Digital Logic Design (CSNB163)

Module 9

Categories of Logic Circuit

- Digital logic circuits can be categorized based on the nature of their inputs either:
  - **Combinational logic circuit**

    It consists of logic gates whose outputs at any time are determined from the present combination of inputs. It can perform an operation that can be specified logically by a set of Boolean functions. (The subsequent modules are based on combinational logic circuit)
  - **Sequential logic circuit**

    It employs storage elements in addition to logic gates.

    Their outputs are a function of the inputs and the state of the storage elements. (the last module on flip-flops & latches is on sequential logic circuits)

Construction of Combinational Logic Circuit

- Combinational logic circuits can be constructed based on the following steps:
  1. Derive the **Truth Table** (to perform the given function)
  2. Derive the **Karnaugh Map** (from the truth table)
  3. Derive the **simplified Boolean expression(s)** (from the K-map(s))
  4. Draw the **circuit diagram(s)** (to implement the simplified Boolean expression(s))

Binary Adder

- Binary adder performs the addition of two or more binary digits.
- Two types of binary adder:
  - **Half adder**

    The **input** is **two** binary variables (***a, b***)

    The **output** is **two** binary variables (***sum***, c_{out} being the output carry)
  - **Full adder**

    The input is **three** binary variables (***a, b***, c_{in} being the input carry)

    The **output** is **two** binary variables (***sum***, c_{out} being the output carry)

Half Adder

- This circuit needs:
  - 2 inputs; augend *(a)* and addend *(b)*
  - 2 outputs; sum *(sum)* and output carry (c_{out}).
- Truth table:

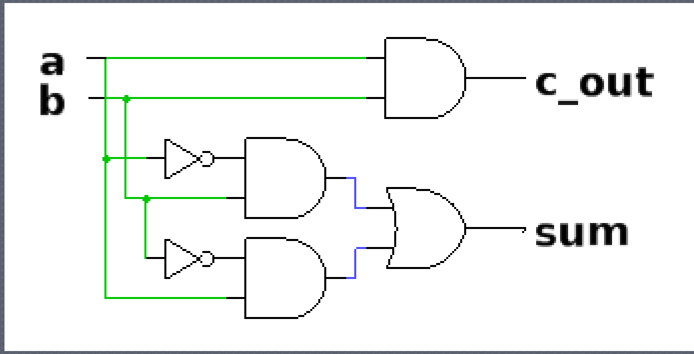
a	***b***	***sum***	c_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$sum = a'b + ab'$

$c_{out} = ab$

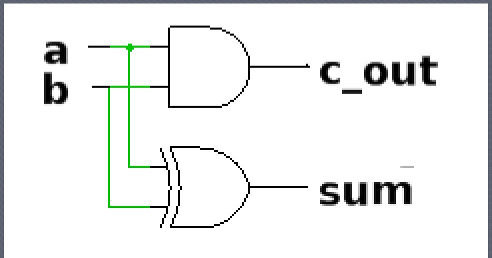
Half Adder (cont.)

- Circuit diagram:



$c_{out} = ab$

$sum = a'b + ab'$



Full Adder

- This circuit needs:
  - 3 inputs; augend *(a)*,addend *(b)* and input carry *(c_{in})*
  - 2 outputs; sum *(sum)* and output carry *(c_{out})*.
- Truth table:

a	*b*	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

$$sum = a'b'c_{in} + a'bc_{in}' + ab'c_{in}' + abc_{in}$$

$$c_{out} = a'bc_{in} + ab'c_{in} + abc_{in}' + abc_{in}$$

Full Adder (cont.)

- Sum of minterms (derived from Truth Table)

$$sum = a'b'c_{in} + a'bc_{in}' + ab'c_{in}' + abc_{in}$$

$$c_{out} = a'bc_{in} + ab'c_{in} + abc_{in}' + abc_{in}$$

- Karnaugh map:

sum

a \ bc_{in}	00	01	11	10
0		1		1
1	1		1	

c_{out}

a \ bc_{in}	00	01	11	10
0			1	
1		1	1	1

Group1: bc_{in}

Group2: ac_{in} Group3: ab

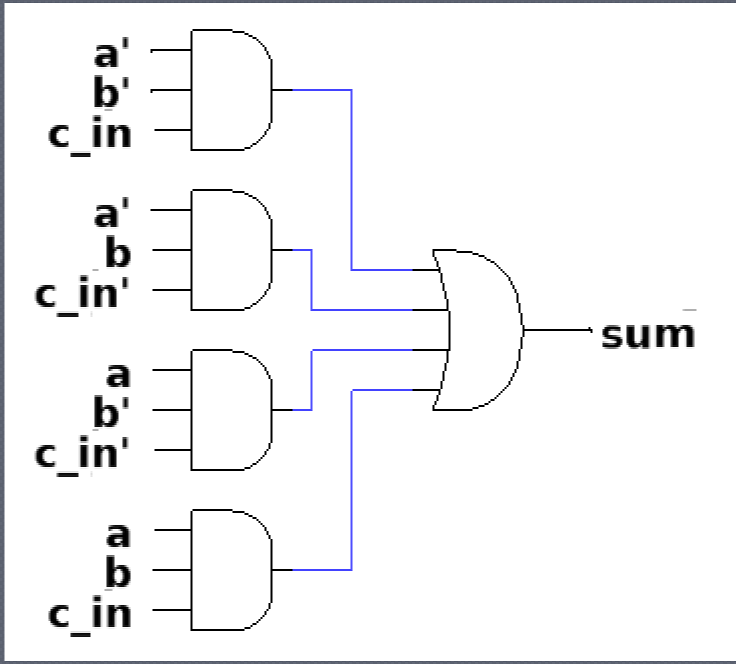
- Simplified Boolean expressions:

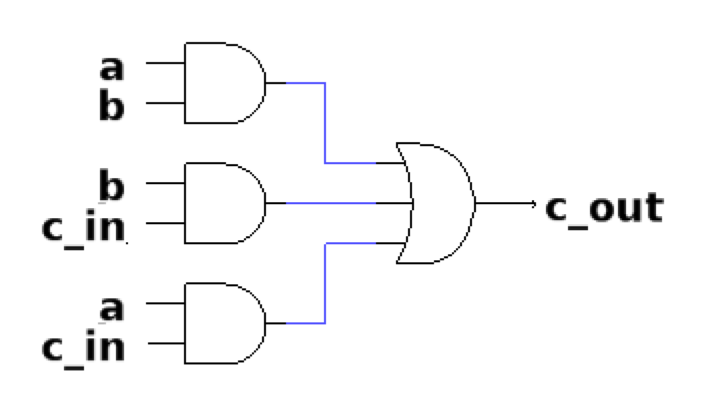
$$sum = a'b'c_{in} + a'bc_{in}' + ab'c_{in}' + abc_{in}$$

$$c_{out} = bc_{in} + ac_{in} + ab$$

Full Adder (cont.)

- Circuit diagram:



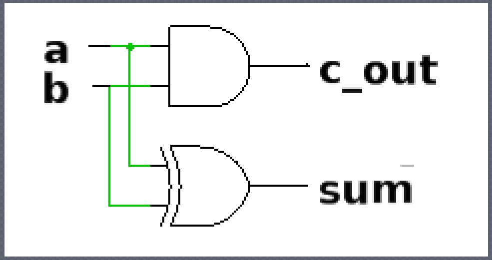


$c_{out} = bc_{in} + ac_{in} + ab$

$sum = a'b'c_{in} + a'bc_{in}' + ab'c_{in}' + abc_{in}$

Constructing a Full Adder using Half Adders

- A full adder can also be implemented by using two half adders and an OR gate.
- Recaps half adder consists of **AND** and **XOR** operation:



$c_{out} = ab$

$sum = a'b + ab'$

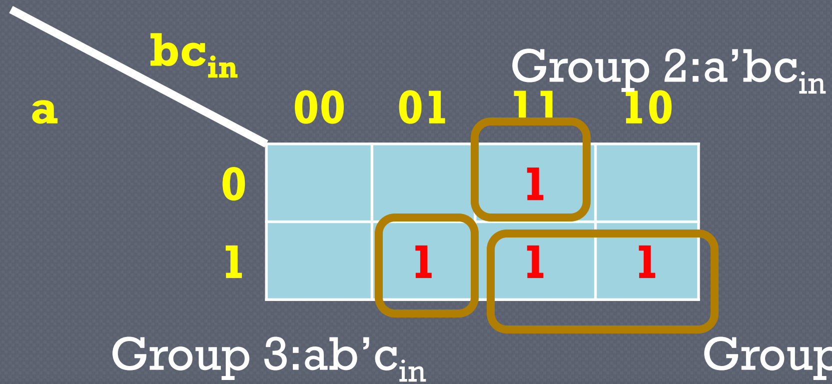
- Thus, in order to construct a full adder using half adders, we need to represent the Boolean expressions of a full adder using AND and XOR operations.

Constructing a Full Adder using Half Adders (cont.)

- Sum Boolean expression of a full adder:

$$sum = a'b'c_{in} + a'bc_{in}' + ab'c_{in}' + abc_{in}$$
$$sum = c_{in}(a'b' + ab) + c_{in}'\ (a'b + ab')$$
$$sum = c_{in}(a \oplus b)' + c_{in}'\ (a \oplus b)$$
$$sum = (a \oplus b) \oplus c$$

- C_{out} Boolean expression of a full adder:

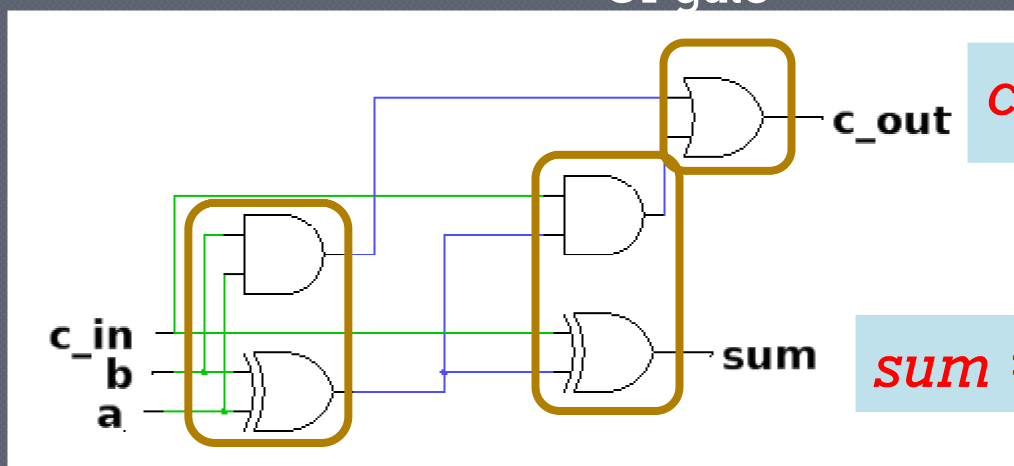


$$c_{out} = a'bc_{in} + ab'c_{in} + abc_{in}' + abc_{in}$$
$$c_{out} = ab + a'bc_{in} + ab'c_{in}$$
$$c_{out} = ab + c_{in}(a'b + ab')$$
$$c_{out} = ab + c_{in}(a \oplus b)$$

Constructing a Full Adder using Half Adders (cont.)

- Circuit diagram of a full adder (which is made up of 2 half adders and 1 OR gate):

Or gate



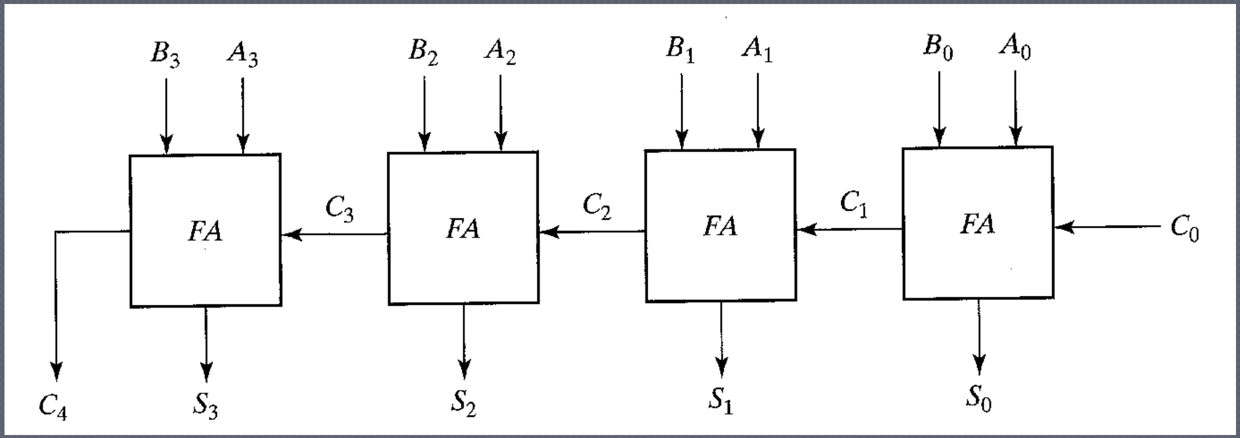
$c_{out} = ab + c_{in}(a \oplus b)$

$sum = (a \oplus b) \oplus c$

Half adder   Half adder

n-bits Cascaded Binary Adder

- **An *n*-bits binary adder** is capable of performing addition of binary digits up to ***n* bits**.
- **An *n*-bits binary adder** requires for ***n* full adders** setup in a **cascaded manner.**
- E.g. 4-bits binary adder with 4 full adders:



n-bits Cascaded Binary Adder (cont.)

- A 4-bits cascaded binary adder is capable of performing 4 bits binary operation as follows:

i	**4**	**3**	**2**	**1**	**0**
Input Carry (c_i)	-	1	1	0	0
Augend (a_i)	-	1	1	1	0
Addend (b_i)	-	1	0	1	1
Sum (*sum*)	-	1	0	0	1
Output carry (c_{i+1})	1	1	1	1	0

E.g. Augend = 1110

E.g. Addend = 1011

Sum = 1001 with output carry c_4 = 1

n-bits Cascaded Binary Adder (cont.)

- Properties:
  - The bits are added with full adder, **starting** from the **least significant** position to form the sum bit and carry bit then proceeds to the higher significant bits.
  - The innitial input carry c_0 must be **0**.
  - The **output carry c_{i+1}** is **transferred** into the **input carry c_i** of the full adder that adds the higher significant bits.
  - The sum bits are generated starting from the least significant bit and are available **as soon as the corresponding previous carry bit is generated**.
  - Thus, **all the carries must be generated for the correct sum bits to appear at the outputs**.
  - If each full adder has a delay of t_{delay};

$$\text{Total delay} = n \times t_{delay}$$

Digital Logic Design (CSNB163)

End of Module 9