Combinational Logic Circuits

Hyunok Oh

XOR and XNOR

- XOR : Exclusive OR
- XNOR : Exclusive NOT-OR

$$x \oplus y = (x \wedge y') \vee (x' \wedge y) = \begin{cases} 1 & \text{when } x \neq y \\ 0 & \text{otherwise} \end{cases}$$

$$x \odot y = (x \wedge y) \vee (x' \wedge y') = \begin{cases} 1 & \text{when } x = y \\ 0 & \text{otherwise} \end{cases}$$

XOR		
x	y	x⊕y
0	0	0
0	1	1
1	0	1
1	1	0

XNOR		
x	y	x⊙y
0	0	1
0	1	0
1	0	0
1	1	1

NOR and NAND

- NOR : NOT-OR

$$(x \vee y)'$$

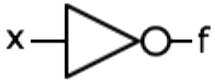
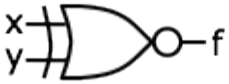
- NAND : NOT-AND

$$(x \wedge y)'$$

NOR		
x	y	$(x \vee y)'$
0	0	1
0	1	0
1	0	0
1	1	0

NAND		
x	y	$(x \wedge y)'$
0	0	1
0	1	1
1	0	1
1	1	0

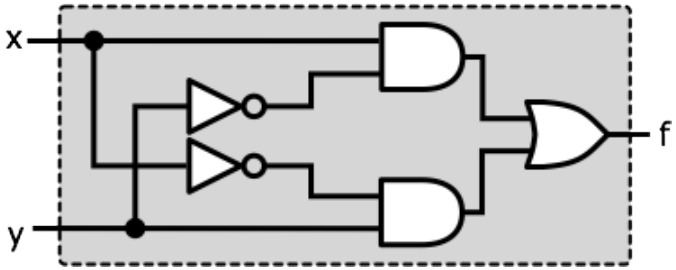
Logic Gates

Name	Symbol	Function
NOT		$f = x'$
OR		$f = x \vee y$
AND		$f = x \wedge y$
NOR		$f = (x \vee y)'$
NAND		$f = (x \wedge y)'$
XOR		$f = x \oplus y$
XNOR		$f = x \odot y$

Combination of Logic Gates

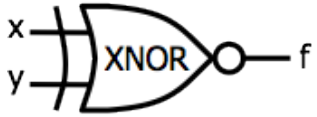
- More complex logic gates can be constructed by combining the basic logic gates

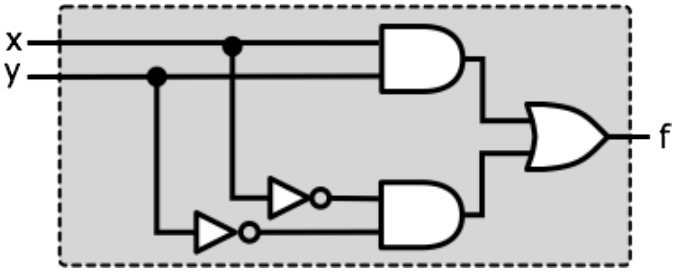
- eg) Implement XOR using AND, OR, NOT gates



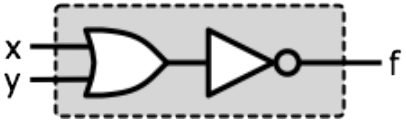
Combination of Logic Gates(Continue)

