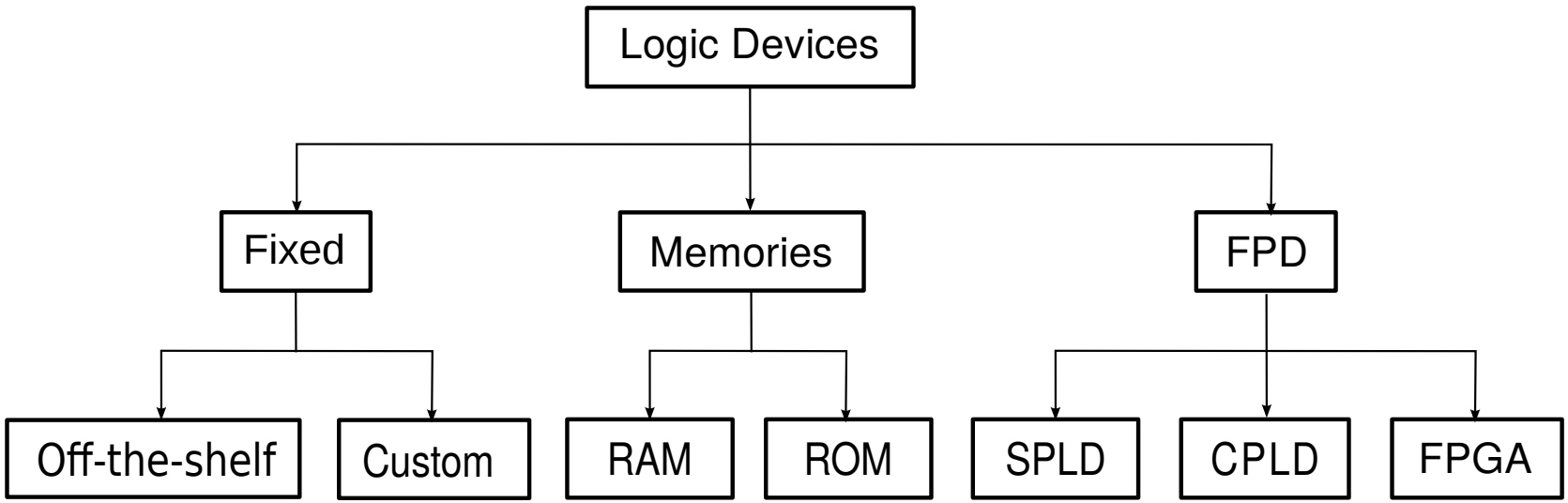
Chapter 3
Logic Implementations
SKEE2263 Digital Systems

Mun'im/Ismahani/Izam

{munim@utm.my,e-izam@utm.my,ismahani@fke.utm.my}

February 23, 2016

Taxonomy of Logic Devices



Logic Devices

Fixed – Functions defined during manufacture.

1. Off-the-shelf – designer can order from a catalog
2. Custom – designed for one customer, made by semiconductor foundry

Memory – Holds big amounts of data. Separate fabrication to get highest density

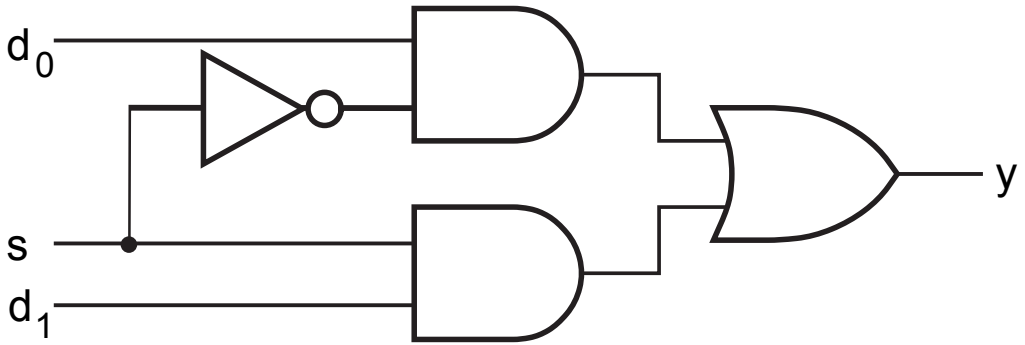
1. RAM – random access memory
2. ROM – read-only memory

Logic Devices

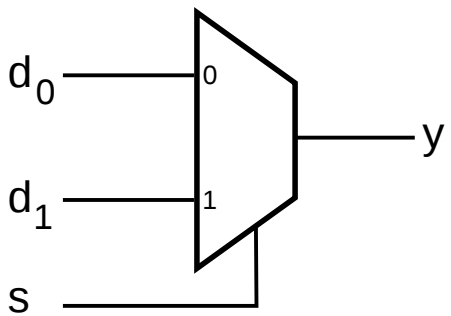
FPD – Field-Programmable Devices. Blank devices that must be configured/programmed before use. Also called Programmable Logic Devices (PLD)

1. SPLD – Simple PLD. A few hundred gates.
2. CPLD – Complex PLD. A few thousand gates. Fundamental technology same as SPLD.
3. FPGA – Field-Programmable Gate Array. A few million gates.

