

Warmup

Use a k-map to find a minimal implementation of this truth table:

A	B	C	D		Y
0	0	0	0		0
0	0	0	1		0
0	0	1	0		1
0	0	1	1		1
0	1	0	0		0
0	1	0	1		0
0	1	1	0		1
0	1	1	1		1

A	B	C	D		Y
1	0	0	0		0
1	0	0	1		1
1	0	1	0		0
1	0	1	1		1
1	1	0	0		0
1	1	0	1		1
1	1	1	0		0
1	1	1	1		1

Hand-drawn Karnaugh map for the truth table above. The map is a 4x4 grid with columns labeled AB (00, 01, 11, 10) and rows labeled CD (00, 01, 11, 10). The cells contain the output Y values. A red box highlights the prime implicant $\bar{A}C$ (covering cells where A=0, C=1). A blue box highlights the prime implicant AD (covering cells where A=1, D=1).

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

$$\bar{A}C + AD$$

EE 201: Multiplexers and FPGAs

Steven Bell

30 January 2024

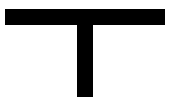
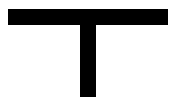
By the end of class today, you should be able to:

- Explain what a multiplexer is
- Draw a logic diagram using a 2^N -input multiplexer to implement an N-variable boolean equation
- Describe the basic structure of an FPGA

Both [multiplexers and decoders] seem like obscure ways to implement things we already have. Why use them?

Some schematic terminology

V_{dd}

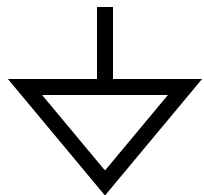
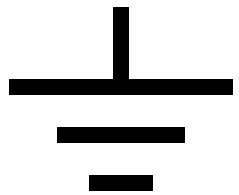


VDD

5V

HIGH

1



Ground

0V

LOW

0

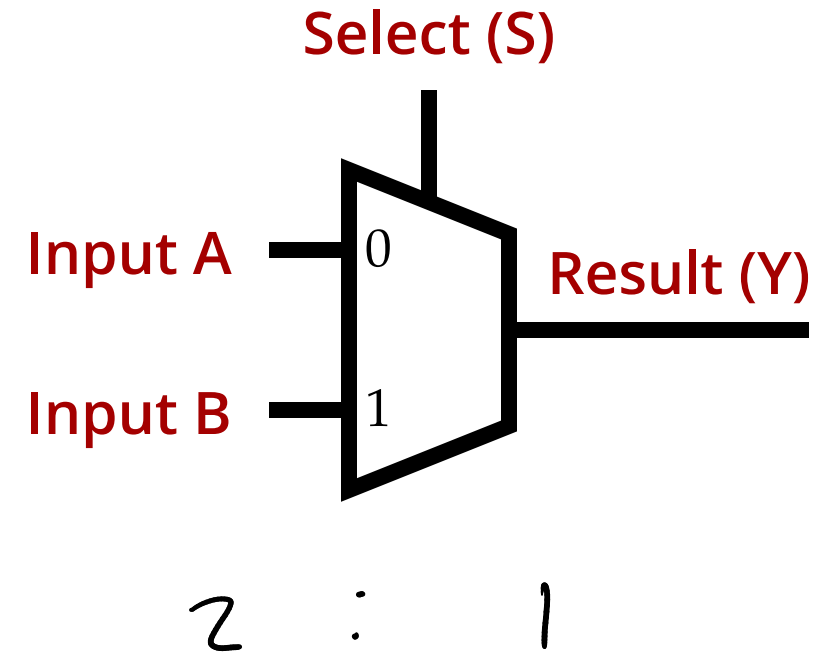


Earth ground

What is a multiplexer?



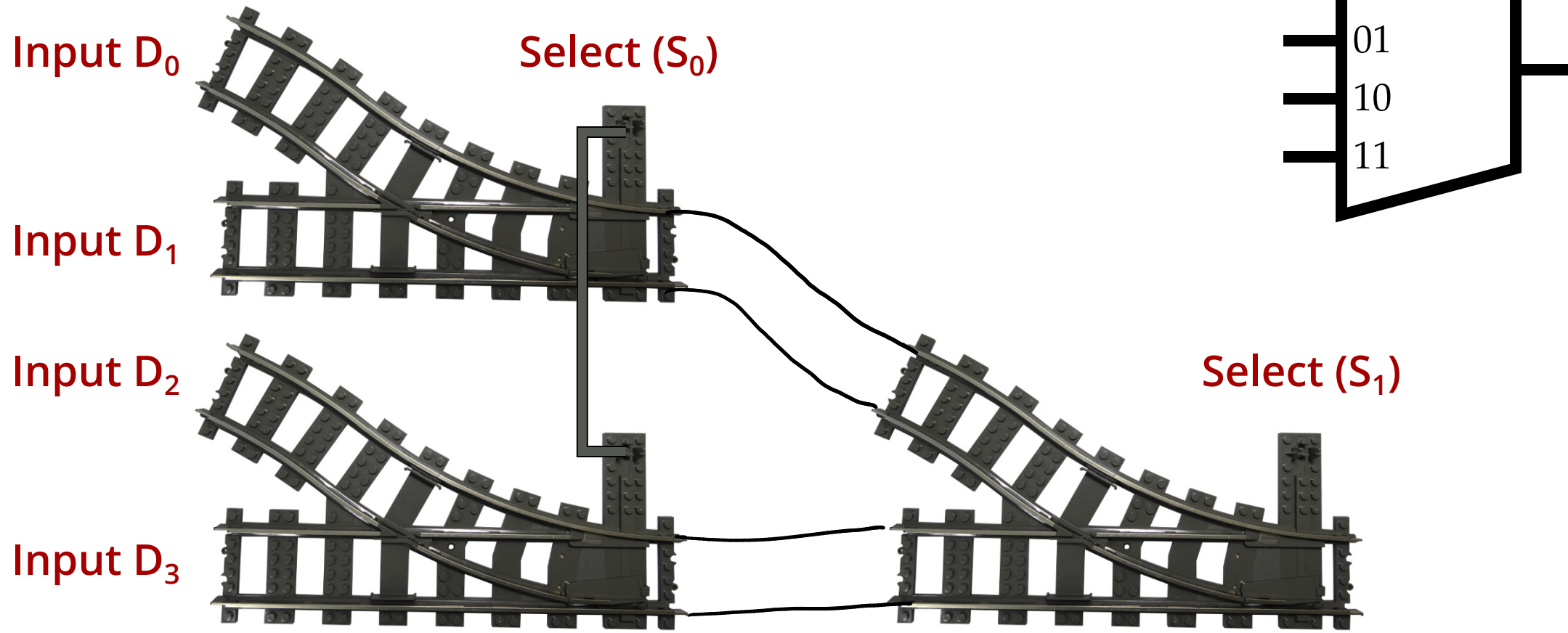
What is a multiplexer?



