Digital Logic
II. *Combinational Circuits*

Jalal Kawash

Outline

1. Types of Circuits
2. Basic Combinational Circuits
   a. Adders
   b. Decodres
3. ALUs
4. More Combinational Circuits
   a. Multiplexers
   b. Multipliers
   c. Comparators

Section 1

Types of Digital Circuits

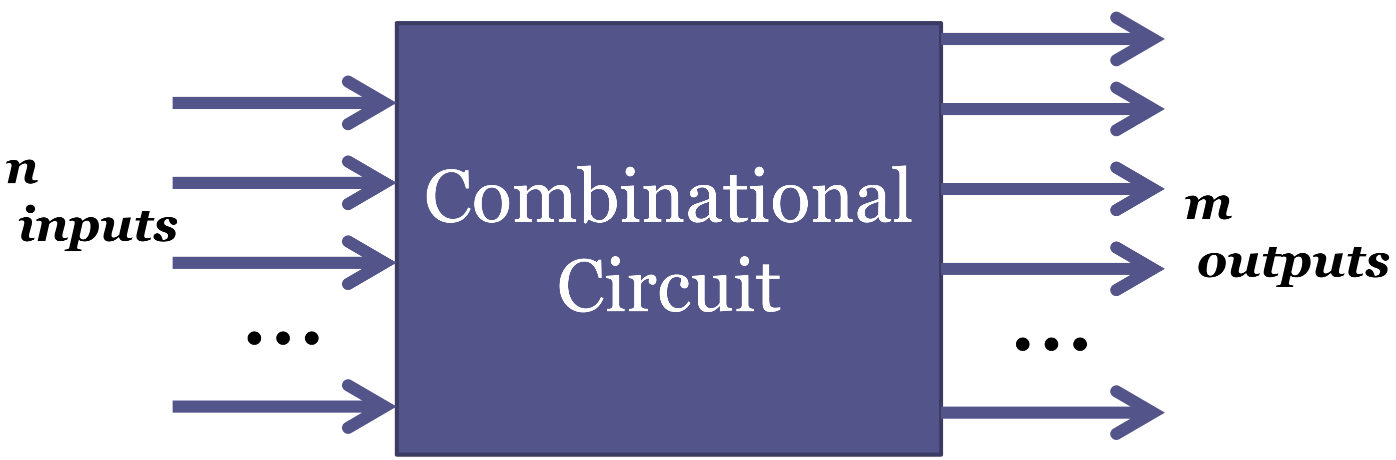
Section 1 Objective

At the end of this section you will

1. Understand the difference between combinational and sequential circuits

Combinational Circuits

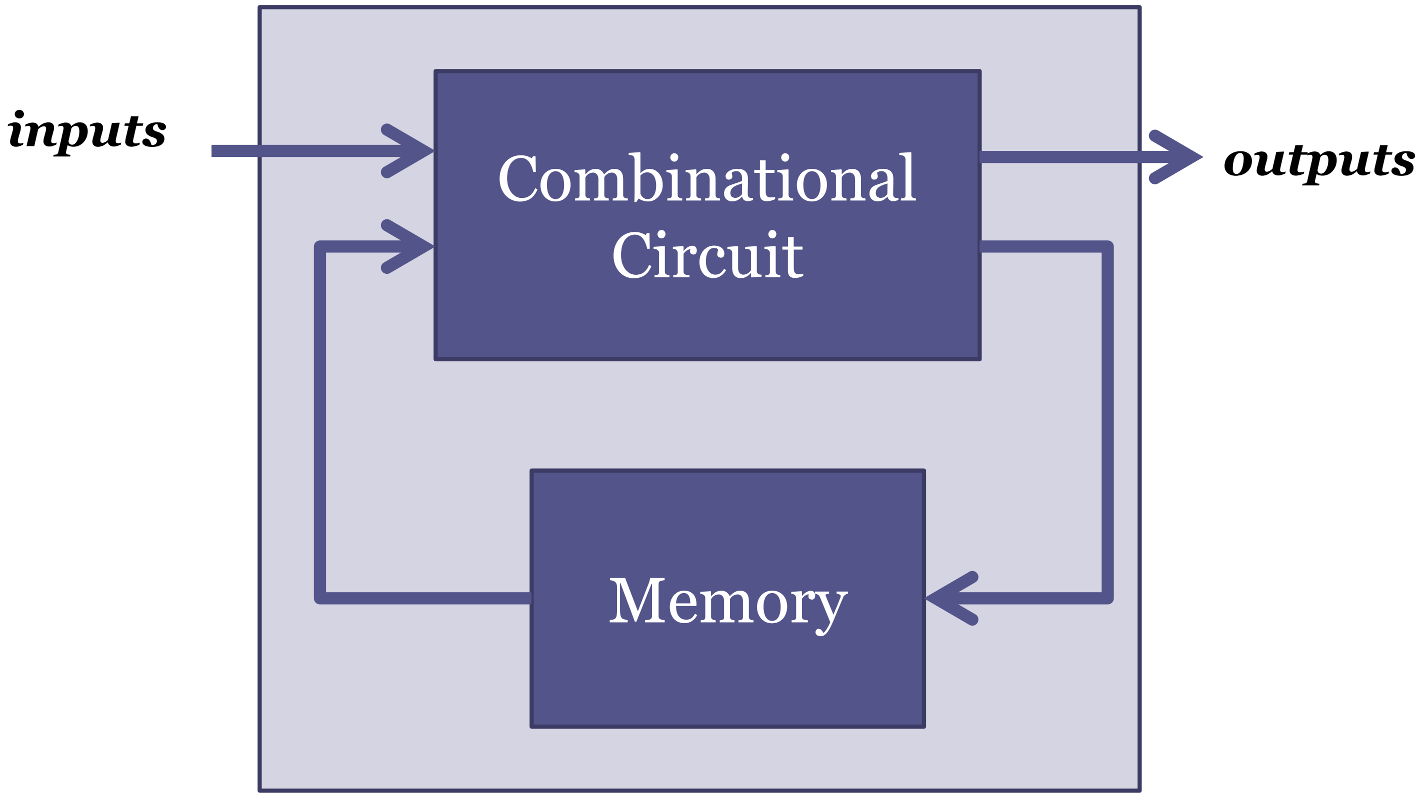
- A combinational circuit is a logic circuit whose output is solely determined by the inputs



Sequential Circuits

- Unlike a combinational circuit, the output of a sequential circuit at time *t* can affect the circuits output at time *t+1*

Sequential Circuits



Section 2

Basic Circuits

Section 2 Objectives

At the end of this section you will

1. Be able to build half-adder (HA) circuits
2. Understand HA limitations
3. Be able to build full-adder (FA) circuits
4. Chain combinational circuits
5. Be able to build decoders
6. Understand how to use decoders as “switches” to build mult-function circuits

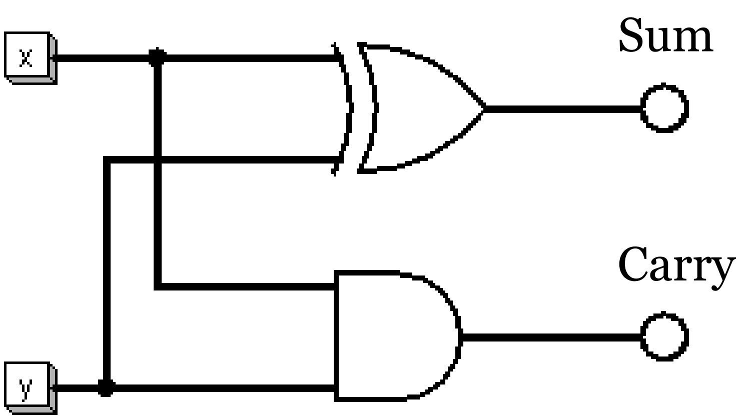
Half Adder (HA)

- Input: two bits, x and y
- Output: $x + y$

x	y	**Sum**	**Carry**
0	0	**0**	**0**
0	1	**1**	**0**
1	0	**1**	**0**
1	1	**0**	**1**

HA Circuit

x	y	Sum	Carry
0	0	**0**	**0**
0	1	**1**	**0**
1	0	**1**	**0**
1	1	**0**	**1**



- Sum = $x'y + y'x = x \oplus y$
- Carry = xy

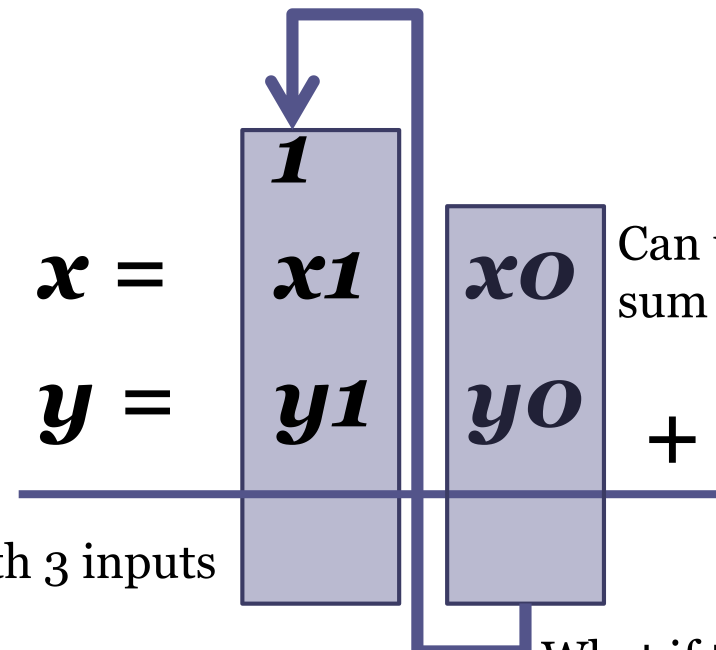
Source: DL2.circ (HA)

Limitation of a HA

- Can we build an adder for 2-bit numbers from the 1-bit HAs?

