Computer Architecture

Week 2: Logic Gates and Arithmetic

Fenerbahçe University

Professor & TAs

Prof: Dr. Vecdi Emre Levent

Office: 311

Email: emre.levent@fbu.edu.tr

TA: Arş. Gör. Uğur Özbalkan

Office: 311

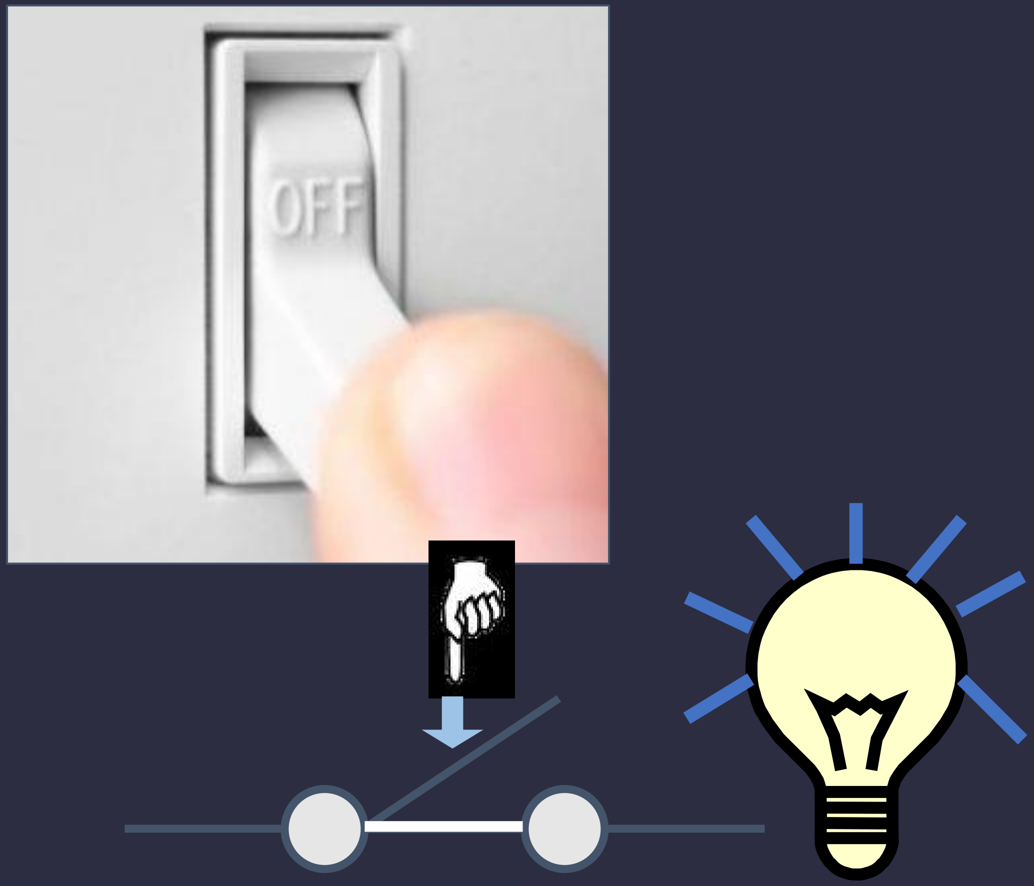
Email: ugur.ozbalkan@fbu.edu.tr

Course Plan

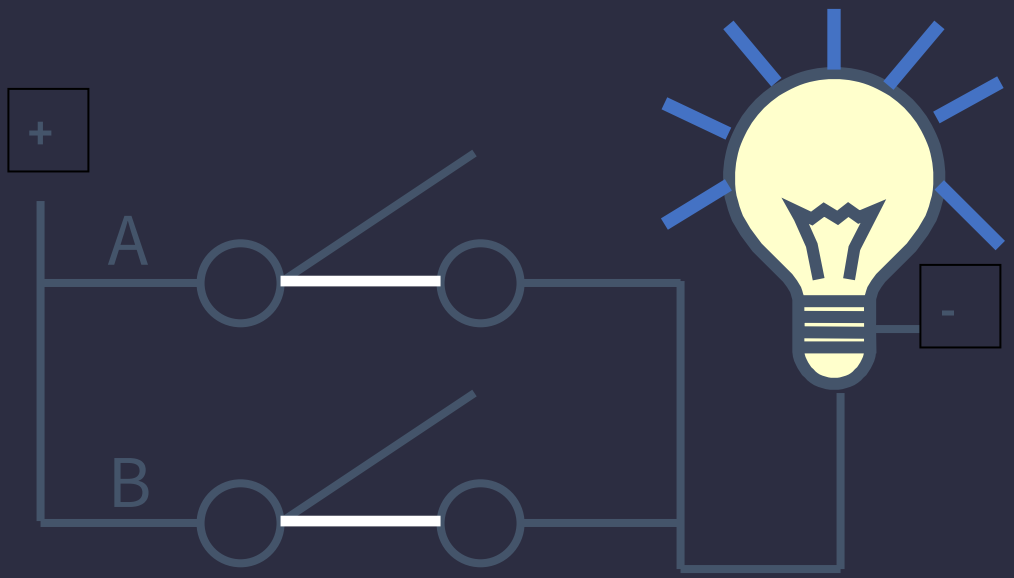
- Logic Gates and Arithmetic

Switch

Conductor and insulator



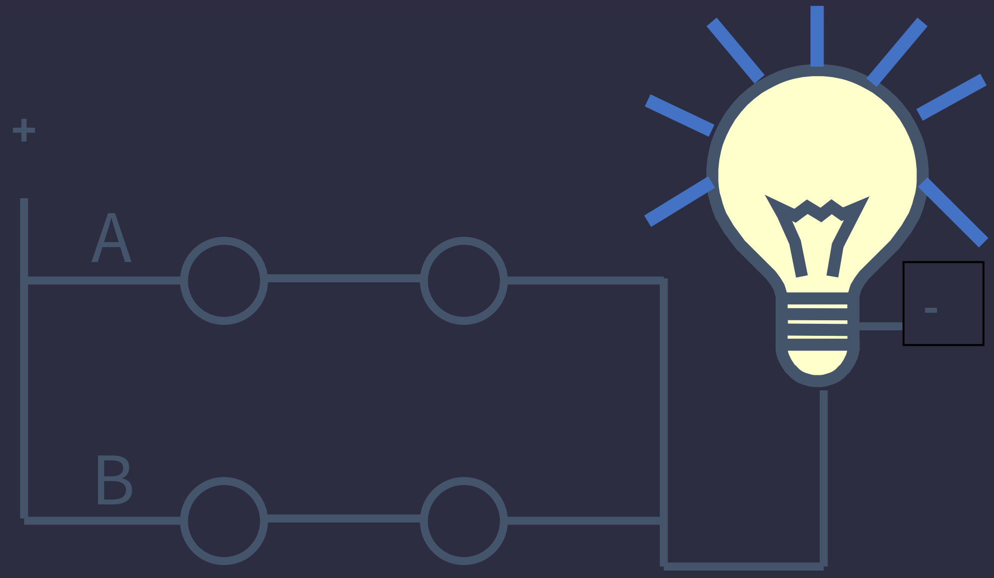
Switches to Gates



Truth Table

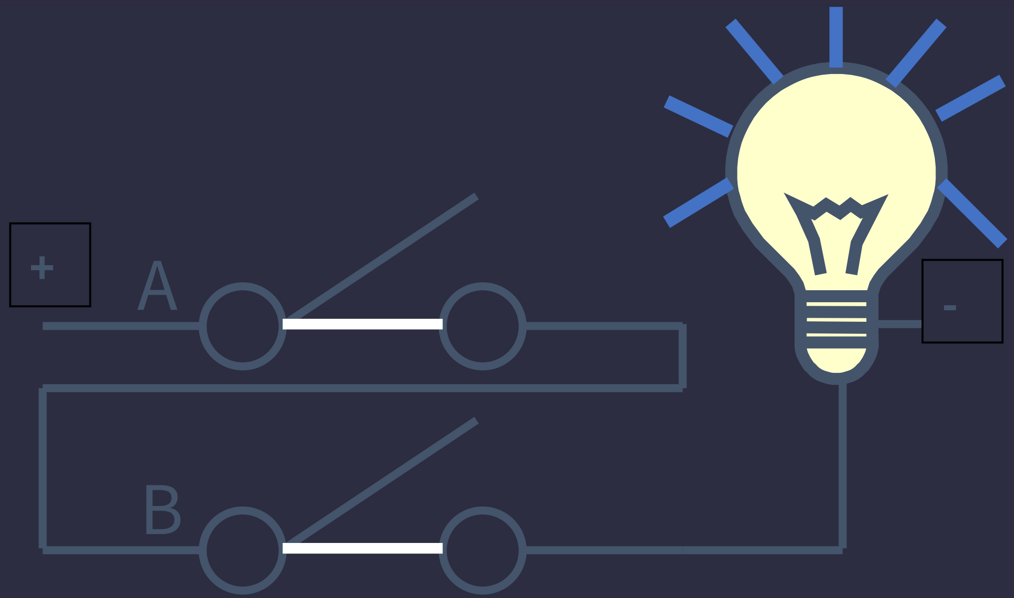
A	B	Light
OFF	OFF	OFF
OFF	ON	ON
ON	OFF	ON
ON	ON	ON

Switches to Gates



Truth Table

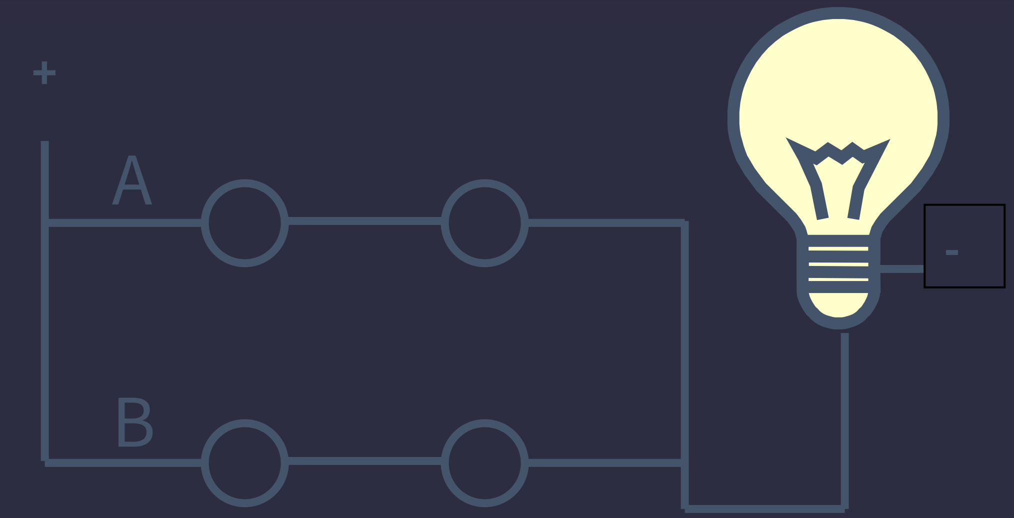
A	B	Light
OFF	OFF	OFF
OFF	ON	ON
ON	OFF	ON
ON	ON	ON



