

# 6.2: Combinational Circuits

## Let's Make an Adder Circuit

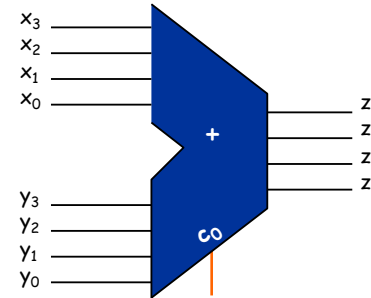
Goal:  $x + y = z$ .

- We build 4-bit adder: 9 inputs, 4 outputs. (Same idea scales to 128-bit adder.)
- Key computer component.

	1	1	1	0
	2	4	8	7
+	3	5	7	9
	6	1	6	6

Step 1.

- Represent input and output in binary.



	1	1	0	0
	0	0	1	0
+	0	1	1	1
	1	0	0	1

	$x_3$	$x_2$	$x_1$	$x_0$
+	$y_3$	$y_2$	$y_1$	$y_0$
	$z_3$	$z_2$	$z_1$	$z_0$

## Let's Make an Adder Circuit

Goal:  $x + y = z$ .

Step 2.

- Build truth table.
- Why is this a bad idea?
  - 128-bit adder:  $2^{256+1}$  rows > # electrons in universe!

				$c_0$
	$x_3$	$x_2$	$x_1$	$x_0$
+	$y_3$	$y_2$	$y_1$	$y_0$
	$z_3$	$z_2$	$z_1$	$z_0$

4-Bit Adder Truth Table												
$c_0$	$x_3$	$x_2$	$x_1$	$x_0$	$y_3$	$y_2$	$y_1$	$y_0$	$z_3$	$z_2$	$z_1$	$z_0$
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	0	1	1	0	0	1
0	0	0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	1	0	1	0	1	0	1
.	.	.	.	.	.	.	.	.	.	.	.	.
1	1	1	1	1	1	1	1	1	1	1	1	1

$2^{8+1} = 512$  rows!

## Let's Make an Adder Circuit

Goal:  $x + y = z$ .

Step 2.

- Build truth table for carry bit.
- Build truth table for summand bit.

	$c_3$		$c_1$	$c_0 = 0$
	$x_3$		$x_1$	$x_0$
+	$y_3$		$y_1$	$y_0$
	$z_3$		$z_1$	$z_0$

Carry Bit			
$x_i$	$y_i$	$c_i$	$c_{i+1}$
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

Summand Bit			
$x_i$	$y_i$	$c_i$	$z_i$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

## Let's Make an Adder Circuit

Goal:  $x + y = z$ .

Step 3.

- Derive (simplified) Boolean expression.

$$\begin{array}{r}
 c_3 \quad c_2 \quad c_1 \quad c_0 = 0 \\
 x_3 \quad x_2 \quad x_1 \quad x_0 \\
 + \quad y_3 \quad y_2 \quad y_1 \quad y_0 \\
 \hline
 z_3 \quad z_2 \quad z_1 \quad z_0
 \end{array}$$

Carry Bit				
$x_i$	$y_i$	$c_i$	$c_{i+1}$	MAJ
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Summand Bit				
$x_i$	$y_i$	$c_i$	$z_i$	ODD
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

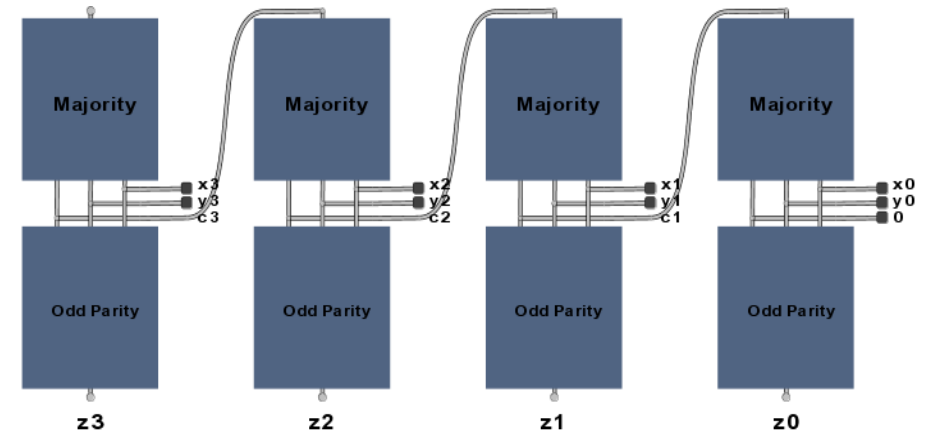
5

## Let's Make an Adder Circuit

Goal:  $x + y = z$ .