A 4:1 multiplexer

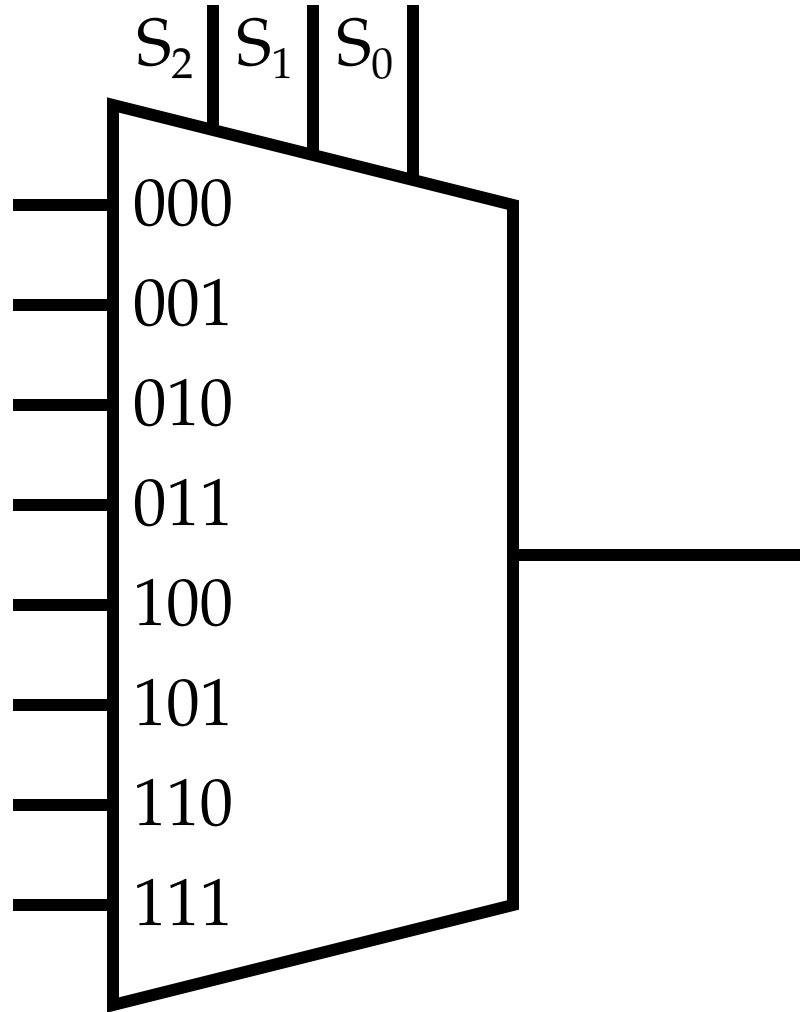
N selects $\Rightarrow 2^N$ data

A B C



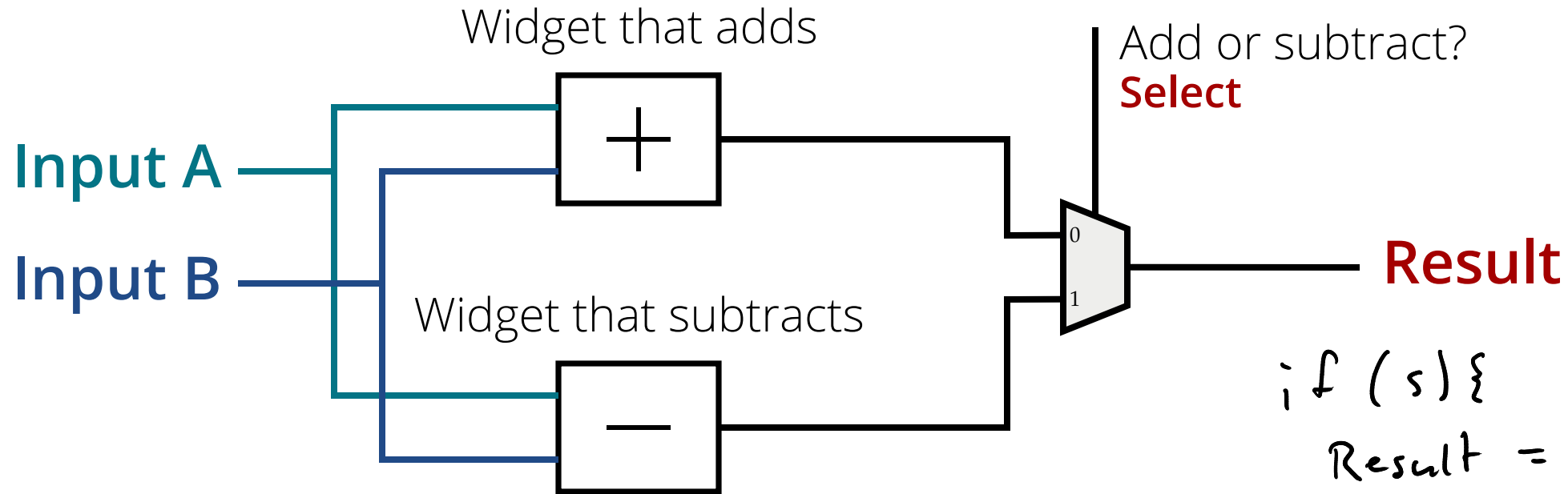
More than 4:1?