Truth Table

A	B	Light
OFF	OFF	OFF
OFF	ON	OFF
ON	OFF	OFF
ON	ON	ON

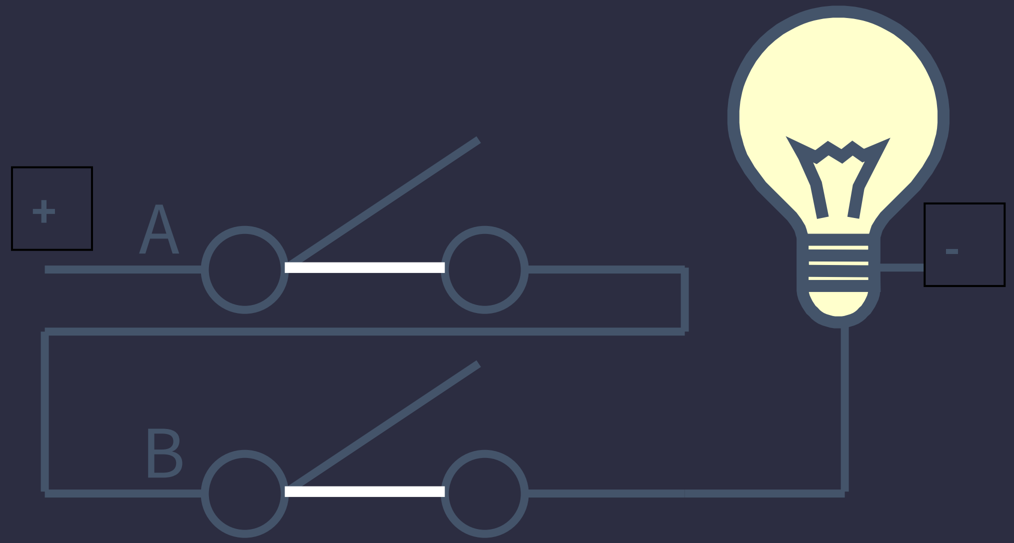
Switches to Gates



- OR

Truth Table

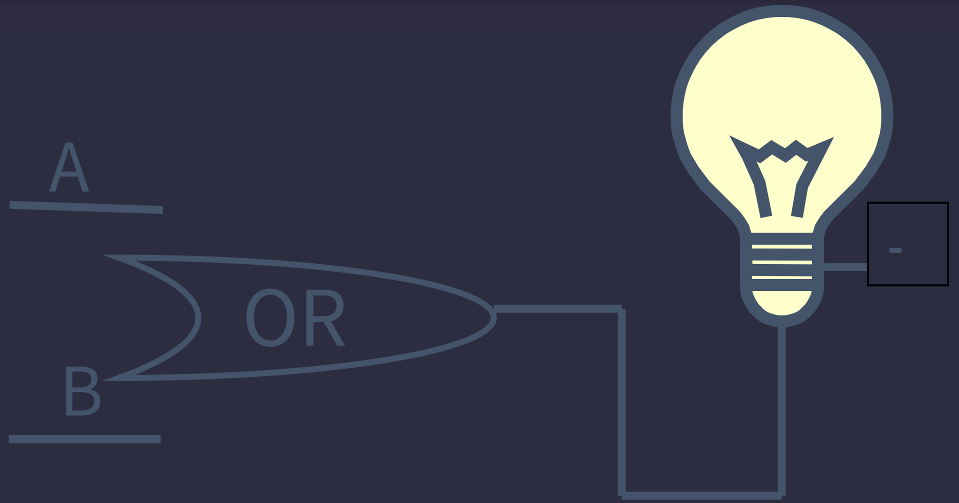
A	B	Light
OFF	OFF	OFF
OFF	ON	ON
ON	OFF	ON
ON	ON	ON



- AND

A	B	Light
OFF	OFF	OFF
OFF	ON	OFF
ON	OFF	OFF
ON	ON	ON

Switches to Gates



- OR

Truth Table

A	B	Light
OFF	OFF	OFF
OFF	ON	ON
ON	OFF	ON
ON	ON	ON

- AND

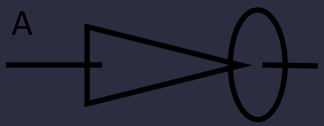
A	B	Light
OFF	OFF	OFF
OFF	ON	OFF
ON	OFF	OFF
ON	ON	ON

Switches to Gates

- Binary
  - There are two symbols: true and false
  - Fundamental for logic design

Logic Gates

- NOT:



A	Out
0	1
1	0

- AND:

A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

- OR:

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

Logic Gates

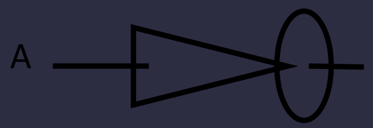
They are using for building logic functions

Commonly using gates:

AND, OR, NOT

Logic Gates

NOT:



A	Out
0	1
1	0

AND:

A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

OR:

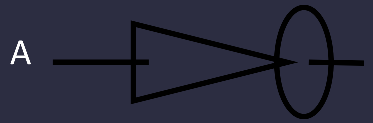
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

XOR:

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

Logic Gates

NOT:



A	Out
0	1
1	0

AND:

A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

OR:

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

XOR:

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

NAND:

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

NOR:

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

XNOR:

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	1

Logic Gates

a	b	c	out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Truth Table to Function

a	b	c	out	minterm
0	0	0	0	$\bar{a}\,\bar{b}\,\bar{c}$
0	0	1	1	$\bar{a}\,\bar{b}\,c$
0	1	0	0	$\bar{a}\,b\,\bar{c}$
0	1	1	1	$\bar{a}\,b\,c$
1	0	0	0	$a\,\bar{b}\,\bar{c}$
1	0	1	1	$a\,\bar{b}\,c$
1	1	0	0	$a\,b\,\bar{c}$
1	1	1	0	$a\,b\,c$

1. Write minterms
2. Write sum of products of elements which minterm column is 1

Truth Table to Function

a	b	c	out	minterm
0	0	0	0	$\bar{a}\,\bar{b}\,\bar{c}$
0	0	1	1	$\bar{a}\,\bar{b}\,c$
0	1	0	0	$\bar{a}\,b\,\bar{c}$
0	1	1	1	$\bar{a}\,b\,c$
1	0	0	0	$a\,\bar{b}\,\bar{c}$
1	0	1	1	$a\,\bar{b}\,c$
1	1	0	0	$a\,b\,\bar{c}$
1	1	1	0	$a\,b\,c$

1. Write minterms
2. Write sum of products of elements which minterm column is 1
   - So, $out = \bar{a}\bar{b}c + \bar{a}bc + a\bar{b}c$

Logical Expressions

- NOT:
  - out = $\bar{a}$ = !a = $\neg a$
- AND:
  - out = $a \cdot b$ = a & b = $a \wedge b$
- OR:
  - out = $a + b$ = a | b = $a \vee b$
- XOR:
  - out = $a \oplus b = a\bar{b} + \bar{a}b$

NAND:

- out = $\overline{a \cdot b}$ = !(a & b) = $\neg (a \wedge b)$

NOR:

- out = $\overline{a + b}$ = !(a | b) = $\neg (a \vee b)$

XNOR:

- out = $\overline{a \oplus b} = ab + \overline{ab}$

Logical Expressions

$$a + 0 = a$$
$$a + 1 = 1$$
$$a + \bar{a} = 1$$

$$a \cdot 0 = 0$$
$$a \cdot 1 = a$$
$$a \cdot \bar{a} = 0$$

Logical Expressions

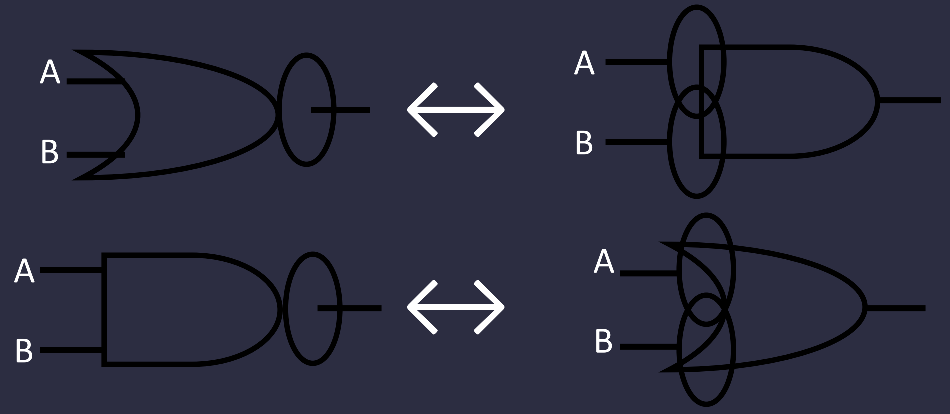
$$\overline{(a+b)} = \bar{a} \cdot \bar{b}$$

$$\overline{(a \cdot b)} = \bar{a} + \bar{b}$$

$$a + a\,b = a$$

$$a(b+c) = ab + ac$$

$$\overline{a(b+c)} = \bar{a} + \bar{b} \cdot \bar{c}$$



Minimization Example

$a + 0 = a$

$a + 1 = 1$

$a + \bar{a} = 1$

$a \cdot 0 = 0$

$a \cdot 1 = a$

$a \cdot \bar{a} = 0$

$a + a\,b = a$

$a\,(b+c) = ab + ac$

$$\overline{(a+b)} = \overline{a} \bullet \overline{b}$$

$$\overline{(ab)} = \overline{a} + \overline{b}$$

$$\overline{a(b+c)} = \overline{a} + \overline{b} \bullet \overline{c}$$

Minimize the function below

$$\begin{aligned}(a+b)(a+c) &= (a+b)a + (a+b)c \\ &= aa + ba + ac + bc \\ &= a + a(b+c) + bc \\ &= a + bc\end{aligned}$$

Minimization Example

$a + 0 = a$

$a + 1 = 1$

$a + \bar{a} = 1$

$a \cdot 0 = 0$

$a \cdot 1 = a$

$a \cdot \bar{a} = 0$

$a + a\,b = a$

$a\,(b+c) = ab + ac$

$$\overline{(a+b)} = \overline{a} \bullet \overline{b}$$

$$\overline{(ab)} = \overline{a} + \overline{b}$$

$$\overline{a(b+c)} = \overline{a} + \overline{b} \bullet \overline{c}$$

Minimize the function below

$$\begin{aligned}(a+b)(a+c) &= (a+b)a + (a+b)c \\ &= aa + ba + ac + bc \\ &= a + a(b+c) + bc \\ &= a + bc\end{aligned}$$

Required logic gates

Before	After
2 OR, 1 AND	1 OR, 1 AND

Equality Check with Truth tables

Example: $(a+b)(a+c) = a + bc$

a	b	c					
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

Equality Check with Truth tables

Example: $(a+b)(a+c) = a + bc$

a	b	c	a+b	a+c	Sol	bc	Sağ
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Karnaugh Maps

How to reach most optimum circuit?