- E.g.) XNOR using AND, OR, NOT gates

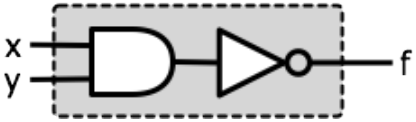




- E.g.) NOR using OR, NOT gates



- E.g.) NAND using AND, NOT gates

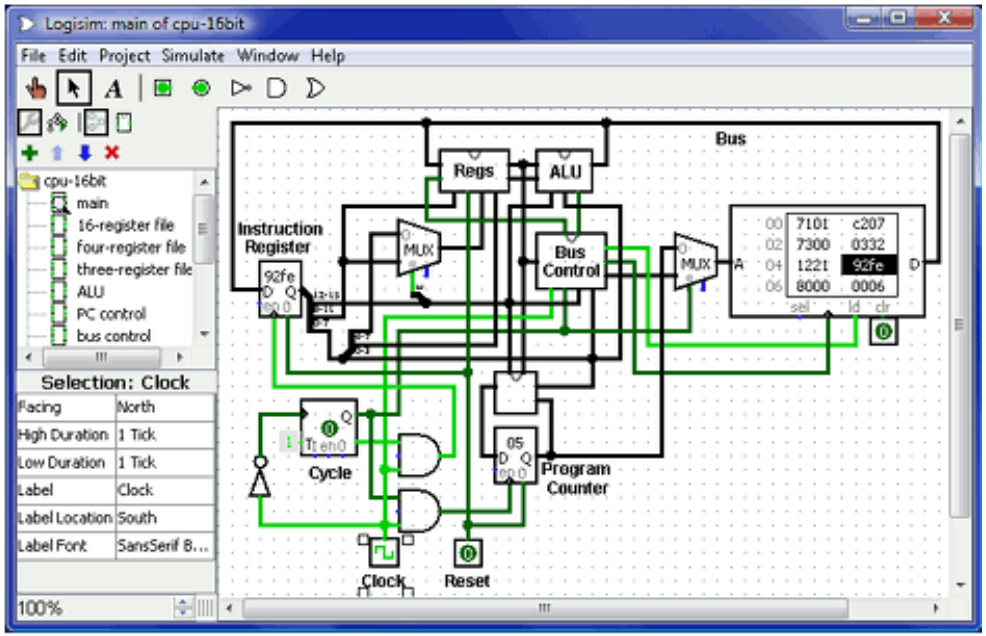


Bus

- More than one signal lines
- 두 개 이상의 신호라인의 모음을 버스라고 함

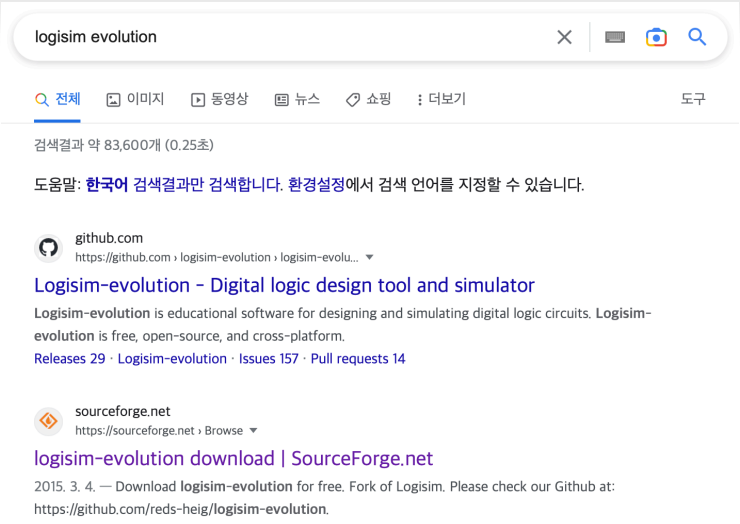
Logisim

- A software tool to design a logic circuit and simulate it

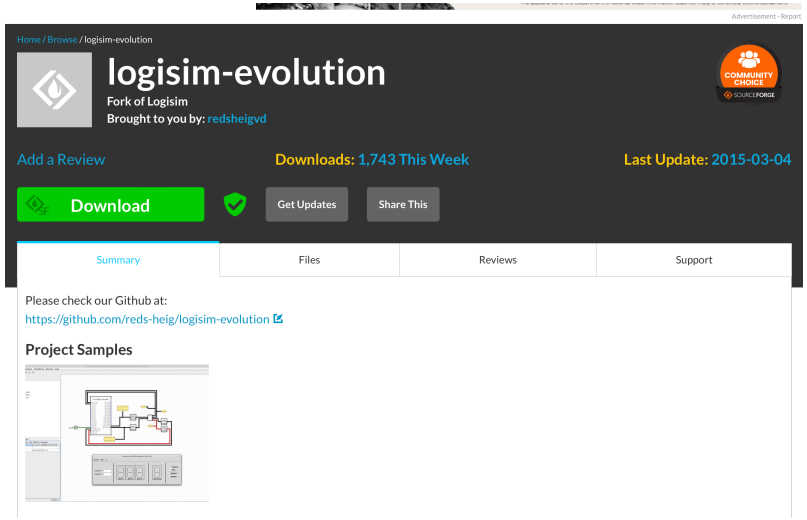


Installation of Logisim-evolution

1. Search



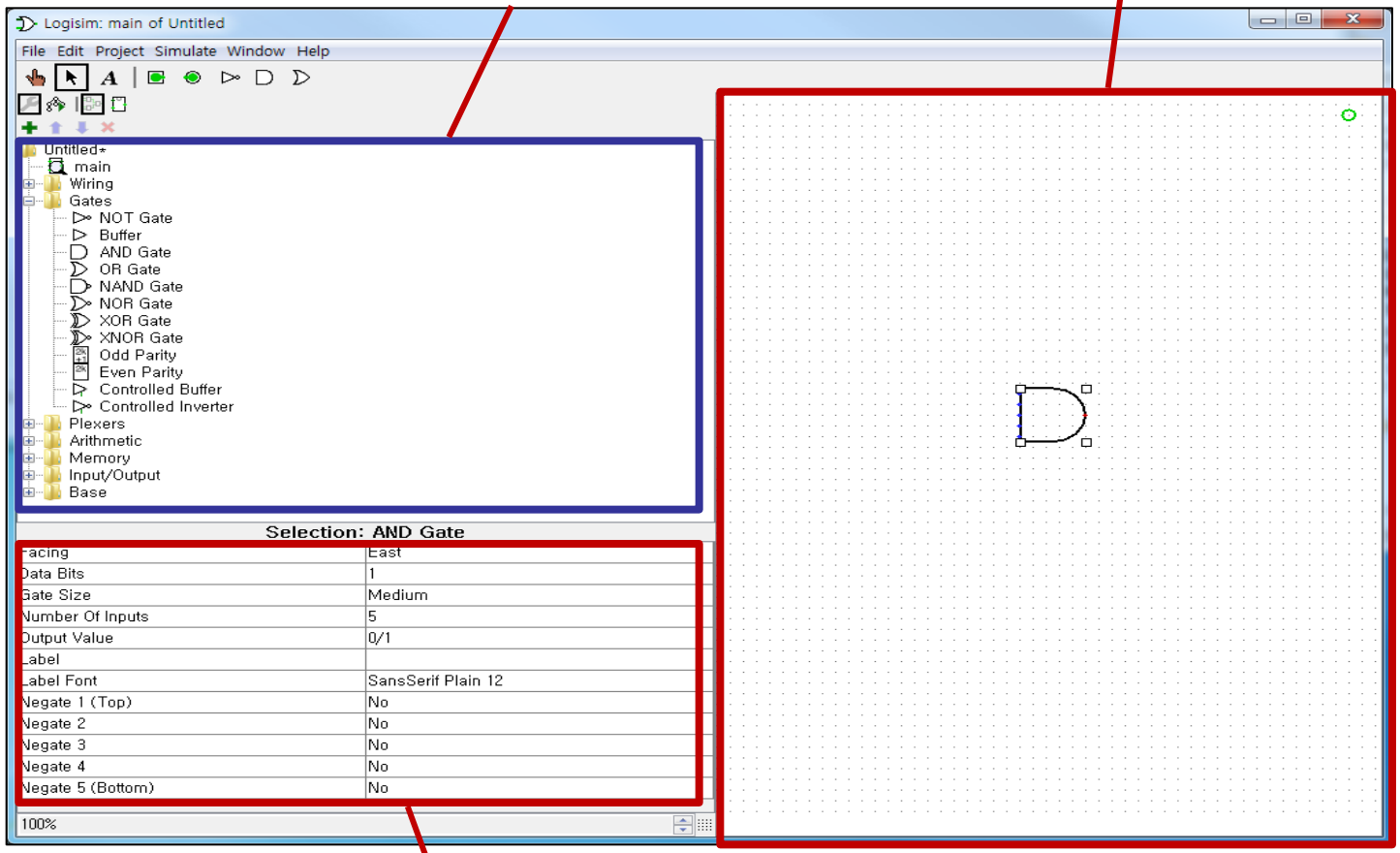
2. Download a file



Logisim Screen

Logic gates provided by Logisim

Schematic editor

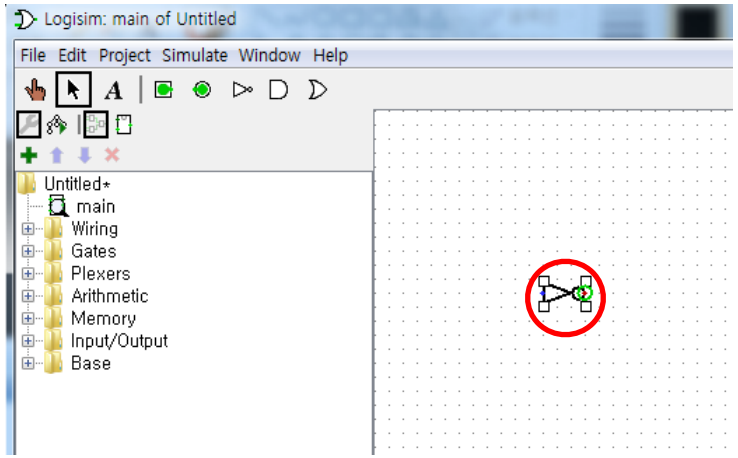


characteristics of a gate

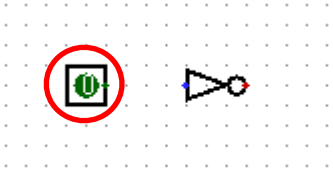
Toolbar Menu

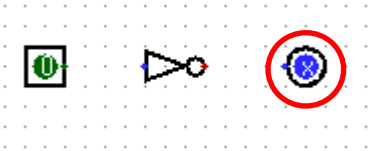
	Change values within circuit
	Edit selection and add wires
	Edit text in circuit
	Add Pin for input
	Add Pin for output
	Add (NOT, AND, OR) gate