Step 4.

- Transform Boolean expression into circuit.

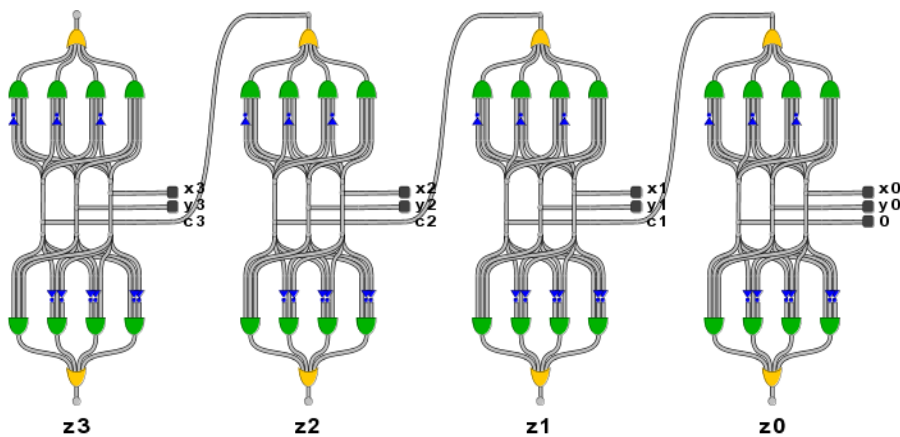


## Let's Make an Adder Circuit

Goal:  $x + y = z$ .

Step 4.

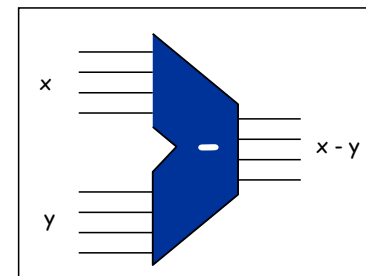
- Transform Boolean expression into circuit.



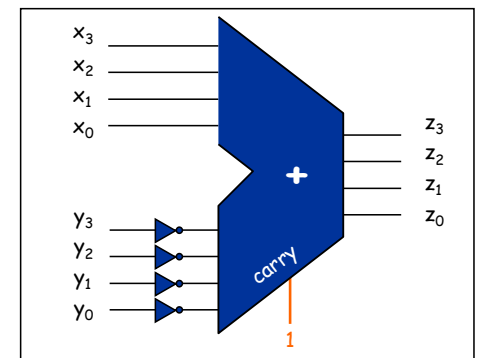
## Subtractor

Subtractor circuit:  $z = x - y$ .

- One approach: design like adder circuit.
- Better idea: reuse adder circuit.
  - 2's complement: to negate an integer, flip bits, then add 1



4-Bit Subtractor Interface



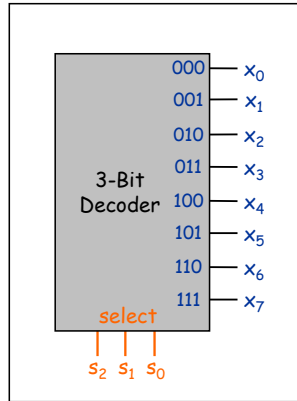
4-Bit Subtractor Implementation

8

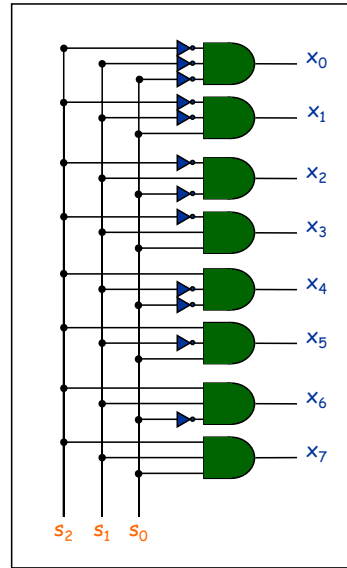
## N-Bit Decoder

### N-bit decoder.

- N address inputs,  $2^N$  data outputs.
- Addressed output bit is 1; all others are 0.