2:1 mux

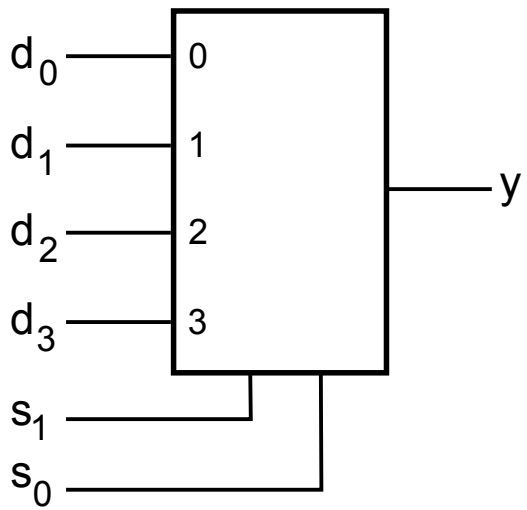


$$y = s'd_0 + sd_1$$



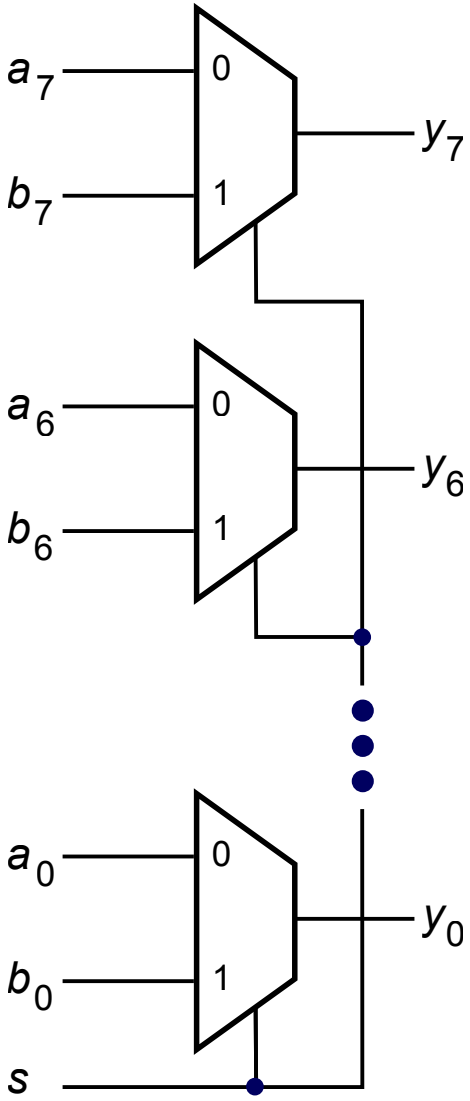
Distinctive symbol

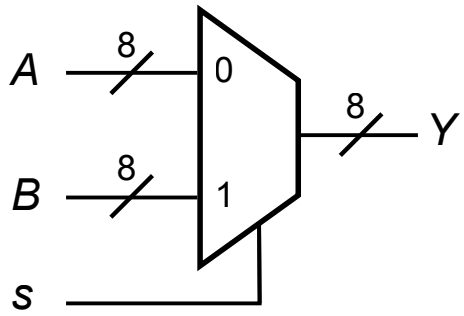
4:1 mux



$$y = s_1' s_0' d_0 + s_1' s_0 d_1 + s_1 s_0' d_2 + s_1 s_0 d_3$$

Word Multiplexer

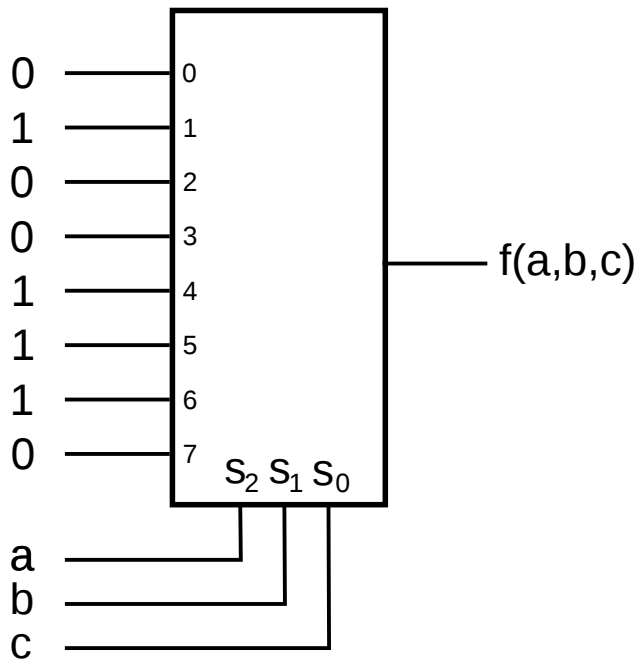




Mux Implementation of Logic Functions #1

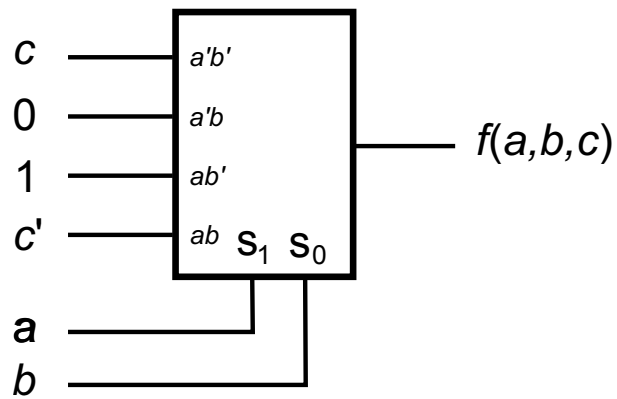
Direct input

a	b	c	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



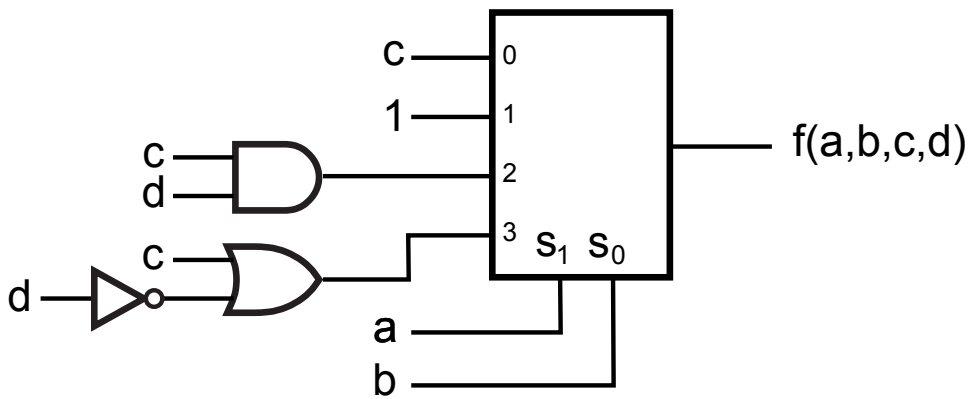
Mux Implementation of Logic Functions #2

a	b	c	f	
0	0	0	0	c
0	0	1	1	
0	1	0	0	0
0	1	1	0	
1	0	0	1	1
1	0	1	1	
1	1	0	1	c'
1	1	1	0	