If we have N select lines, we can choose from 2^N inputs:



What good are multiplexers?

1) Allow you to select one signal out of many



```
if (s) {  
    Result = A - B ;  
}  
else {  
    Result = A + B ;  
}
```

if (case 1)

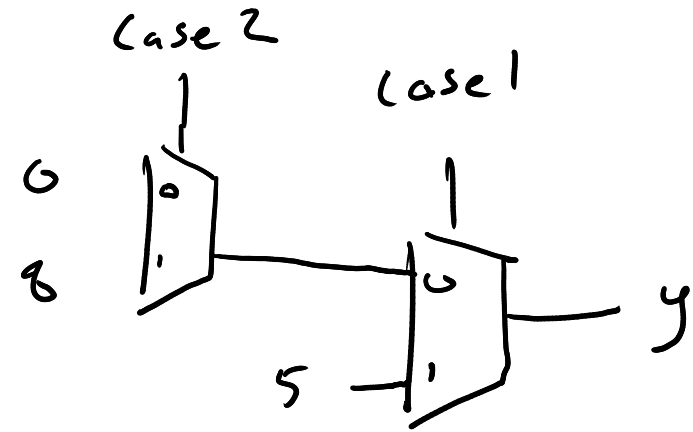
$y = 5$

else if (case 2)

$y = 8$

else

$y = 0$



What good are multiplexers?

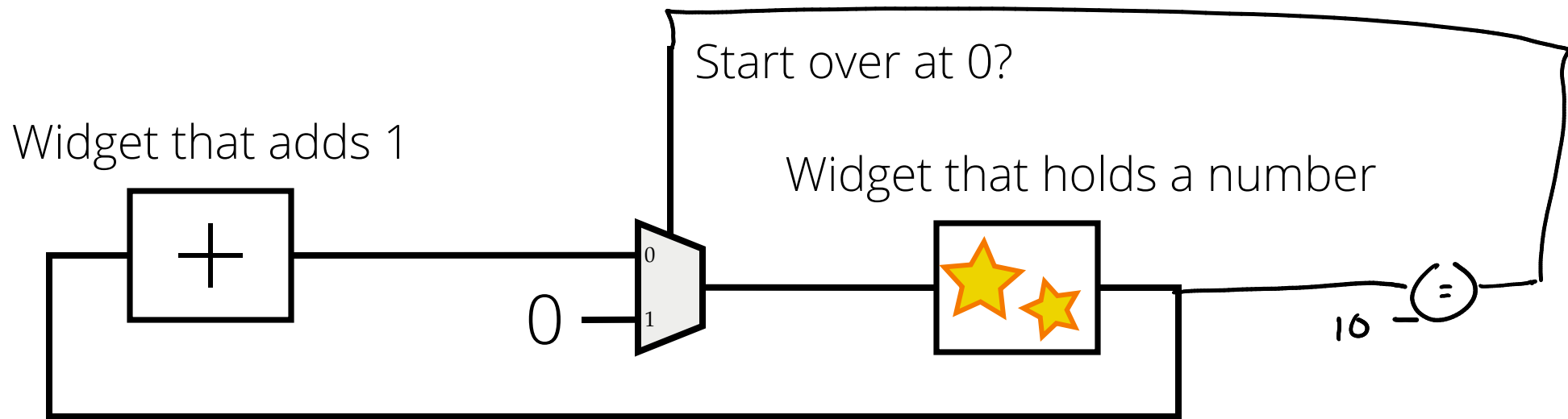
1) Allow you to select one signal out of many

1 B) Allow you to make a choice based on a control value

It's like an **if** or **case** statement in software

Another multiplexer example

1 B) Like an **if** statement in hardware



And back in the old days...

1) Allow you to select one signal out of many

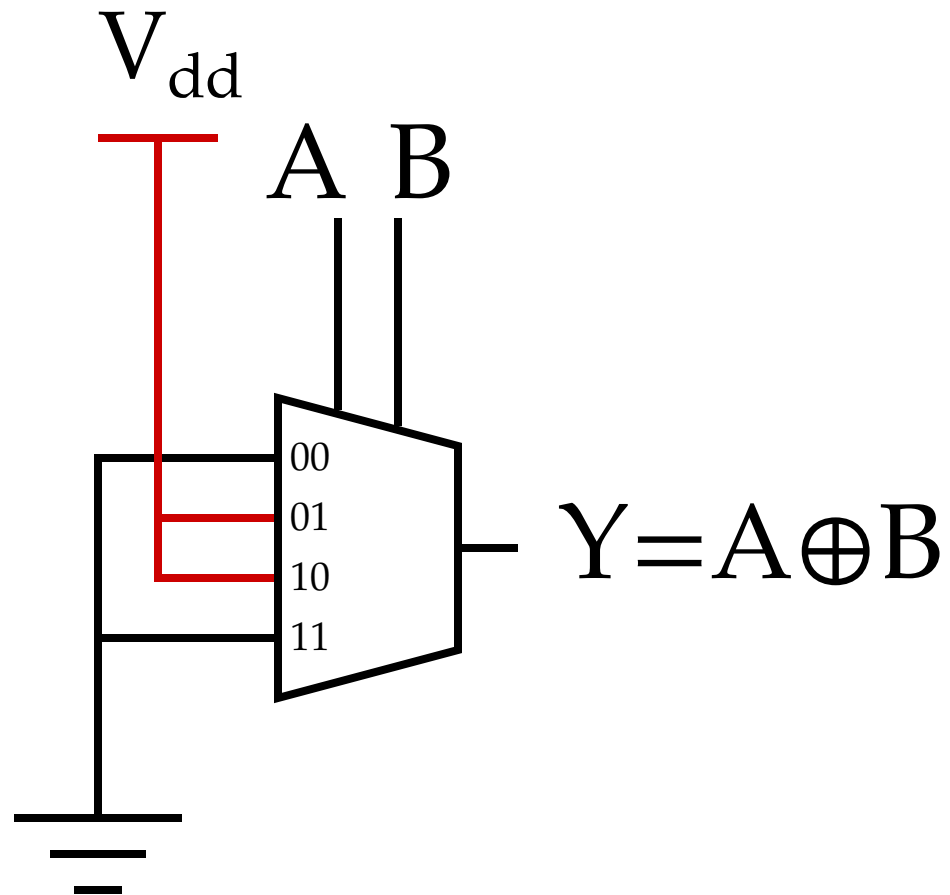
1 B) Allow you to make a choice based on a control value

