CPT_S 260 Intro to Computer Architecture
Lecture 24

Digital Design IV
March 7, 2022

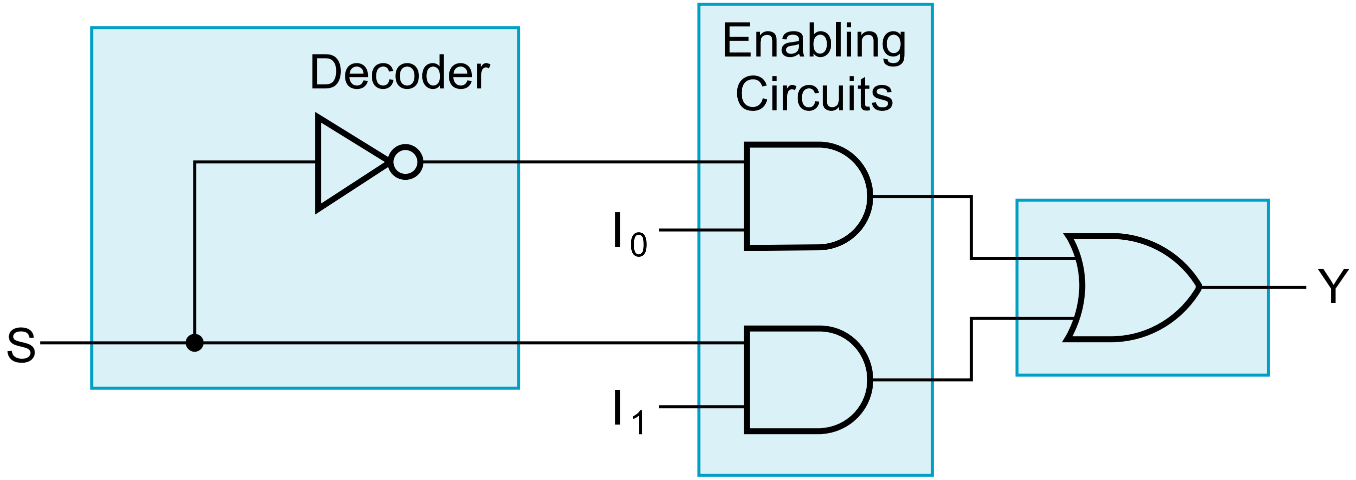
Ganapati Bhat
School of Electrical Engineering and Computer Science
Washington State University

Recap: 2-to-1-Line Multiplexer

- **Since 2 = 2^1, n = 1**
- **The single selection variable S has two values:**
  - S = 0 selects input I_0
  - S = 1 selects input I_1
- **The equation:**

$$Y = \bar{S}I_0 + SI_1$$

- **The circuit:**

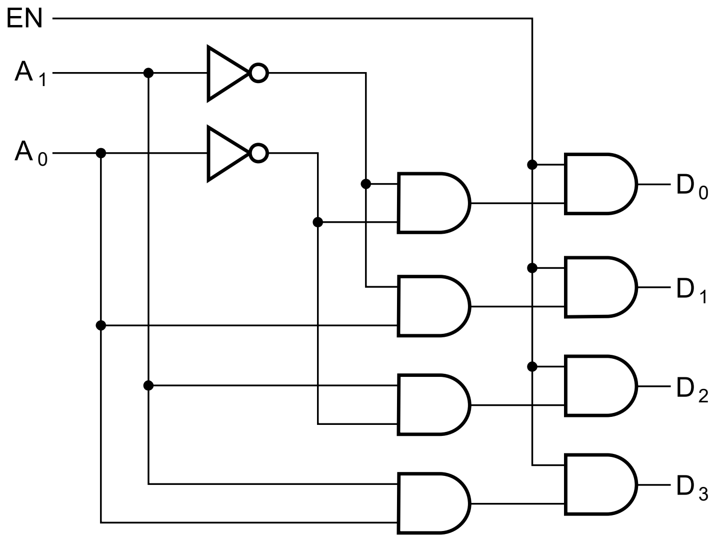


Recap: Decoders

- **Decoding - the conversion of an *n*-bit input code to an *m*-bit output code with $n \le m \le 2^n$ such that each valid code word produces a unique output code**
- **Circuits that perform decoding are called *decoders***
- **Here, functional blocks for decoding are**
  - called n-to-m line decoders, where $m \le 2^n$, and
- **Example: 2-to-4 decoder:**