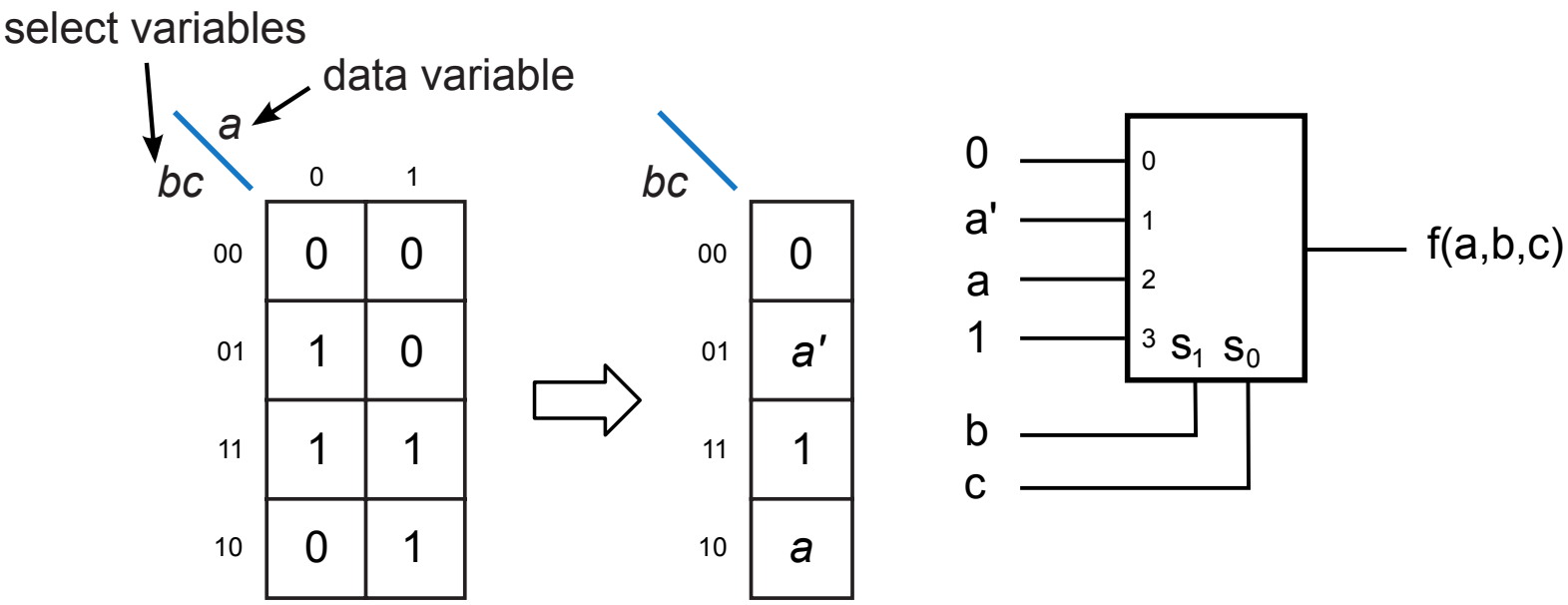
Mux Implementation of Logic Functions #3

a	b	c	d		f
0	0	0	0	0	c
0	0	0	1	0	
0	0	1	0	1	
0	0	1	1	1	
0	1	0	0	1	1
0	1	0	1	1	
0	1	1	0	1	
0	1	1	1	1	
1	0	0	0	0	cd
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	1	
1	1	0	0	1	$c + d'$
1	1	0	1	0	
1	1	1	0	1	
1	1	1	1	1	



Mux Implementation of Logic Functions #4

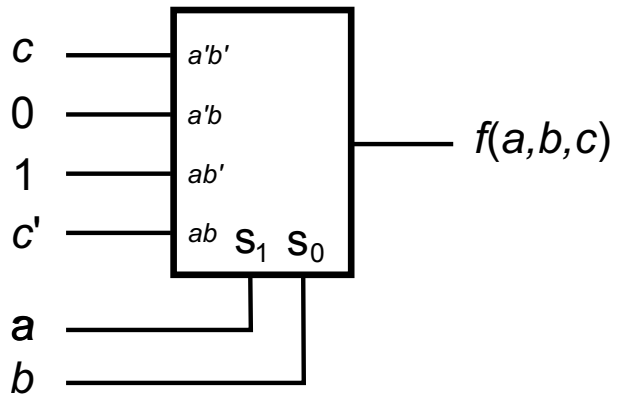
K-map



Mux Implementation of Logic Functions #5

Algebraic Manipulation

$$
\begin{aligned}
f(a,b,c) &= ab' + b'c + ac' \\
&= ab'(c + c') + b'c(a + a') + ac'(b + b') \\
&= ab'c + ab'c' + ab'c + a'b'c + abc' + a'b'c' \\
&= a'b'c + ab'c + ab'c' + abc' \\
&= a'b'c + ab'(c + c') + abc' \\
&= a'b'(c) + a'b(0) + ab'(1) + ab(c')
\end{aligned}
$$

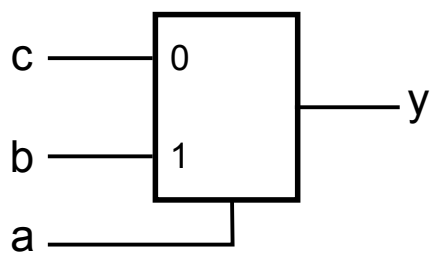


Mux Implementation of Logic Functions #6

Variable Substitution

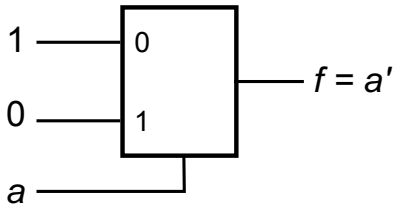
The output function of the 2-to-1 multiplexer is $y = s'd_0 + sd_1$
The expression $f(a, b, c) = a'c + ab$ fits the mux equation pattern.