It's like an **if** or **case** statement in software

2) Make it easy to implement arbitrary logic functions

Implementing XOR

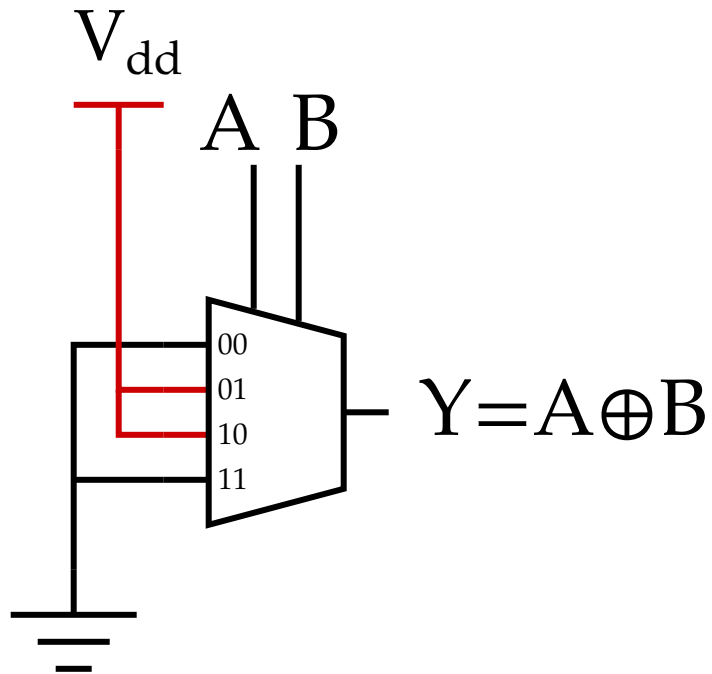
Using a mux to implement a logic function



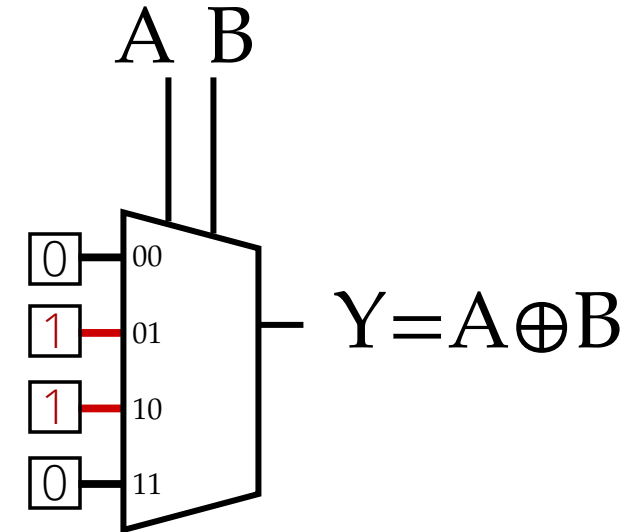
Look-up tables (LUTs)

A look-up-table is basically a mux where the inputs are little memory boxes statically configured to be 0 or 1.

With a mux



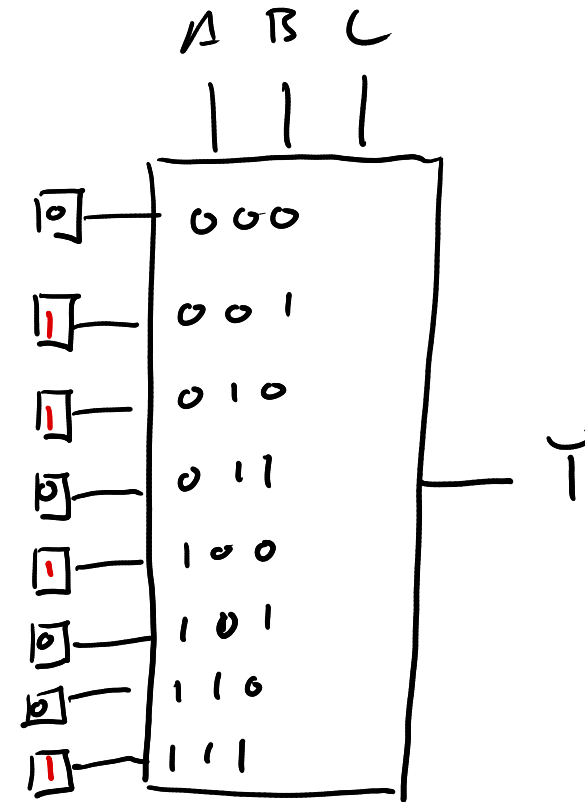
With a LUT



Look-up table practice

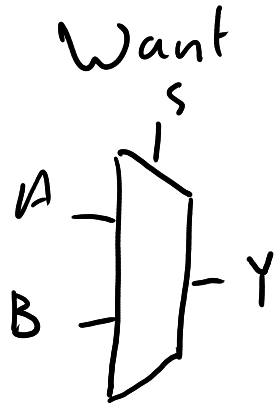
Use a 3-input LUT to implement $A \oplus B \oplus C$