3-Bit Decoder Interface



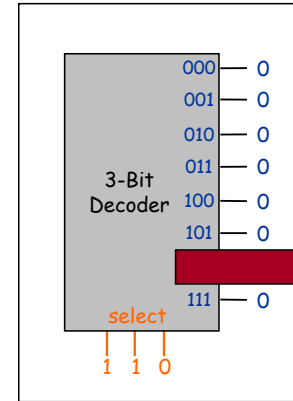
3-Bit Decoder Implementation

9

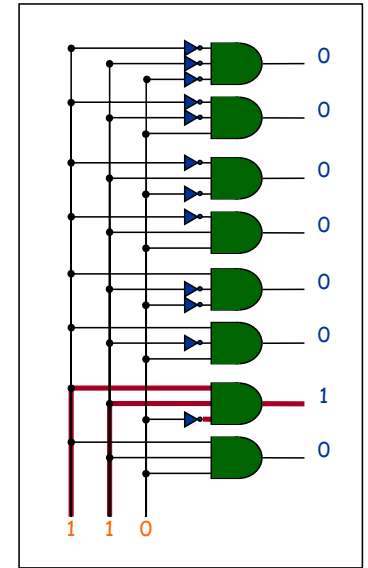
## N-Bit Decoder

### N-bit decoder.

- N address inputs,  $2^N$  data outputs.
- Addressed output bit is 1; all others are 0.



3-Bit Decoder Interface



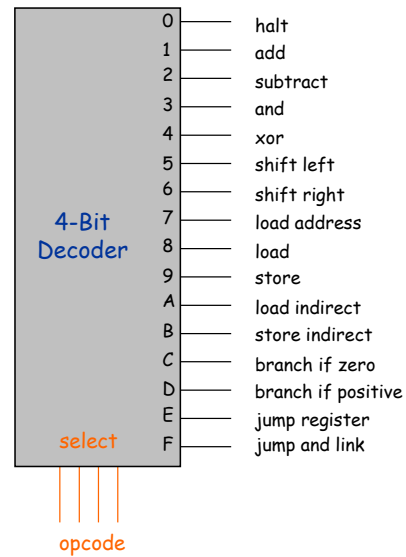
3-Bit Decoder Implementation

10

## N-Bit Decoder

### Application.

- Convert from binary to "unary."
- Decode opcode to instruction type.

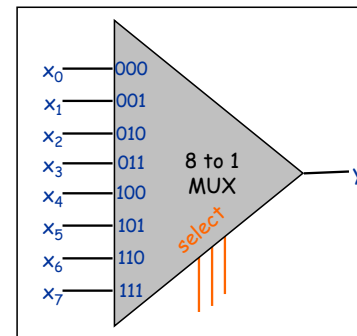


11

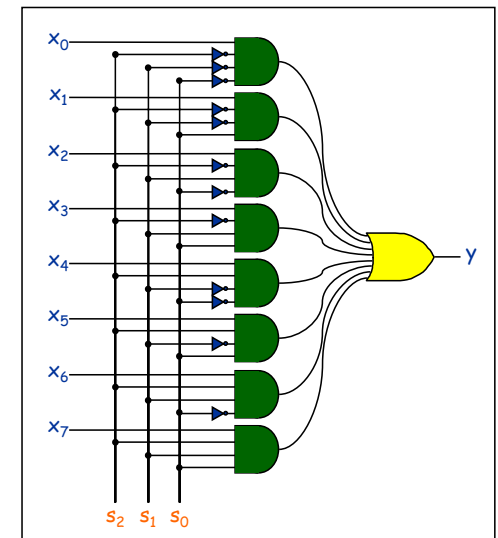
## 8-to-1 Multiplexer

### $2^N$ -to-1 multiplexer.

- N select inputs,  $2^N$  data inputs, 1 output.
- Copies "selected" data input bit to output.



8-to-1 Mux Interface



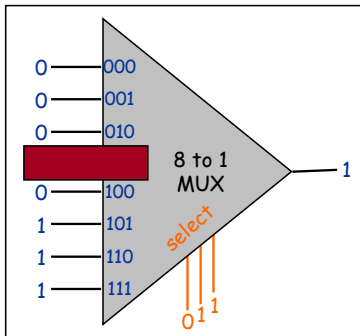
8-to-1 Mux Implementation

12

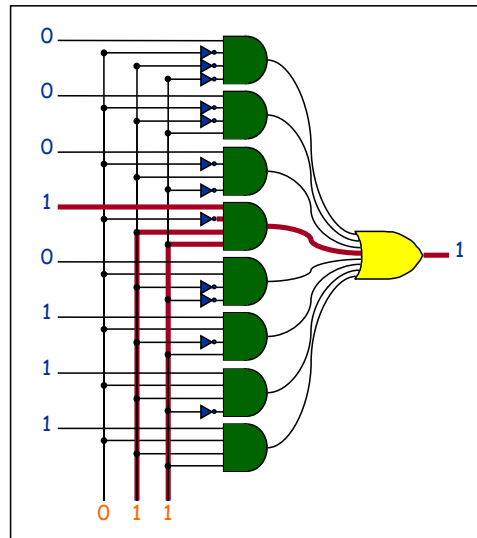
## 8-to-1 Multiplexer

$2^N$ -to-1 multiplexer.