– Algebraic minimizations
– Karnaugh Map
– Computer aided minimization softwares (Totally that way in 2021)

Karnaugh Maps

a	b	c	out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

- Sum of Minterms
  - $out = \overline{ab}c + \overline{a}bc + a\overline{bc} + a\overline{b}c$

Karnaugh Maps

a	b	c	out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

c \ ab	00	01	11	10
0				
1				

- Sum of Minterms
  - $out = \overline{ab}c + \bar{a}bc + a\overline{bc} + a\bar{b}c$
- Kmaps are usefull for detecting which inputs are relevant for output
- When creating Kmap table, only 1 bit can change on row and columns

Karnaugh Maps

a	b	c	out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

c \ ab	00	01	11	10
0	0	0	0	1
1	1	1	0	1

Sum of Minterms

- $out = \bar{a}\bar{b}c + \bar{a}bc + a\bar{b}\bar{c} + a\bar{b}c$

Kmap minimization

- Group ones with twos power group
- Eliminate unnecessary inputs
  - $out = a\bar{b} + \bar{a}c$

Karnaugh Maps

c \ ab	00	01	11	10
0	0	1	1	1
1	0	0	1	0

c \ ab	00	01	11	10
0	1	1	1	1
1	0	0	1	0

- Minterms can be used different groups
  - out = $b\bar{c} + a\bar{c} + ab$

- Minters can be group by twos power
  - out = $\bar{c} + ab$

Karnaugh Maps

cd \ ab	00	01	11	10
00	0	0	0	0
01	1	0	0	1
11	1	0	0	1
10	0	0	0	0

- $out = \bar{b}d$

cd \ ab	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	0	0	0
10	1	0	0	1

- $out = \bar{b}\,\bar{d}$

Karnaugh Maps

cd \ ab	00	01	11	10
00	0	0	0	0
01	1	x	x	x
11	1	x	x	1
10	0	0	0	0

- X ("Don't care") can be evaluated zero or one.
  - If all X's are 1,
  - out = d

cd \ ab	00	01	11	10
00	1	0	0	x
01	0	x	x	0
11	0	x	x	0
10	1	0	0	1

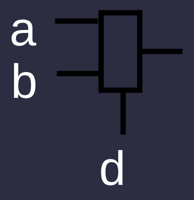
  - If all X's are 0,
  - And only right top X is 1, then
  - out = $\overline{b}\,\overline{d}$

Karnaugh Maps

c \ ab	00	01	11	10
0	0	0	0	1
1	1	1	0	1

$$= a\bar{b} + \bar{a}c$$

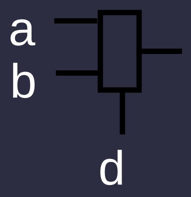
Multiplexer



a	b	d	out
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

- Multiplexer kendisine verilen girişleri seçerek dışarı çıkarır
  - Eğer d=0, out = a
  - Eğer d= 1, out = b

Multiplexer

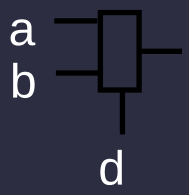


- Doğruluk tablosu oluşturun

$out = \bar{a}bd + a\overline{bd} + ab\bar{d} + abd$

a	b	d	out
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

Multiplexer

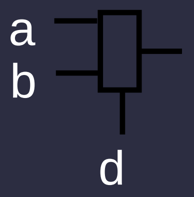


a	b	d	out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

- KMap

d \ ab	00	01	11	10
0				
1				

Multiplexer

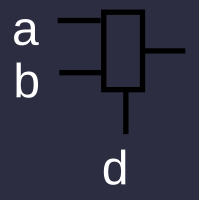


a	b	d	out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

- KMap

d \ ab	00	01	11	10
0	0	0	1	1
1	0	1	1	0

Multiplexer

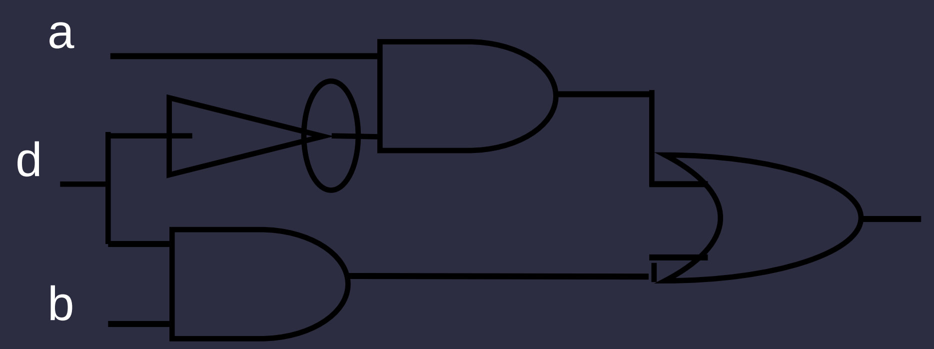


a	b	d	out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

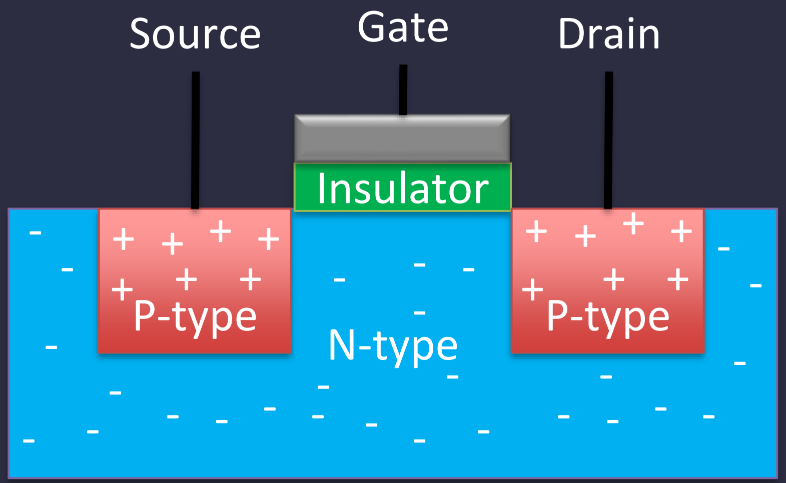
- Minimized Equation

d \ ab	00	01	11	10
0	0	0	1	1
1	0	1	1	0

- $out = a\bar{d} + bd$

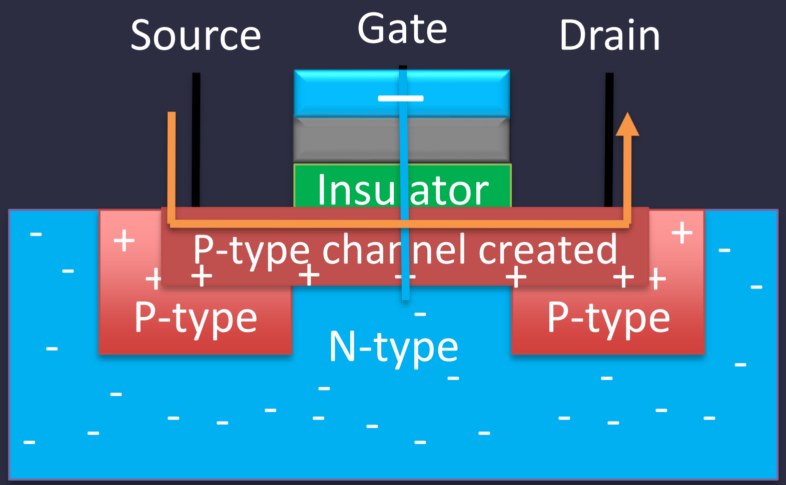


Transistors



P-Transistor

Closed



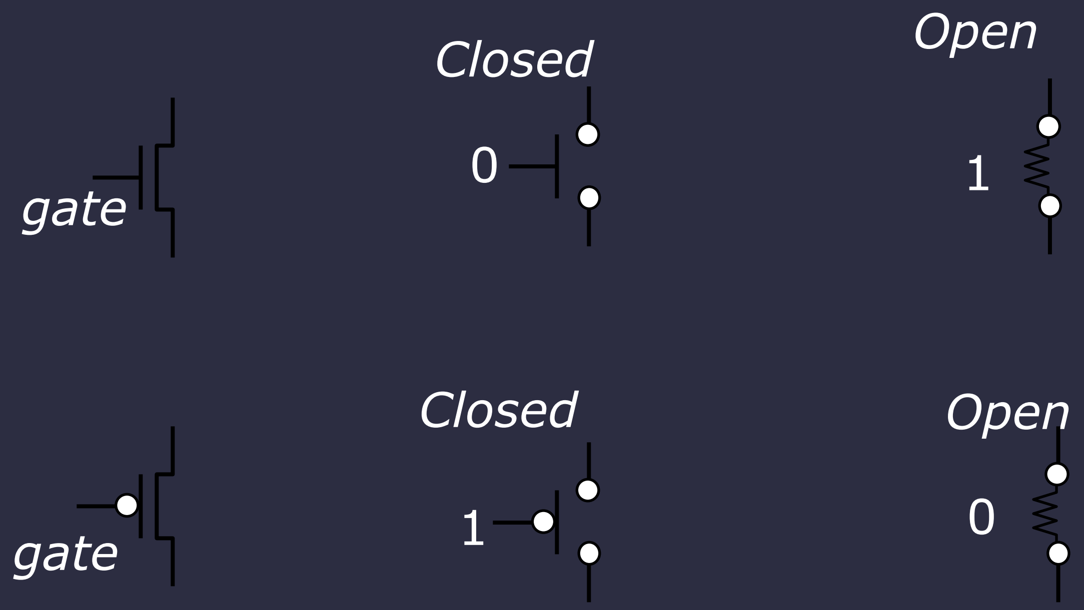
P-Transistor

Open

CMOS Notation

N-Type

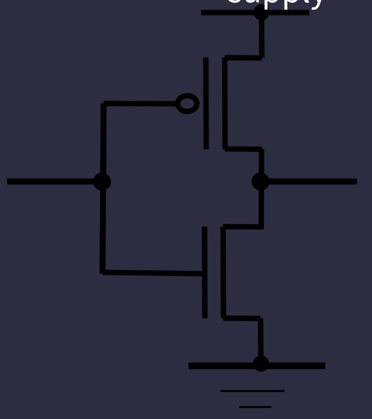
P-Type

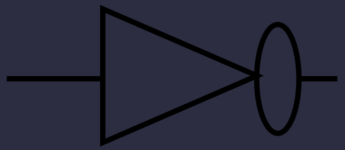


Electric flow controls with gate input.

NOT Gate

NOT

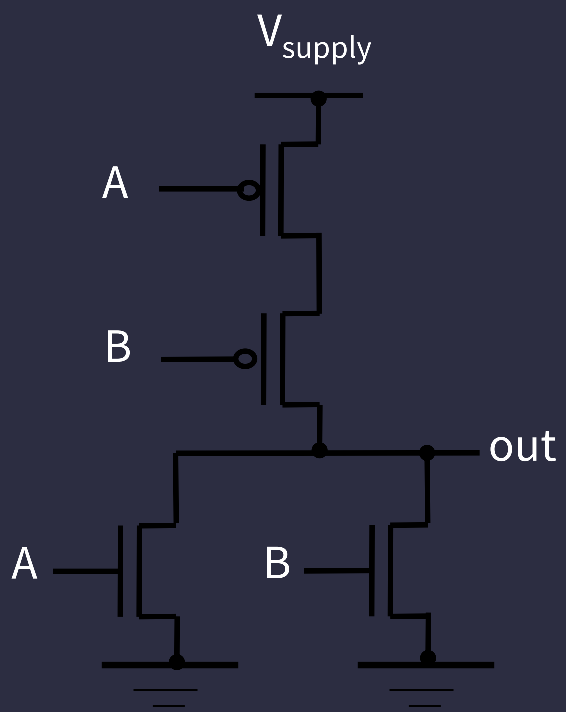




Truth Table

In	Out
0	1
1	0

NOR Gate



NOR

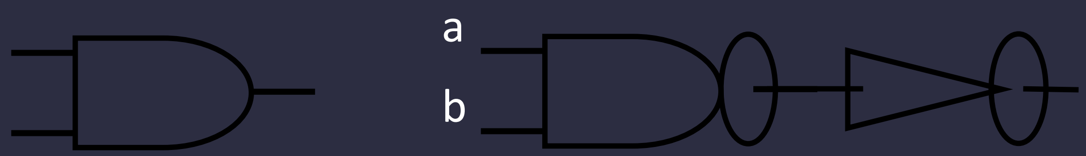
Truth Table

A	B	out
0	0	1
0	1	0
1	0	0
1	1	0

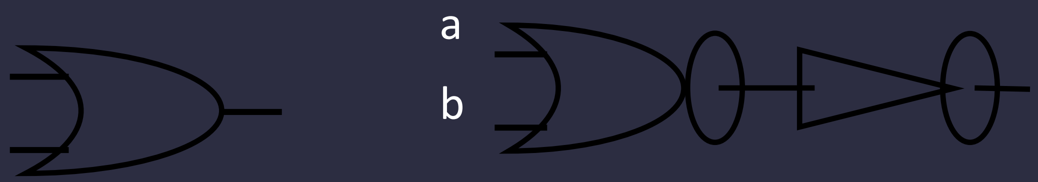
Logic Gates

- NOT:

- AND:



- OR:



- Generally ASIC designs builds combinations of NAND gates

Number Representations

Binary

- There are two symbols: true and false; 1 and 0
- It is fundamental of digital systems