A	B	$A \oplus B$	C	Y
0	0	0	0	0
0	0	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1



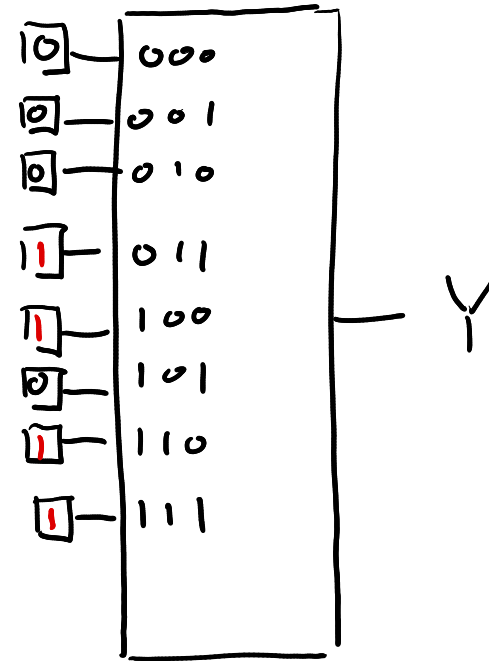
Look-up table practice

Use a 3-input LUT to implement a 2:1 multiplexer



S	Y
0	A
1	B

A	B	S	Y
0	0	0	A → 0
0	0	1	B → 0
0	1	0	A → 0
0	1	1	B → 1
1	0	0	A → 1
1	0	1	B → 0
1	1	0	A → 1
1	1	1	B → 1



Introducing FPGAs

An FPGA is like a big chip full of logic gates that can be wired together by "programming" it.

iCE40UP block diagram

Clock stuff

Fixed-function multipliers

Memory

Logic "fabric"