Limitation of a HA

- Can we build an adder for 2-bit numbers from the 1-bit HAs? Does not work if *xo* + *yo* produce a carry!



Can use a HA to produce sum and possibly a carry

Need an adder with 3 inputs
HA has only 2!

What if there is anon-zero carry?

Full Adder (FA)

- Input: three bits, x, y, and z
- Output: $x + y + z$

x	y	z	**Sum**	**Carry**
0	0	0	**0**	**0**
0	0	1	**1**	**0**
0	1	0	**1**	**0**
0	1	1	**0**	**1**
1	0	0	**1**	**0**
1	0	1	**0**	**1**
1	1	0	**0**	**1**
1	1	1	**1**	**1**

Full Adder – BSP for Carry

x	*y*	*z*	Sum	Carry	
0	0	0	**0**	**0**	
0	0	1	**1**	**0**	
0	1	0	**1**	**0**	
0	1	1	**0**	**1**	$x'yz$
1	0	0	**1**	**0**	
1	0	1	**0**	**1**	$xy'z$
1	1	0	**0**	**1**	xyz'
1	1	1	**1**	**1**	xyz

Carry Function

- $Carry = x'yz + xy'z + xyz' + xyz$

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

- $Carry = x(y'z + yz') + yz$
- $Carry = x(y \oplus z) + yz$

Full Adder – BSP for Sum

x	y	z	Sum	Carry	
0	0	0	**0**	**0**	
0	0	1	**1**	**0**	$x'y'z$
0	1	0	**1**	**0**	$x'yz'$
0	1	1	**0**	**1**	
1	0	0	**1**	**0**	$xy'z'$
1	0	1	**0**	**1**	
1	1	0	**0**	**1**	
1	1	1	**1**	**1**	xyz

Sum Function

- Sum = $x'y'z + x'yz' + xy'z' + xyz$
- ***Map is useless!!***

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

Sum Function

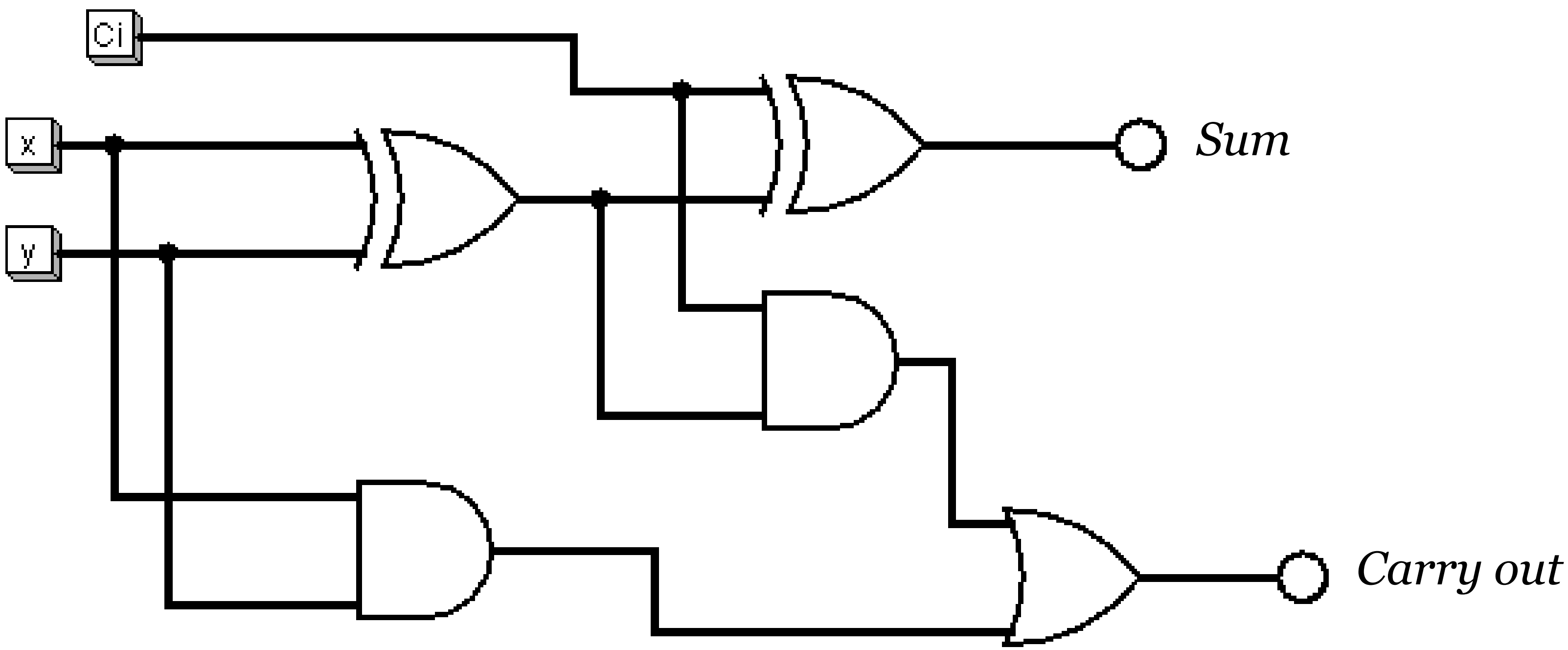
- Observation: $(a \oplus b)' = a'b' + ab$
- *Proof*:
  - $(a \oplus b)' = (a'b + b'a)'$ *xor definition*
  - $(a \oplus b)' = (a'b)'(b'a)'$ *DeMorgan's law*
  - $(a \oplus b)' = (a'' + b')(b'' + a')$ *DeMorgna's law*
  - $(a \oplus b)' = (a + b')(b + a')$ *Double negation*
  - $(a \oplus b)' = ab + a'a + b'b + a'b'$ *Distribution*
  - $(a \oplus b)' = ab + a'b'$ $xx' = 0,\ x+0=x$

Sum Function

- $Sum = x'y'z + x'yz' + xy'z' + xyz$
- $Sum = x'(y'z + yz') + x(y'z' + yz)$
- $Sum = x'(y \oplus z) + x(y \oplus z)'$
- $Sum = x \oplus y \oplus z$
  - $Let\ a = (y \oplus z), Sum = x'a + xa' = x \oplus a$

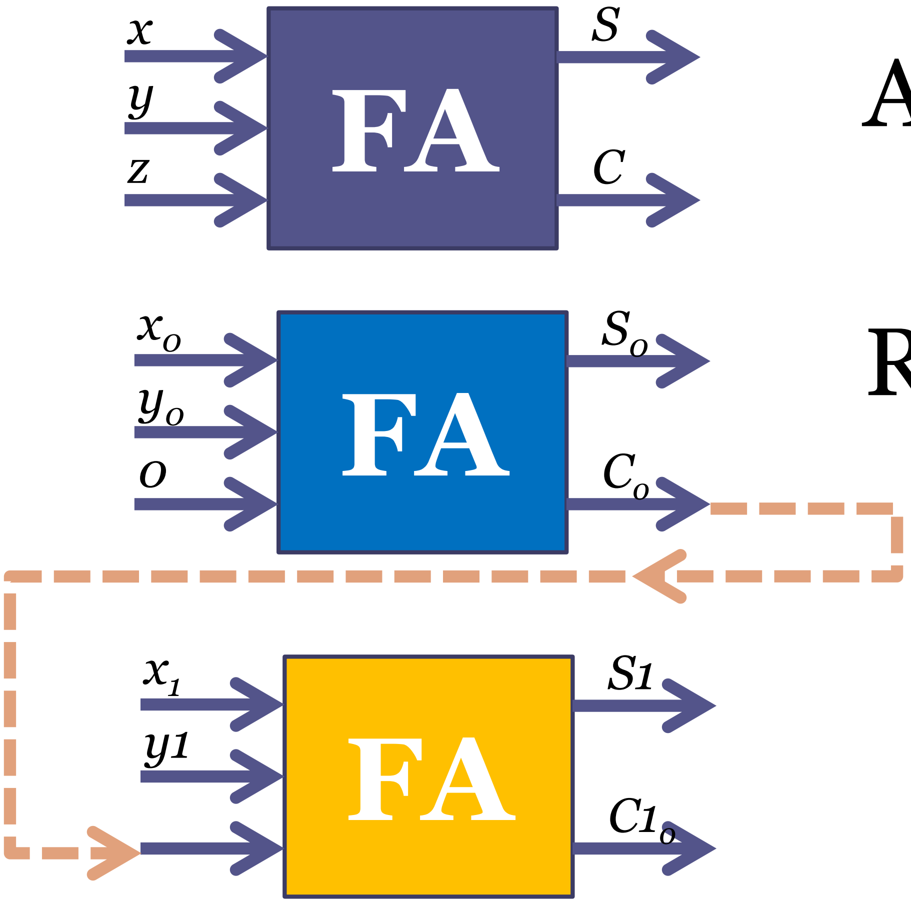
FA Circuit

- $Sum = x \oplus y \oplus z$
- $Carry = x(y \oplus z) + yz$



Source: DL2.circ (FA)