Drawing a Gate



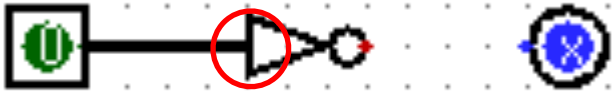
Input/Output

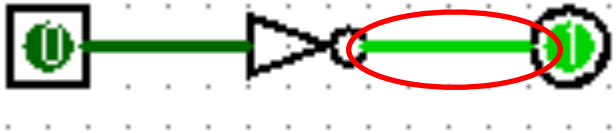




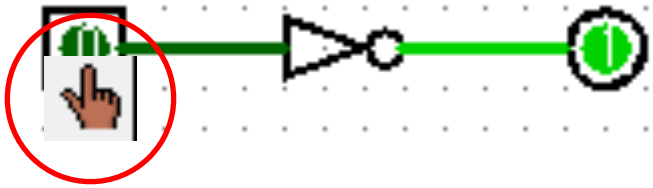
Connection

클릭한 다음 끌기 click and drag

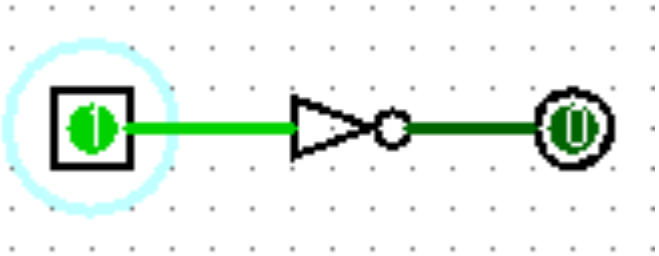




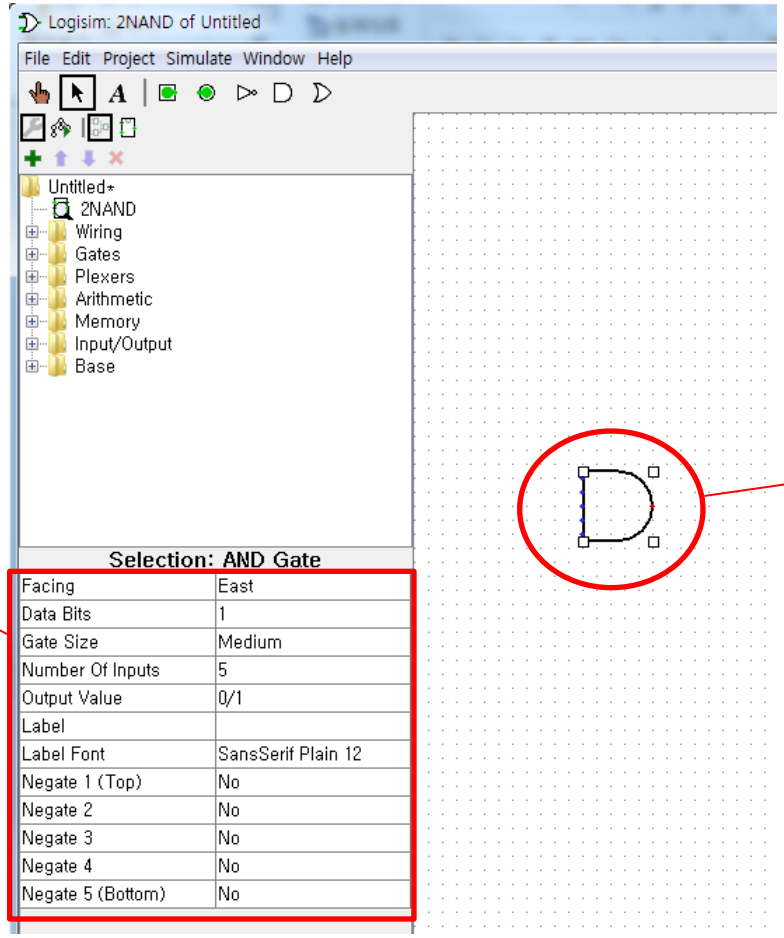
Input Toggling



클릭 Click



Change the Characteristics of a Gate



1. 클릭! Click

2. 특성 수정

Characterics

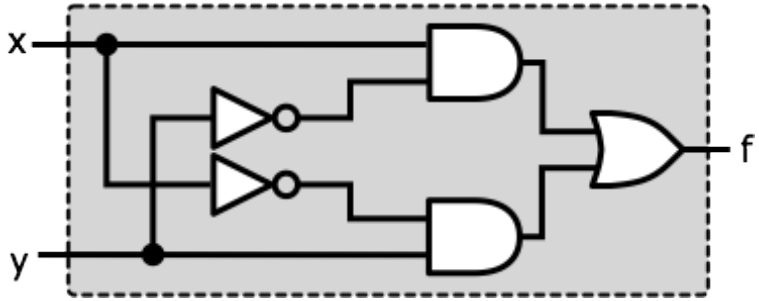
Modification

Facing	East	해당 개체의 방향(90도 씩 회전한 것과 같은 효과)
Data Bits	1	Data의 bit 수
Gate Size	Medium	회로에서 보이는 개체의 크기
Number Of Inputs	5	Input pin의 개수
Output Value	0/1	Output 값의 종류(0/1로 고정)
Label		개체의 이름
Label Font	SansSerif Plain 12	개체의 이름을 보여주는 폰트
Negate 1 (Top)	No	각 Input을 negate 시키는지 여부
Negate 2	No	
Negate 3	No	
Negate 4	No	
Negate 5 (Bottom)	No	

Summary : H/W

- Verify the functionality of basic logic gates of AND, OR, NOT in Logisim

- Implement XOR gate using AND, OR, NOT gates in Logisim

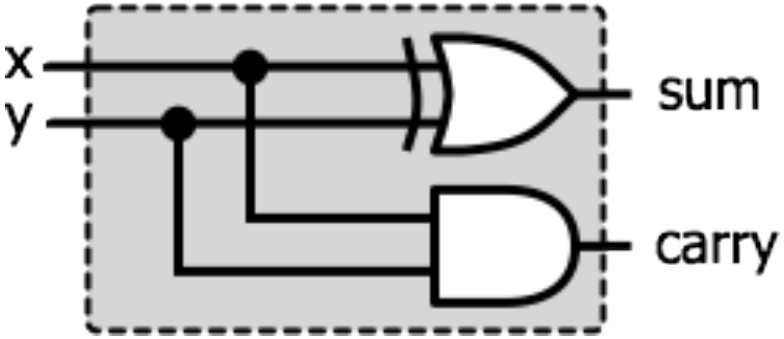


Half Adder

- A gate adding a single bit x and y
  - Outputs : sum, carry

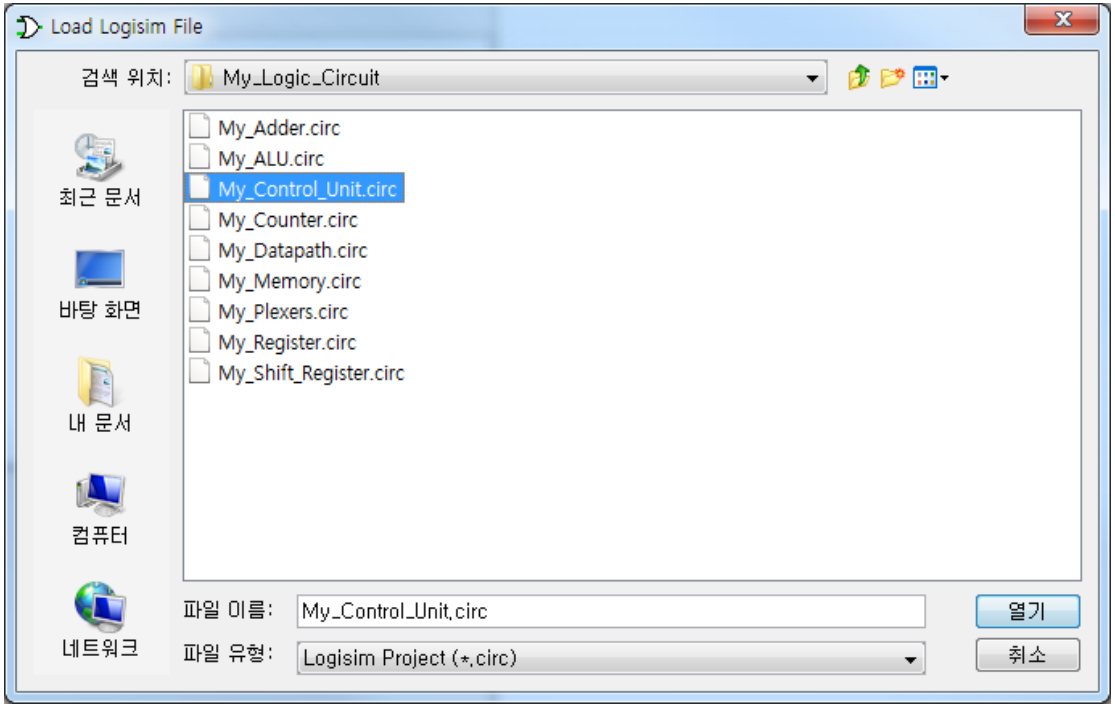
$$\begin{aligned} \text{sum} &= x \oplus y \\ \text{carry} &= x \wedge y \end{aligned}$$