Fixed-function I/O modules

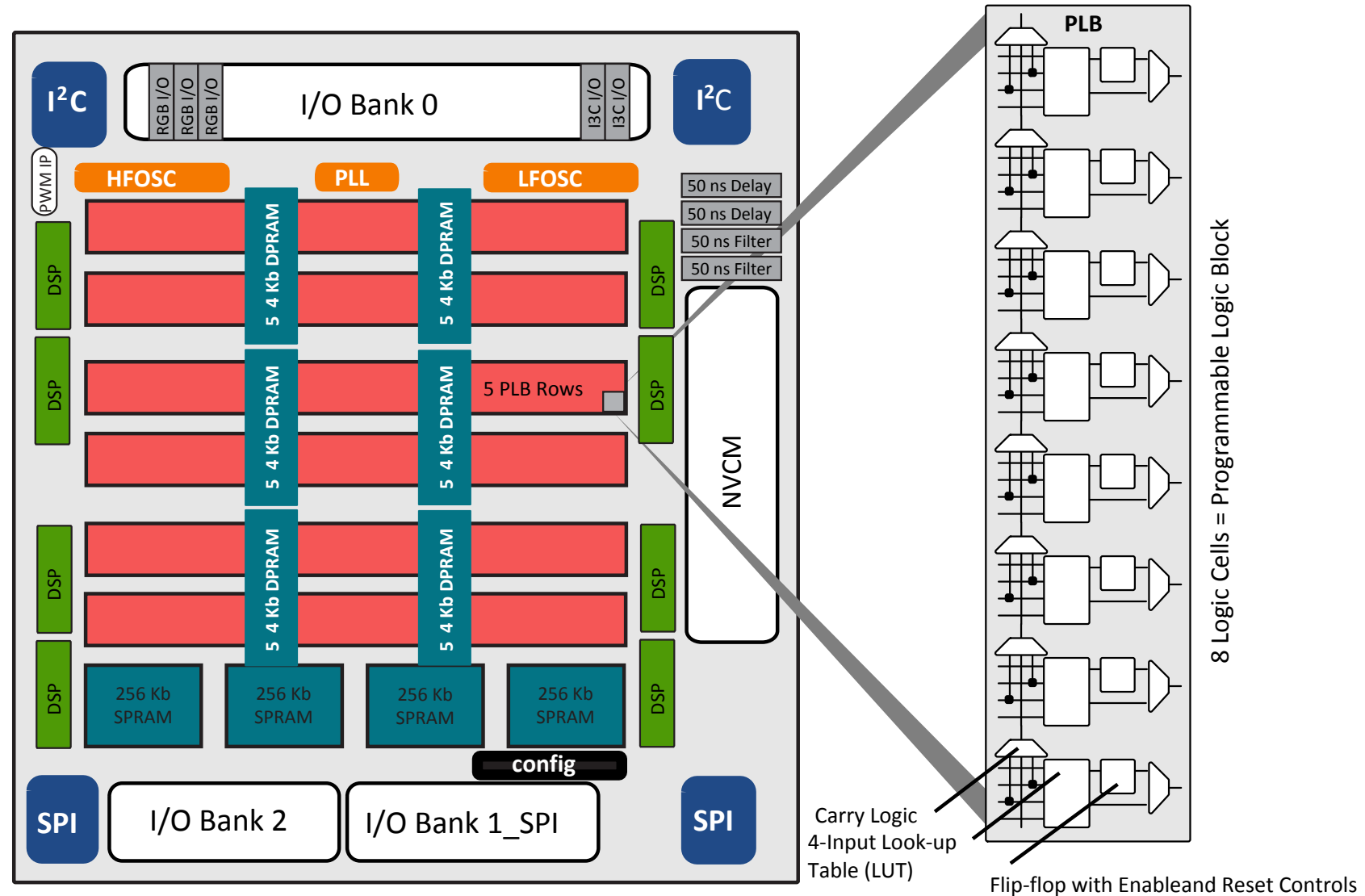


Figure 3.1. iCE40UP5K Device, Top View

iCE40UP logic element

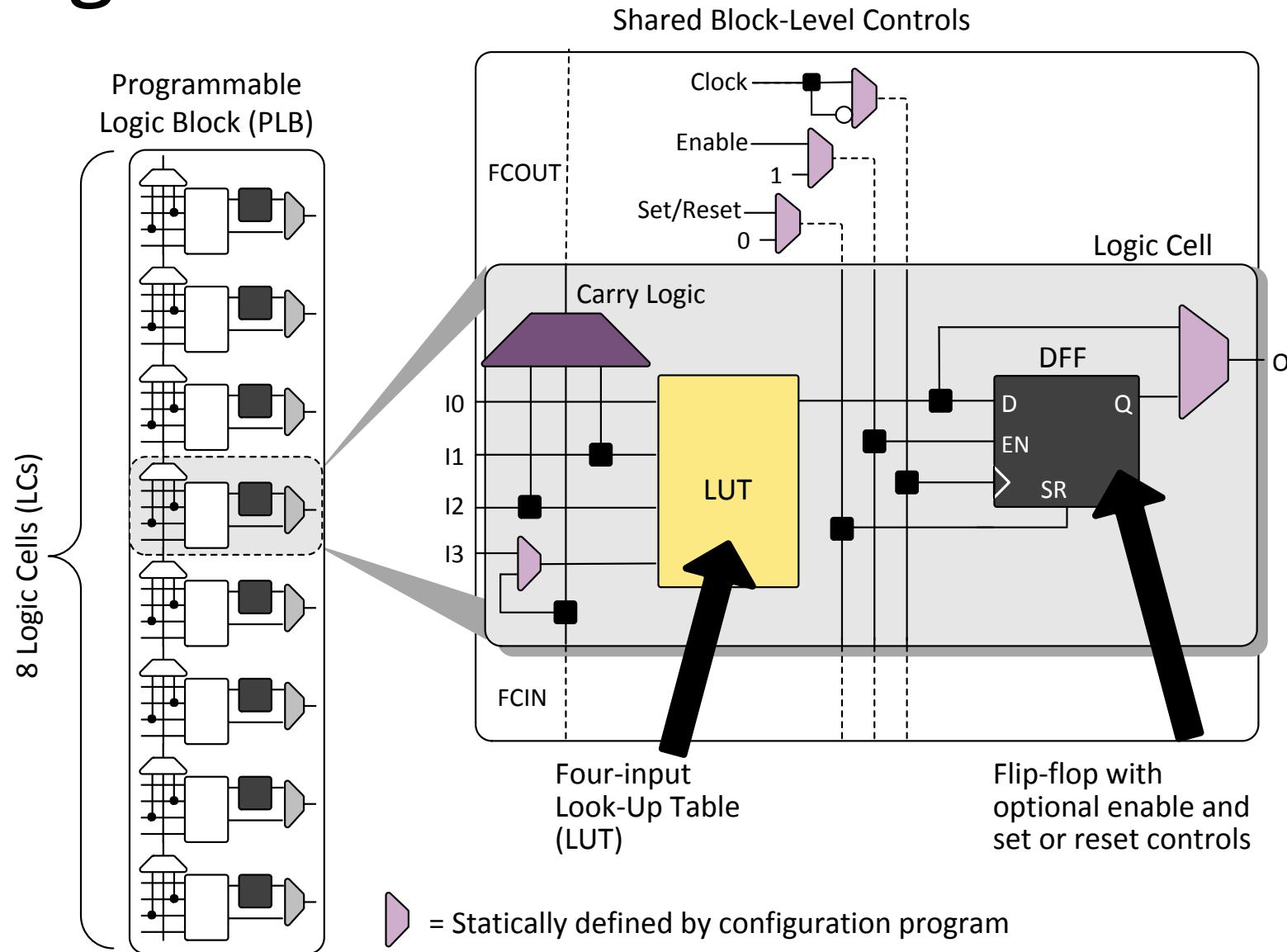


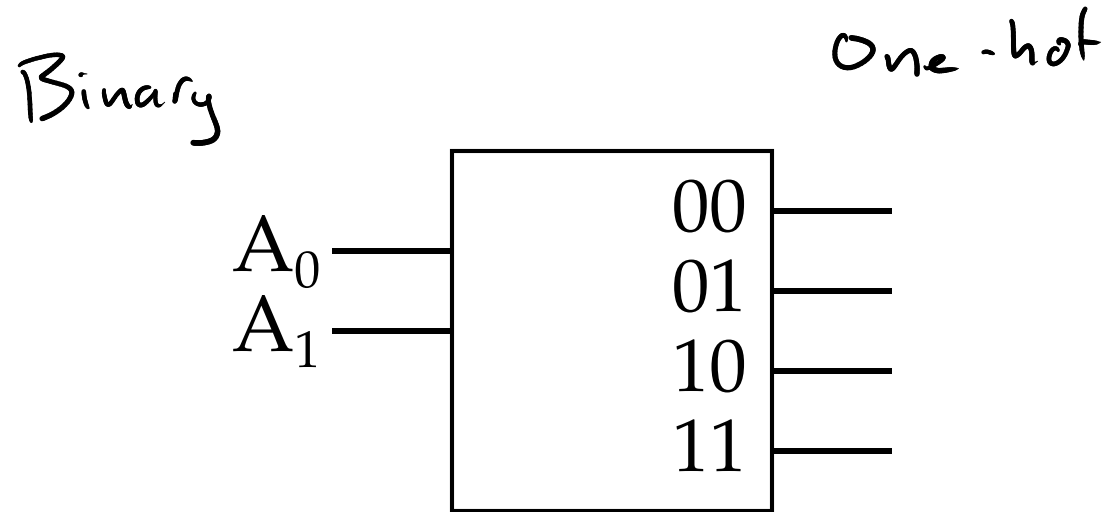