$$\begin{array}{ccccccc} f & = & a' & c & + & a & b \\ \downarrow & & \downarrow & \downarrow & & \downarrow & \downarrow \\ y & = & s' & d_0 & + & s & d_1 \end{array}$$

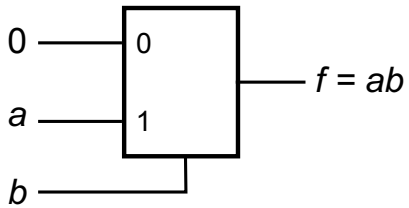


Mux Implementation of Logic Functions #7

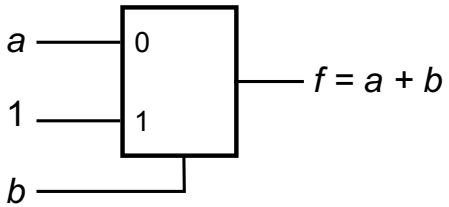
Implementing Basic Logic Gates



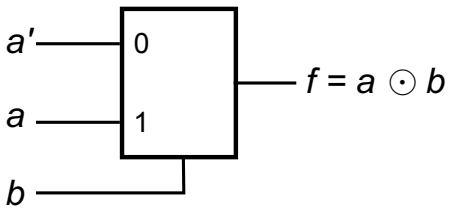
NOT



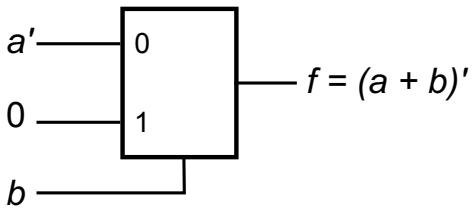
AND



OR



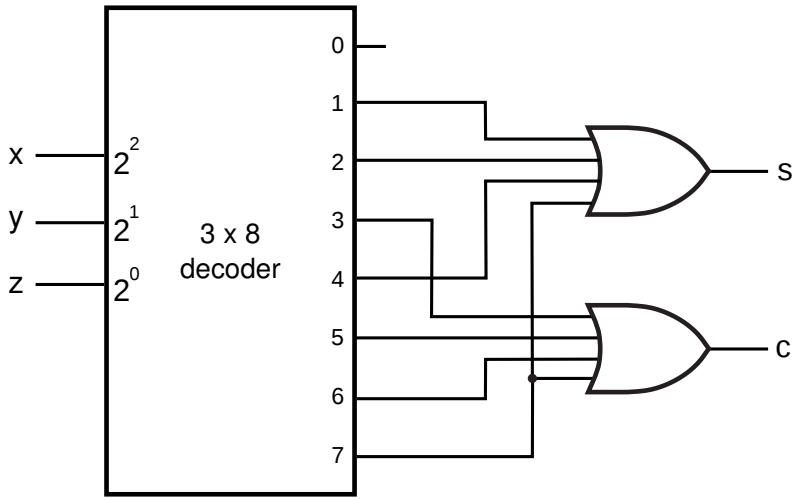
XNOR



NOR

