BASIC LOGIC DESIGN

LOGICAL OPERATIONS

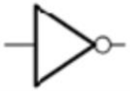
- Logical operations (Boolean algebra)
- named after George Bool, a famous mathematician
- AND, OR, NOT, XOR, NAND, NOR

LOGICAL OPERATIONS

NOT Gate

- operates on one bit
- logical reverse (usually denoted by ”!” or ”~”)
- !0 = ˜0 = 1
- !1 = ˜1 = 0

NOT

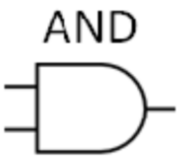


INPUT	OUTPUT
A	
0	1
1	0

LOGICAL OPERATIONS

AND Gate

- operates on two bits
- logical and (usually denoted by ”&” or ”.”): both have to be true



INPUT		OUTPUT
A	B	
0	0	0
1	0	0
0	1	0
1	1	1

LOGICAL OPERATIONS

OR Gate

- operates on two bits
- logical or (usually denoted by ”|” or ”+”): at least one has to be true

OR

INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	1

LOGICAL OPERATIONS

XOR Gate

- operates on two bits
- logical exclusive or (usually denoted by ”⊕”): only one is true

XOR

INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	0

LOGICAL OPERATIONS

NAND and NOR

- NAND = NOT AND
- NOR = NOT OR

- A NAND B = !(A.B)
- A NOR B = !(A+B)

NAND

INPUT		OUTPUT
A	B	
0	0	1
1	0	1
0	1	1
1	1	0

NOR

INPUT		OUTPUT
A	B	
0	0	1
1	0	0
0	1	0
1	1	0

LOGICAL OPERATIONS

NAND and NOR are very convenient

- You can build any other gate out of NANDs and NORs
- So, any circuit can be built out of just NANDs or NORs

NAND

INPUT		OUTPUT
A	B	
0	0	1
1	0	1
0	1	1
1	1	0

NOR

INPUT		OUTPUT
A	B	
0	0	1
1	0	0
0	1	0
1	1	0

LOGICAL OPERATIONS

NOT out of NAND

!(A) = A NAND A

AND out of NAND

A AND B = (A NAND B) NAND (A NAND B)

OR out of NAND

A OR B = (A NAND A) NAND (B NAND B)

DE MORGAN'S LAWS

$$\overline{(A \cdot B)} = \bar{A} + \bar{B}$$

$$\overline{(A + B)} = \bar{A} \cdot \bar{B}$$

DIGITAL CIRCUITS

With basic gates and logical operations, you can build any logical functions or arithmetic functions.

For example, if you want to choose something based on a condition

i.e. if S = 0, choose A,
if S = 1, choose B.

SIMPLE DIGITAL CIRCUIT

If S = 0, choose A,
if S = 1, choose B.

A	B	S	Output
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

SIMPLE DIGITAL CIRCUIT

If S = 0, choose A,
if S = 1, choose B.

A	B	S	Output
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**

SIMPLE DIGITAL CIRCUIT

If S = 0, choose A,
if S = 1, choose B.

A	B	S	Output
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**

SIMPLE DIGITAL CIRCUIT

If S = 0, choose A,
if S = 1, choose B.

A	B	S	Output
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**

$$\boldsymbol{Output = \overline{A} \cdot B \cdot S + A \cdot \overline{B} \cdot \overline{S} + A \cdot B \cdot \overline{S} + A \cdot B \cdot S}$$

SIMPLE DIGITAL CIRCUIT

Can you simplify this?

$$\mathbf{Output} = \overline{\mathbf{A}} \cdot \mathbf{B} \cdot \mathbf{S} + \mathbf{A} \cdot \overline{\mathbf{B}} \cdot \overline{\mathbf{S}} + \mathbf{A} \cdot \mathbf{B} \cdot \overline{\mathbf{S}} + \mathbf{A} \cdot \mathbf{B} \cdot \mathbf{S}$$

If S = 0, choose A,
if S = 1, choose B.