Figure 3.2. PLB Block Diagram

A better definition

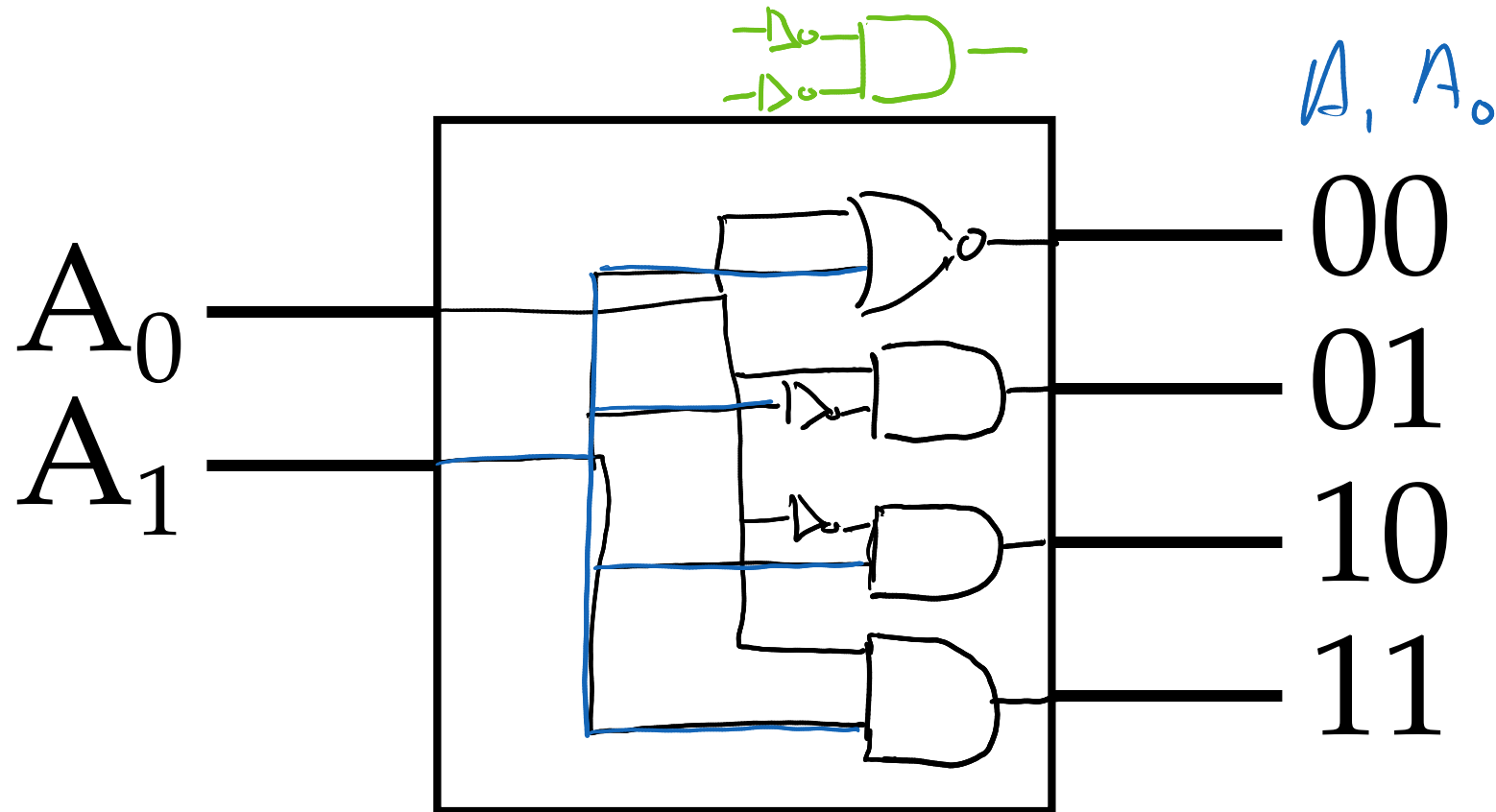
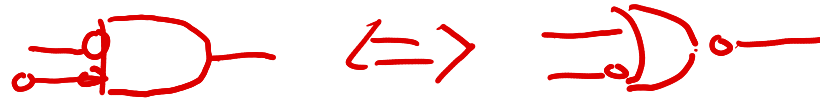
An FPGA is a chip full of configurable **look-up-tables** with configurable **interconnections** and storage.