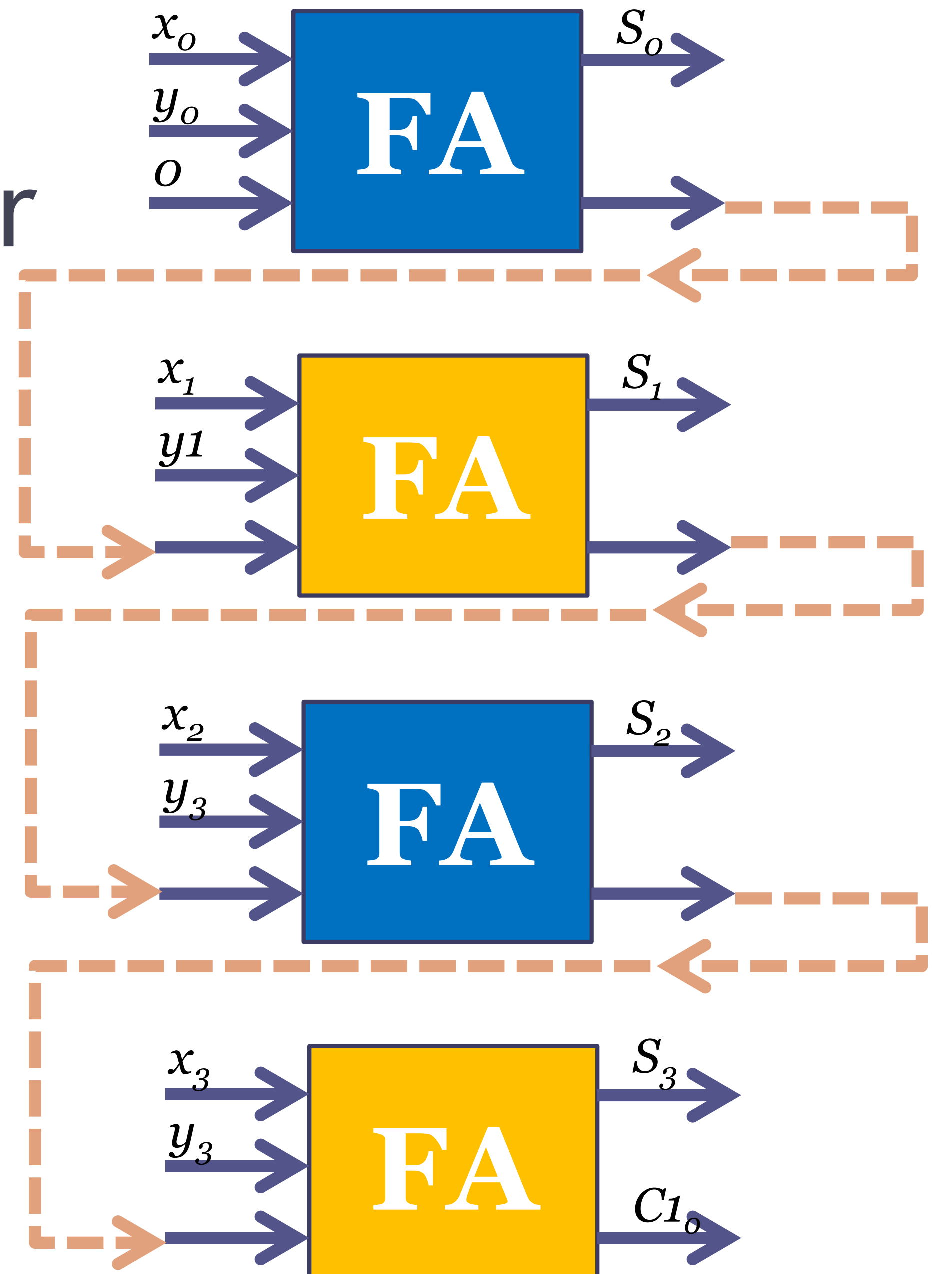
Chaining FAs (2-bit number FA)



Add x_1x_0 to y_1y_0

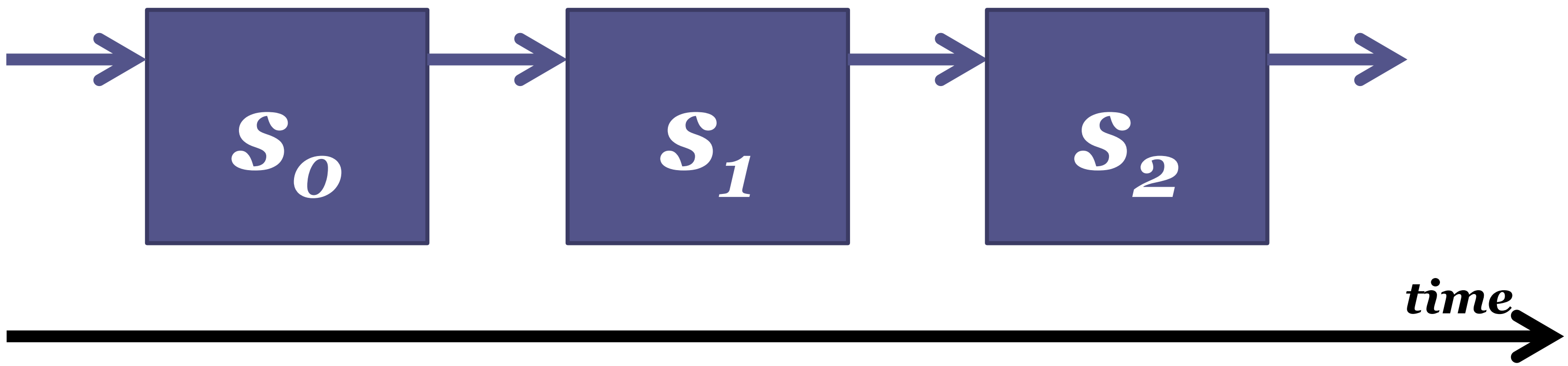
Result is $C_1S_1S_0$

4-bit Adder



Chaining

- Chaining is an easy way to build "higher-order" circuits from "lower-order" ones
- However, it is not the best
- The longer the chain the slower the circuit