Number Representations

Binary

- There are two symbols: true and false; 1 and 0
- It is fundamental of digital systems

Base 10 Representation

$10^2\ 10^1\ 10^0$

- Example. 6 3 7 $\quad 6 \cdot 10^2 + 3 \cdot 10^1 + 7 \cdot 10^0 = \mathbf{637}$

- Other Bases
  - Base 2 — Binary
  - Base 8 — Octal
  - Base 16 — Hexadecimal

$$1 \cdot 2^9 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^0 = \mathbf{637}$$

$$1 \cdot 8^3 + 1 \cdot 8^2 + 7 \cdot 8^1 + 5 \cdot 8^0 = \mathbf{637}$$

$$2 \cdot 16^2 + 7 \cdot 16^1 + d \cdot 16^0 = \mathbf{637}$$

$$2 \cdot 16^2 + 7 \cdot 16^1 + 13 \cdot 16^0 = \mathbf{637}$$

Number Representations

Conversion between bases

Base conversion via repetitive division

- Divide by base, write remainder, move left with quotient
- Example: Base 10 to 8 (octal)

- **637** $\div$ 8 = 79 remainder 5 lsb
- 79 $\div$ 8 = 9 remainder 7
- 9 $\div$ 8 = 1 remainder 1
- 1 $\div$ 8 = 0 remainder 1 msb

lsb (least significant bit)

msb (most significant bit)

- 637 = 0o 1175

Number Representations

Convert a base 10 number to a base 2 number

Base conversion via repetitive division

- Divide by base, write remainder, move left with quotient
- 637 ÷ 2 = 318 remainder 1
- 318 ÷ 2 = 159 remainder 0
- 159 ÷ 2 = 79 remainder 1
- 79 ÷ 2 = 39 remainder 1
- 39 ÷ 2 = 19 remainder 1
- 19 ÷ 2 = 9 remainder 1
- 9 ÷ 2 = 4 remainder 1
- 4 ÷ 2 = 2 remainder 0
- 2 ÷ 2 = 1 remainder 0
- 1 ÷ 2 = 0 remainder 1

637 = 10 0111 1101 (can also be written as 0b10 0111 1101)

msb lsb

Convert a base 10 number to a base 16 number

Base conversion via repetitive division

- Divide by base, write remainder, move left with quotient
- 657 ÷ 16 = 41 remainder 1
- 41 ÷ 16 = 2 remainder 9
- 2 ÷ 16 = 0 remainder 2

lsb

msb

Thus, 657 = 0x291

Number Representations

Convert a base 10 number to a base 16 number

Base conversion via repetitive division

- Divide by base, write remainder, move left with quotient
- $637 \div 16 = 39$ remainder 13
- $39 \div 16 = 2$ remainder 7
- $2 \div 16 = 0$ remainder 2

lsb

msb

dec = hex		= bin
10	= 0xa	= 1010
11	= 0xb	= 1011
12	= 0xc	= 1100
13	= 0xd	= 1101
14	= 0xe	= 1110
15	= 0xf	= 1111

637 = 0x 2 7 13 = ?

Thus, 637 = 0x27d

Number Representations Summary

- Base 10 – Decimal

$$\underline{6}\ \underline{3}\ \underline{7}$$
$10^2\ 10^1\ 10^0$

$$6 \cdot 10^2 + 3 \cdot 10^1 + 7 \cdot 10^0 = 637$$

- Base 2 — Binary

$$\underline{1}\ \underline{0}\ \ \underline{0}\ \underline{1}\ \underline{1}\ \underline{1}\ \ \underline{1}\ \underline{1}\ \underline{0}\ \underline{1}$$
$2^9\ 2^8\ \ 2^7\ 2^6\ 2^5\ 2^4\ \ 2^3\ 2^2\ 2^1\ 2^0$

$$1 \cdot 2^9 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^0 = 637$$

- Base 8 — Octal

$$0o\ \underline{1}\ \underline{1}\ \underline{7}\ \underline{5}$$
$8^3\ 8^2\ 8^1\ 8^0$

$$1 \cdot 8^3 + 1 \cdot 8^2 + 7 \cdot 8^1 + 5 \cdot 8^0 = 637$$

- Base 16 — Hexadecimal

$$0x\ \underline{2}\ \underline{7}\ \underline{d}$$
$16^2 16^1 16^0$

$$2 \cdot 16^2 + 7 \cdot 16^1 + d \cdot 16^0 = 637$$
$$2 \cdot 16^2 + 7 \cdot 16^1 + 13 \cdot 16^0 = 637$$

Binary Addition

- Addition works the same way regardless of base
  - Add the digits in each position
  - Propagate the carry

Unsigned binary addition is pretty easy

- Combine two bits at a time
- Along with a carry

How do we do arithmetic in binary?

$$\begin{array}{r} 1 \\ 183 \\ +\ \underline{254} \\ 437 \end{array}$$

Carry-in

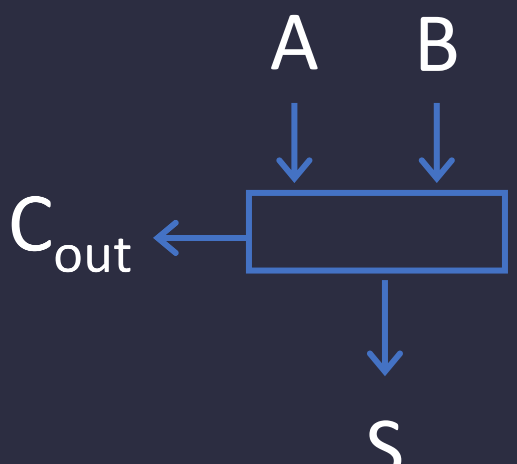
Carry-out

$$\begin{array}{r} 111 \\ 001110 \\ +\ \underline{011100} \\ 101010 \end{array}$$

Binary Addition

- Binary addition requires
  - Add of ***two bits*** PLUS ***carry-in***
  - Also, ***carry-out*** if necessary

1-bit Adder



A	B	C_{out}	S
0	0		
0	1		
1	0		
1	1		

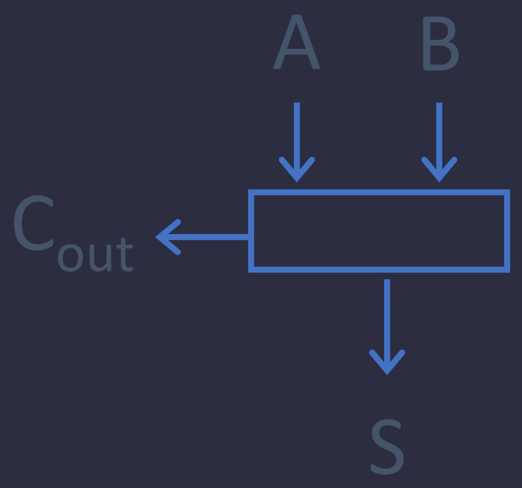
Half Adder

- Adds two 1-bit numbers
- Computes 1-bit result and 1-bit carry
- No carry-in

What is the equation for C_{out}?

a) $A + B$
b) AB
c) $A \oplus B$
d) $A + !B$
e) $!A!B$

1-bit Adder

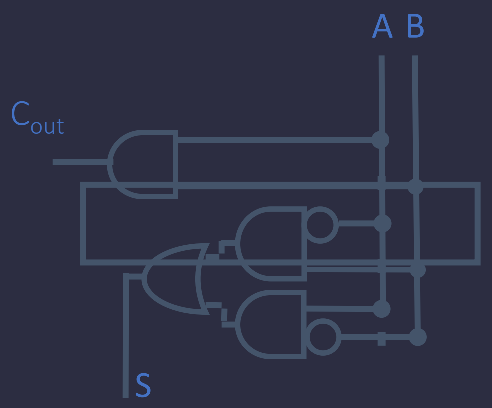


Half Adder

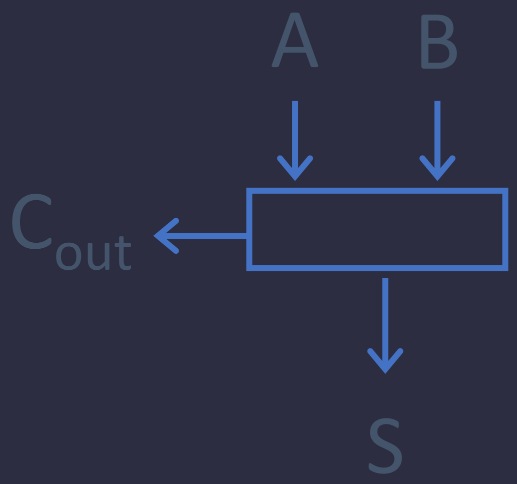
- Adds two 1-bit numbers
- Computes 1-bit result and 1-bit carry
- No carry-in

- $S = \overline{A}B + A\overline{B}$
- $C_{out} = AB$

A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



1-bit Adder

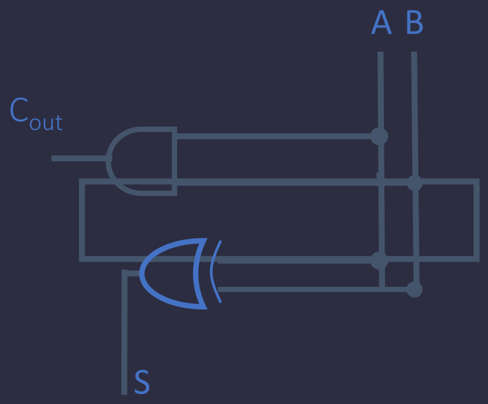


A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

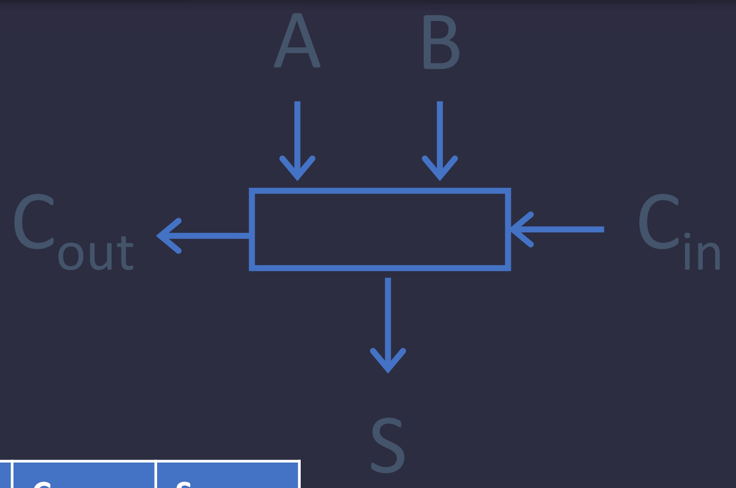
Half Adder

- Adds two 1-bit numbers
- Computes 1-bit result and 1-bit carry
- No carry-in

- $S = \overline{A}B + A\overline{B} = A \oplus B$
- $C_{out} = AB$



1-bit Adder with Carry



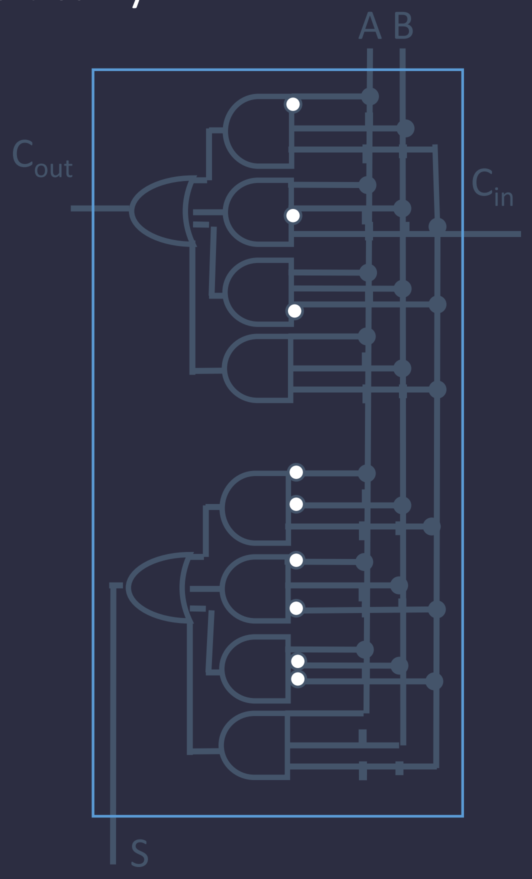
A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Full Adder

