Combinational Logic

Karnaugh maps

Building blocks: encoders, decoders, multiplexers

Abstraction!

But first…

Recall: *sum of products*

logical sum (OR)

of products (AND)

of inputs or their complements (NOT).

A	B	C	M
0	0	0	**0**
0	0	1	**0**
0	1	0	**0**
0	1	1	**1**
1	0	0	**0**
1	0	1	**1**
1	1	0	**1**
1	1	1	**1**

Construct with:

- 1 code detector per 1-valued output row
- 1 large OR of all code detector outputs

Is it minimal?

Gray Codes = reflected binary codes

Alternate binary encoding
designed for electromechanical switches and counting.

00	01	11	10
0	1	2	3

000	001	011	010	110	111	101	100
0	1	2	3	4	5	6	7

How many bits change when incrementing?

Karnaugh Maps: find (minimal) sums of products

ex

A	B	C	D	F(A, B, C, D)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

gray code order

CD

AB

	00	01	11	10
00	0	0	0	0
01	0	0	0	1
11	1	1	0	1
10	1	1	1	1

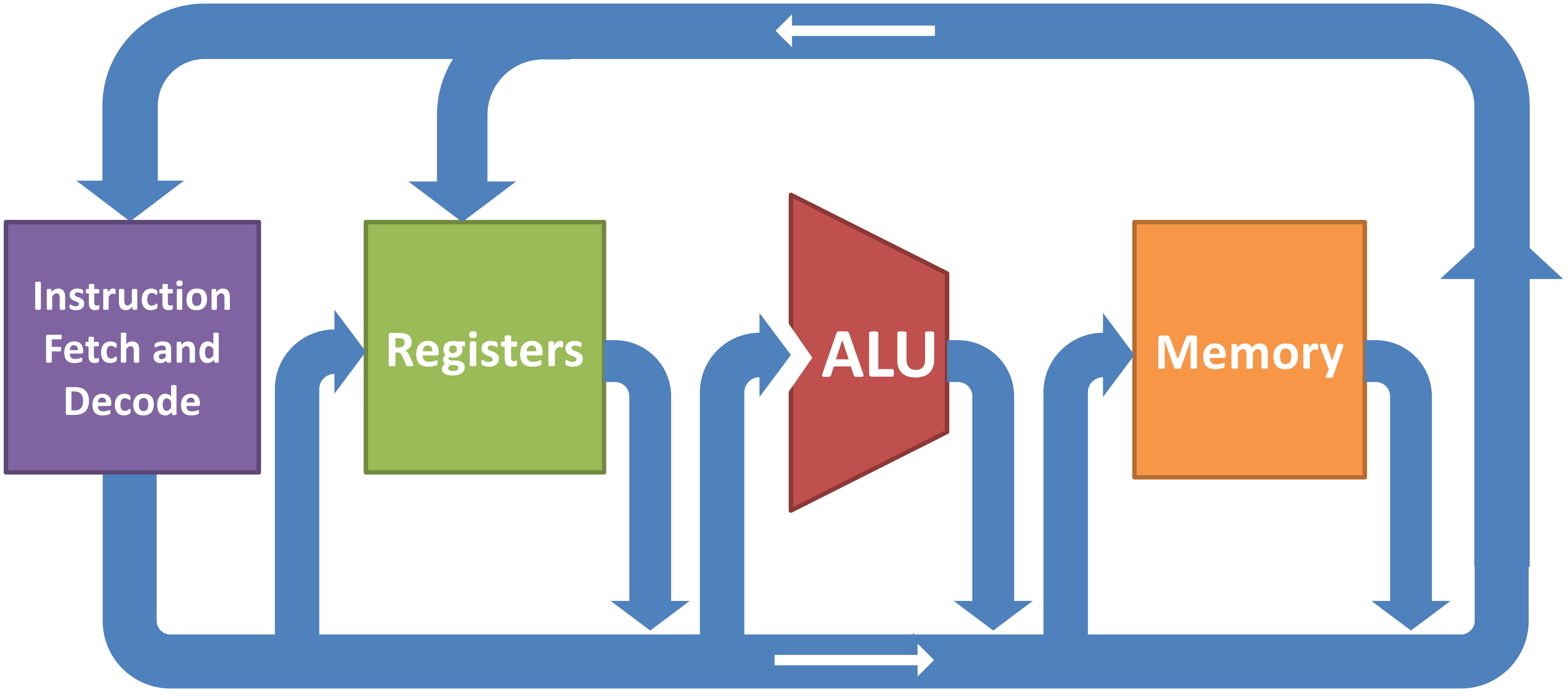
1. Cover exactly the 1s by drawing a (minimum) number of maximally sized rectangles whose dimensions (in cells) are powers of 2. (They may overlap or wrap around!)
2. For each rectangle, make a *product* of the inputs (or complements) that are 1 for all cells in the rectangle. (*minterms*)
3. Take the *sum* of these products.