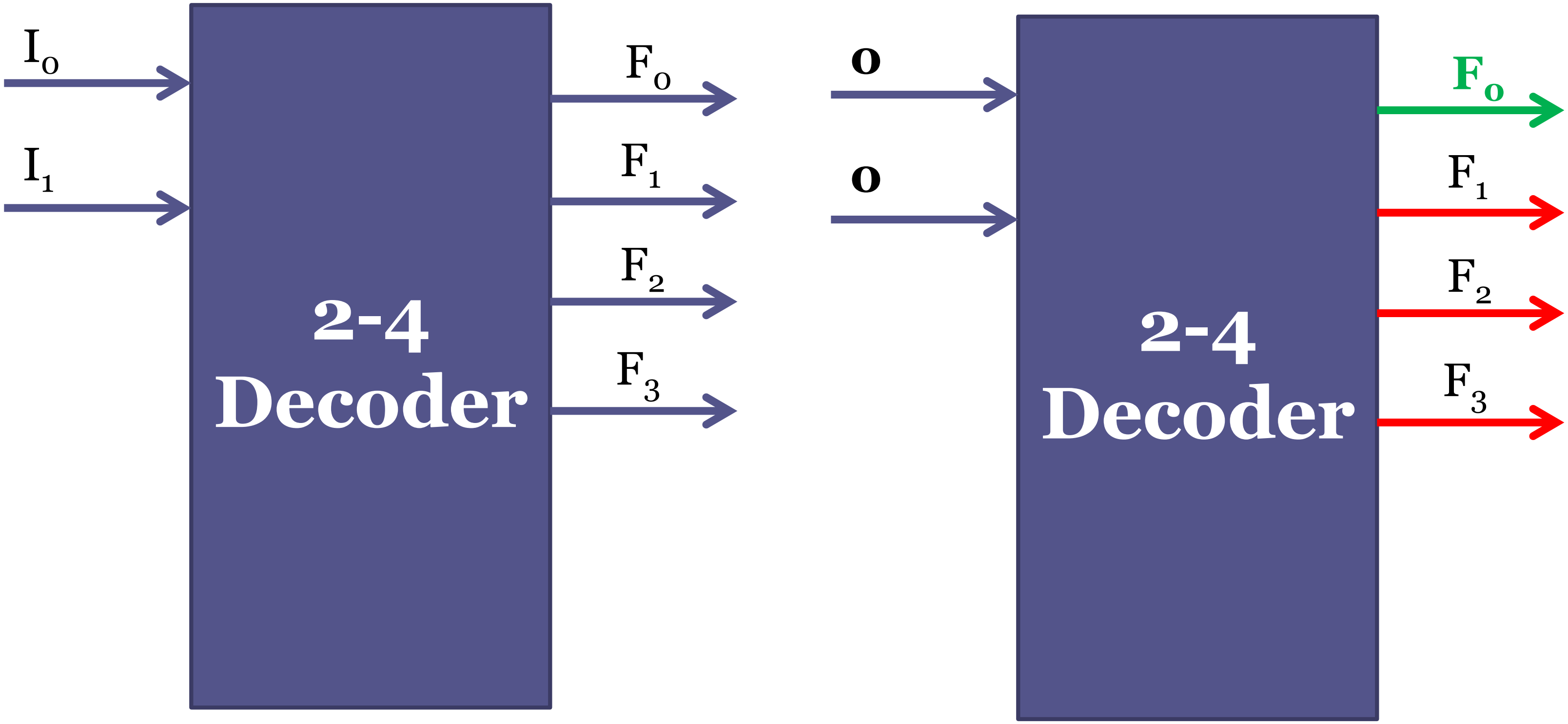
Full Subtractor

- Can be built from a FA
- $x - y = x\ (y \oplus 1) + 1$
- *Why?*

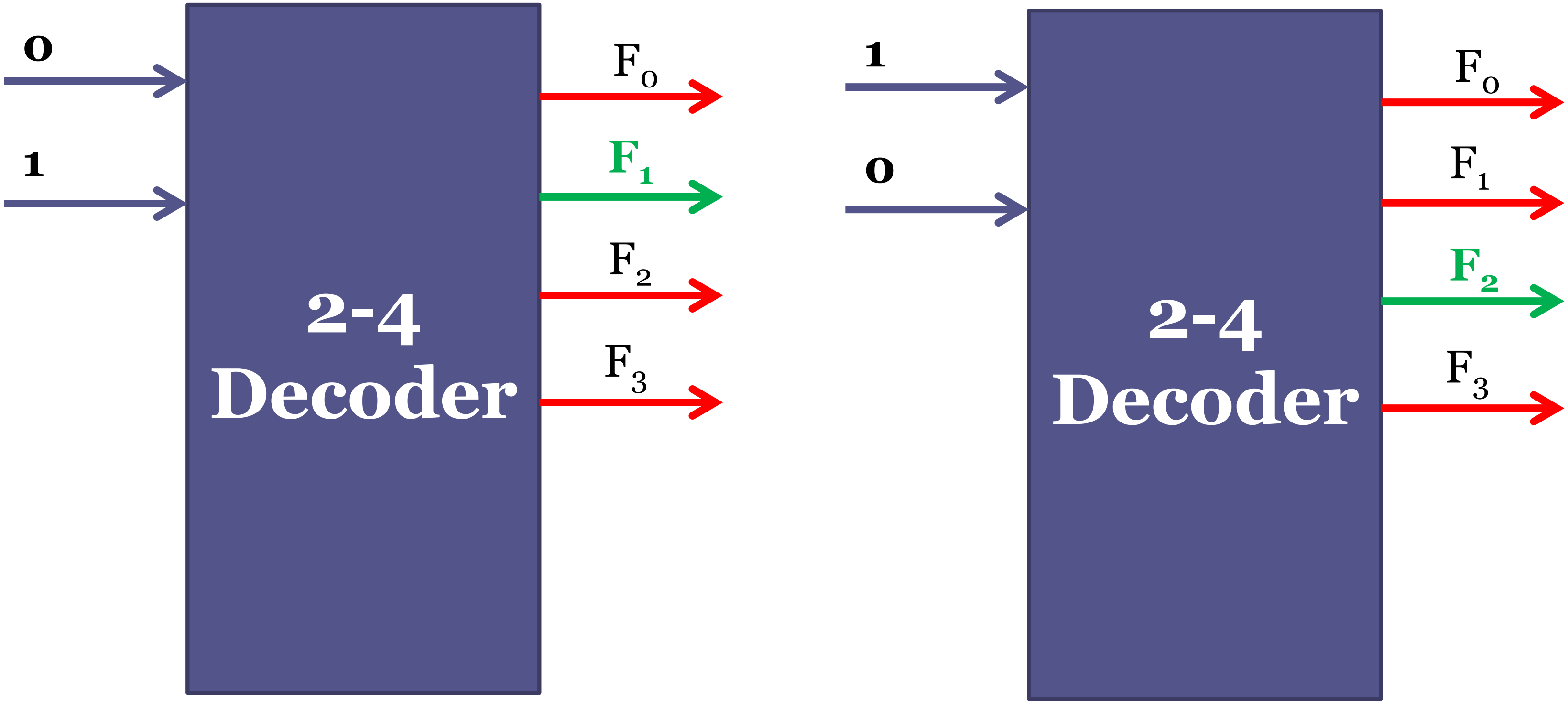
Decoders

- A decoder is a combinational circuit such that
  - It has n inputs
  - It has 2^n outputs
  - For each input a unique associated output line carries 1
  - All other lines carry 0
  - Let b_n be the binary number corresponding to the input bits
  - If the input is b_n, then line b_n is set (carries 1)

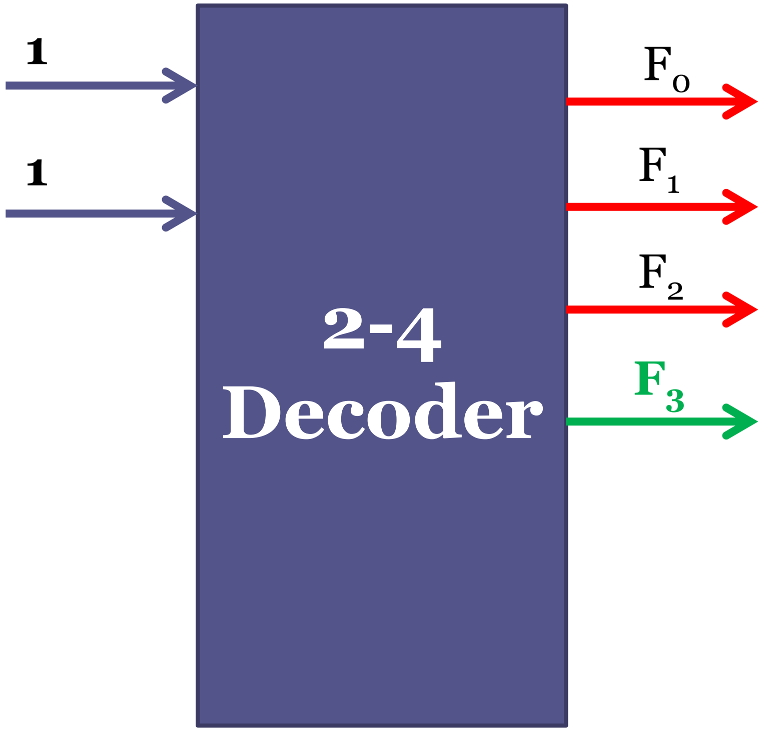
2-4 Decoder



2-4 Decoder



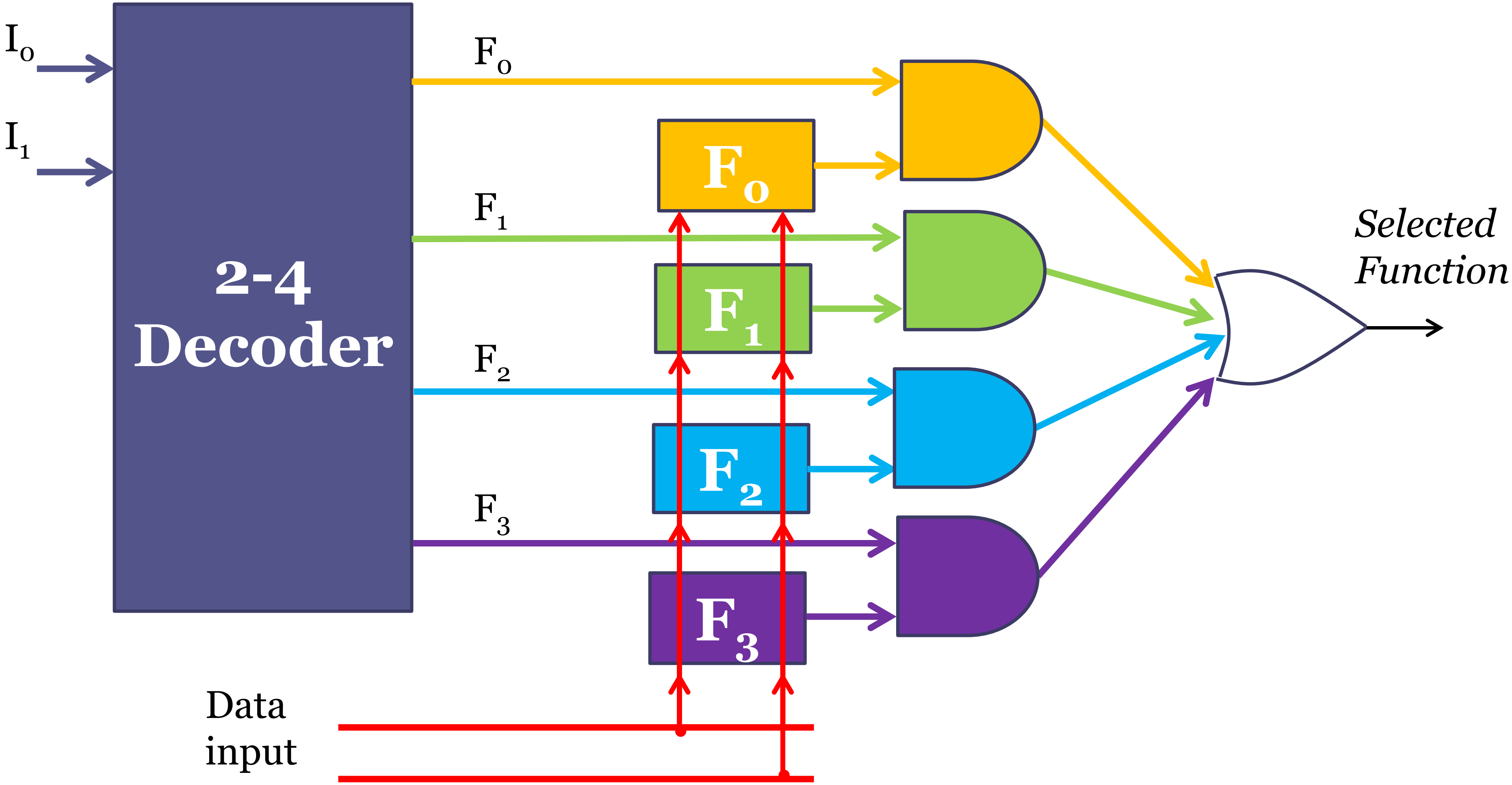
2-4 Decoder



Multi-Function Circuits

- Hardware runs all the time
- A multi-function circuit will compute all of its functions
- A decoder is used to “filter out” the functions that are not required