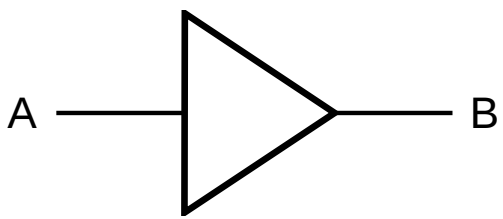
Decoders

$$s(x, y, z) = \sum m(1, 2, 4, 7)$$
$$c(x, y, z) = \sum m(3, 5, 6, 7)$$



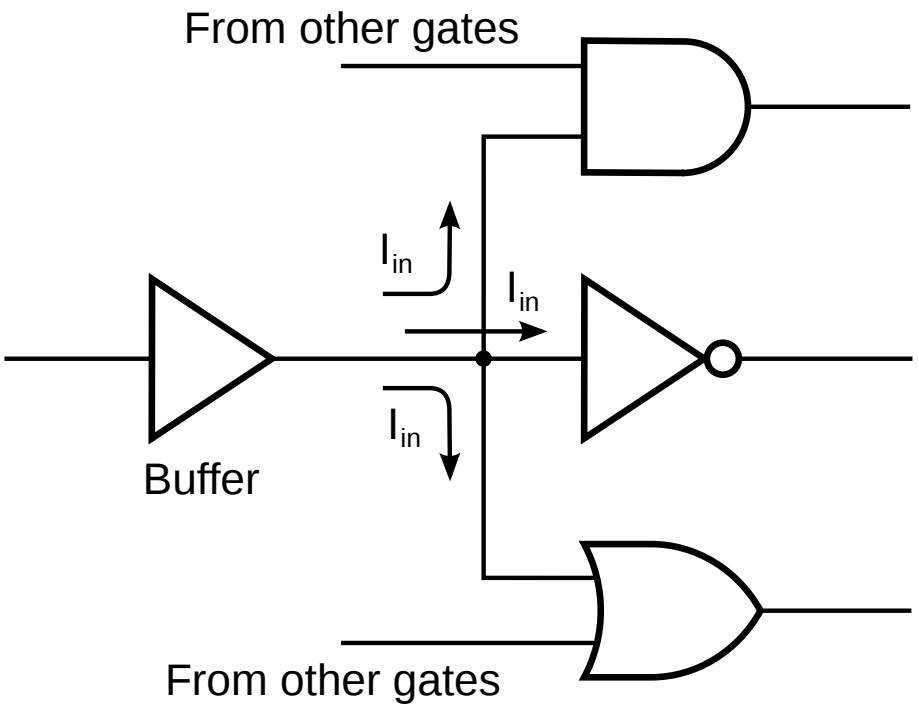
Buffer Truth Table

A	B
0	0
1	1



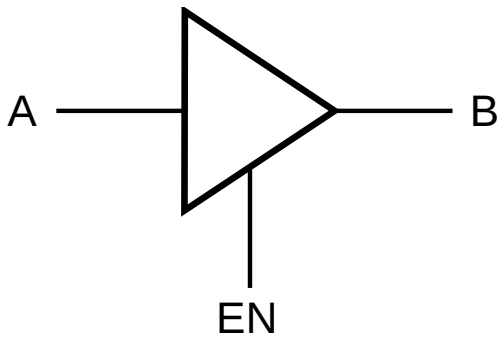
Buffer Application

Using buffer to increase current

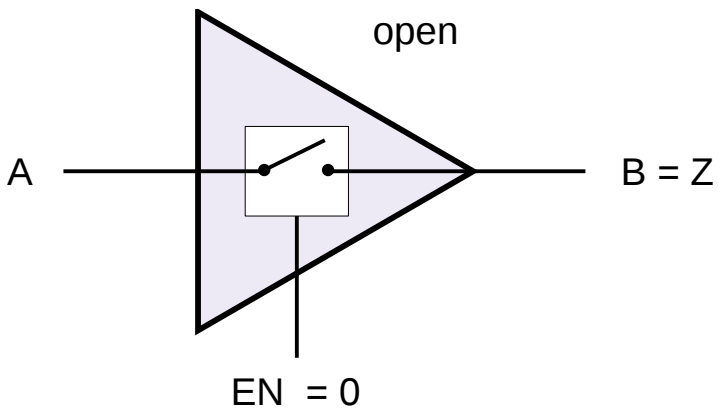


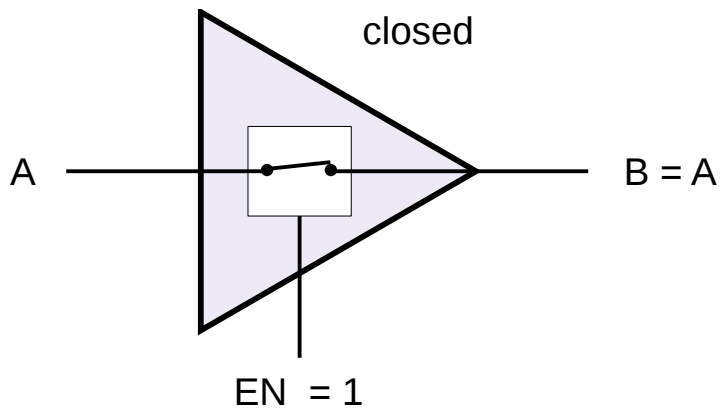
Tristate Buffer Truth Table

A	EN	B
0	1	0
1	1	1
×	0	Z

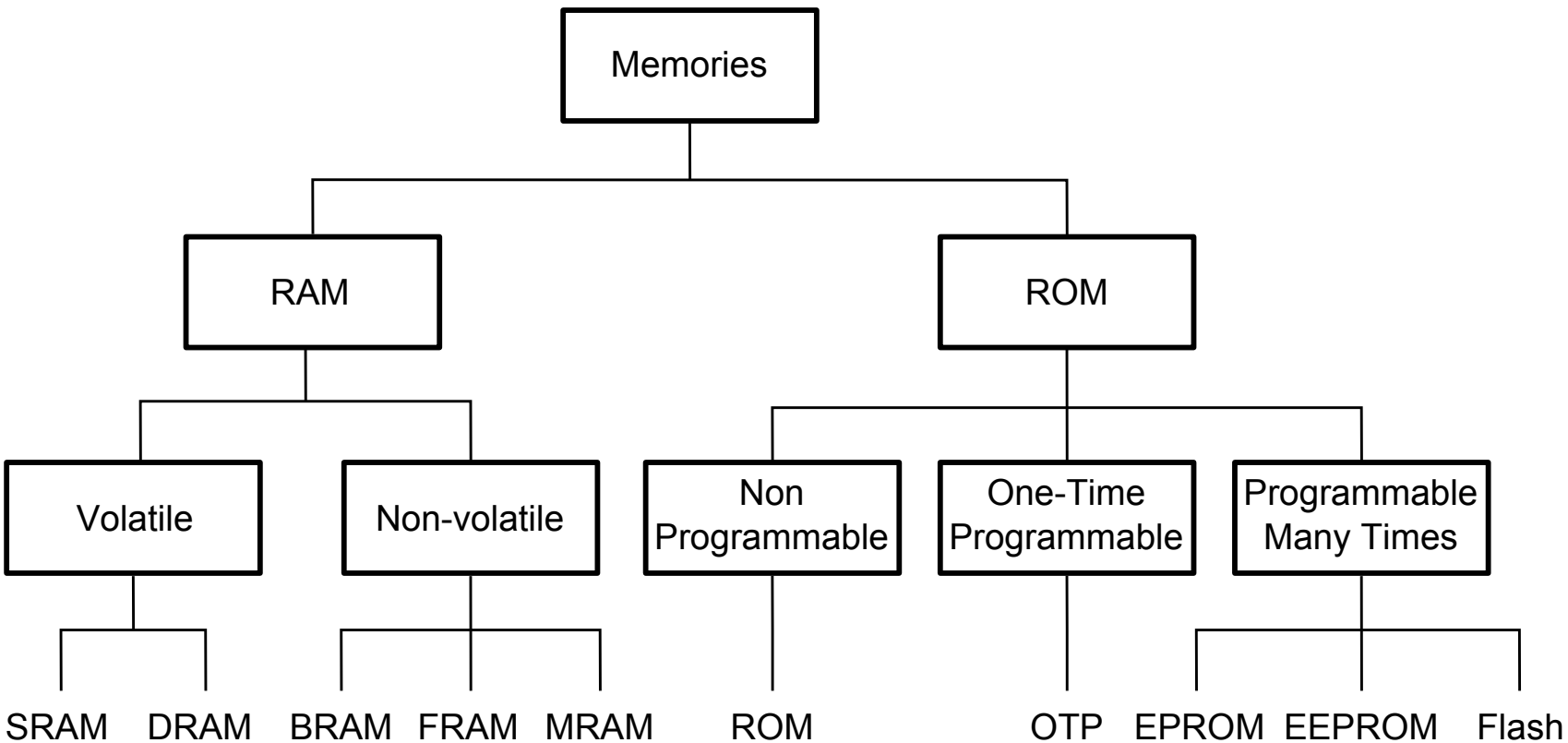