Decoders

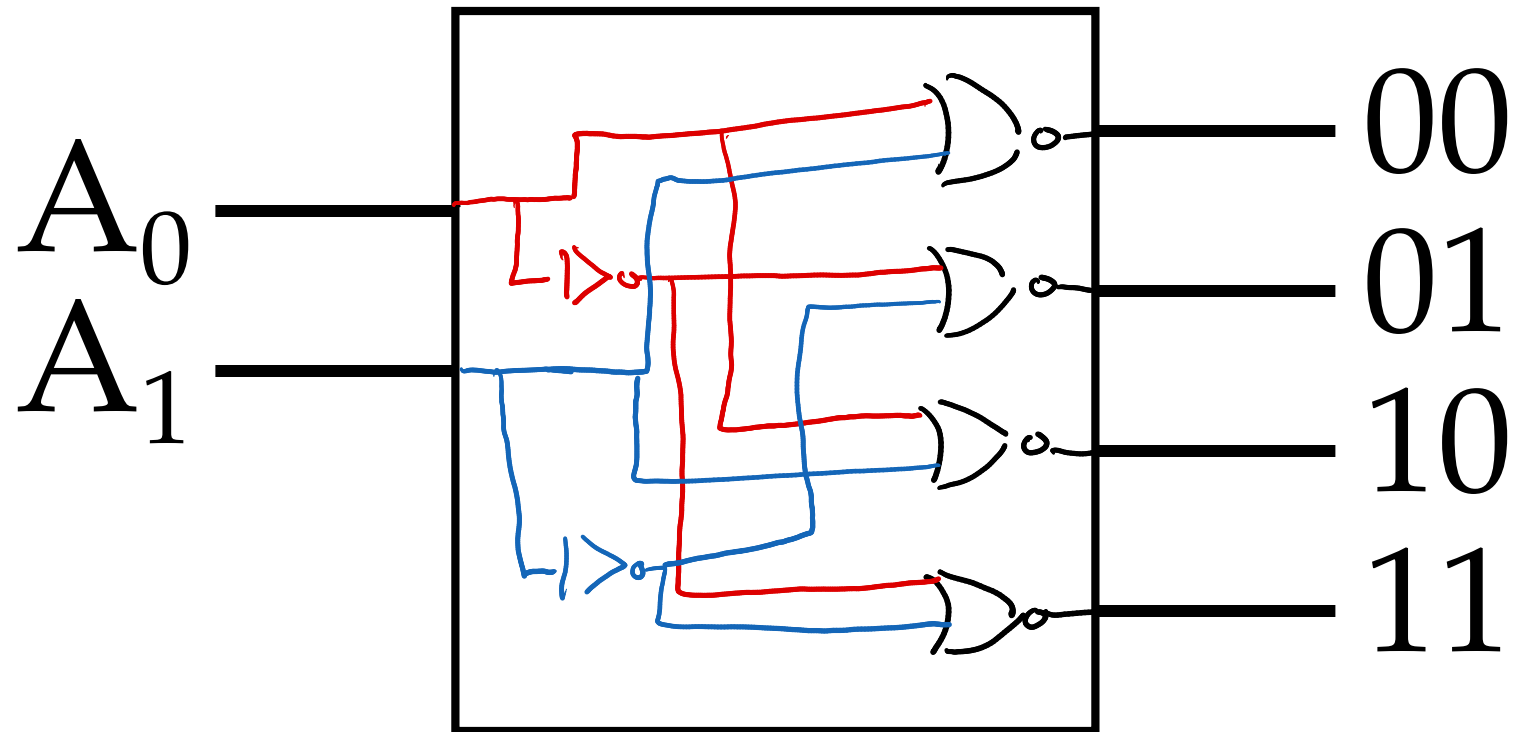
Take a binary number as an input, and set the corresponding output high.



Building a decoder

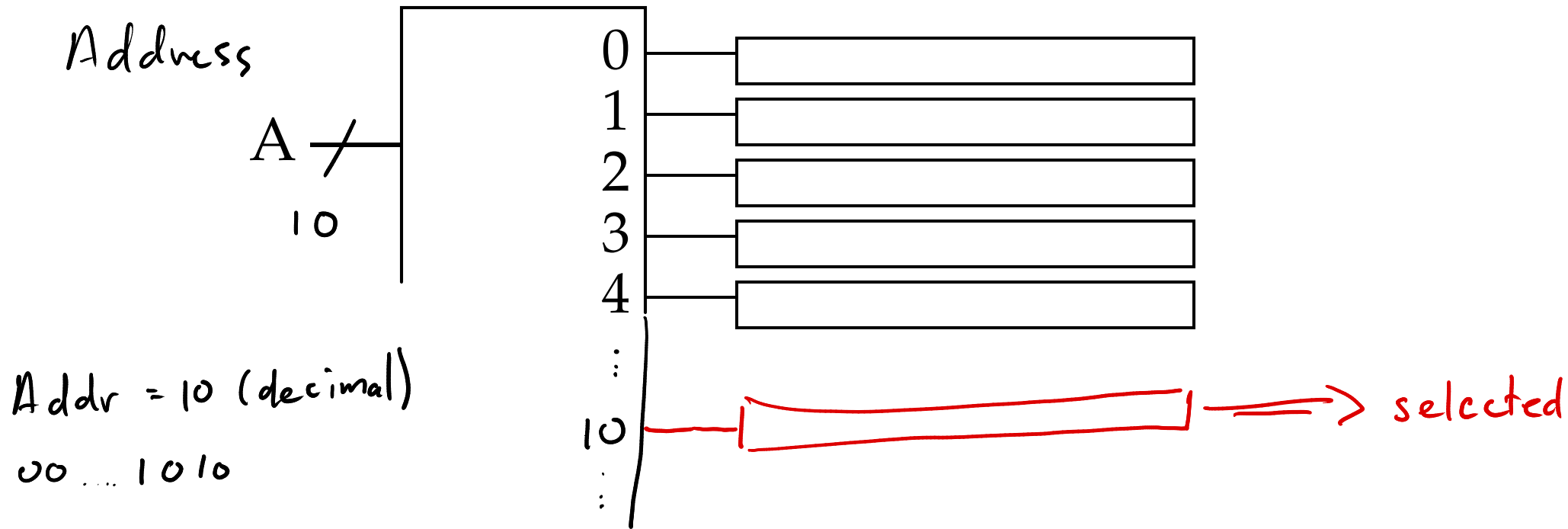


Building a decoder *with fewer transistors*



What could you use a decoder for?

You have a whole bunch of things to activate one at a time
(say, memory cells)



For Thursday

1. Read the book (2.9) and complete the reading check