Voting again with Karnaugh Maps

ex

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Goal for next 2 weeks: **Simple Processor**



Toolbox: Building Blocks

Abstraction!

Microarchitecture

Processor datapath

Instruction Decoder
Arithmetic Logic Unit

Memory

Digital Logic

Adders
Multiplexers
Demultiplexers
Encoders
Decoders

Registers

Flip-Flops
Latches

Gates

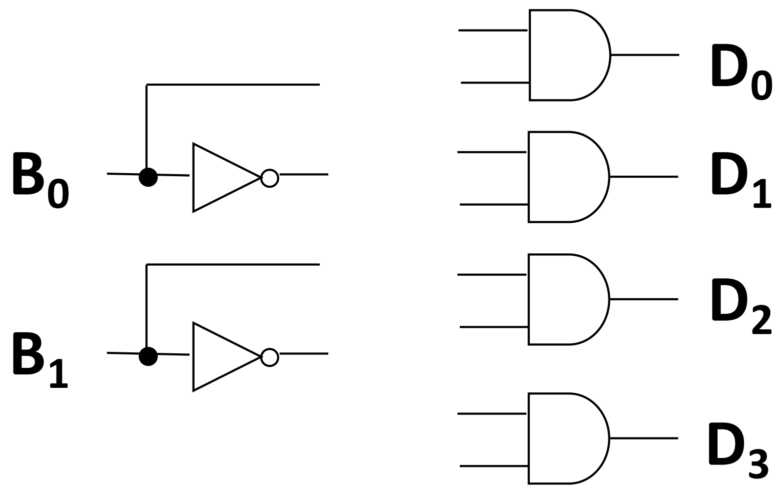
Devices (transistors, etc.)

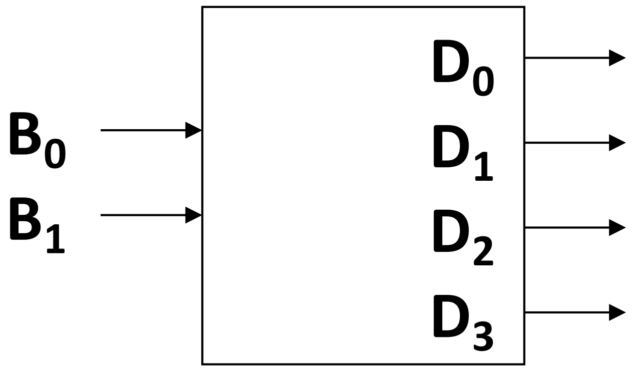
Decoders

ex

Decodes input number, asserts corresponding output.

- *n*-bit input (an unsigned number)
- 2^n outputs
- Built with code detectors.

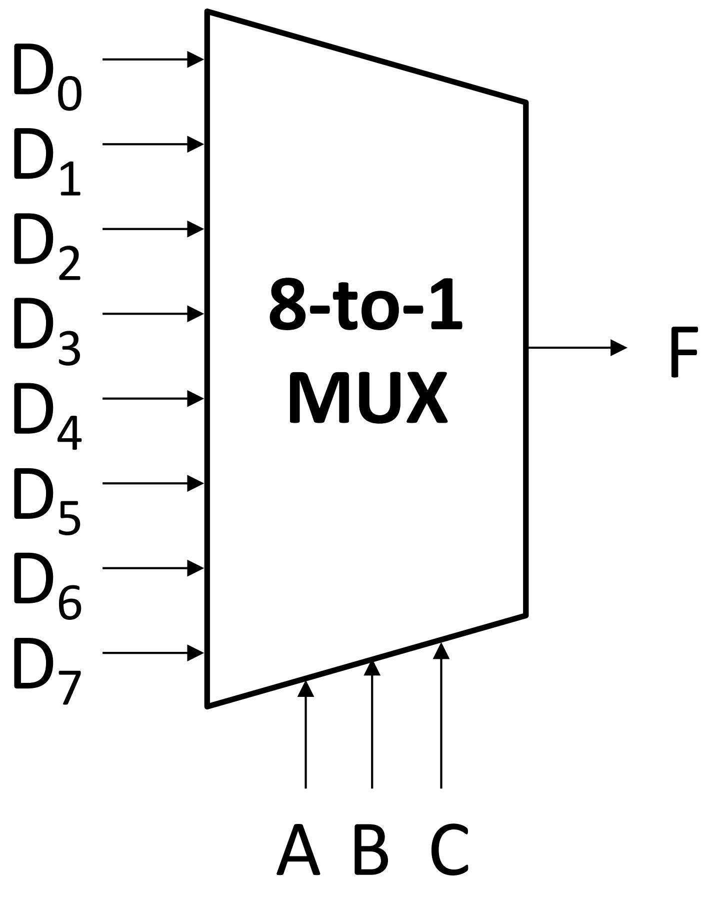




Multiplexers

Select one of several inputs as output.