x	y	carry	sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

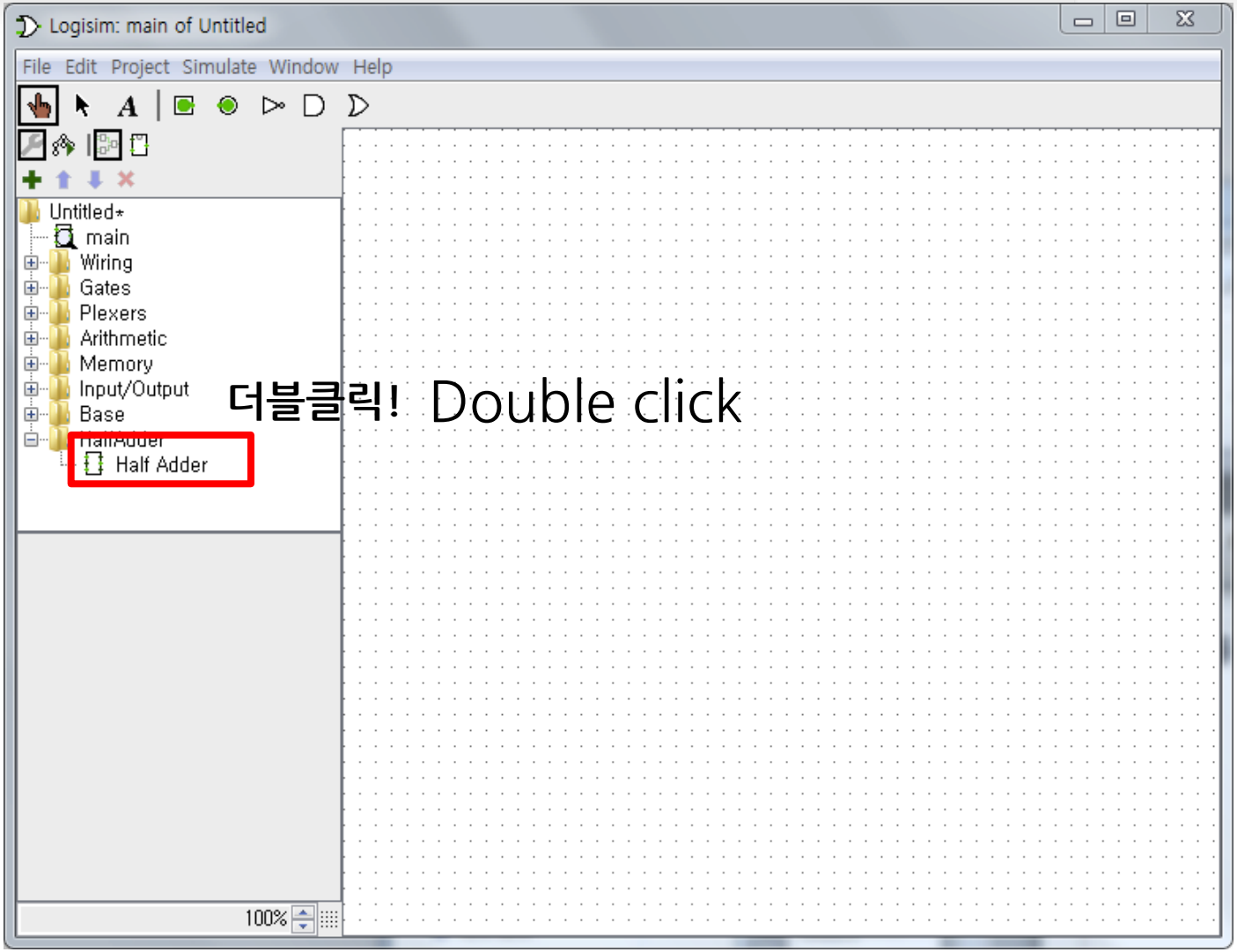


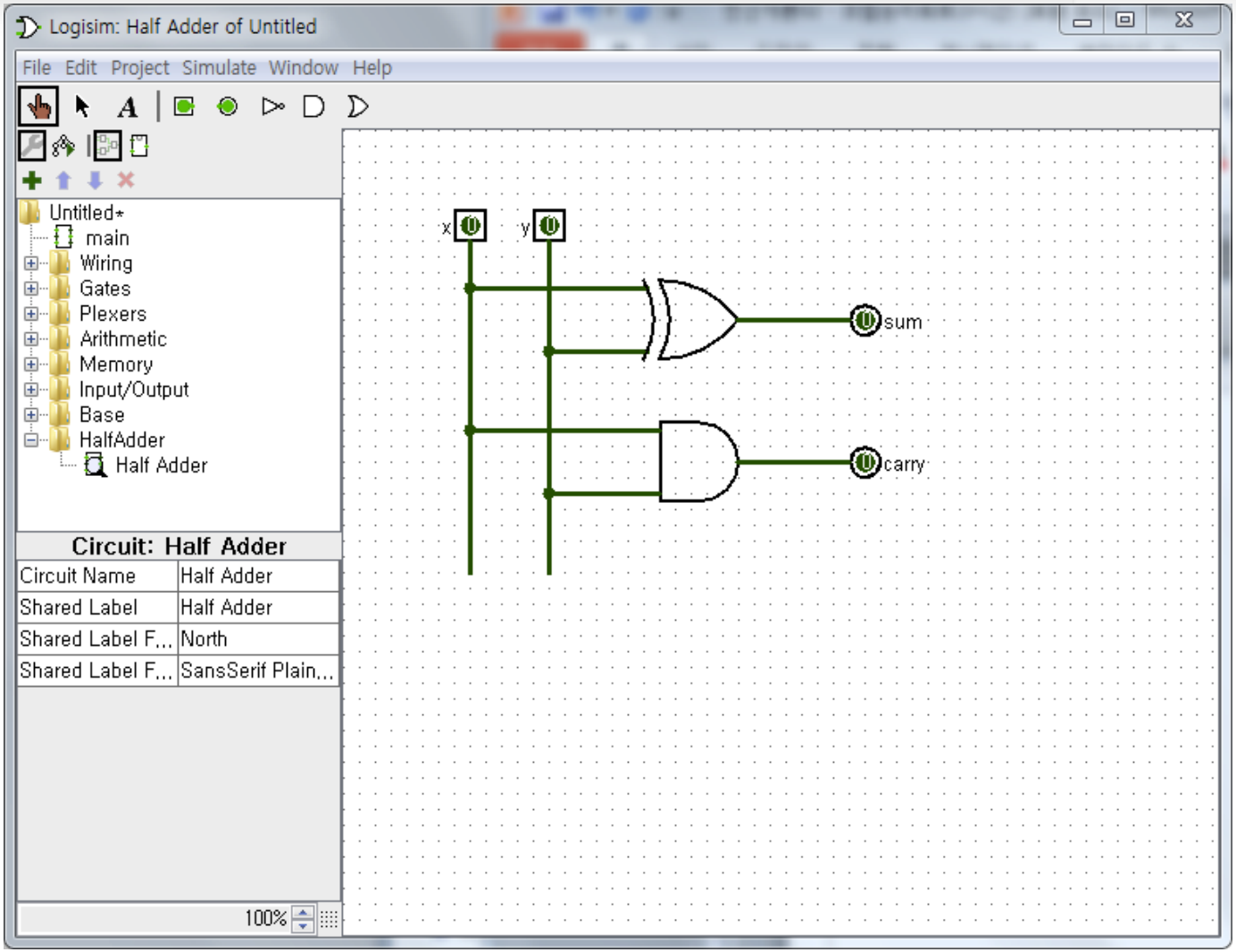
Load an existing Gate in Logisim

- Project → Load library → Logisim library
- Select a file



See the internal circuit of the gate




Logisim: Half Adder of Untitled
File Edit Project Simulate Window Help
Untitled*
main
Wiring
Gates
Plexers
Arithmetic
Memory
Input/Output
Base
HalfAdder
Half Adder
Circuit: Half Adder
Circuit Name
Half Adder
Shared Label
Half Adder
Shared Label F...
North
Shared Label F...
SansSerif Plain...
100%
x
y
sum
carry

(Homework1) Check the Halfadder in Logisim

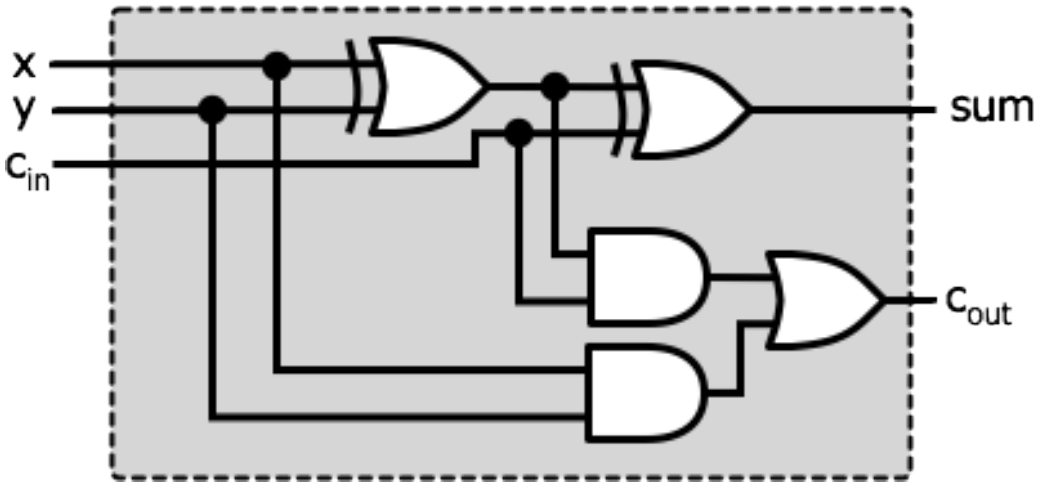
- Load combinational.circ and run the halfadder.