- N select inputs,  $2^N$  data inputs, 1 output.
- Copies "selected" data input bit to output.



8-to-1 Mux Interface



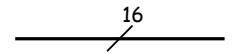
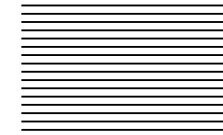
8-to-1 Mux Implementation

13

## Bus

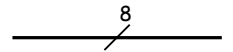
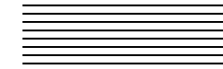
16-bit bus.

- Bundle of 16 wires.
- Memory transfer, register transfer.



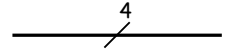
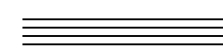
8-bit bus.

- Bundle of 8 wires.
- TOY memory address.



4-bit bus.

- Bundle of 4 wires.
- TOY register address.

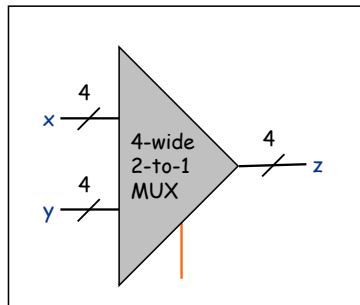


14

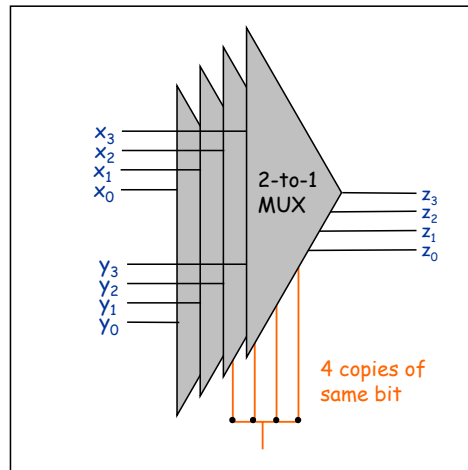
## 4-Wide 2-to-1 Multiplexer

Goal: select from one of two 4-bit buses.

- Implement by layering 4 2-to-1 multiplexers.



Interface



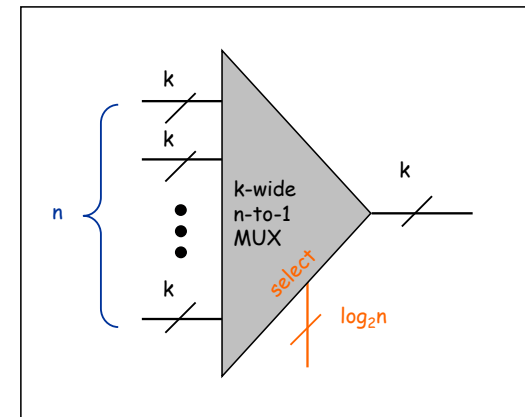
Implementation

15

## k-Wide n-to-1 Multiplexer

Goal: select from one of n k-bit buses.

- Implement by layering k n-bit muxes.



Interface

16

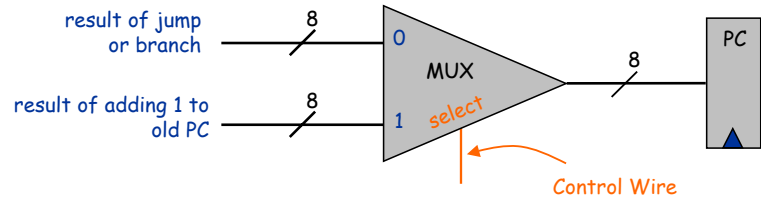
## Multiplexer: Application

### Program counter

- Result of jump or branch instruction.
- Adding 1 to old program counter.

Use 8-wide 2-to-1 mux to route appropriate 8-bit address to PC.

- Unspecified detail: how to set control wire?