2^n data inputs



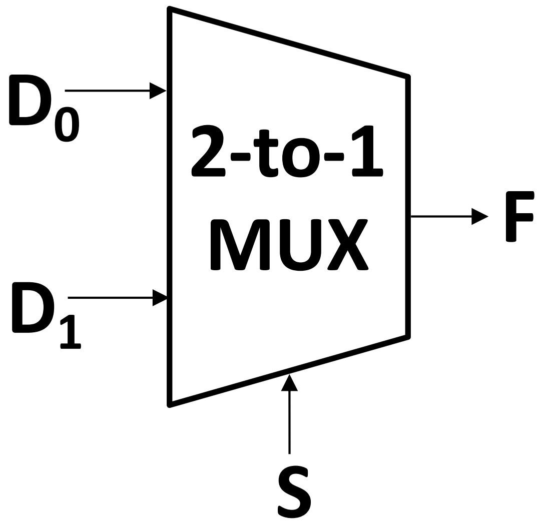
1 data output

***n* selector lines**

Build a 2-to-1 MUX from gates

ex

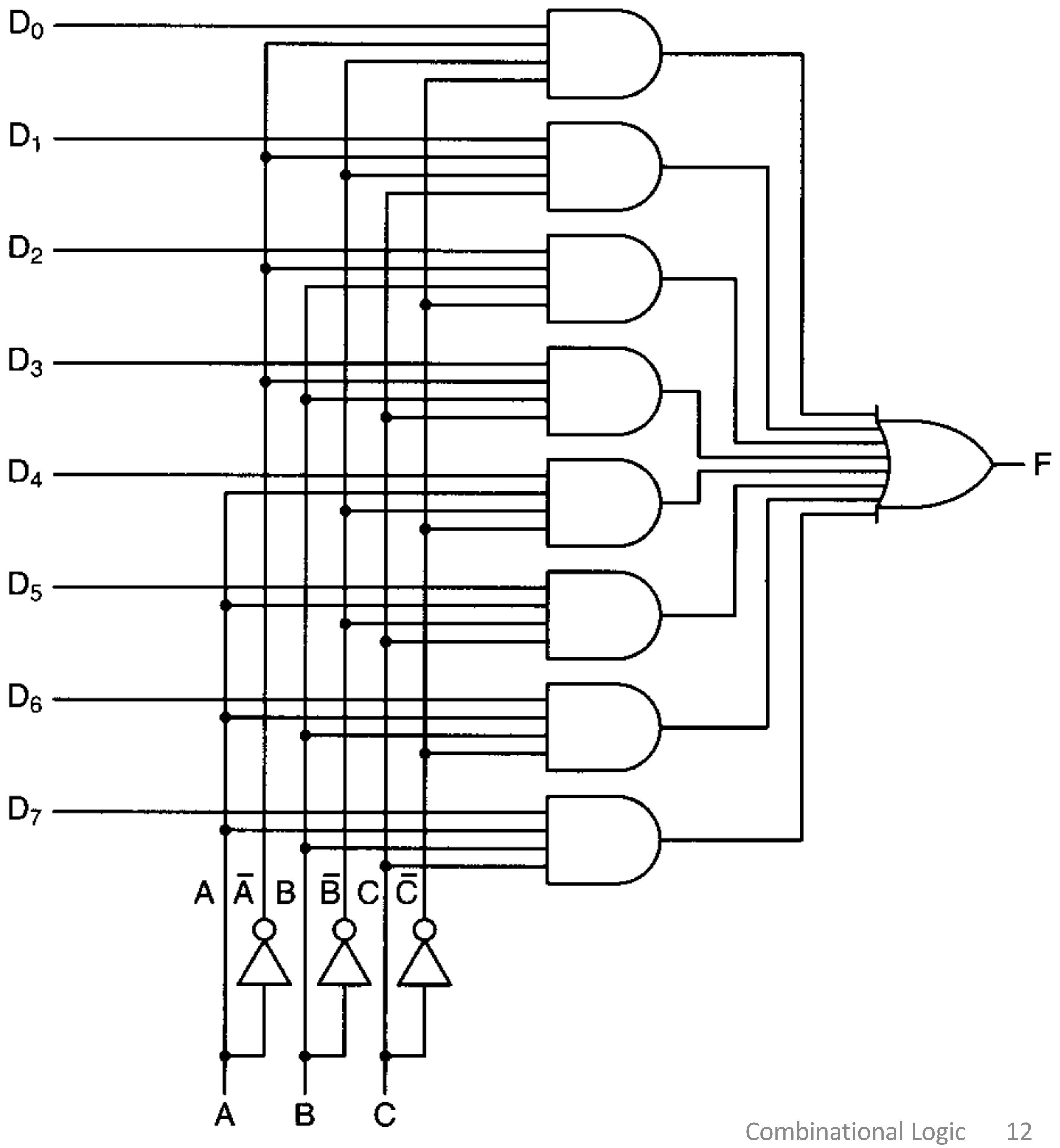
If S=0, then F=D_0.
If S=1, then F=D_1.