전가산기(Full Adder)

- A circuit to add three inputs with a single bit
  - 두 개의 출력 : sum, C_{out} (carry)

$$\begin{aligned} \text{sum} &= x \oplus y \oplus c_{in} \\ \text{carry} &= (x \wedge y) \vee (c_{in} \wedge (x \oplus y)) \end{aligned}$$

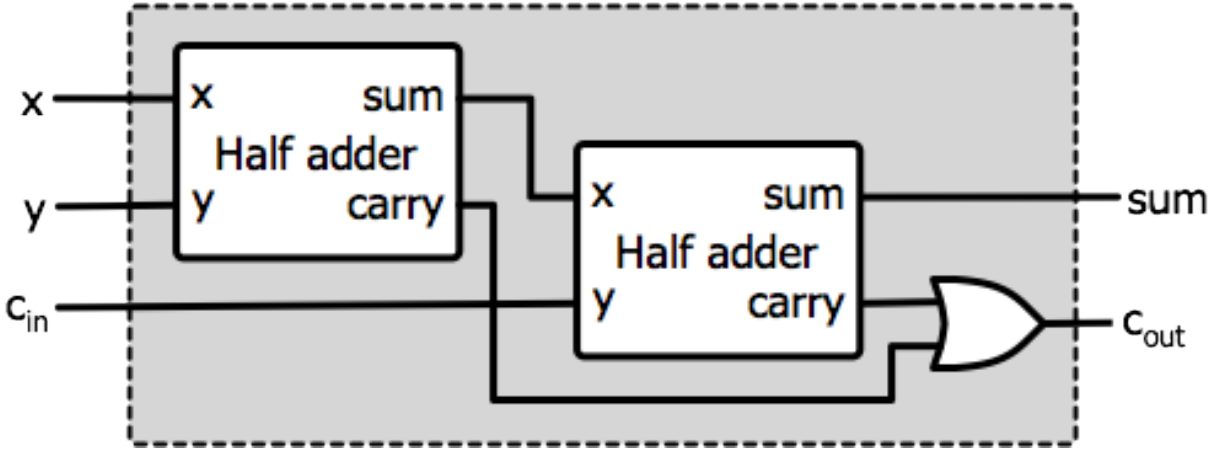
x	y	c_{in}	sum	c_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



전가산기(계속)

- 반 가산기 두 개로 전가산기 구현

x	y	c_{in}	sum	c_{out}
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



(HW2) Check FullAdder in

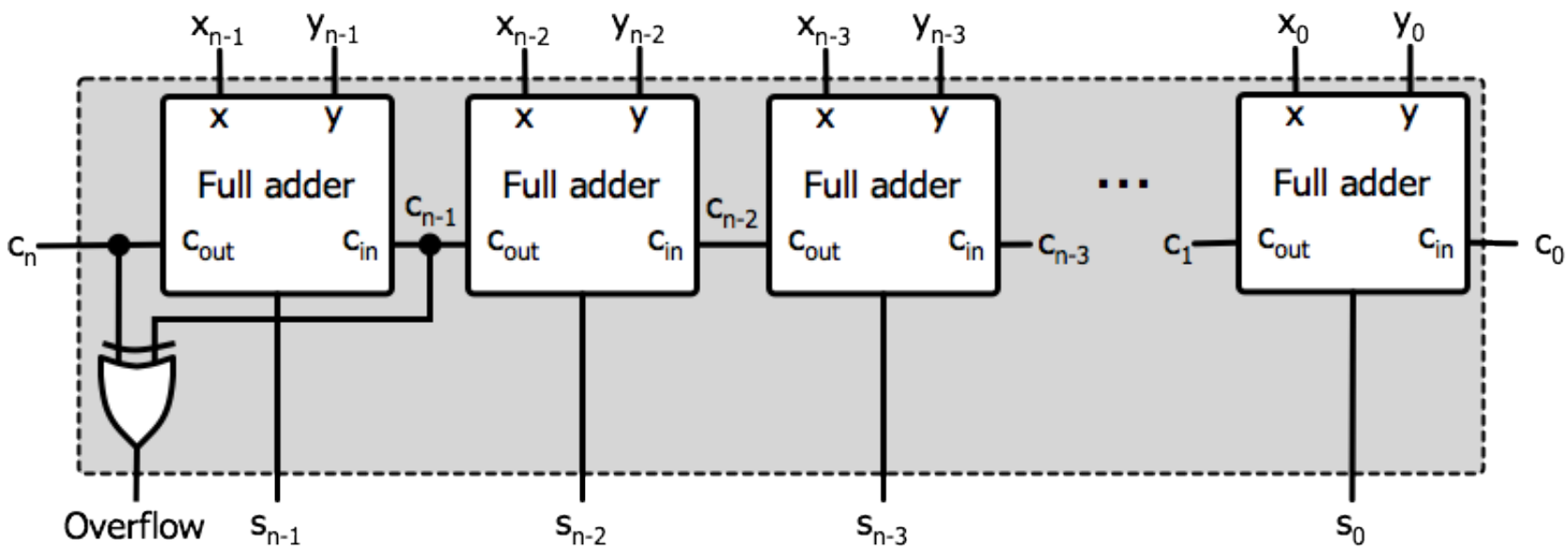
- Load combinational.circ and run the fulladder

리플캐리 가산기(Ripple Carry Adder)

- n 개의 전가산기를 연속적으로 붙여서 n 비트 이진 가산기를 구현
  - 이전 전가산기의 c_{out} 이 다음 전가산기의 c_{in} 으로 연결됨

- 2의 보수 표현에서 오버플로우 감지는 다음과 같음

$$\text{Overflow} = c_n \oplus c_{n-1}$$



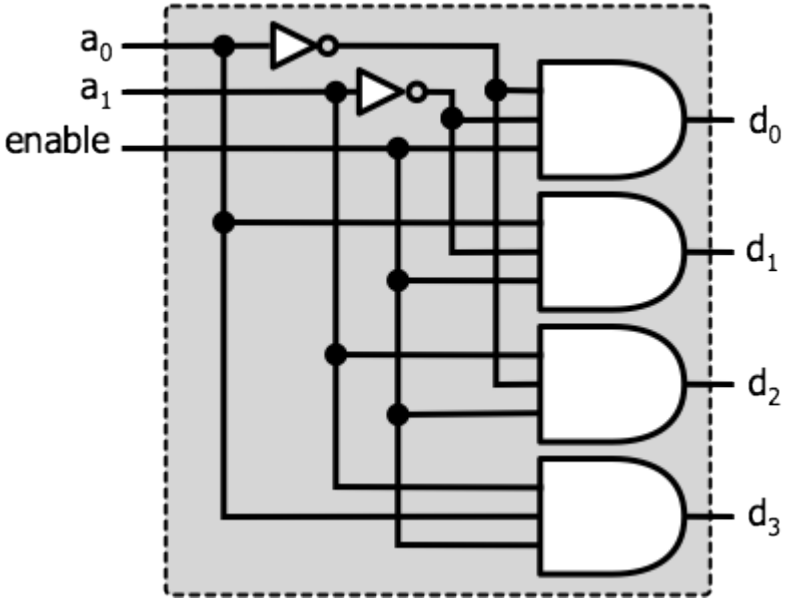
(HW3) Logisim에서 리플 캐리 가산기 동작 확인

- Load combinational.circ and run 4bit ripple carry adder
  - 1100 + 1010 = ?
  - 0111 + 0011 = ?

