit

Follow implementation procedure



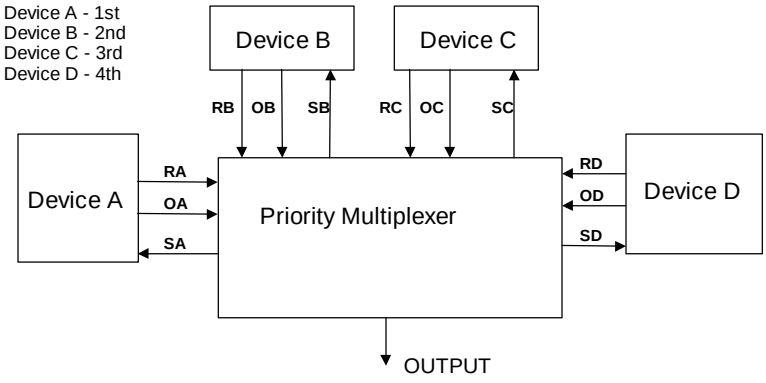
F = C2' A' B' + C0' A B'
+ C0' A' B + C1' A B

5 gates, 5 inverters

Also four packages:
4 x 3-input NAND
1 x 4-input NAND

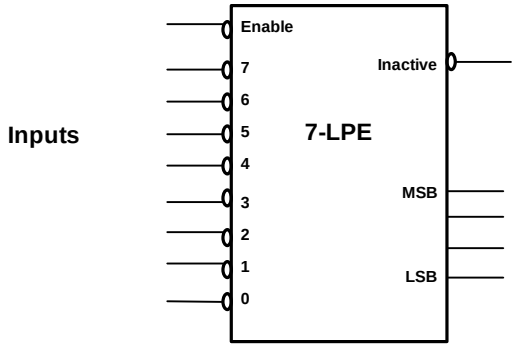
Alternative: 32 x 1-bit ROM

single package



The Priority Mux connects a device output (OA, OB, OC, OD) to the output line (OUTPUT) based on priorities. Each device has a request line (RA, RB, RC, RD) that is asserted when the system output line is requested by that device. The Mux must return a signal (SA, SB, SC, SD) to each device indicating whether the request was accepted. These signals should be asserted only if a request has been received from the device and the device output has been connected to the system output.

7-Level Priority Encoder



The output of this **Priority Encoder** represent the **binary code** of the highest **asserted** input at any point in time. The **Inactive** output indicates when no input is **asserted**.

Chapter Review

- ***Non-Simple Gate Logic Building Blocks:***
  - PALs/PLAs
  - Multiplexers/Selecters
  - Decoders
  - ROMs
  - Tri-state, Open Collector

- ***Combinational Word Problems:***
  - Understand the Problem
  - Formulate in terms of a Truth Table
  - Choose implementation technology
  - Implement by following the design procedure