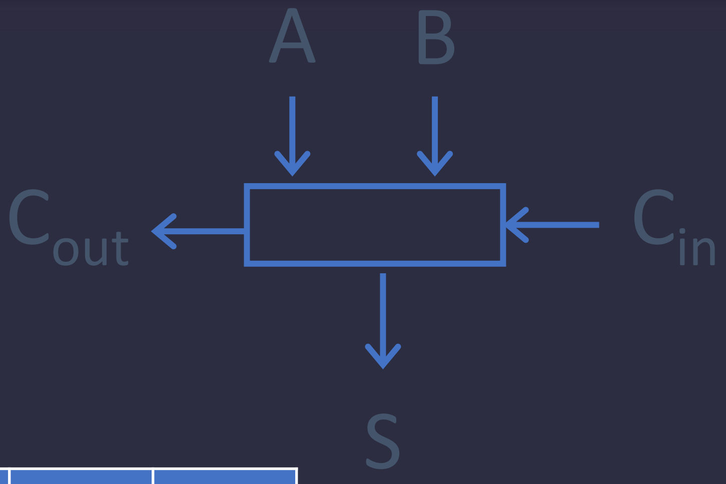
- Adds three 1-bit numbers
- Computes 1-bit result and 1-bit carry
- Can be cascaded

$S = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

$C_{out} = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$



1-bit Adder with Carry



A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

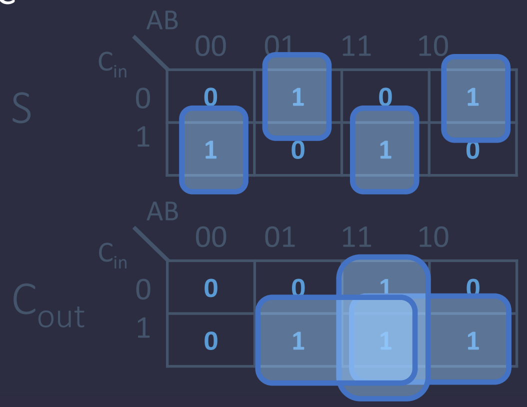
Full Adder

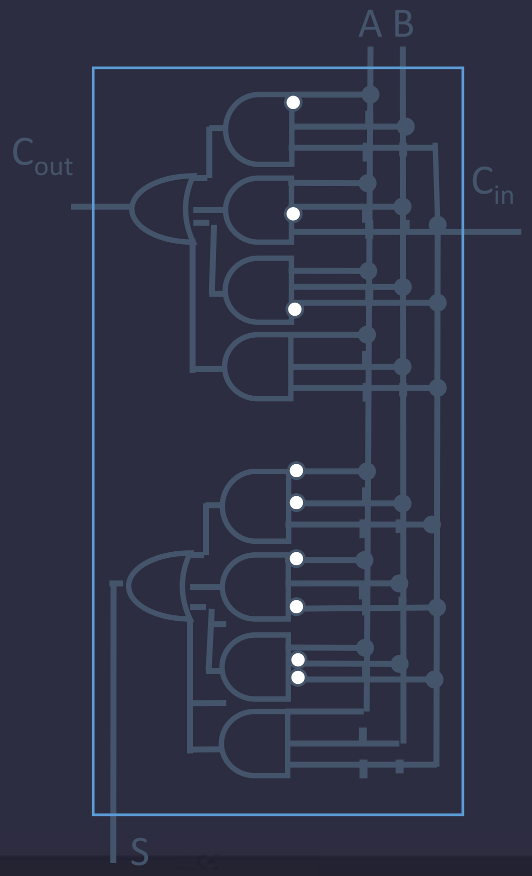
- Adds three 1-bit numbers
- Computes 1-bit result and 1-bit carry
- Can be cascaded

$S = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

$C_{out} = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$

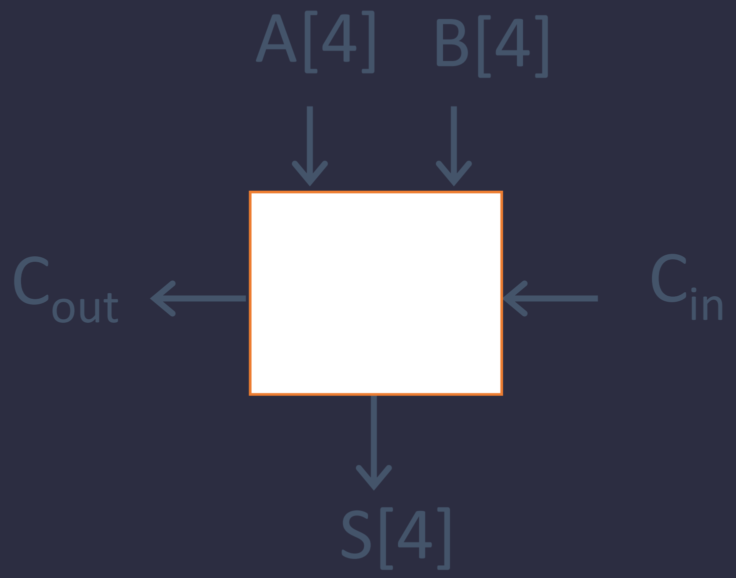
$C_{out} = AB + AC + BC$



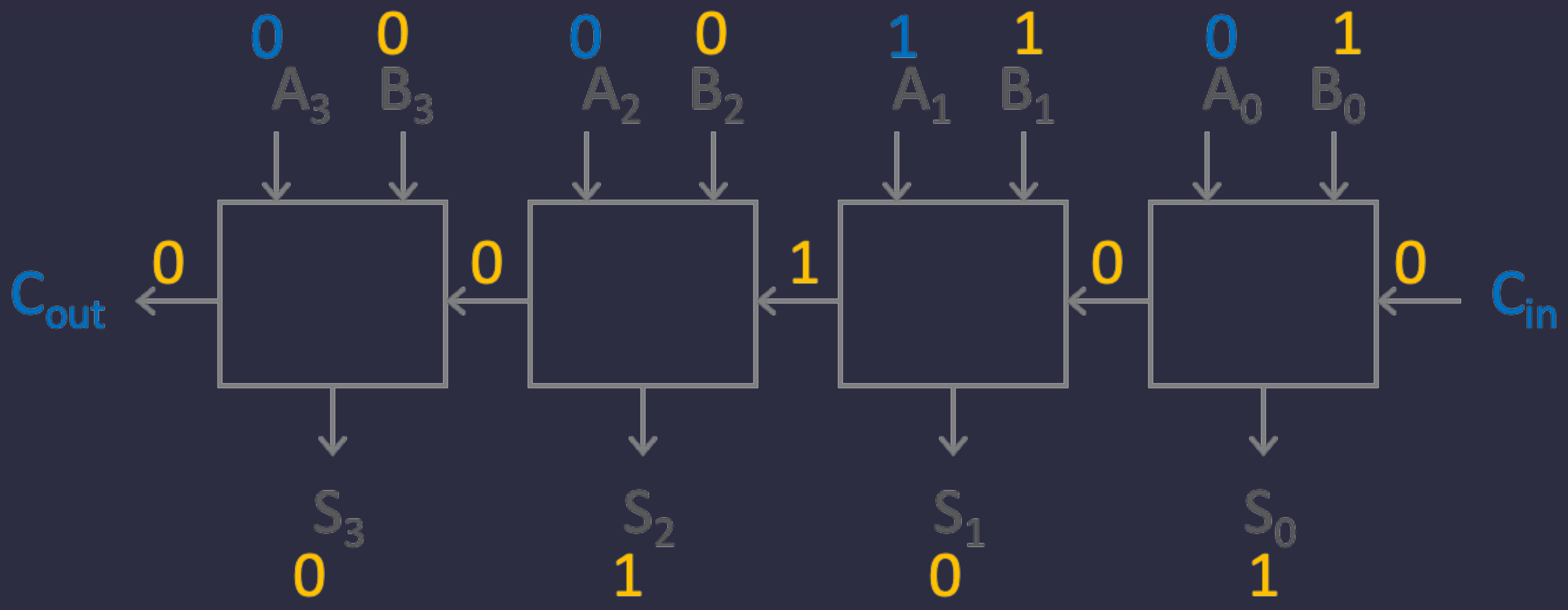


4-Bit Full Adder

- Adds two 4-bit numbers and carry in
- Computes 4-bit result and carry out
- Can be cascaded

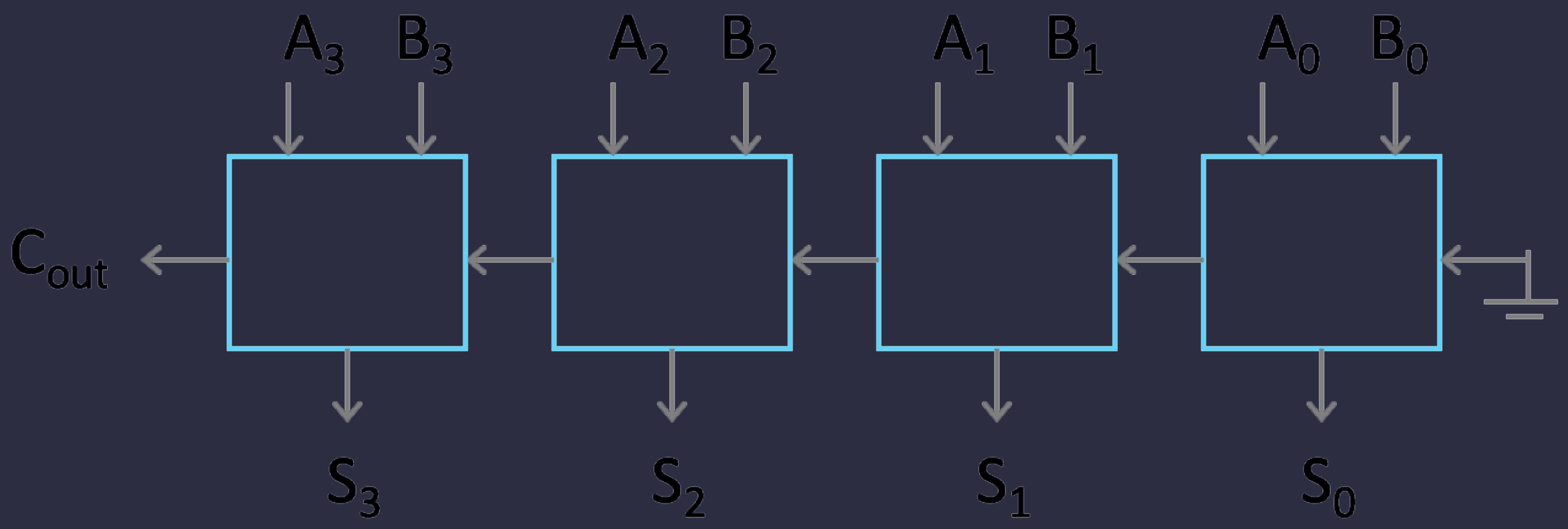


4-bit Adder



- Adds two 4-bit numbers, along with carry-in
- Computes 4-bit result and carry out
- Carry-out = overflow indicates result does not fit in 4 bits

4-bit Adder



- Adds two 4-bit numbers, along with carry-in
- Computes 4-bit result and carry out
- Carry-out = overflow indicates result does not fit in 4 bits

1st Attempt: Sign/Magnitude Representation

- First Attempt: Sign/Magnitude Representation

  - 1 bit for sign (0=positive, 1=negative)
  - N-1 bits for magnitude

<u>0</u>111 = 7
<u>1</u>111 = -7

Problem?