디코더 (Decoder)

- Demultiplexer 라고도 불림
- n 개의 이진 입력코드에 대해서 최대 2^n 개의 출력
  - 2-to-4 디코더, 3-to-8 디코더, 4-to-16 디코더, …
- 활성화(enable) 입력을 가지기도 함
  - 활성화 입력이 1이면, 디코더의 출력이 활성화 됨
  - 그렇지 않은 경우는 모든 출력이 0이 됨

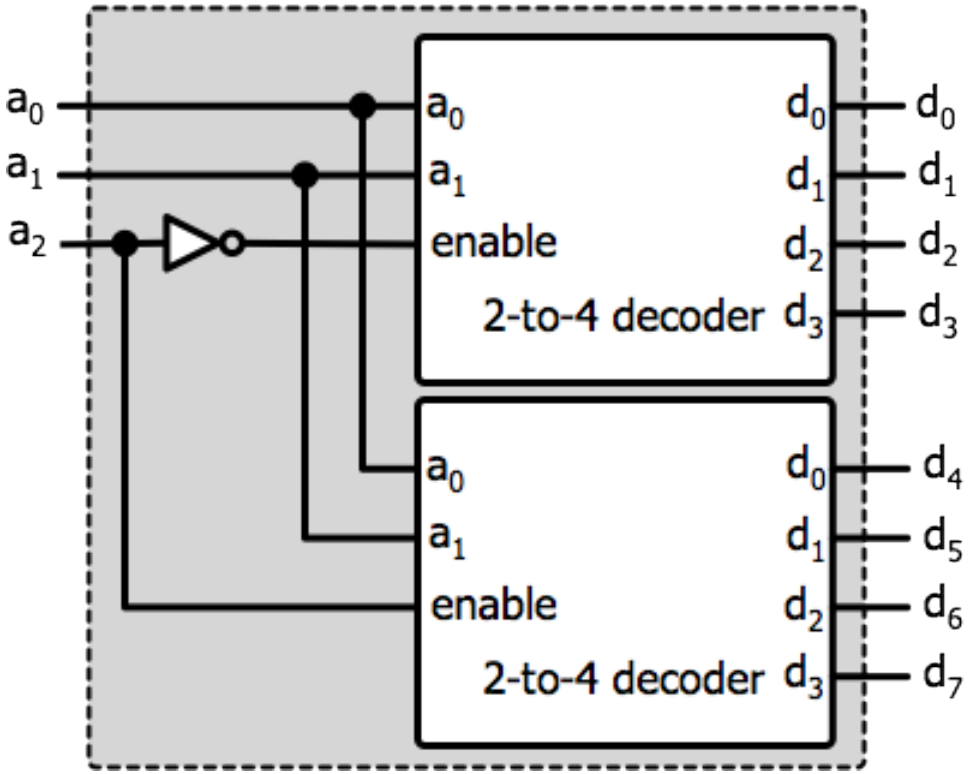
enable	a_1	a_0	d_3	d_2	d_1	d_0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0
0	X	X	0	0	0	0



디코더들의 조합

- 큰 디코더는 작은 디코더를 여러 개 결합해서 만들 수 있음

- 예) 활성화 입력이 있는 두 개의 2-to-4 디코더를 이용하여 3-to-8 디코더 만들기



(HW4) Logisim에서 디코더 동작 확인

- Load combinational.circ and run 2-to-4 decoder.
- Load combinational.circ and run 3-to-8 decoder.

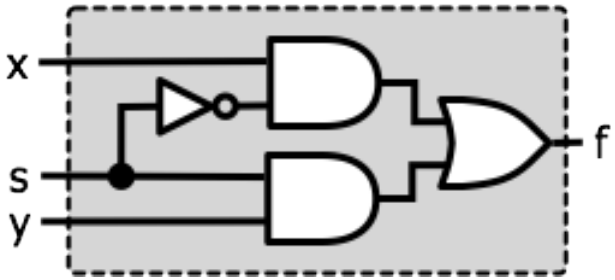
멀티플렉서(Multiplexer)

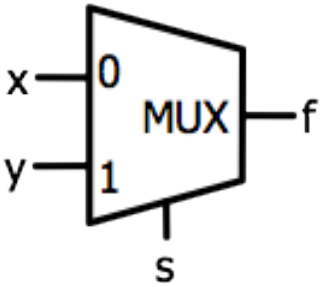
- 선택기(selector)라고도 불림
- n 개의 입력 중에 한 개를 선택해 출력해주는 디지털 스위치
- MUX : 멀티플렉서의 약칭

x	y	s	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

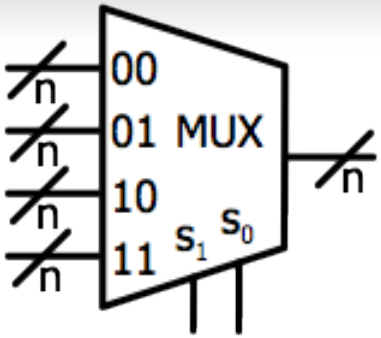
$$f(x, y, s) = (x \wedge s') \vee (y \wedge s)$$