- Identity law: $A+0=A$ and $A \cdot 1=A$
- Zero and One laws: $A+1=1$ and $A \cdot 0=0$
- Inverse laws: $A+\overline{A}=1$ and $A \cdot \overline{A}=0$
- Commutative laws: $A+B=B+A$ and $A \cdot B=B \cdot A$
- Associative laws: $A+(B+C)=(A+B)+C$ and $A \cdot (B \cdot C)=(A \cdot B) \cdot C$
- Distributive laws: $A \cdot (B+C)=(A \cdot B)+(A \cdot C)$ and $A+(B \cdot C)=(A+B) \cdot (A+C)$

SIMPLE DIGITAL CIRCUIT

Can you simplify this?

$$\mathbf{Output} = \bar{\mathbf{A}} \cdot \mathbf{B} \cdot \mathbf{S} + \mathbf{A} \cdot \bar{\mathbf{B}} \cdot \bar{\mathbf{S}} + \mathbf{A} \cdot \mathbf{B} \cdot \bar{\mathbf{S}} + \mathbf{A} \cdot \mathbf{B} \cdot \mathbf{S}$$

$$\mathbf{Output} = \mathbf{S}\left(\bar{\mathbf{A}}\mathbf{B} + \mathbf{AB}\right) + \bar{\mathbf{S}}\left(\mathbf{A}\bar{\mathbf{B}} + \mathbf{AB}\right)$$

If S = 0, choose A,
if S = 1, choose B.

$$\mathbf{Output} = \mathbf{SB}\left(\bar{\mathbf{A}} + \mathbf{A}\right) + \bar{\mathbf{S}}\mathbf{A}\left(\bar{\mathbf{B}} + \mathbf{B}\right)$$

$$\mathbf{Output} = \mathbf{SB} + \bar{\mathbf{S}}\mathbf{A}$$

- Identity law: $A+0=A$ and $A \cdot 1=A$
- Zero and One laws: $A+1=1$ and $A \cdot 0=0$
- Inverse laws: $A+\bar{A}=1$ and $A \cdot \bar{A}=0$
- Commutative laws: $A+B=B+A$ and $A \cdot B=B \cdot A$
- Associative laws: $A+(B+C)=(A+B)+C$ and $A \cdot (B \cdot C)=(A \cdot B) \cdot C$
- Distributive laws: $A \cdot (B+C)=(A \cdot B)+(A \cdot C)$ and $A+(B \cdot C)=(A+B) \cdot (A+C)$

SIMPLE DIGITAL CIRCUIT

If S = 0, choose A,
if S = 1, choose B.

Can you simplify this?

Using K-MAP

A \ BS	00	01	11	10
0	0	0	1	0
1	1	0	1	1

- Identity law: $A+0=A$ and $A \cdot 1=A$
- Zero and One laws: $A+1=1$ and $A \cdot 0=0$
- Inverse laws: $A+\overline{A}=1$ and $A \cdot \overline{A}=0$
- Commutative laws: $A+B=B+A$ and $A \cdot B=B \cdot A$
- Associative laws: $A+(B+C)=(A+B)+C$ and $A \cdot (B \cdot C)=(A \cdot B) \cdot C$
- Distributive laws: $A \cdot (B+C)=(A \cdot B)+(A \cdot C)$ and $A+(B \cdot C)=(A+B) \cdot (A+C)$

MULTIPLEXOR 2:1

very common circuit where the name is MUX (# of inputs) : (# of output)