Tristate Buffer Equivalent Circuit

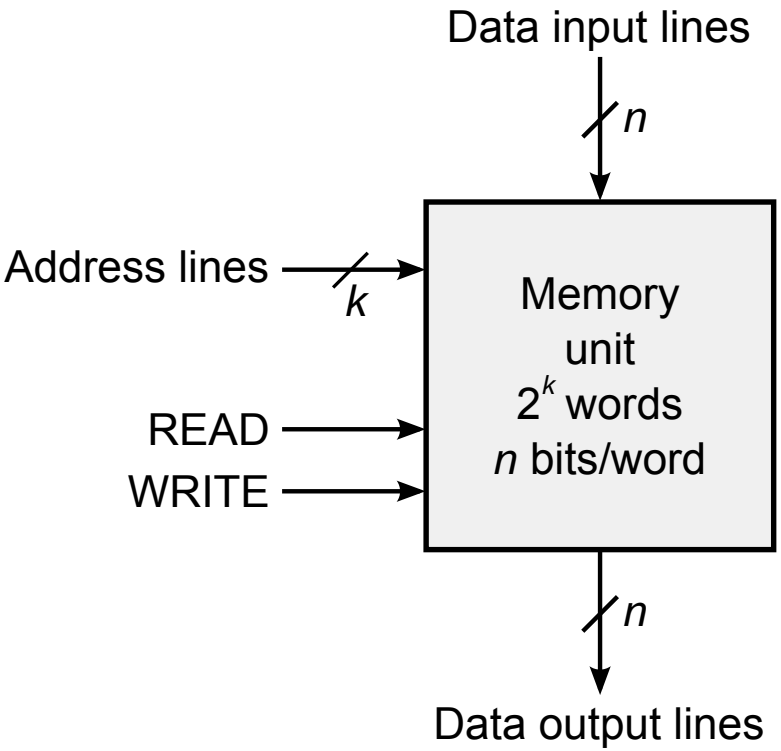




Memory Taxonomy



Simplified Memory Model



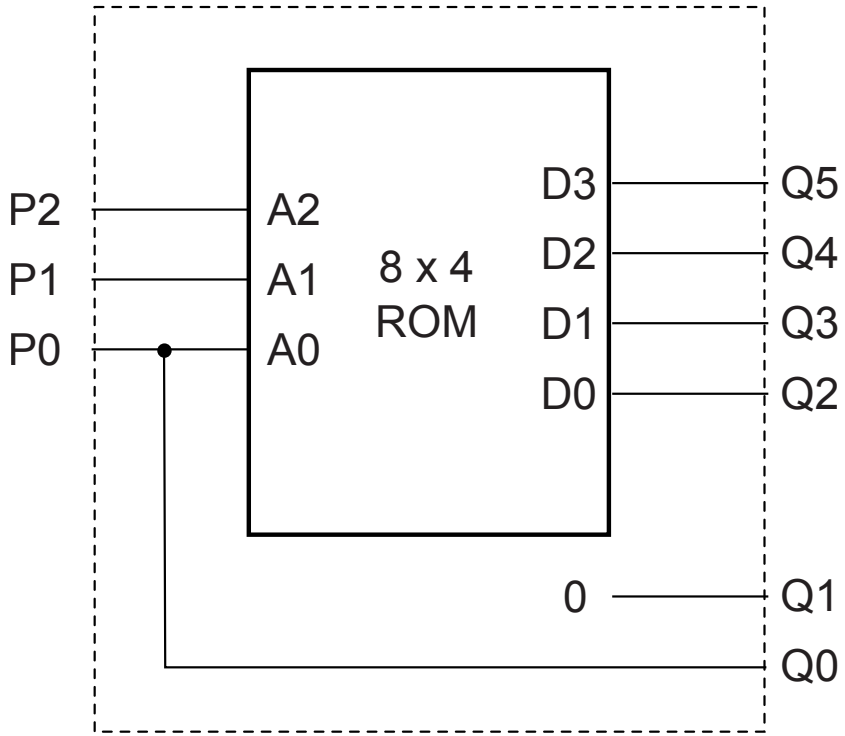
Terms:

- Organization
  = $2^k \times n$
- #locations (a.ka. entry, row or “word”)
  = 2^k
- index is called “address”
- #bits/location
  = word size
  = n

ROM-based Lookup Table (LUT)

Inputs			Outputs						
P2	**P1**	**P0**	**Q5**	**Q4**	**Q3**	**Q2**	**Q1**	**Q0**	**Decimal**
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