s	f
0	x
1	y

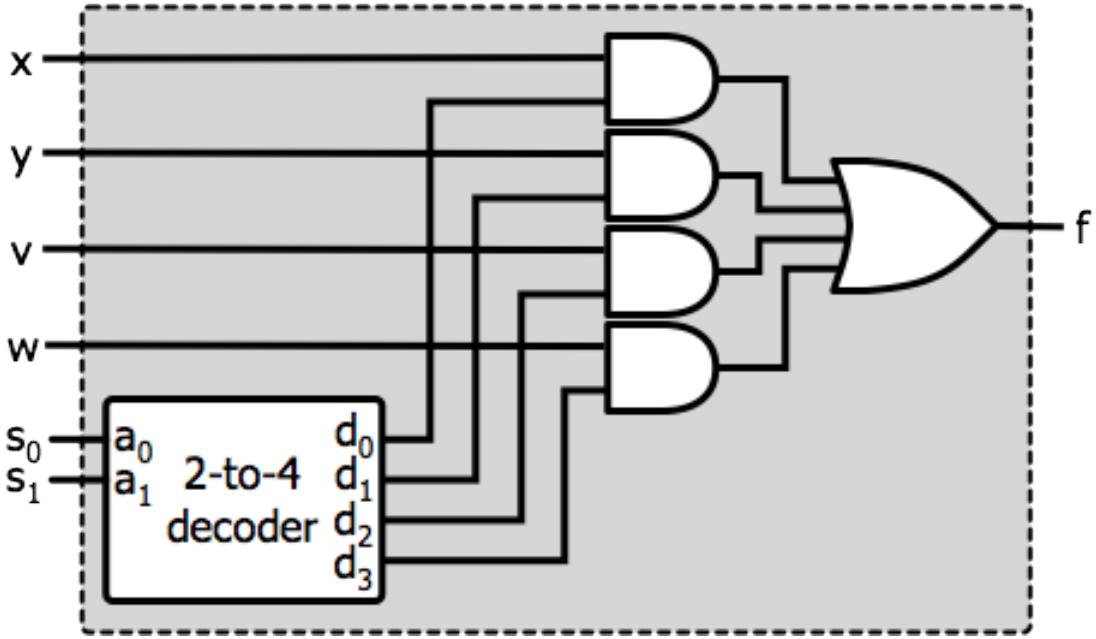




4-to-1 MUX

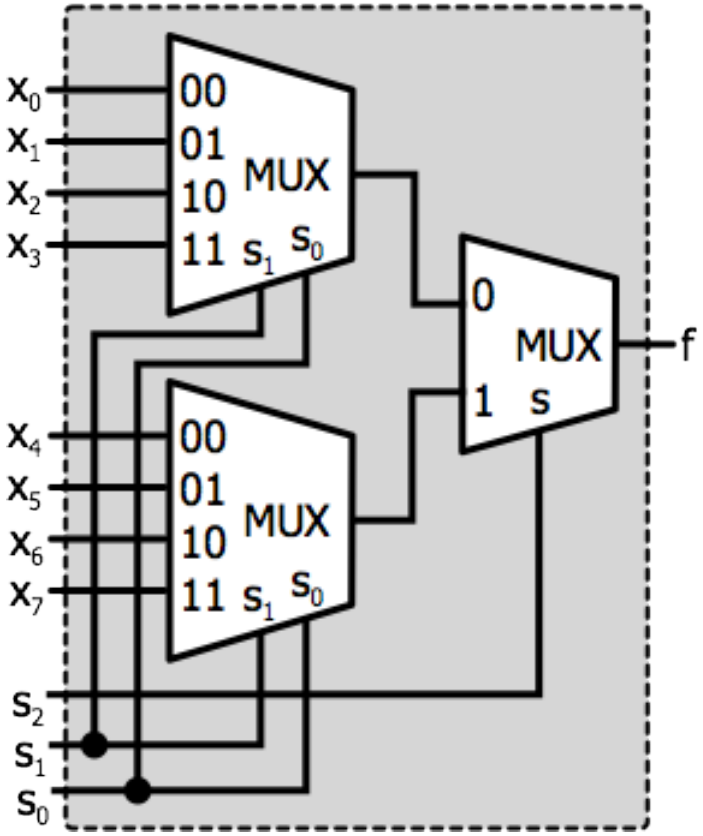


s_1	s_0	f
0	0	x
0	1	y
1	0	v
1	1	w



MUX의 결합

- 큰 MUX는 작은 MUX를 여러 개 결합해서 만들 수 있음



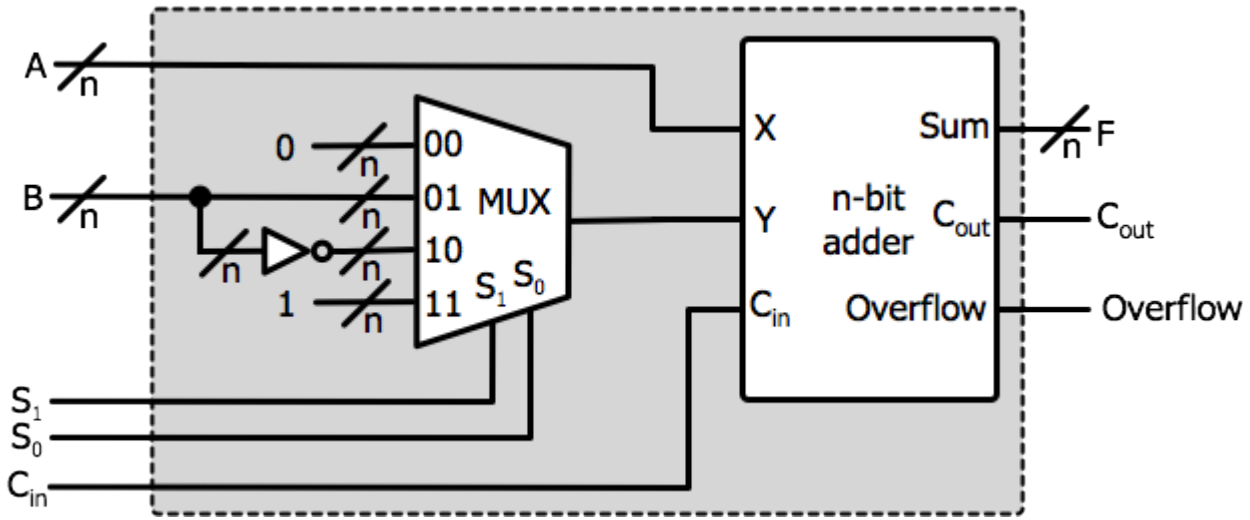
(HW5) Logisim에서 멀티플렉서 동작 확인

- Load combinational.circ and run 4-to-1 MUX.

Arithmetic Unit

- Addition and subtraction for integer values
- 2's complement number system is used

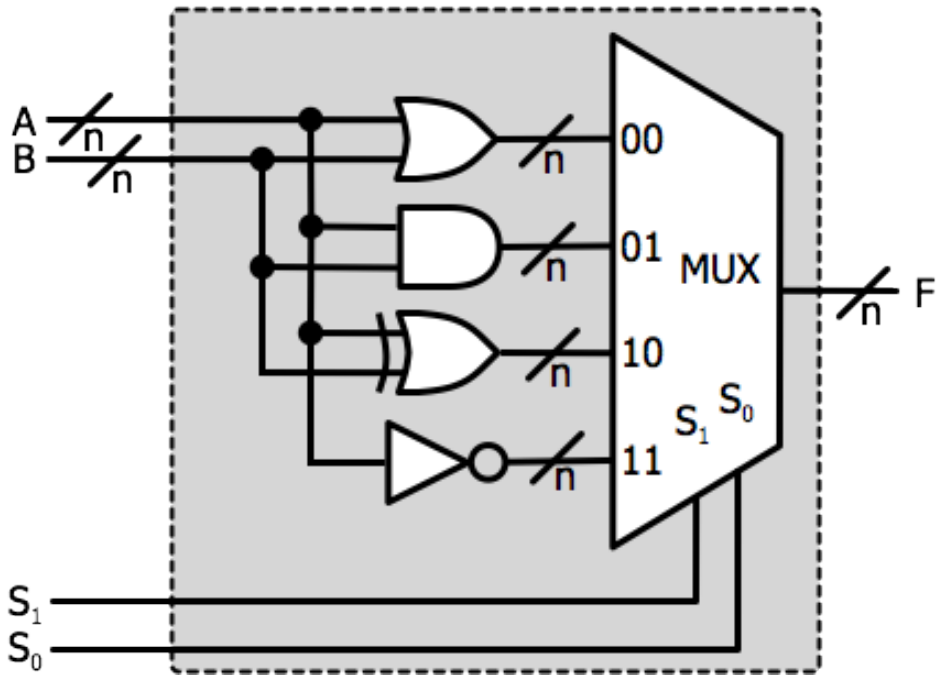
S_1	S_0	Y	$C_{in} = 0$	$C_{in} = 1$
0	0	00⋯0	F = A	F = A+1
0	1	B	F = A+B	F = A+B+1
1	0	B′	F = A+B′	F = A+B′+1
1	1	11⋯1	F = A-1	F = A



논리연산 장치(Logic Unit)

- 주어진 두 개의 n 비트 워드에 대하여 비트 단위 OR, AND, XOR, NOT 연산 수행

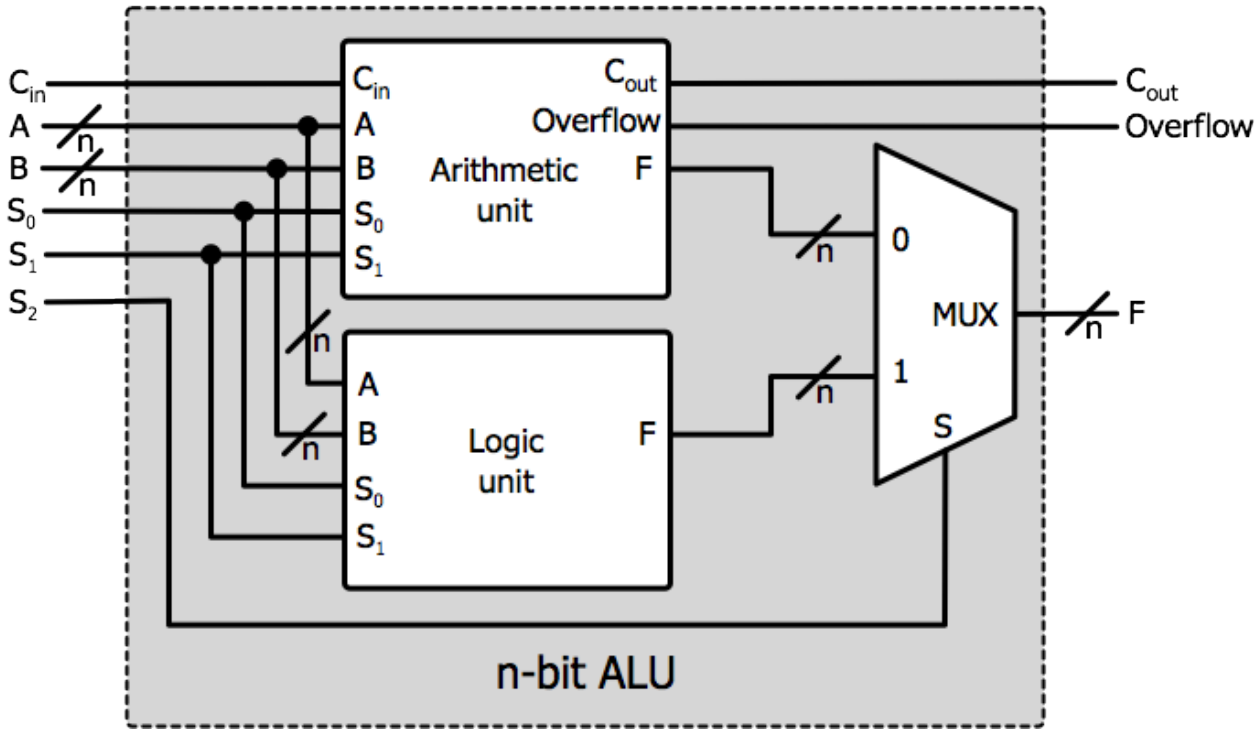
S_1	S_0	output	operation
0	0	$F = A \vee B$	OR
0	1	$F = A \wedge B$	AND
1	0	$F = A \oplus B$	XOR
1	1	$F = A'$	NOT