- Two zero's: +0 different than -0
- Complicated circuits
- -2 + 1 = ???

<u>0</u>000 = +0
<u>1</u>000 = -0

Second Attempt: One's complement

- Second Attempt: One's complement
  - Leading 0's for positive and 1's for negative
  - Negative numbers: complement the positive number

0111 = 7
1000 = -7

- Problem?
  - Two zero's still: +0 different than -0
  - -1 if offset from two's complement
  - Complicated circuits
    - Carry is difficult

0000 = +0
1111 = -0

Two's Complement Representation

What is used: Two's Complement Representation

Nonnegative numbers are represented as usual

- 0 = 0000, 1 = 0001, 3 = 0011, 7 = 0111

Leading 1's for negative numbers

To negate any number:

- complement *all* the bits (i.e. flip all the bits)
- then add 1
- -1: 1 ⇒ 0001 ⇒ 1110 ⇒ 1111
- -3: 3 ⇒ 0011 ⇒ 1100 ⇒ 1101
- -7: 7 ⇒ 0111 ⇒ 1000 ⇒ 1001
- -8: 8 ⇒ 1000 ⇒ 0111 ⇒ 1000
- -0: 0 ⇒ 0000 ⇒ 1111 ⇒ 0000 (this is good, -0 = +0)

Two's Complement

Non-negatives
(as usual):

+0 = 0000
+1 = 0001
+2 = 0010
+3 = 0011
+4 = 0100
+5 = 0101
+6 = 0110
+7 = 0111
+8 = 1000

Negatives (two's complement)

flip	then add 1
$\bar{0}$ = 1111	-0 = 0000
$\bar{1}$ = 1110	-1 = 1111
$\bar{2}$ = 1101	-2 = 1110
$\bar{3}$ = 1100	-3 = 1101
$\bar{4}$ = 1011	-4 = 1100
$\bar{5}$ = 1010	-5 = 1011
$\bar{6}$ = 1001	-6 = 1010
$\bar{7}$ = 1000	-7 = 1001
$\bar{8}$ = 0111	-8 = 1000

Two’s Complement vs. Unsigned

-1 =	1111	= 15
-2 =	1110	= 14
-3 =	1101	= 13
-4 =	1100	= 12
-5 =	1011	= 11
-6 =	1010	= 10
-7 =	1001	= 9
-8 =	1000	= 8
+7 =	0111	= 7
+6 =	0110	= 6
+5 =	0101	= 5
+4 =	0100	= 4
+3 =	0011	= 3
+2 =	0010	= 2
+1 =	0001	= 1
0 =	0000	= 0

4 bit
Two’s
Complement
-8 … 7

4 bit
Unsigned
Binary
0 … 15

2s Complement

Calculate the following twos complement number value in decimal base

11010

$$\begin{array}{r} 11010 \\ \\ 00101 \quad \text{(flip)} \\ \underline{+1} \\ -6 = 00110 \end{array}$$

Two's Complement Facts

Signed two's complement

- Negative numbers have leading 1's
- zero is unique: +0 = - 0
- wraps from largest positive to largest negative

N bits can be used to represent

- unsigned: range $0 \ldots 2^N-1$
  - eg: 8 bits ⇒ 0…255
- signed (two's complement): $-(2^{N-1}) \ldots (2^{N-1} - 1)$
  - E.g.: 8 bits ⇒ (1000 000) … (0111 1111)
  - -128 … 127

Sign Extension & Truncation

Extending to larger size

- 1111 = -1
- 1111 1111 = -1
- 0111 = 7
- 0000 0111 = 7

Truncate to smaller size

- 0000 1111 = 15
- BUT, ~~0000~~ 1111 = 1111 = -1

Two’s Complement Addition

- Addition with two’s complement signed numbers
- Addition as usual. Ignore the sign. It just works!
- Examples
  - 1 + -1 = 0001 + 1111 = 0000 (0)
  - -3 + -1 = 1101 + 1111 = 1100 (-4)
  - -7 + 3 = 1001 + 0011 = 1100 (-4)
  - 7 + (-3) = 0111 + 1101 = 0100 (4)

Clicker Question
Which of the following has problems?

a) 7 + 1 = 1000 overflow
b) -7 + -3 = 1 0110 overflow
c) -7 + -1 = 1000 fine
d) Only (a) and (b) have problems
e) They all have problems

-1 =	1111	= 15
-2 =	1110	= 14
-3 =	1101	= 13
-4 =	1100	= 12
-5 =	1011	= 11
-6 =	1010	= 10
-7 =	1001	= 9
-8 =	1000	= 8
+7 =	0111	= 7
+6 =	0110	= 6
+5 =	0101	= 5
+4 =	0100	= 4
+3 =	0011	= 3
+2 =	0010	= 2
+1 =	0001	= 1
0 =	0000	= 0

Overflow

When can overflow occur?

- adding a negative and a positive?
  - Overflow *cannot occur* (Why?)
  - Always subtract larger magnitude from smaller
- adding two positives?
  - Overflow *can occur* (Why?)
  - Precision: Add two positives, and get a negative number!
- adding two negatives?
  - Overflow *can occur* (Why?)
  - Precision: add two negatives, get a positive number!

Rule of thumb:

- Overflow happens iff
  carry into msb != carry out of msb

Overflow

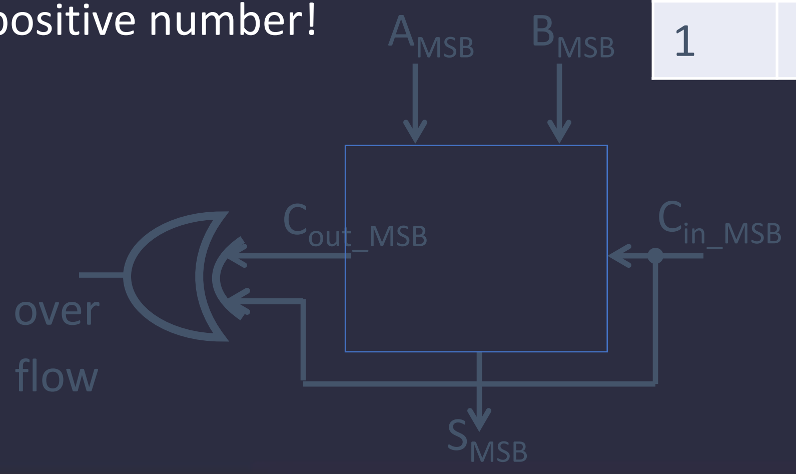
When can overflow occur?

- adding a negative and a positive?
  - Overflow *cannot occur* (Why?)
  - Always subtract larger magnitude from smaller
- adding two positives?
  - Overflow *can occur* (Why?)
  - Precision: Add two positives, and get a negative number!
- adding two negatives?
  - Overflow *can occur* (Why?)
  - Precision: add two negatives, get a positive number!

A	B	C_{in}	C_{out}	S	
0	0	0	0	0	
0	0	1	0	1	Wrong Sign
0	1	0	0	1	
0	1	1	1	0	
1	0	0	0	1	
1	0	1	1	0	
1	1	0	1	0	Wrong Sign
1	1	1	1	1	

Rule of thumb:

- Overflow happens iff
  carry into msb != carry out of msb



Youtube Overflow

YouTube
Shared publicly - Dec 1, 2014

We never thought a video would be watched in numbers greater than a 32-bit integer (=2,147,483,647 views), but that was before we met PSY. "Gangnam Style" has been viewed so many times we had to upgrade to a 64-bit integer (9,223,372,036,854,775,808)!

Hover over the counter in PSY's video to see a little math magic and stay tuned for bigger and bigger numbers on YouTube.

Binary Subtraction

Why create a new circuit?

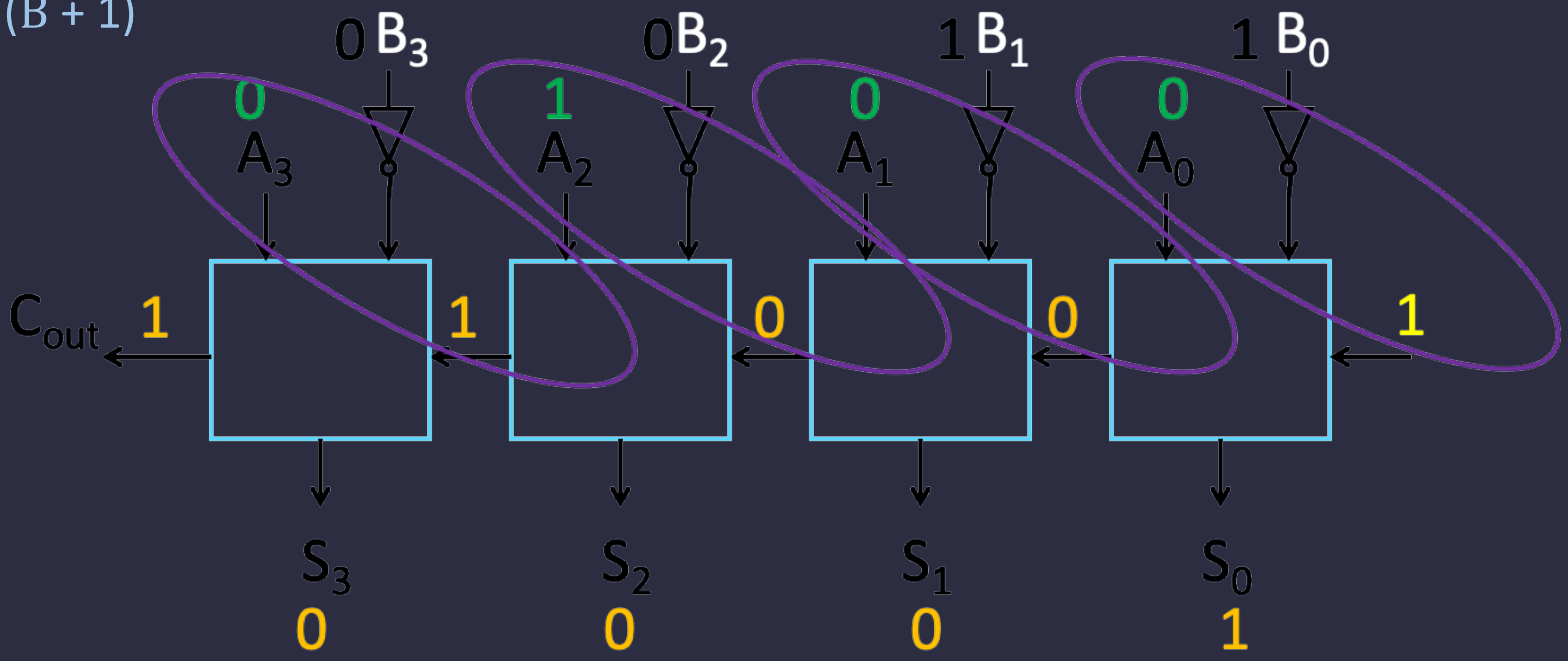
Just use addition using two’s complement math

- How?