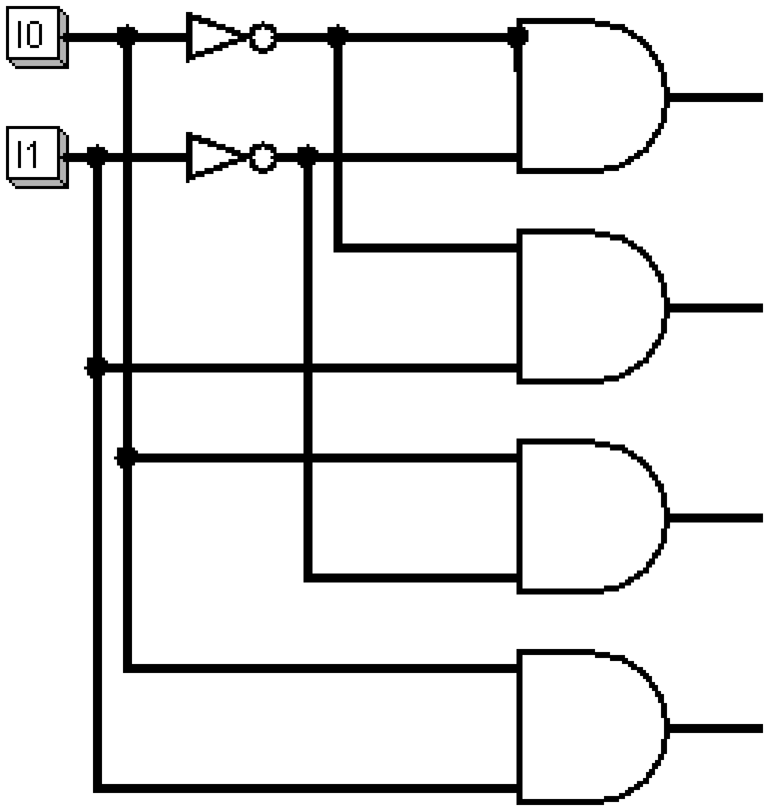
4-Function Circuit



2-4 Decoder (Truth Table)

x	y	F_0	F_1	F_2	F_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

2-4 Decoder (Circuit)



Source: DL2 (2-4 Decoder)

3-8 Decoder

- Exercise

Section 3

A Simple ALU

Section 3 Objectives

At the end of this section you will

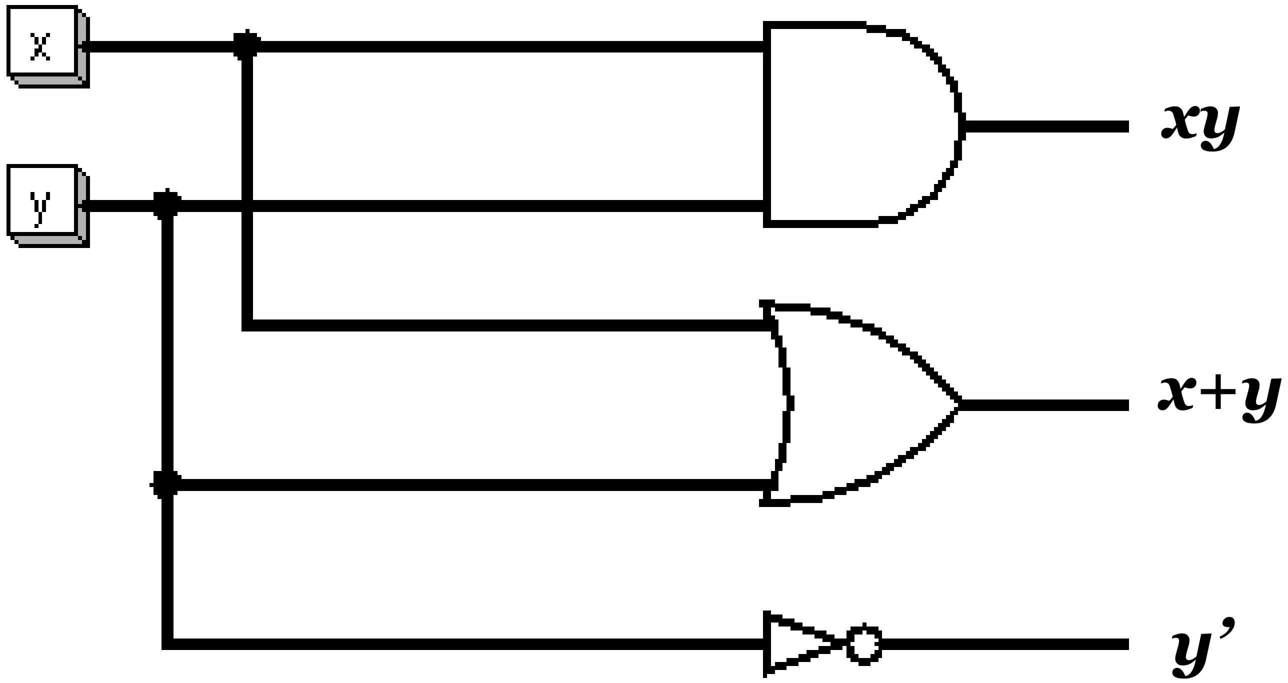
1. Build a simple 1-bit ALU
2. Use chaining to build higher order ALU

Bit Slice

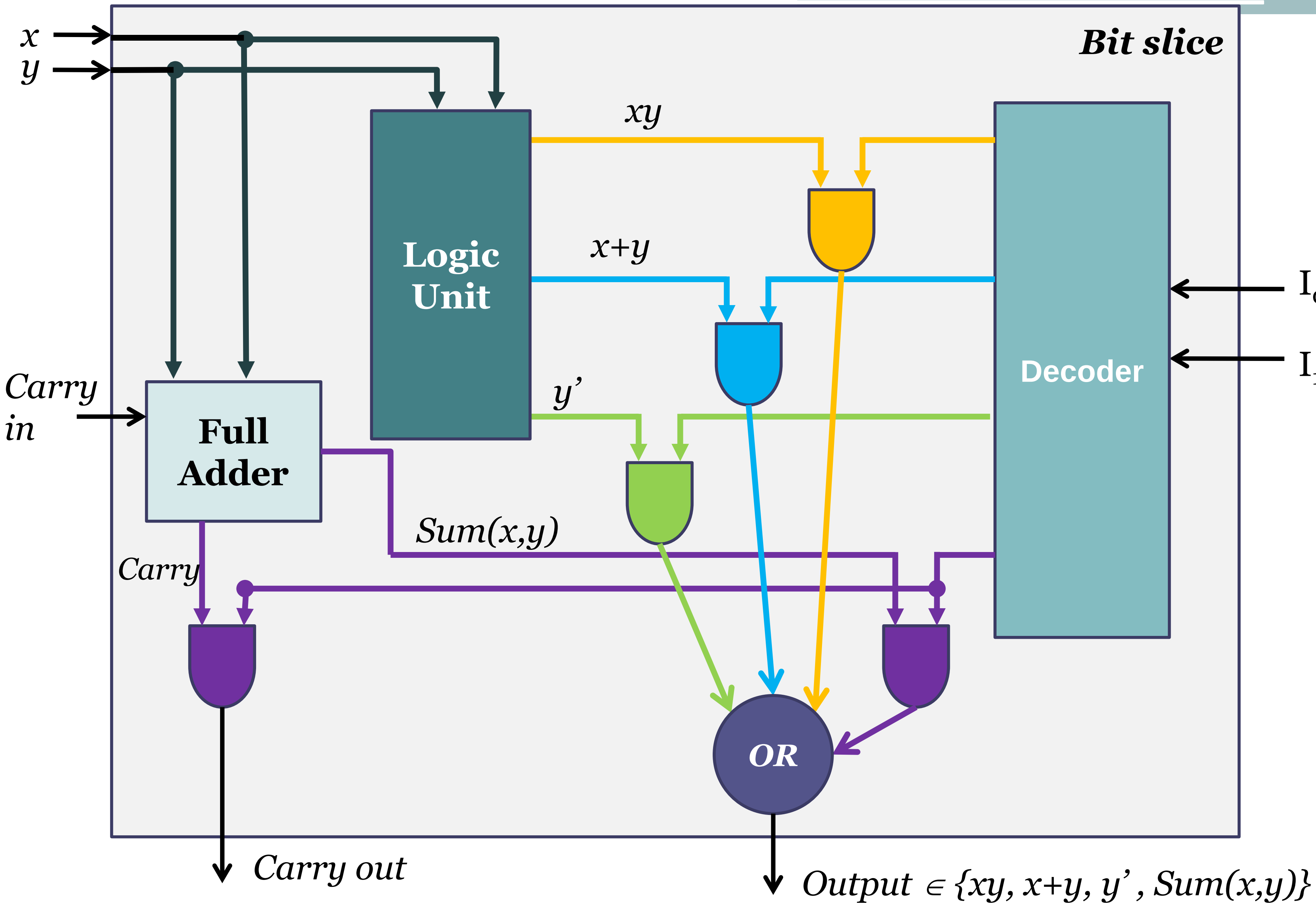
- A bit slice is a 1-bit ALU
- We will use chaining to create a larger ALU