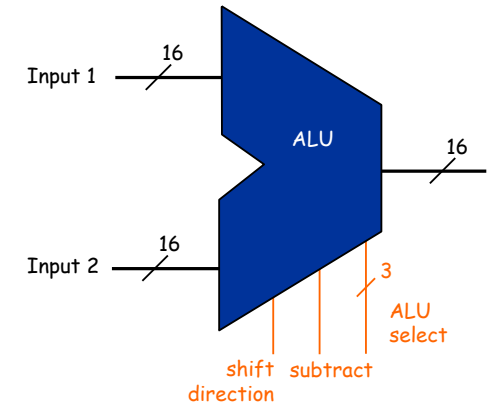
17

## Arithmetic Logic Unit: Interface

### ALU Interface.

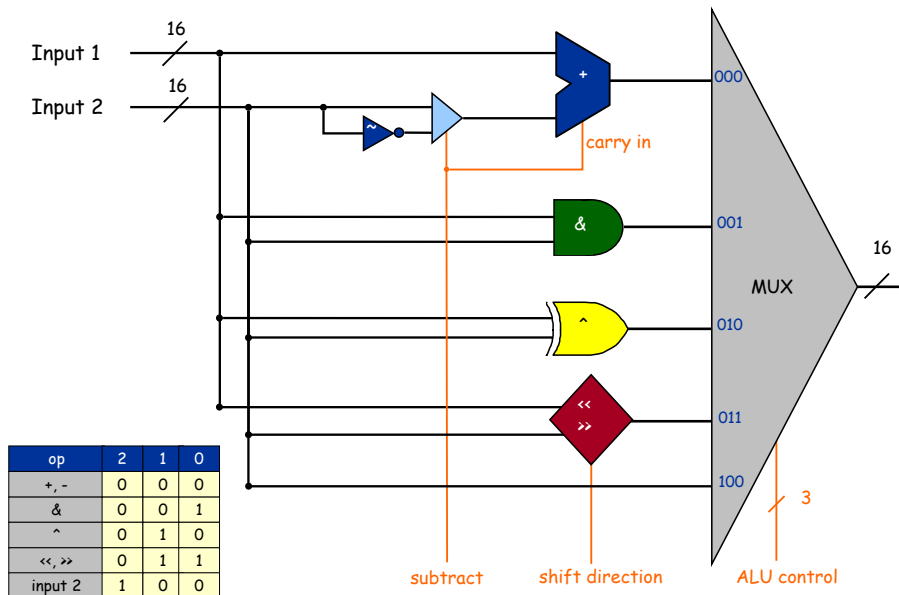
- Add, subtract, bitwise and, bitwise xor, shift left, shift right, copy.
- Associate 3-bit integer with 5 primary ALU operations.
  - ALU performs operations in parallel
  - control wires select which result ALU outputs

op	2	1	0
+, -	0	0	0
&	0	0	1
^	0	1	0
<<, >>	0	1	1
input 2	1	0	0



18

## Arithmetic Logic Unit: Implementation



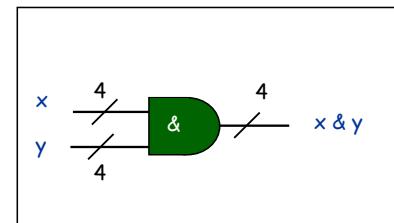
op	2	1	0
+, -	0	0	0
&	0	0	1
^	0	1	0
<<, >>	0	1	1
input 2	1	0	0

19

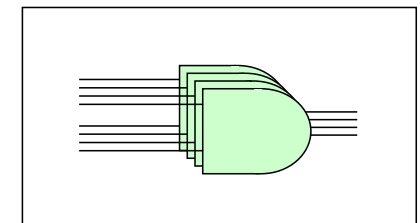
## Bitwise AND, XOR, NOT

### Bitwise logical operations.

- Inputs x and y: n-bits each.
- Output z: n-bits.
- Apply logical operation to each corresponding pair of bits.



Bitwise And Interface



Bitwise And Implementation

20

## Abstraction and Encapsulation

Lessons for ADT apply to hardware!

- Interface describes behavior of circuit.
- Implementation gives details of how to build it.

Layers of abstraction apply with a vengeance!

- TOY ALU is made of:
  - multiplexer, which is made of:
    - ✎ AND, OR, NOT gates
  - adder, which is made of:
    - ✎ AND, OR, NOT gates
    - and some other things also made of gates
- TOY ALU will itself be a component of TOY computer (Lecture A5).