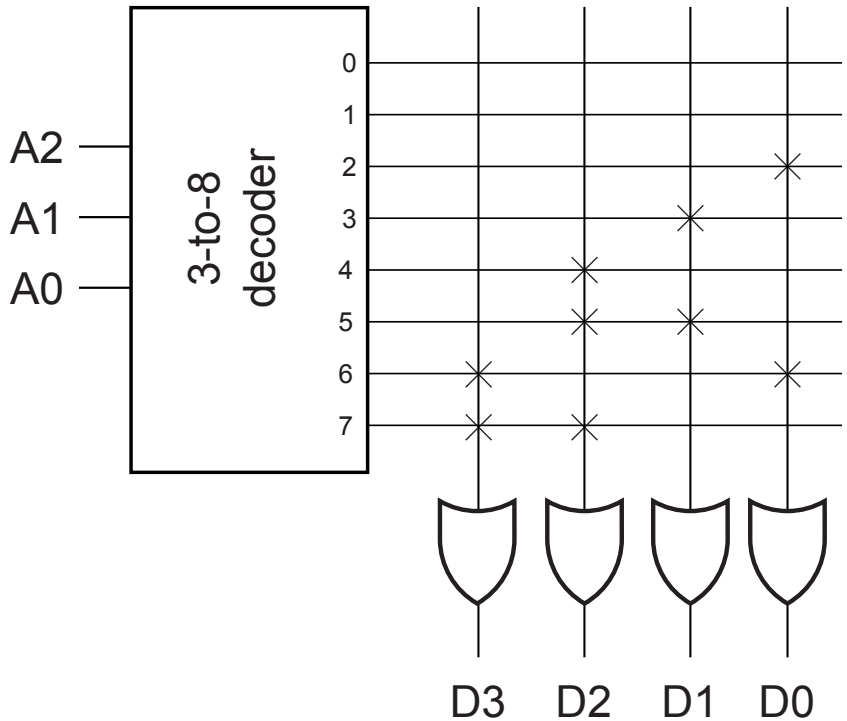
ROM-Based Squarer



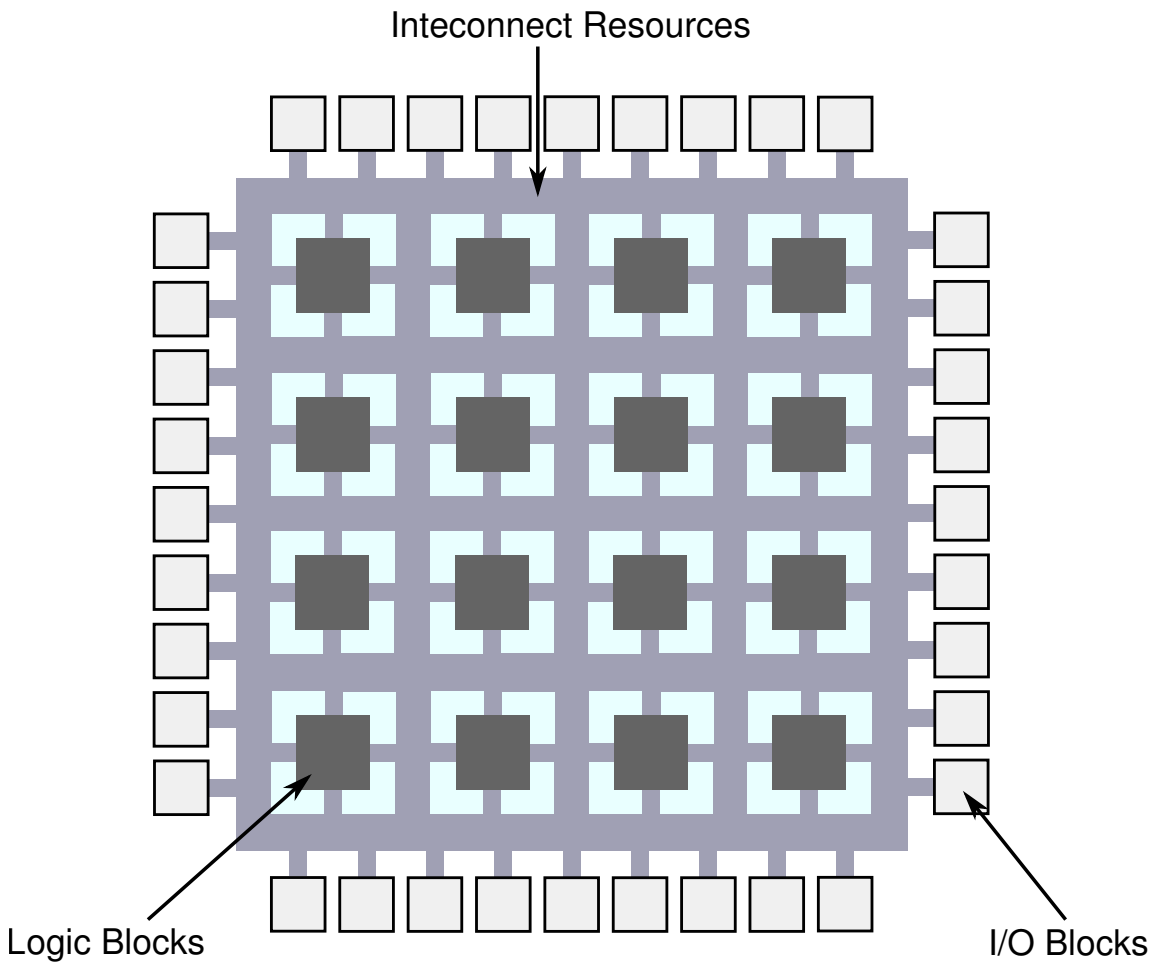
Squarer ROM Contents

Input			Output			
A2	**A1**	**A0**	**D3**	**D2**	**D1**	**D0**
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	0