EN	A_1	A_0	D_0	D_1	D_2	D_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

(a)



(b)

Logic Simplification

Rules of Boolean Algebra

- **Associative Law of multiplication**

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

- **Distributive Law of multiplication**

$$A + BC = (A + B) \cdot (A + C)$$

- **Annulment law:**

$$A \cdot 0 = 0$$
$$A + 1 = 1$$

- **Identity law:**

$$A \cdot 1 = A$$
$$A + 0 = A$$

Rules of Boolean Algebra

- **Complement law:**

$$A + \bar{A} = 1$$
$$A \cdot \bar{A} = 0$$

- **Double negation law:**

$$\bar{\bar{A}} = A$$

- **Absorption law:**

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

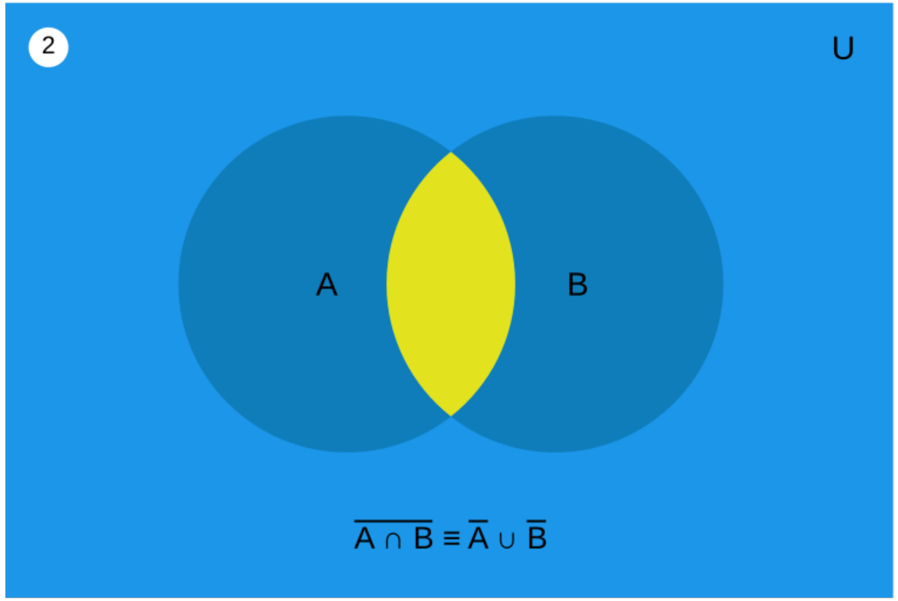
- **Idempotent law:**

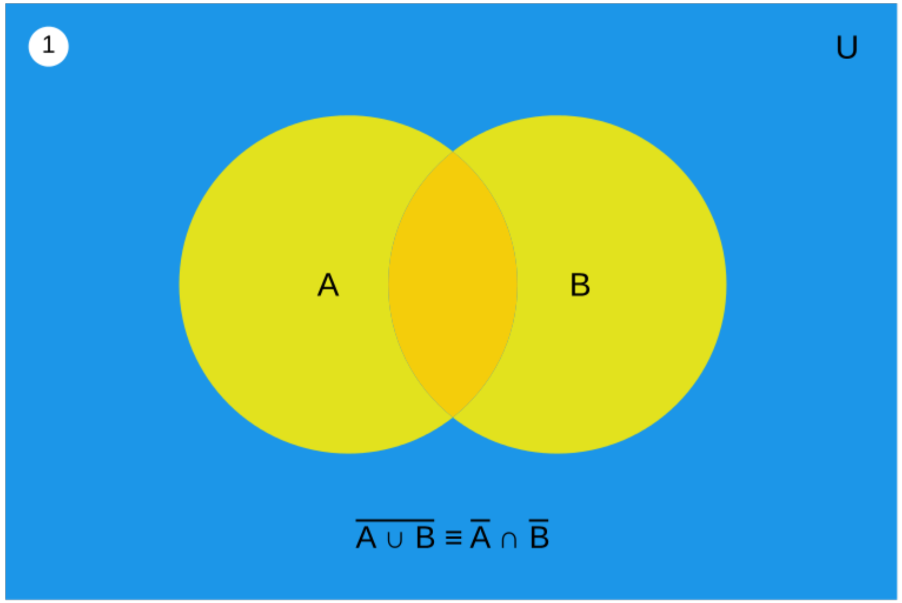
$$A + A = A$$
$$A \cdot A = A$$

De Morgan's Laws

- Transformation rules that help simplification of negations
- Statement:

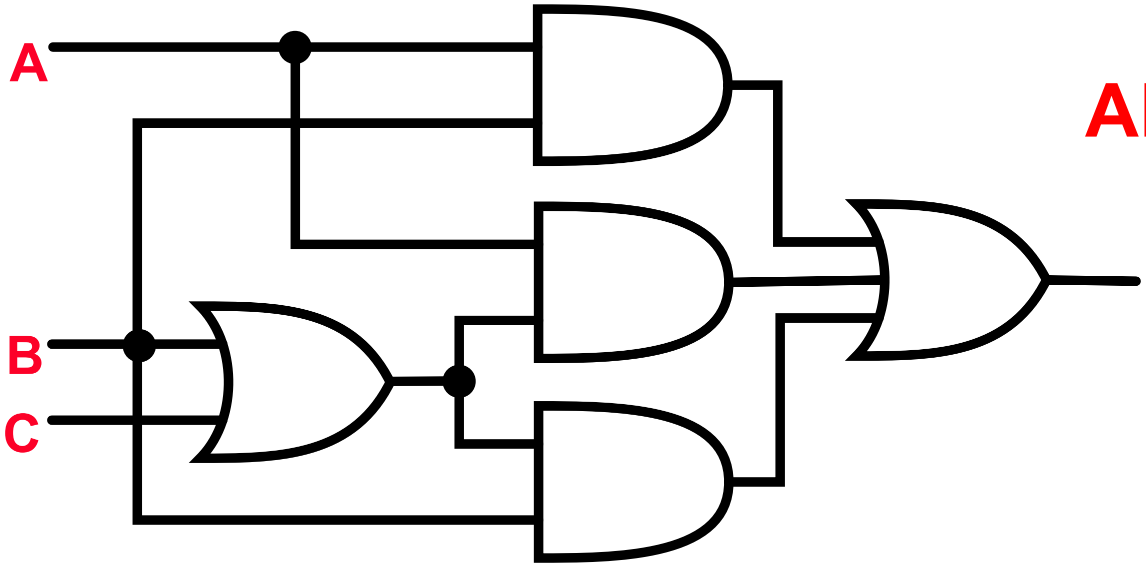
$$\overline{AB} = \bar{A} + \bar{B}$$
$$\overline{(A+B)} = \bar{A} \cdot \bar{B}$$





Simplification Using Boolean Algebra

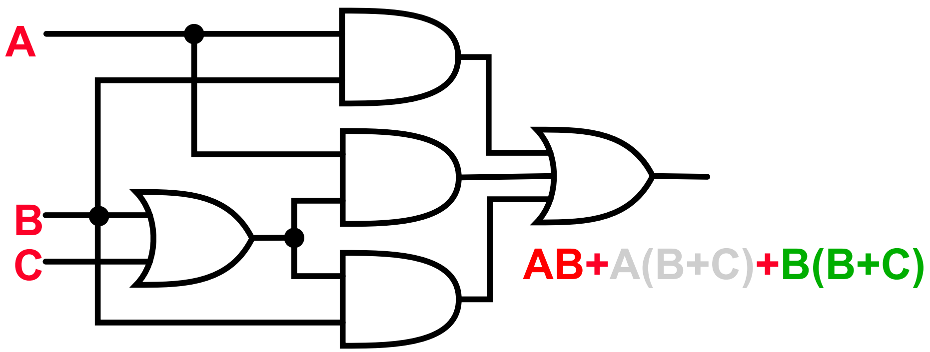
- **A simplified Boolean expression uses the fewest gates possible to implement a given expression.**