1. Construct the truth table.

2. Build the circuit.

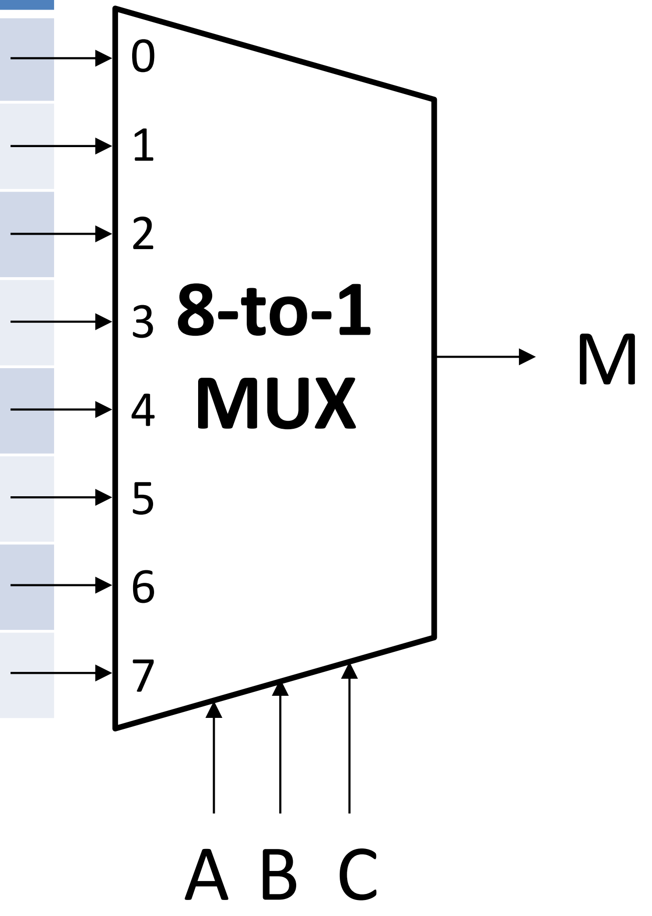
8-to-1 MUX



Costume idea: MUX OX

MUX + voltage source = truth table

A	B	C	M
0	0	0	**0**
0	0	1	**0**
0	1	0	**0**
0	1	1	**1**
1	0	0	**0**
1	0	1	**1**
1	1	0	**1**
1	1	1	**1**



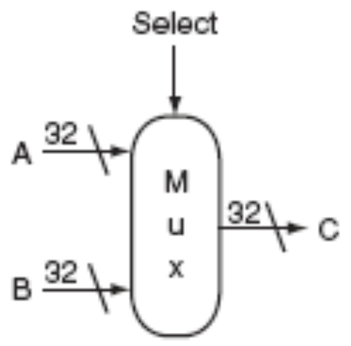
Buses and **Logic Arrays**

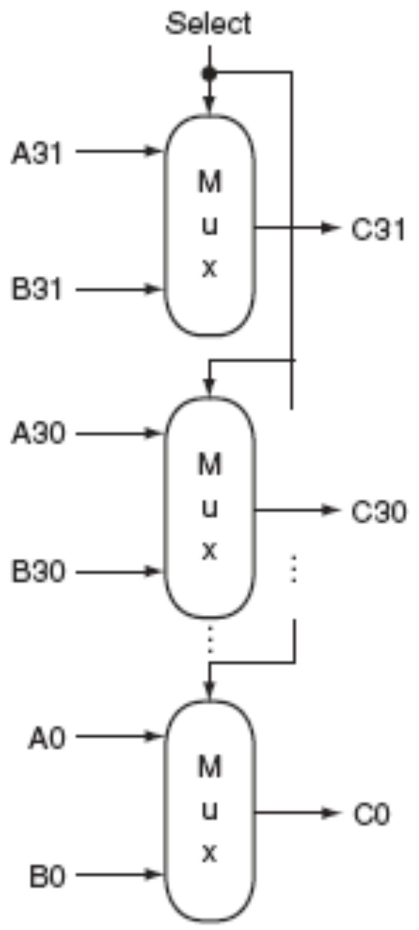
A bus is a collection of data lines treated as a single logical signal.

= fixed-width value

Array of logic elements applies same operation to each bit in a bus.

= bitwise operator


Select
A
32
B
32
M
u
x
32
C


Select
A31
B31
M
u
x
C31
A30
B30
M
u
x
C30
A0
B0
M
u
x
C0