ALU(산술논리연산 장치)

- 산술 장치 + 논리 장치(Arithmetic unit + logic unit)

S_2	S_1	S_0	C_{in}	Operation
0	0	0	0	F = A
0	0	0	1	F = A+1
0	0	1	0	F = A+B
0	0	1	1	F = A+B+1
0	1	0	0	F = A+B′
0	1	0	1	F = A+B′+1
0	1	1	0	F = A-1
0	1	1	1	F = A
1	0	0	X	F = A∨B
1	0	1	X	F = A∧B
1	1	0	X	F = A⊕B
1	1	1	X	F = A′



넓은 관점에서의 ALU

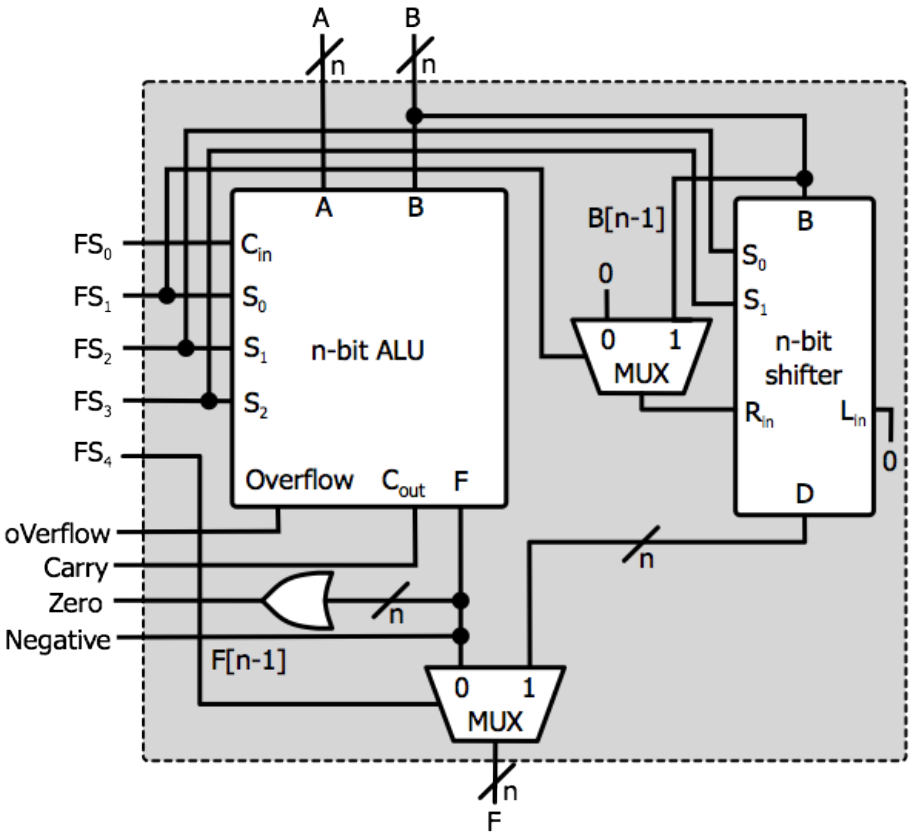
ALU + 쉬프터

FS_4	FS_3	FS_2	FS_1	FS_0	Operation
0	0	0	0	0	F = A
0	0	0	0	1	F = A+1
0	0	0	1	0	F = A+B
0	0	0	1	1	F = A+B+1
0	0	1	0	0	F = A+B′
0	0	1	0	1	F = A+B′+1
0	0	1	1	0	F = A-1
0	0	1	1	1	F = A
0	1	0	0	X	F = A∨B
0	1	0	1	X	F = A∧B
0	1	1	0	X	F = A⊕B
0	1	1	1	X	F = A′
1	0	0	X	X	F = B
1	0	1	0	X	F = logical shift right B
1	0	1	1	X	F = arithmetic shift right B
1	1	0	X	X	F = shift left B
1	1	1	X	X	F = B

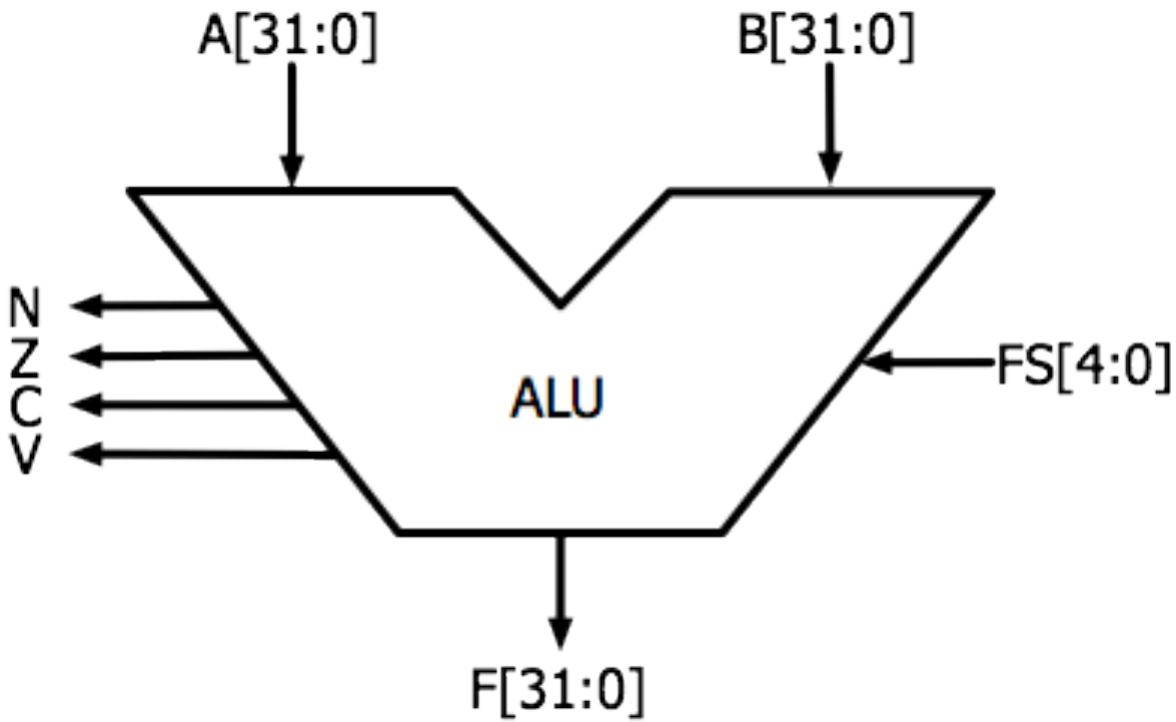
Arithmetic unit

Logic unit

Shifter



ALU 심볼



(HW6)

- **Load combinational.circ and run 4-bit ALU**
  - Which signals are assigned to compute 6 - 1?