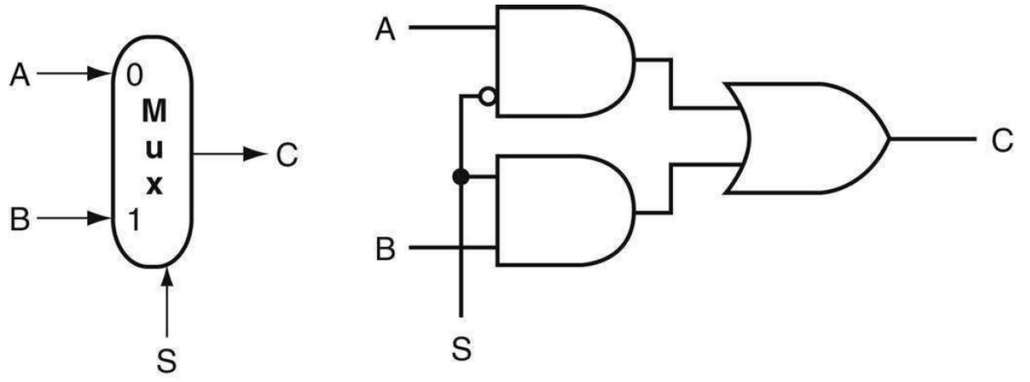


FIGURE A.3.2 **A two-input multiplexor on the left and its implementation with gates on the right.**

BUILDING AN ADDER

- Let's do an arithmetic circuit
- An adder which adds two 1-bit together

A	B	R	Carry-out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

BUILDING AN ADDER

- 1-bit adder actually needs two outputs
- One for the output
- One for the carry-out

A	B	R	Carry-out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

BUILDING AN ADDER

- 1-bit adder actually needs two outputs
- One for the output
- One for the carry-out

A	B	R	Carry-out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Output functions

- $R = A \oplus B$
- $\text{Carry-out} = A.B$

BUILDING AN ADDER

- 1-bit adder actually needs two outputs
- One for the output
- One for the carry-out

A	B	R	Carry-out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

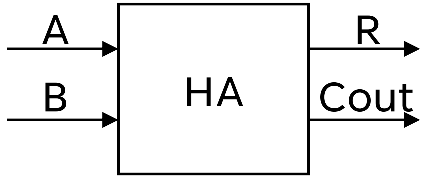
Output functions

- $R = A \oplus B$
- $\text{Carry-out} = A.B$

BUILDING AN ADDER

- This circuit is called, “Half Adder”

A	B	R	Carry-out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

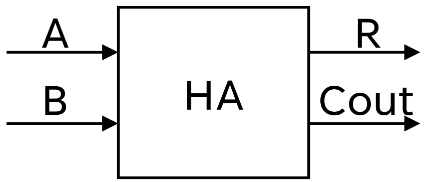


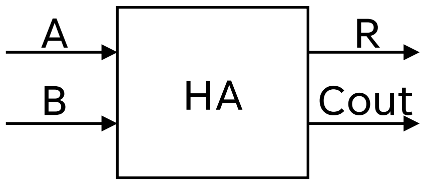
Output functions

- $R = A \oplus B$
- $\text{Carry-out} = A.B$

BUILDING AN ADDER

- To add more than two 1-bits together, the adder must add the carry from the other adder





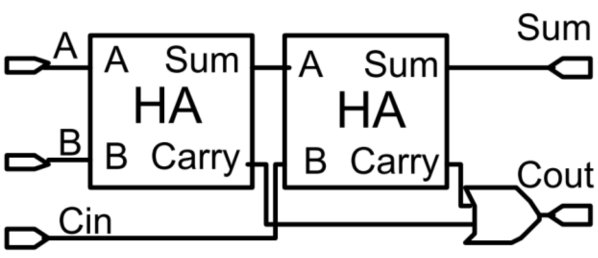
BUILDING AN ADDER

- Full Adder has 3 inputs and 2 outputs.

Carry-in	A	B	R	Carry-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

BUILDING AN ADDER

- Full Adder has 3 inputs and 2 outputs.
- Can use 2 half-adders

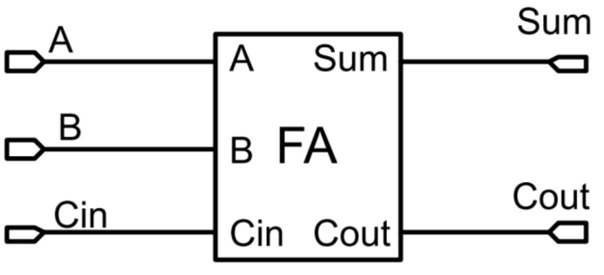


Carry-in	A	B	R	Carry-out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

BUILDING AN ADDER

- Full Adder has 3 inputs and 2 outputs.

Carry-in	A	B	R	Carry-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



Block diagram representation of the Full Adder