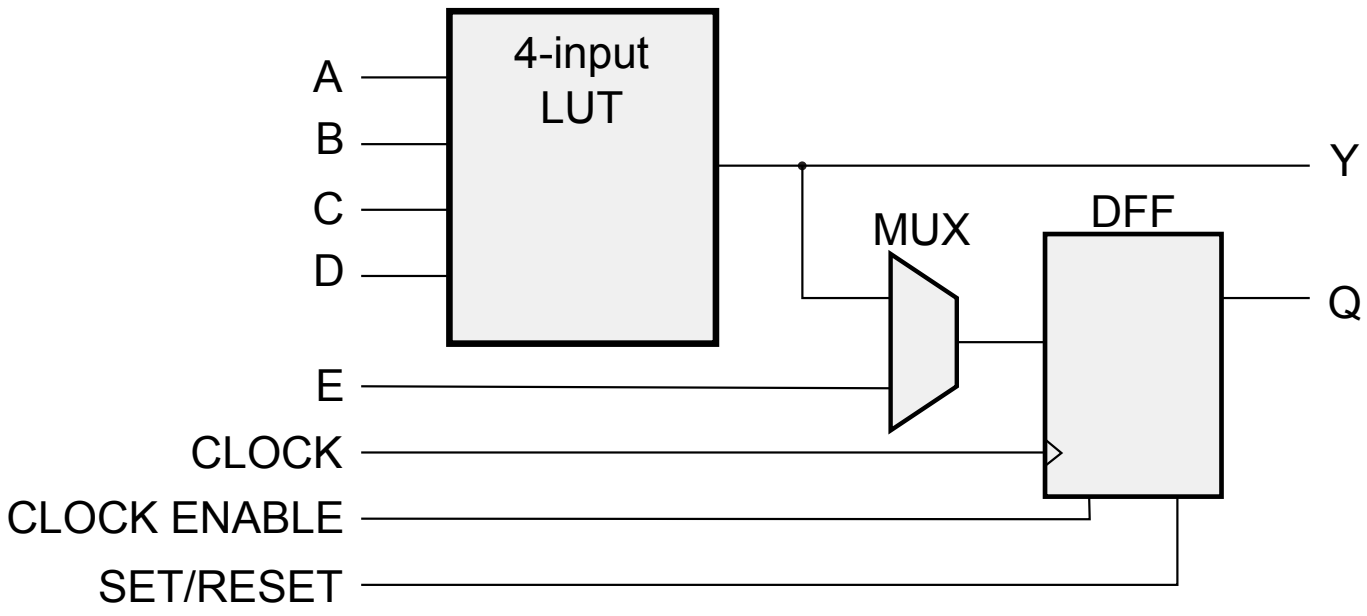
Squarer ROM Programming



Basic Architecture of FPGA



Basic Architecture of Logic Blocks

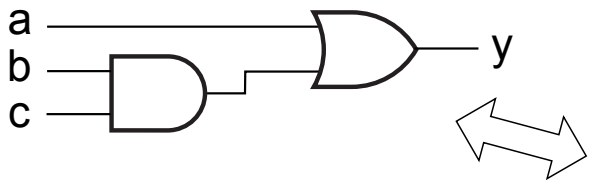


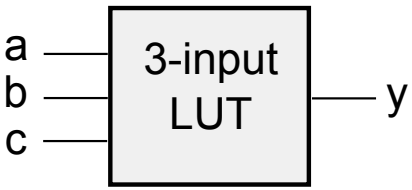
LB Lookup Tables

They are just muxes

Required function

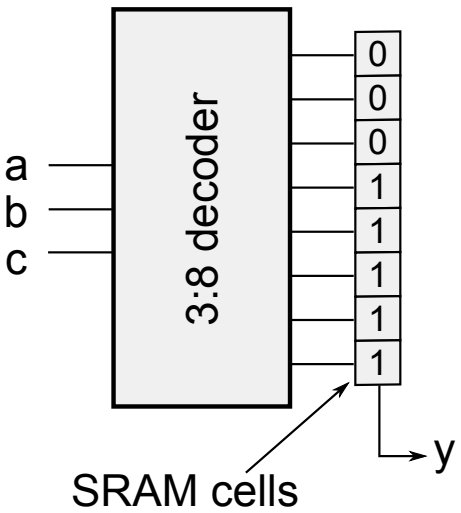
y = a | (b & c)



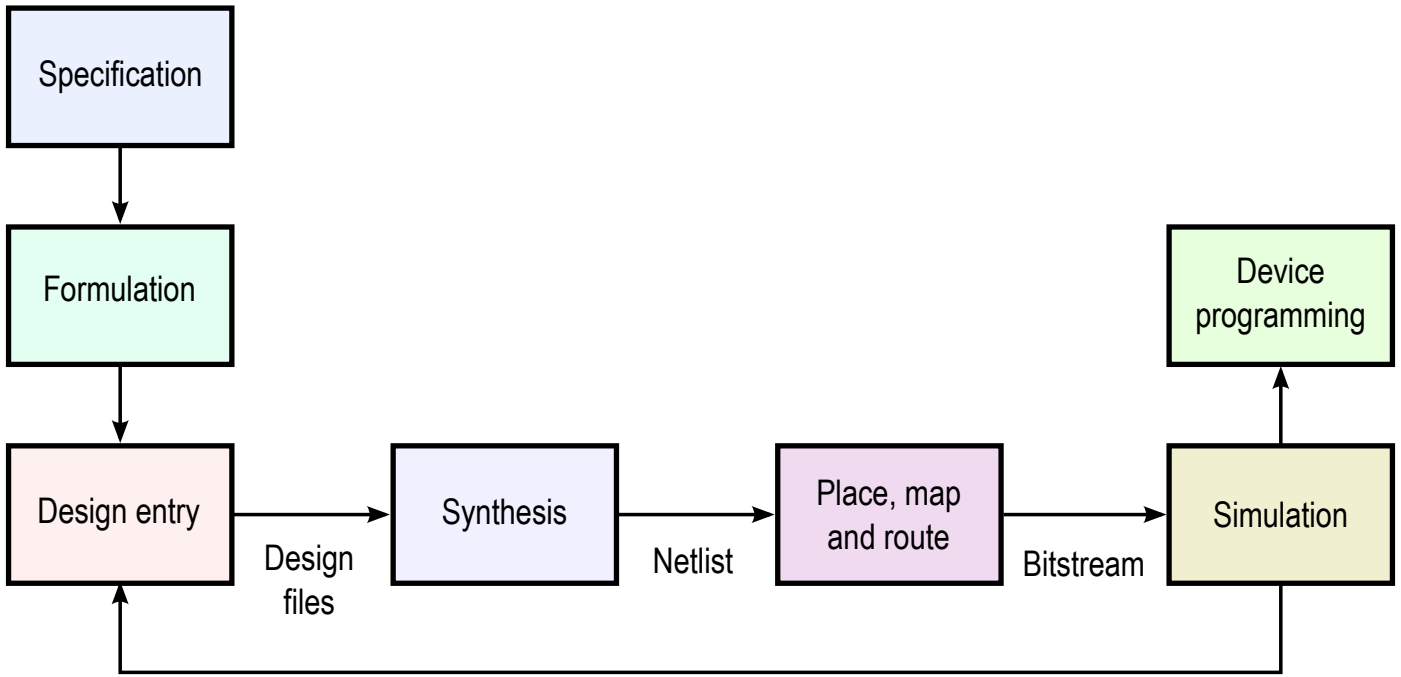


Truth table

a	b	c	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1



CPLD/FPGA Design Flow



Quartus handles both Altera CPLD and FPGA

- Simple designs → use schematic capture
- Complex designs → use hardware design language (HDL) e.g. VHDL or Verilog