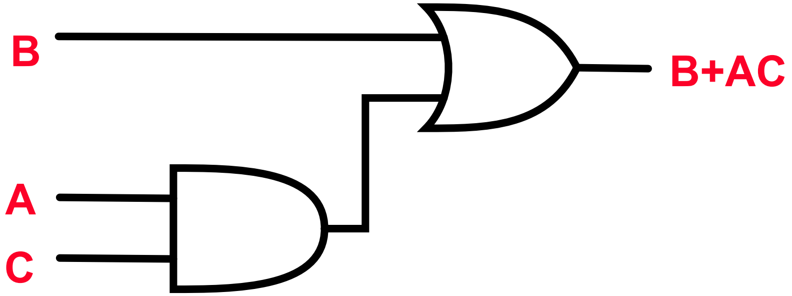


AB+A(B+C)+B(B+C)

Simplification Using Boolean Algebra

- **AB+A(B+C)+B(B+C)**
  - **(distributive law)**
    - » AB+AB+AC+BB+BC
  - (BB=B)
    - » AB+AB+AC+**B**+BC
  - (AB+AB=AB)
    - » **AB**+AC+B+BC
  - (B+BC=B)
    - » AB+AC+**B**
  - (AB+B=B)
    - » **B**+AC





Examples

- $[A\bar{B}(C + BD) + \bar{A}\bar{B}]C$
- $\bar{A}BC + A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C} + A\bar{B}C + ABC$
- $\overline{AB + AC} + \bar{A}\bar{B}C$

Examples

$$
\begin{aligned}
&[\boldsymbol{A\bar{B}(C + BD) + \bar{A}\bar{B}}]\boldsymbol{C} \\
&= [A\bar{B}C + A\bar{B}BD + \bar{A}\bar{B}]C \;(distributive\ law) \\
&= \;A\bar{B}CC + \bar{A}\bar{B}C \;(\bar{B}B = 0\ using\ complement\ law) \\
&= \;\bar{B}C(A + \;\bar{A}) \\
&= \;\bar{B}C \;(Complement\ law)
\end{aligned}
$$

Examples

$$\overline{AB + AC} + \bar{A}\bar{B}C$$
$$= (\overline{AB} \cdot \overline{AC}) + \bar{A}\bar{B}C \; (DeMorgan\ Law)$$
$$= (\bar{A} + \bar{B})(\bar{A} + \bar{C}) + \bar{A}\bar{B}C \; (DeMorgan\ Law)$$
$$= \bar{A}\bar{A} + \bar{A}\bar{C} + \bar{A}\bar{B} + \bar{B}\bar{C} + \bar{A}\bar{B}C$$
$$= \; \bar{A} + \bar{B}\bar{C} \; (Take \bar{A}\ out\ and\ use\ annulment\ law)$$

Standard Forms of Boolean Expressions

- **All Boolean expressions, regardless of their form, can be converted into either of two standard forms:**
  – The sum-of-products (SOP) form (minterms)
  – The product-of-sums (POS) form (maxterms)

- **Standardization makes the evaluation, simplification, and implementation of Boolean expressions much more systematic and easier**

Sum of Products

- Minterm Expressions
- If input is 0 we take the complement of the variable
- If input is 1 we take the variable as is
- To get the desired canonical SOP expression we will add the minterms (product terms) for which the output is 1

$$F = \bar{A}B + A\bar{B} + AB$$

A	B	F	Minterm
0	0	0	A'B'
0	1	1	A'B
1	0	1	AB'
1	1	1	AB

Product of Sums

A	B	F	Minterm
0	0	0	A'B'
0	1	1	A'B
1	0	1	AB'
1	1	1	AB

- **Maxterm Expressions**
- **If input is 1, we take the comp**
- **If input is 0, we take the varial**
- **To get the desired canonical POS expression we will multiply the maxterms (sum terms) for which the output is 0**

$$F = (A + B)$$

A	B	F	Maxterm
0	0	0	$A + B$
0	1	1	$A + \overline{B}$
1	0	1	$\overline{A} + B$
1	1	1	$\overline{A} + \overline{B}$

Designing a Simple ALU

(Arithmetic Logic Unit)

Adder Algorithm

	1	0	0	1
	0	1	0	1
Sum (S)	1	1	1	0
Carry (C)	0	0	0	1

Truth Table for the above operations:

A	B	Cin	Sum	Cout
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Adder Algorithm

	1	0	0	1
	0	1	0	1
Sum	1	1	1	0
Carry	0	0	0	1

Equations:

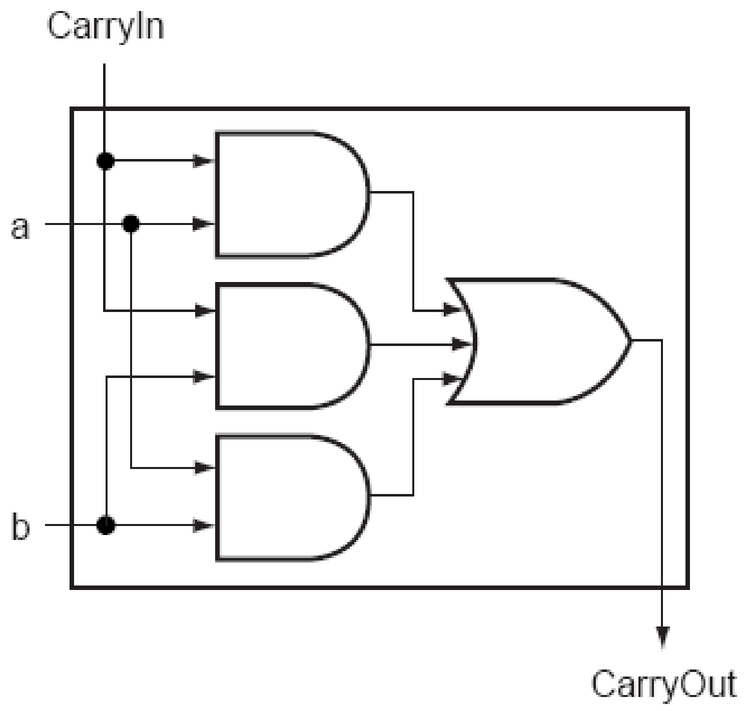
$$
\begin{aligned}
Sum = {} & Cin \cdot \overline{A} \cdot \overline{B} + \\
& B \cdot \overline{Cin} \cdot \overline{A} + \\
& A \cdot \overline{Cin} \cdot \overline{B} + \\
& A \cdot B \cdot Cin
\end{aligned}
$$

Truth Table for the above operations:

A	B	Cin	Sum	Cout
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

$$
\begin{aligned}
Cout = {} & A \cdot B \cdot Cin + \\
& A \cdot B \cdot \overline{Cin} + \\
& A \cdot Cin \cdot \overline{B} + \\
& B \cdot Cin \cdot \overline{A} \\
= {} & A \cdot B + \\
& A \cdot Cin + \\
& B \cdot Cin
\end{aligned}
$$

Carry Out Logic

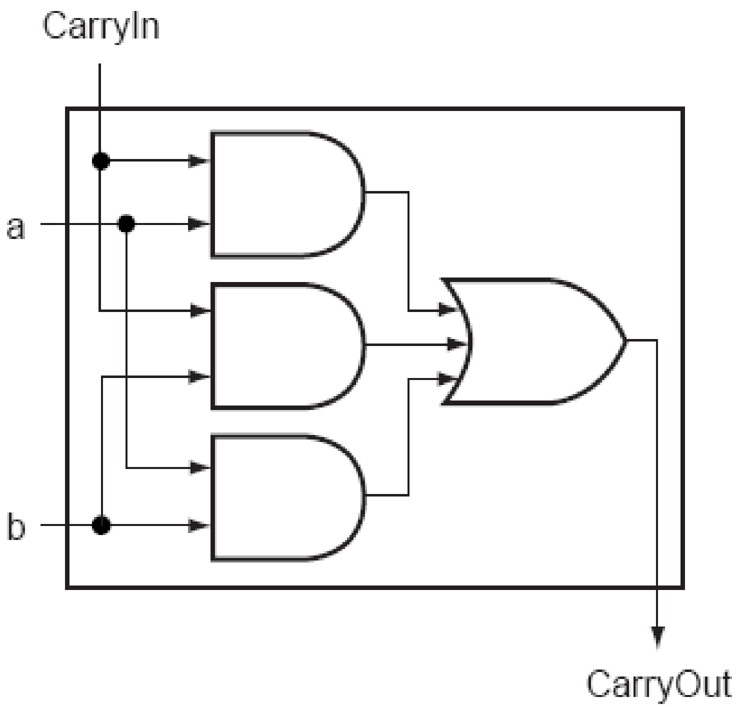


Equations:

Sum = $Cin \,.\, \overline{A} \,.\, \overline{B}$ +
$B \,.\, \overline{Cin} \,.\, \overline{A}$ +
$A \,.\, \overline{Cin} \,.\, \overline{B}$ +
$A \,.\, B \,.\, Cin$

Cout = $A \,.\, B \,.\, Cin$ +
$A \,.\, B \,.\, \overline{Cin}$ +
$A \,.\, Cin \,.\, \overline{B}$ +
$B \,.\, Cin \,.\, \overline{A}$
= $A \,.\, B$ +
$A \,.\, Cin$ +
$B \,.\, Cin$

Carry Out Logic



$$C_{out} = A \cdot B \cdot Cin + A \cdot B \cdot \overline{Cin} + A \cdot Cin \cdot \bar{B} + B \cdot Cin \cdot \bar{A}$$

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

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

Full-Adder (3 inputs, 2 outputs)

- **Full-Adder Truth Table:**

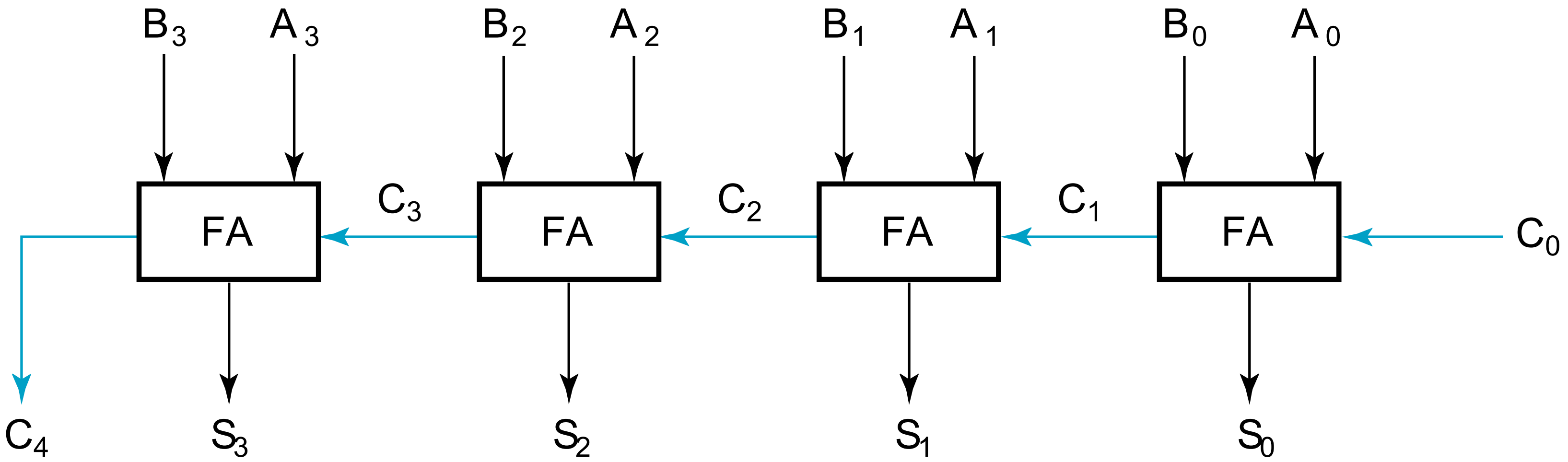
X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

- **Same as the previous example:**

$$S = X\overline{Y}\,\overline{Z} + \overline{X}Y\overline{Z} + \overline{X}\,\overline{Y}Z + XYZ$$
$$C = XY + XZ + YZ$$

4-bit Ripple-Carry Binary Adder

- **A four-bit Ripple Carry Adder made from four 1-bit Full Adders:**



32-bit Ripple Carry Adder

1-bit ALUs are connected “in series” with the carry-out of 1 box going into the carry-in of the next box