Binary Subtraction

- Two's Complement Subtraction
  - Subtraction is simply addition,
    where one of the operands has been negated
    - Negation is done by inverting all bits and adding one

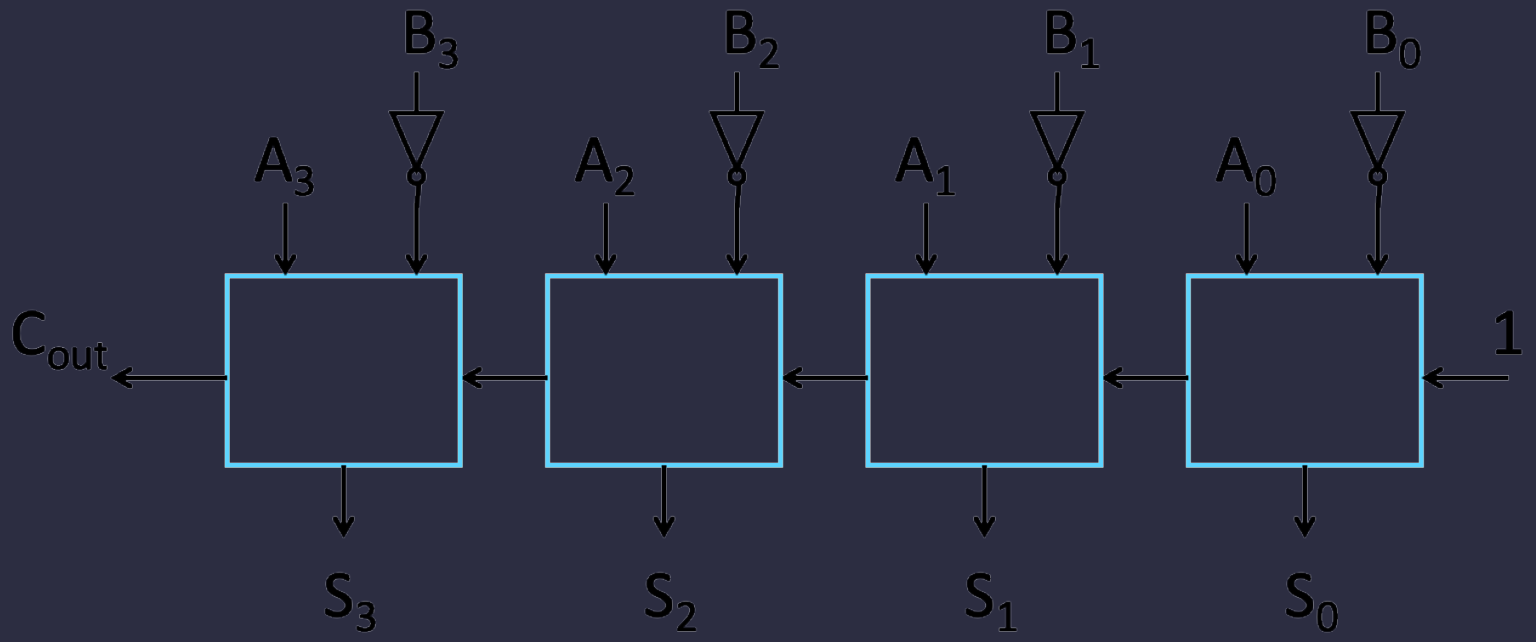
$A - B = A + (-B) = A + (\overline{B} + 1)$



Binary Subtraction

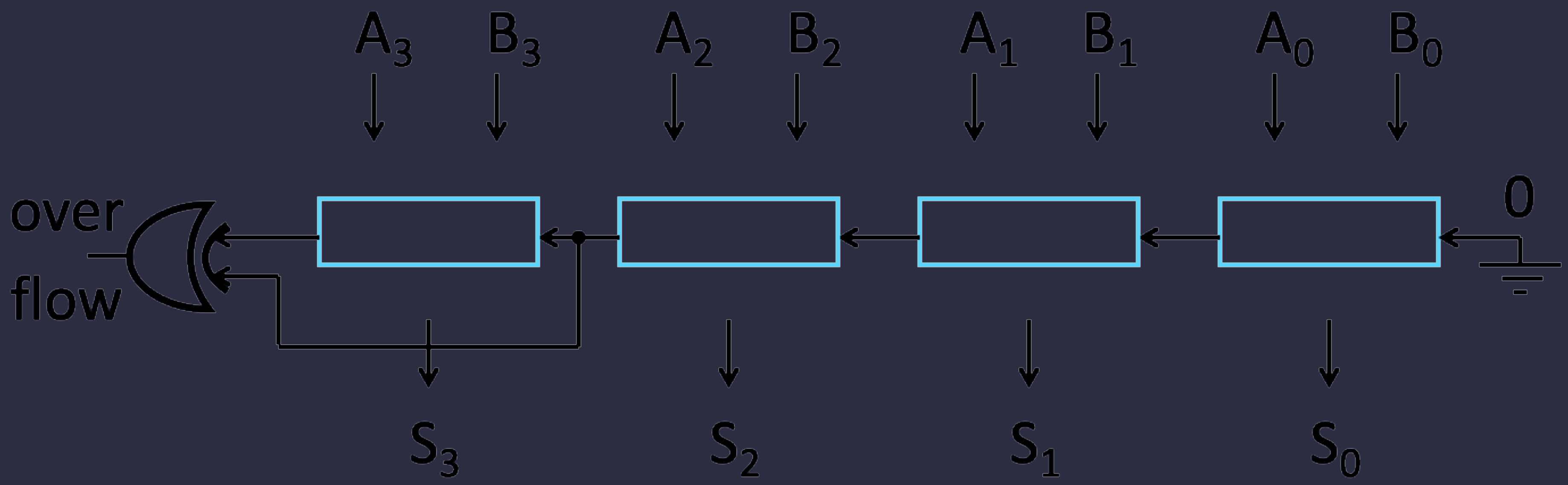
- Two's Complement Subtraction
  - Subtraction is simply addition,
    where one of the operands has been negated
    - Negation is done by inverting all bits and adding one