- ALU can do
  - And
  - Or
  - Not
  - Addition
  - Only

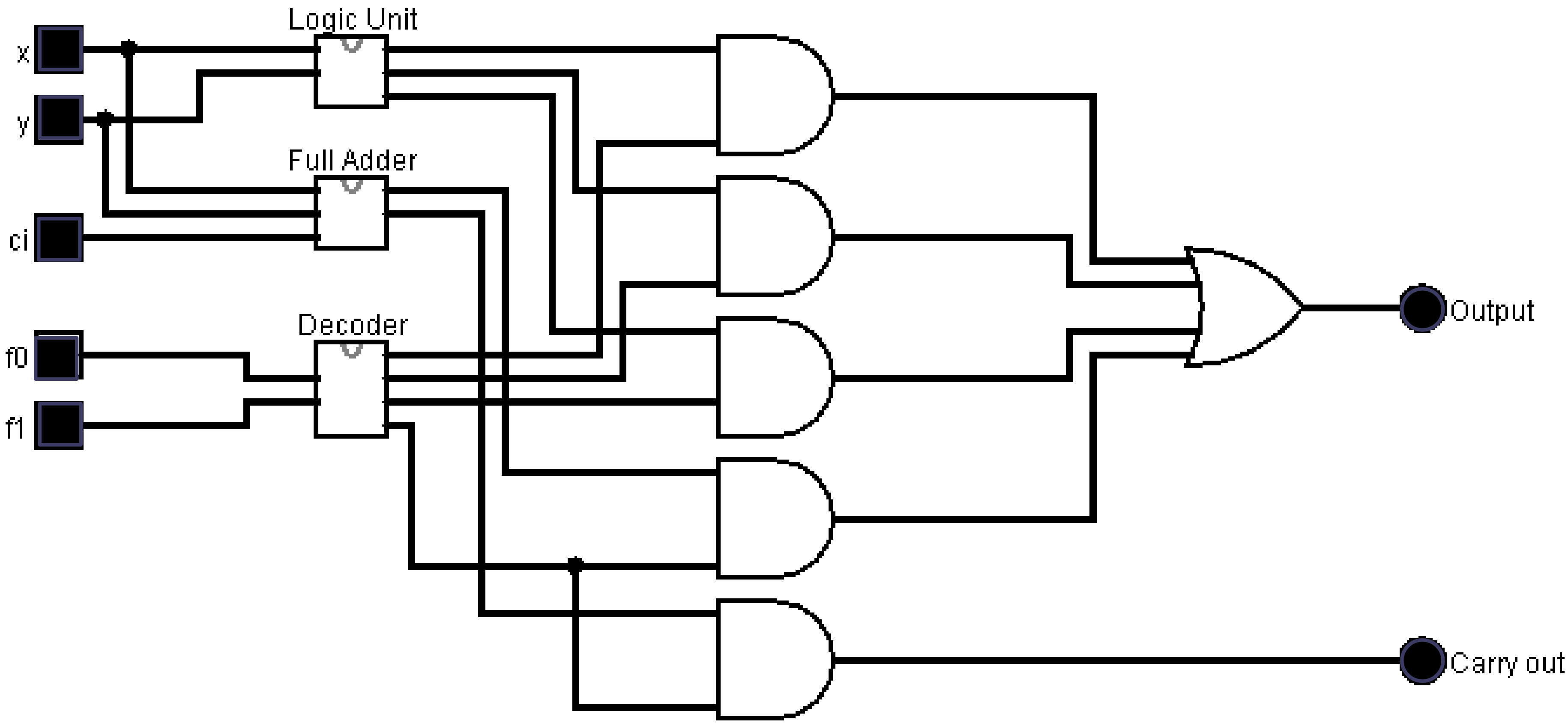
Logic Unit



Source: DL2 (Logic Unit)



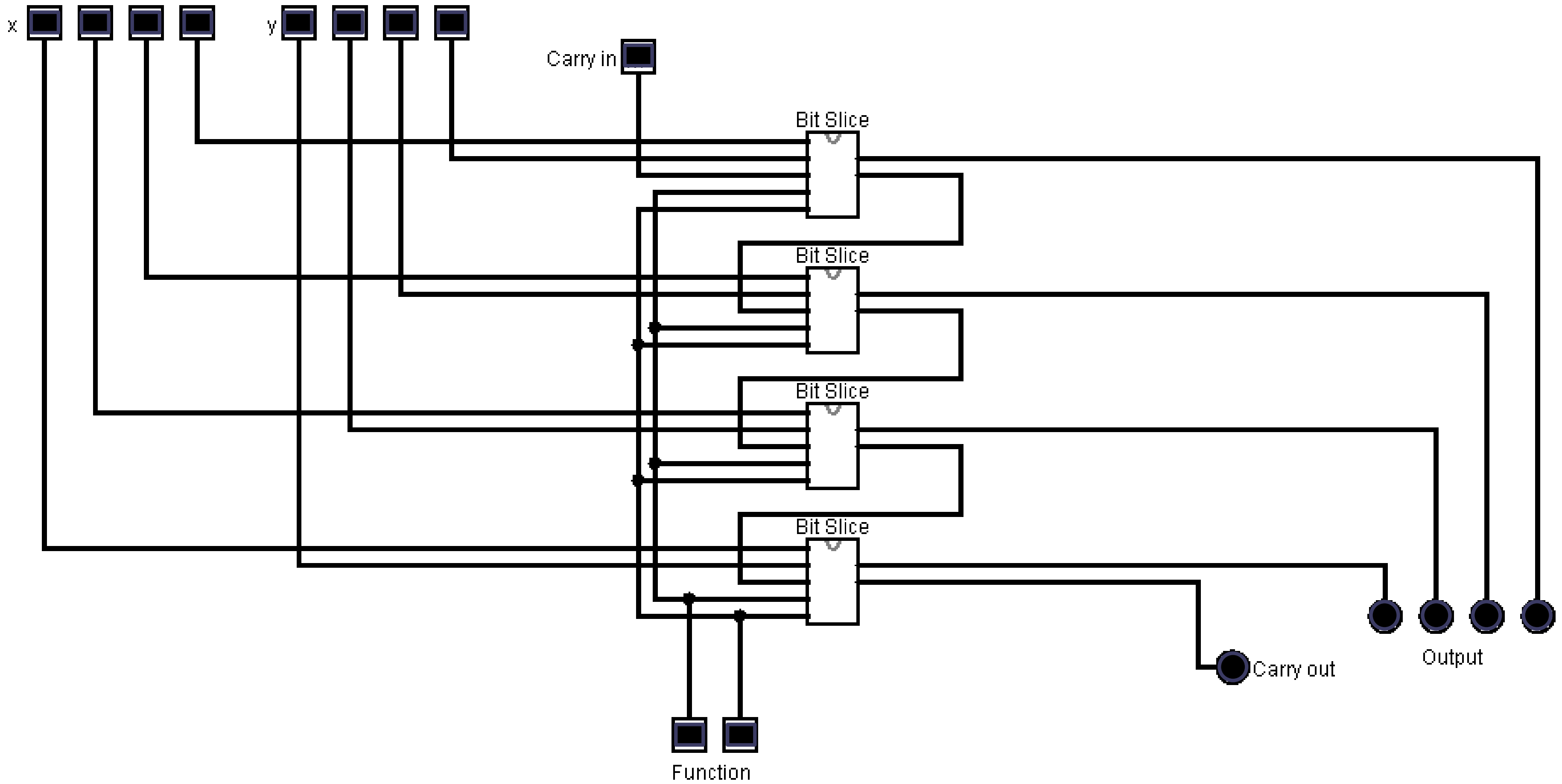
x
y
Bit slice
Logic Unit
xy
x+y
y'
Decoder
I0
I1
Carry in
Full Adder
Sum(x,y)
Carry
OR
Carry out
Output ∈ {xy, x+y, y', Sum(x,y)}



Bit slice demo

Source: DL2.circ (Bit Slice)

4-bit ALU (by chaining)



Source: DL2 (4-bit ALUS)