$A - B = A + (-B) = A + (\overline{B} + 1)$



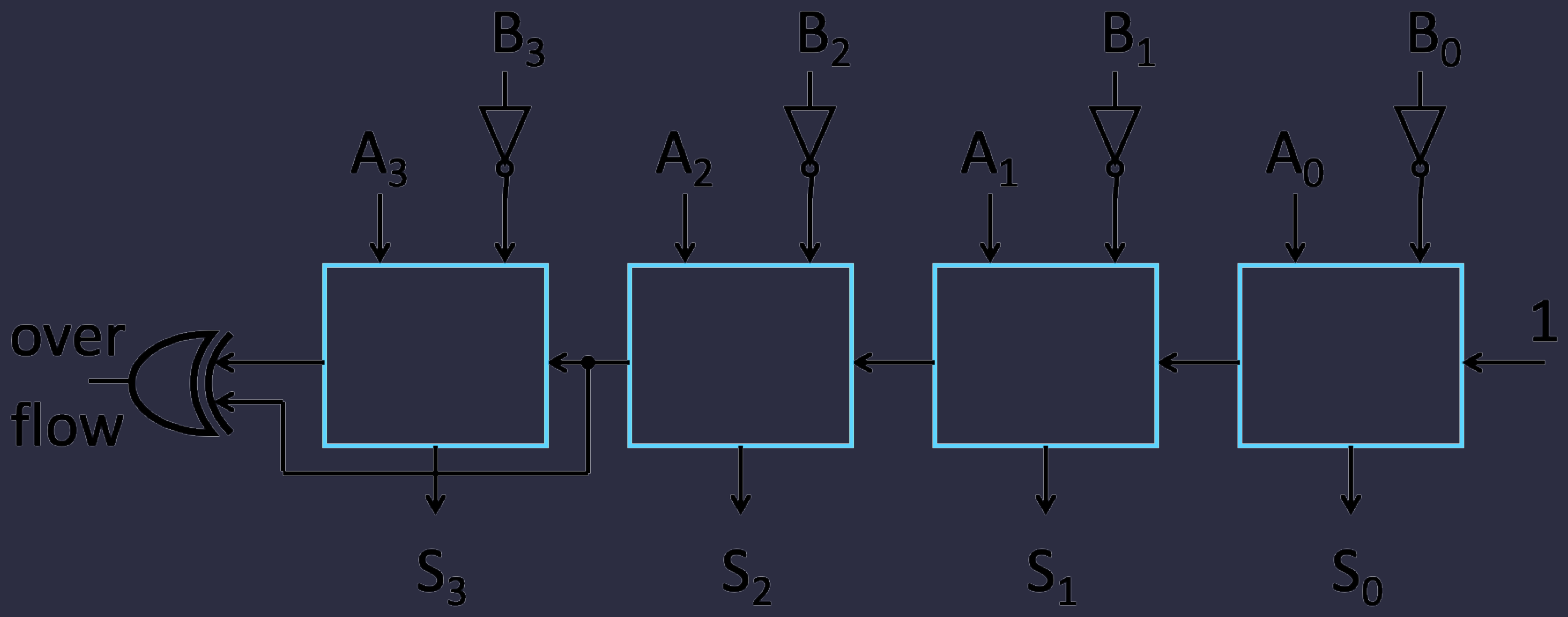
Two's Complement Adder

Two's Complement Adder with overflow detection



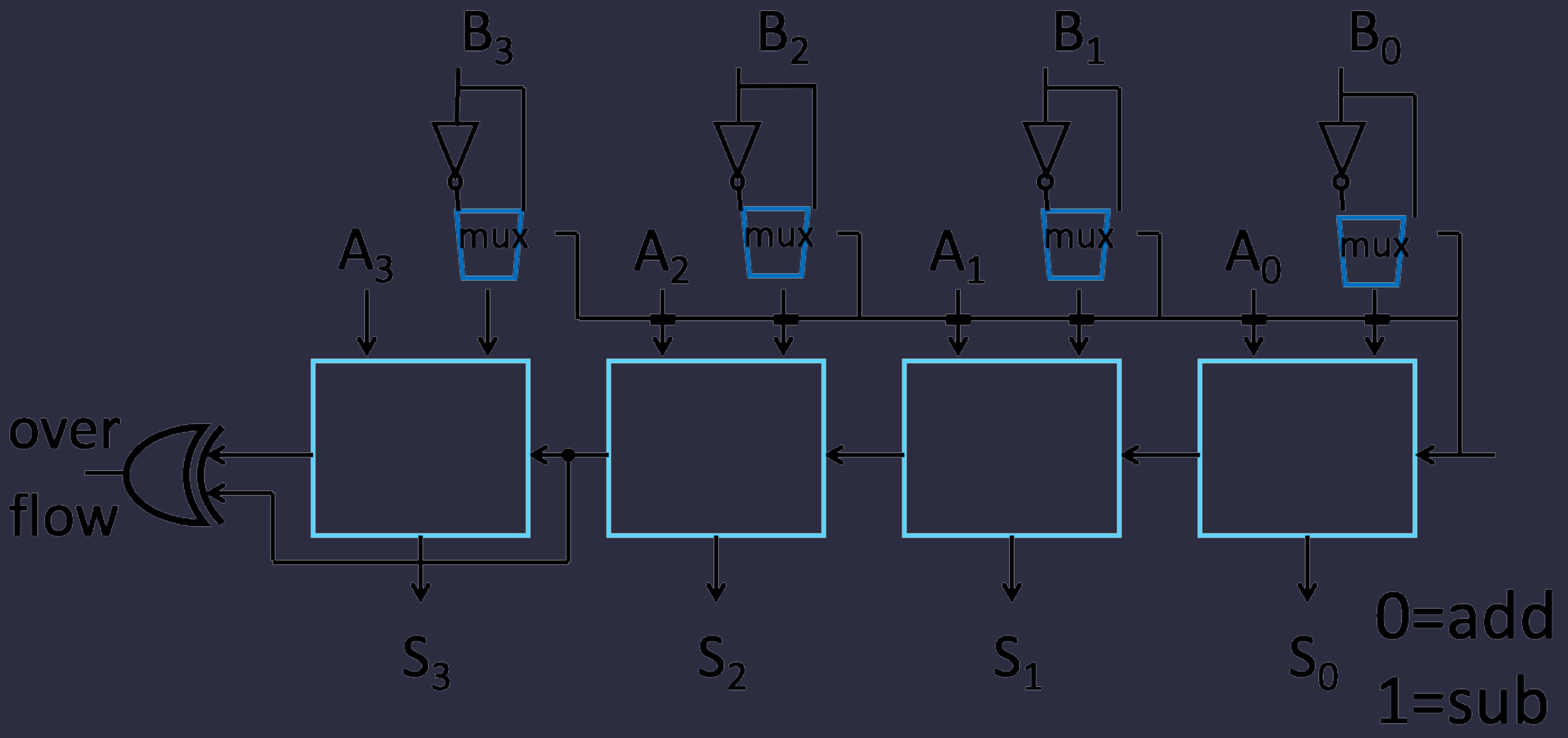
Two’s Complement Adder

Two’s Complement Adder with overflow detection



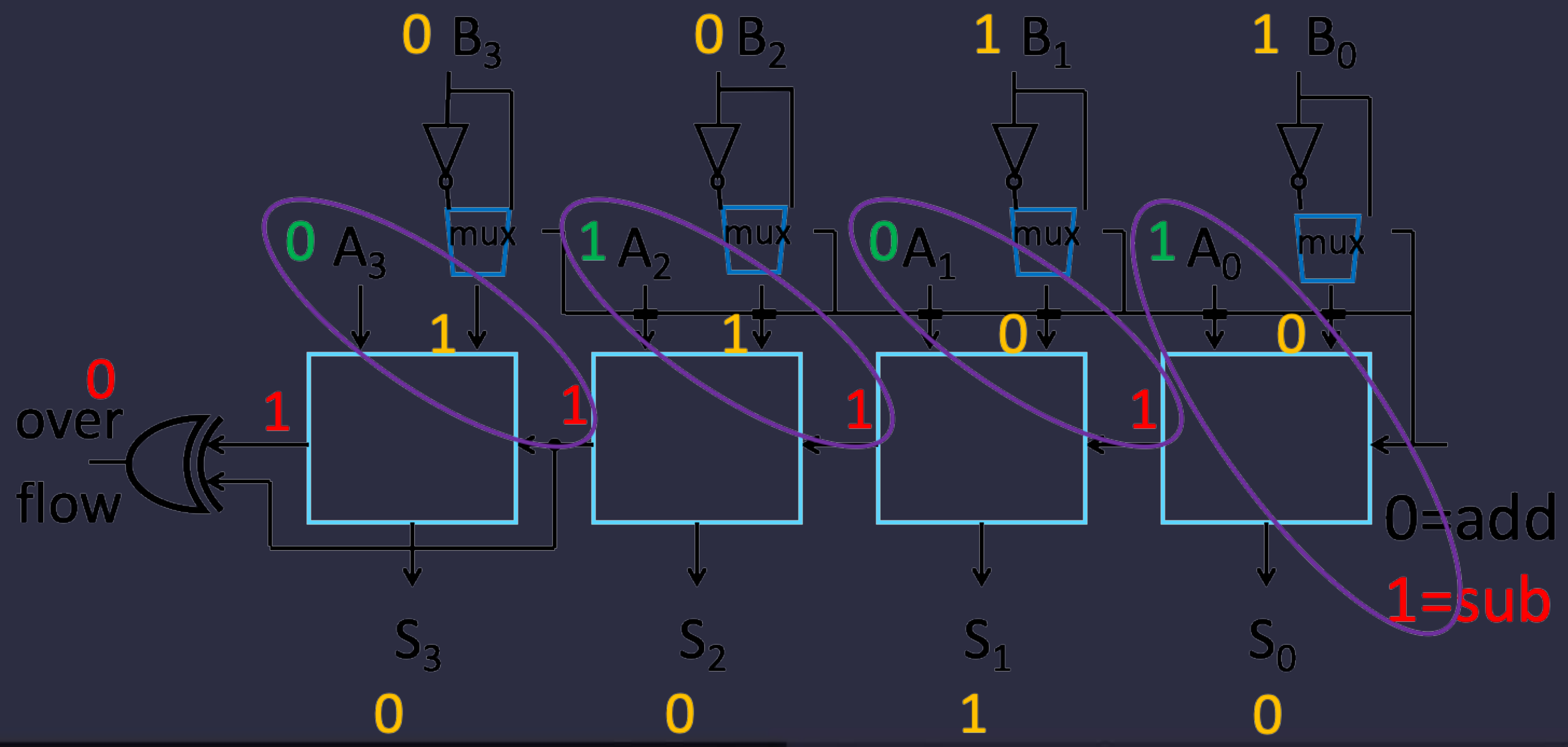
Two’s Complement Adder

Two’s Complement Adder with overflow detection



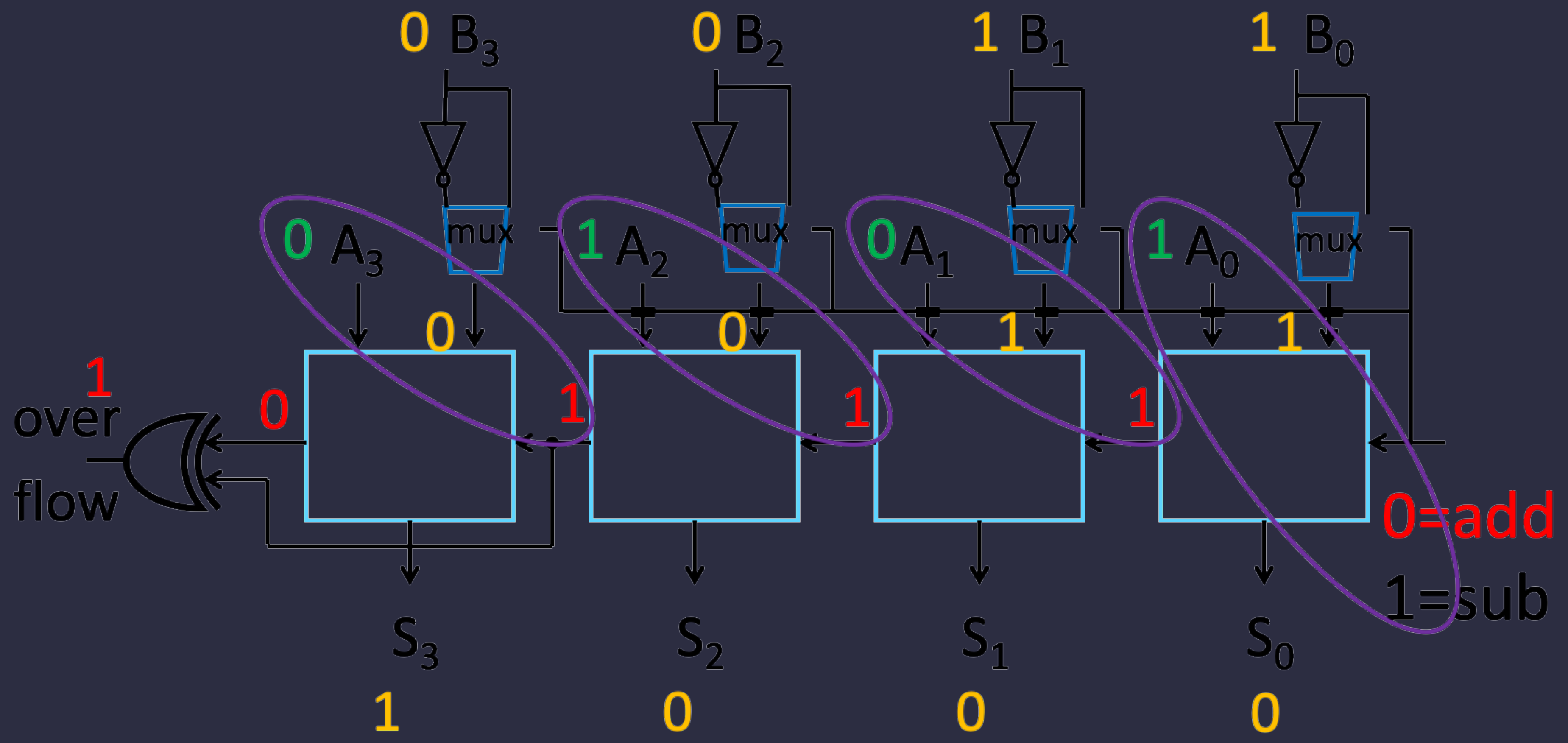
Two’s Complement Adder

Two’s Complement Adder with overflow detection



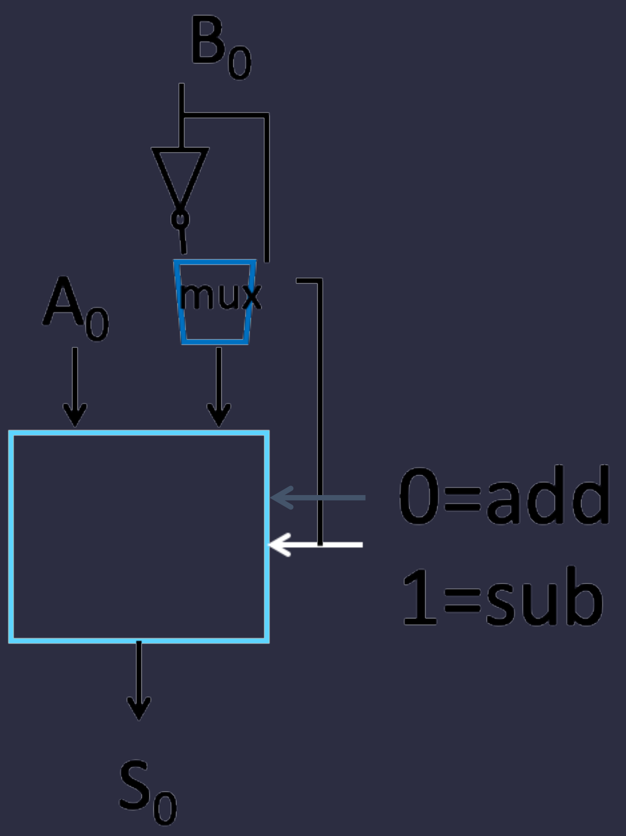
Two's Complement Adder

Two's Complement Adder with overflow detection

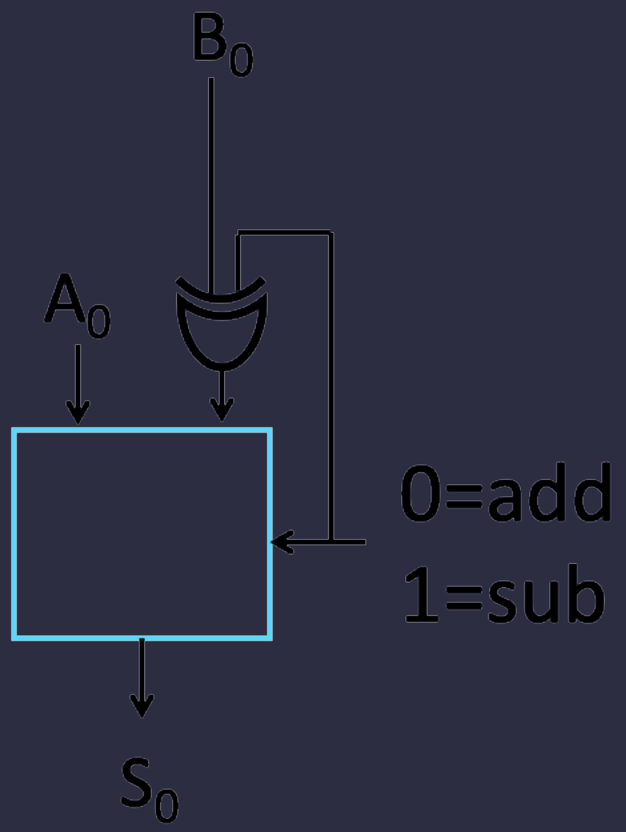


Two’s Complement Adder

Two’s Complement Adder with overflow detection



Before: 2 inverters, 2 AND gates, 1 OR gate



After: 1 XOR gate