Section 4

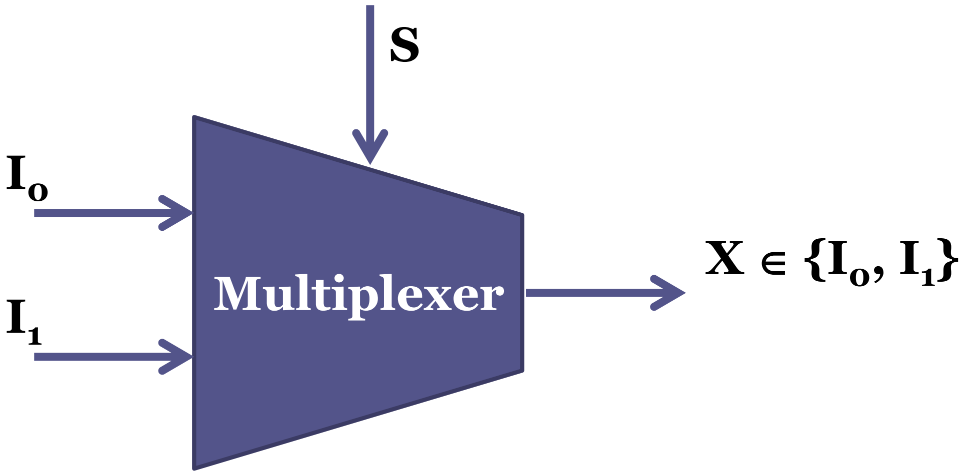
More Combinational Circuits

Section 4 Objective

At the end of this section you will

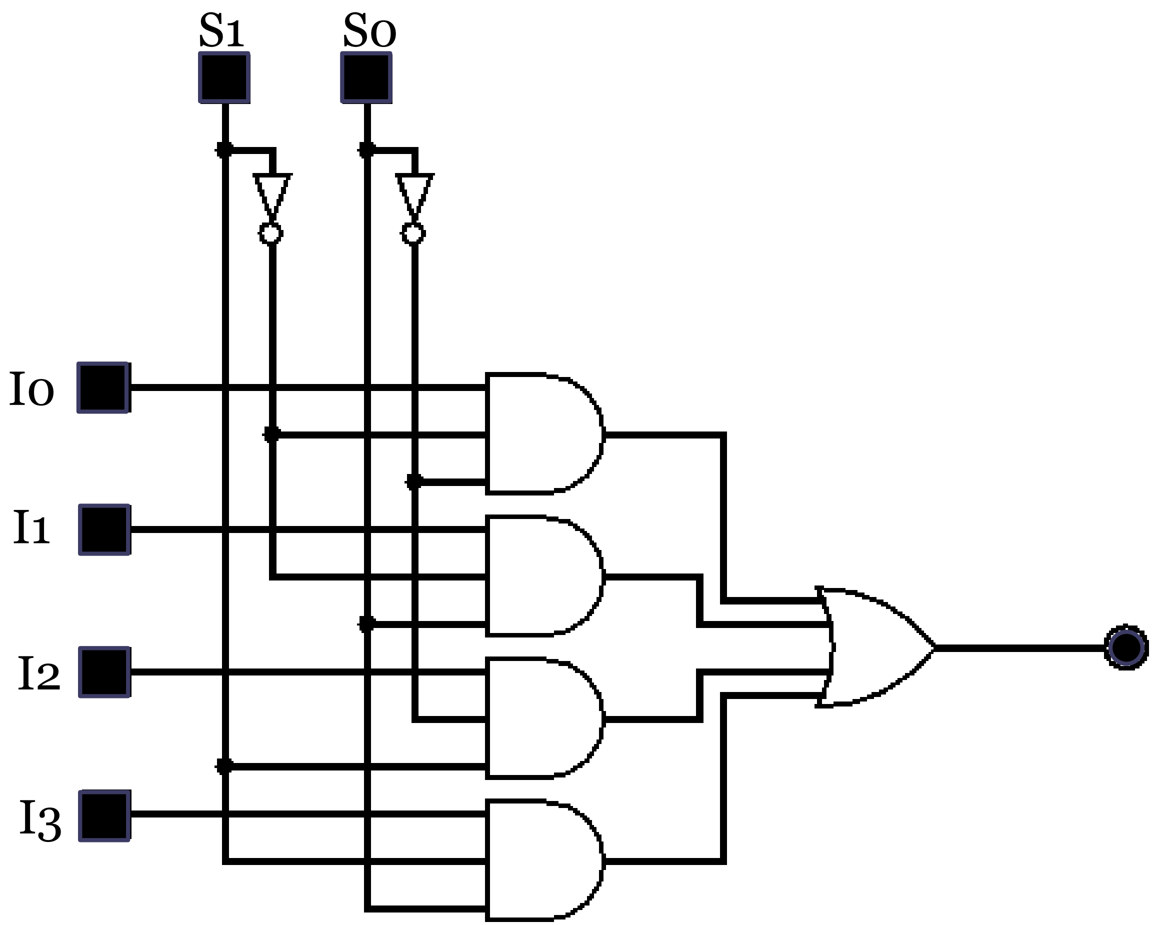
1. Gain more practice with combinational circuits

Multiplexers



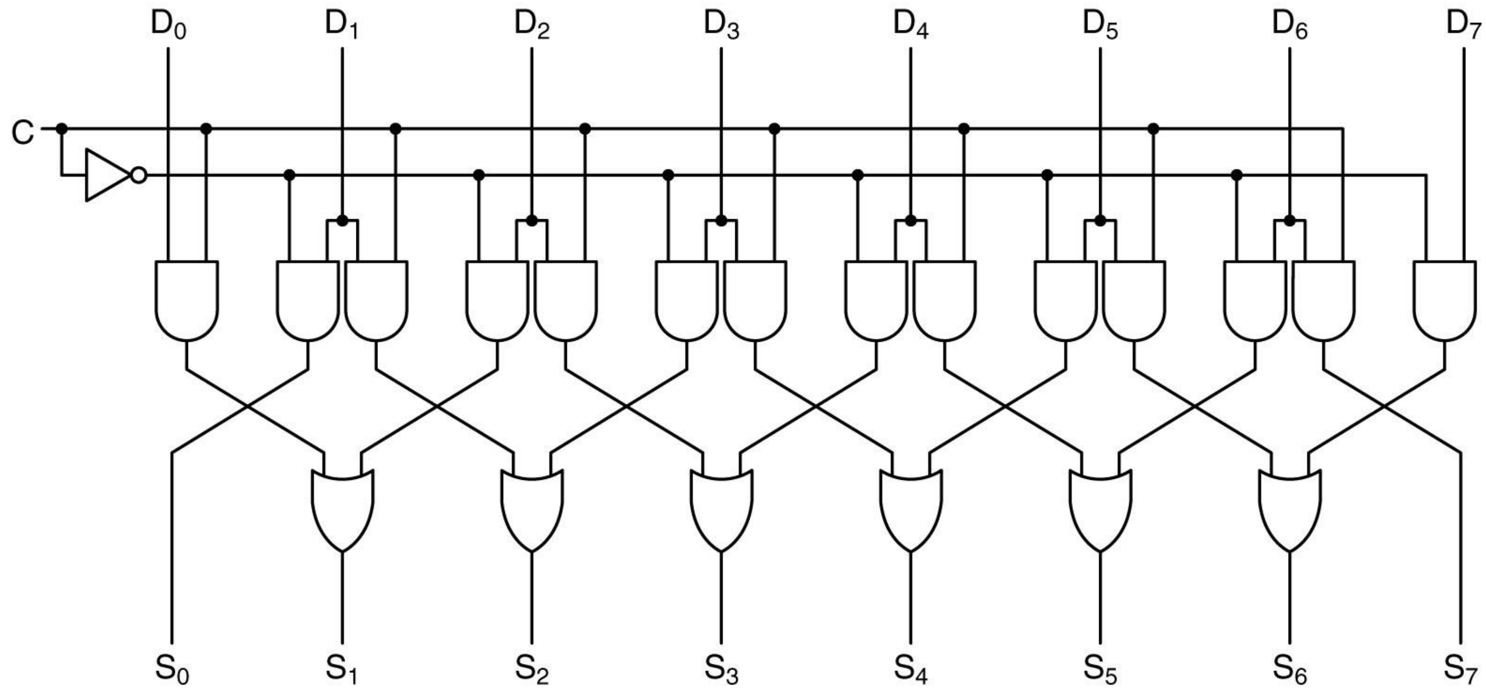
- If $S = 0$, $X = I_0$
- If $S = 1$, $X = I_1$

4-Input Multiplexer



Source: D2l.circ (4-input Multiplexer)

Shifters



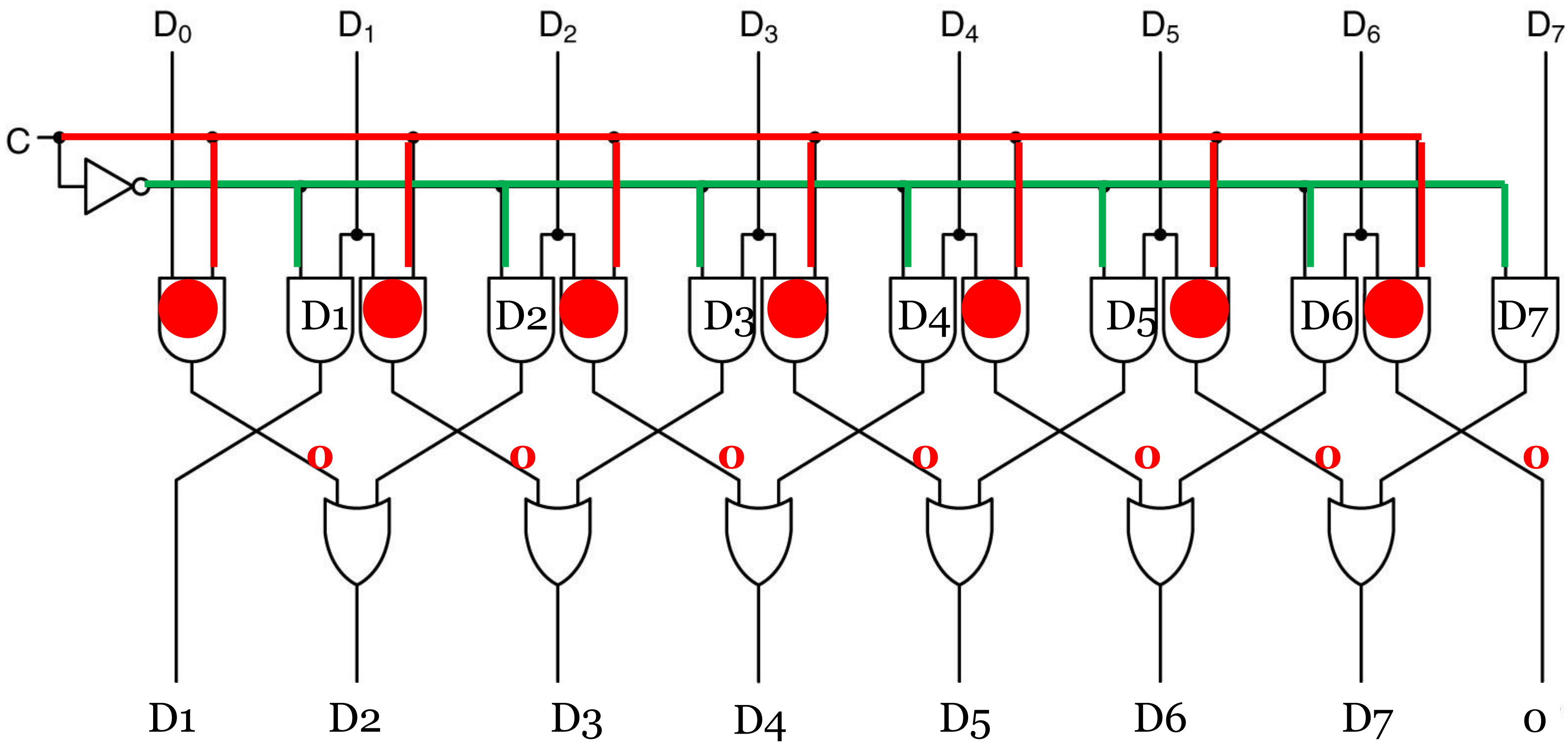
1-bit left/right shifter.
C = 0 for left 1 for right

$C = 0$

$D_0 D_1 D_2 D_3 D_4 D_5 D_6 D_7$

$D_1 D_2 D_3 D_4 D_5 D_6 D_7\ 0$

Left Shift

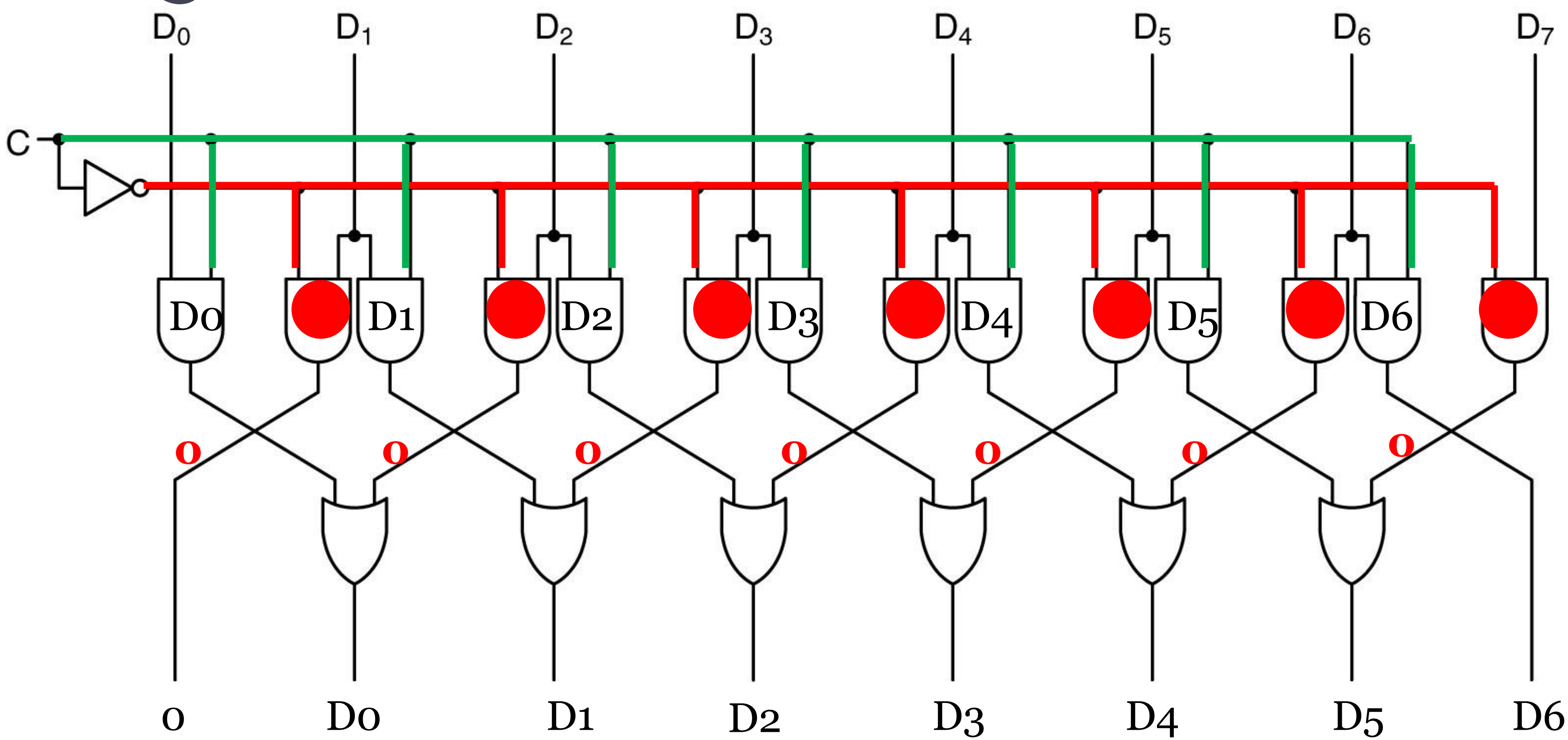


Right Shift

C = 1

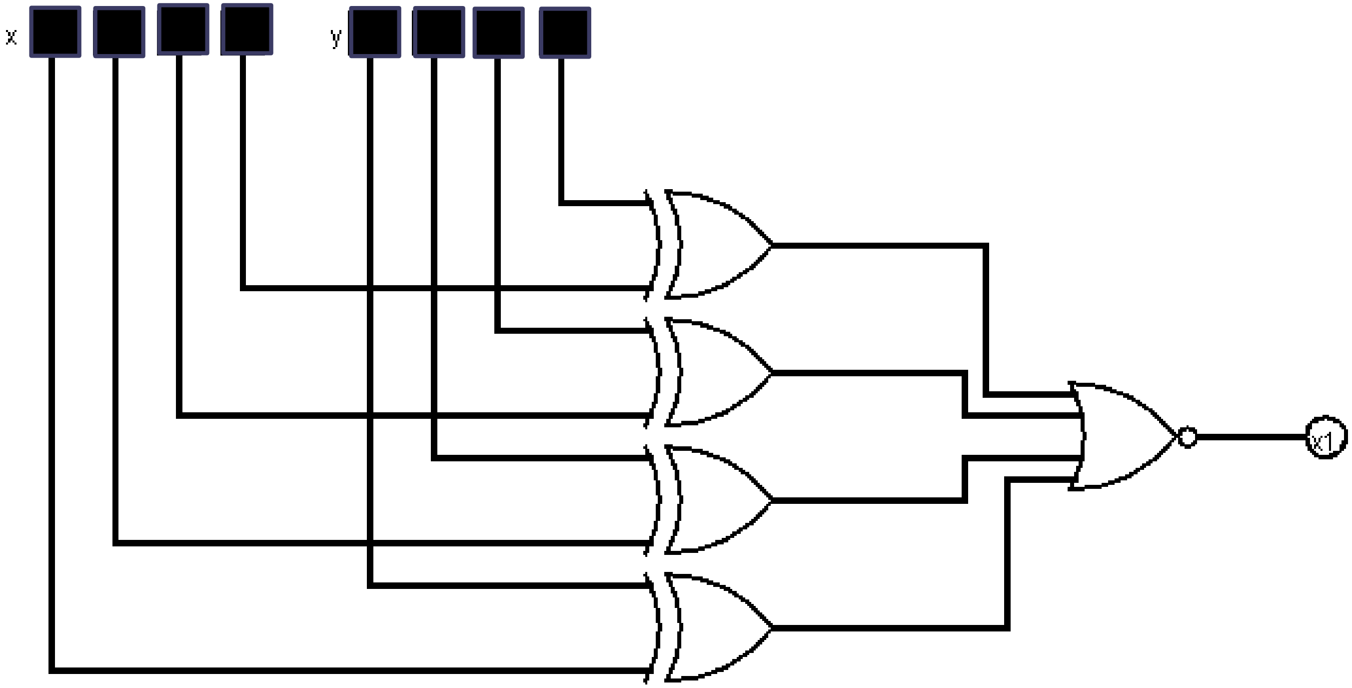
$D_0D_1D_2D_3D_4D_5D_6D_7$

0 $D_0D_1D_2D_3D_4D_5D_6$



Comparators

- Two bits can be compared with a XOR gate
- Two numbers can be compared bit-wise



Magnitude Comparator

- *Challenge* yourself with designing a circuit that inputs 2-bit binary numbers A & B
- It has 3 outputs:
  - First output is 1, iff the numbers are equal
  - Second is 1, iff A < B
  - Third is 1, iff A > B

- Repeat for 4 bit numbers

Multipliers

- Multiplication can be done by addition
- To multiply X and Y, add Y to itself X times; or add X to itself Y times

- Challenge yourself to build a “direct” multiplier
  - i.e., do not use the method described above
  - Hint: fix the size of your inputs; e.g., multiply 2-bit by a 3-bit number

Self-Test Quiz

- Challenge yourself to build a “direct” multiplier that multiplies 2-bit by a 3-bit number