Advanced Boolean Logic

Prof. James L. Frankel
Harvard University

Version of 10:13 AM 2-Dec-2021

Half-Adder

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

Encapsulation of Half-Adder

Full-Adder

A	B	$Carry_{in}$	Sum	$Carry_{out}$
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

Full-Adder: All Zero Inputs

A	B	$Carry_{in}$	Sum	$Carry_{out}$
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

Full-Adder: Only One Input is High

A	B	$Carry_{in}$	Sum	$Carry_{out}$
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

Full-Adder: Exactly Two Inputs are High

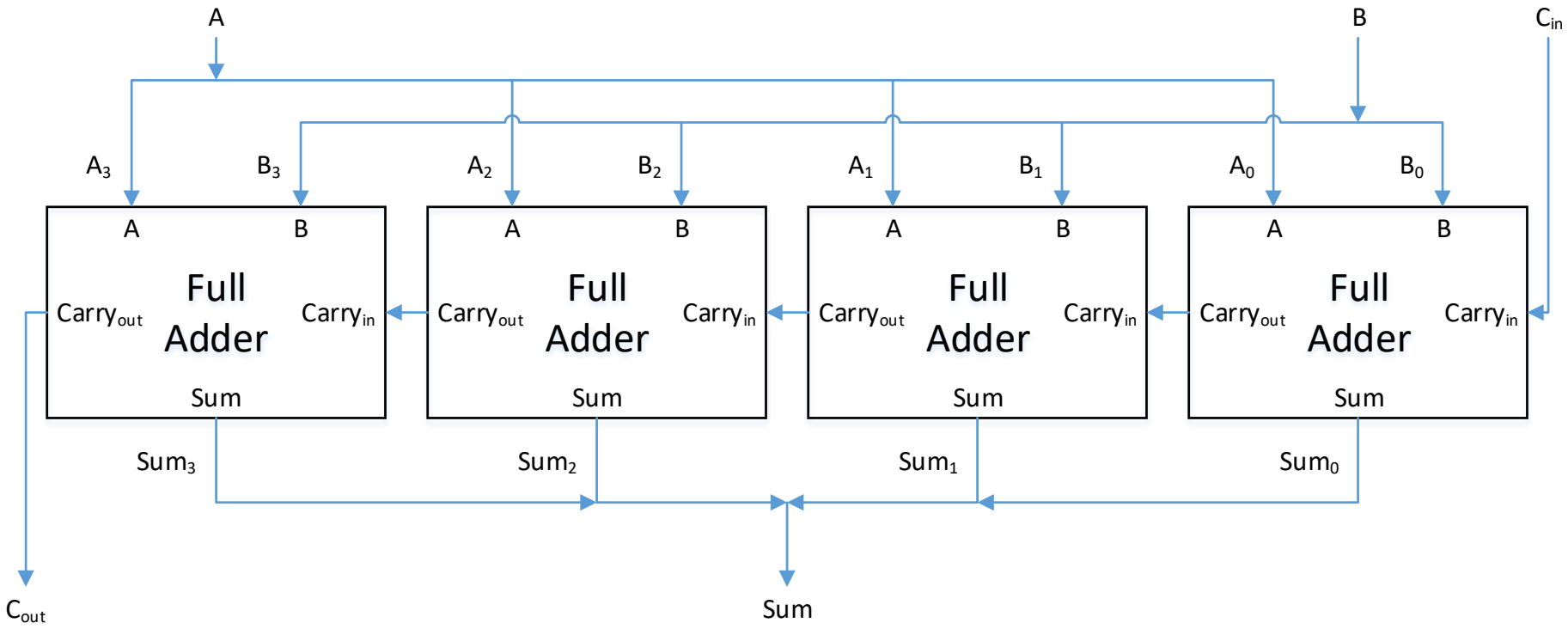
A	B	$Carry_{in}$	Sum	$Carry_{out}$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Full-Adder: All Three Inputs are High

A	B	$Carry_{in}$	Sum	$Carry_{out}$
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

Encapsulation of Full-Adder

Four-bit Adder Built from Full Adders



Sum-of-Products Form

- OR is like SUM (addition)
- AND is like PRODUCT (multiplication)

- So, a sum-of-products Boolean formula has any of the following forms:
  - (A AND B) OR (C AND D) OR (E AND F)
  - (A & B) | (C & D) | (E & F)
  - (A ∧ B) ∨ (C ∧ D) ∨ (E ∧ F)
  - (A · B) + (C · D) + (E · F)
  - (AB) + (CD) + (EF)
  - AB + CD + EF

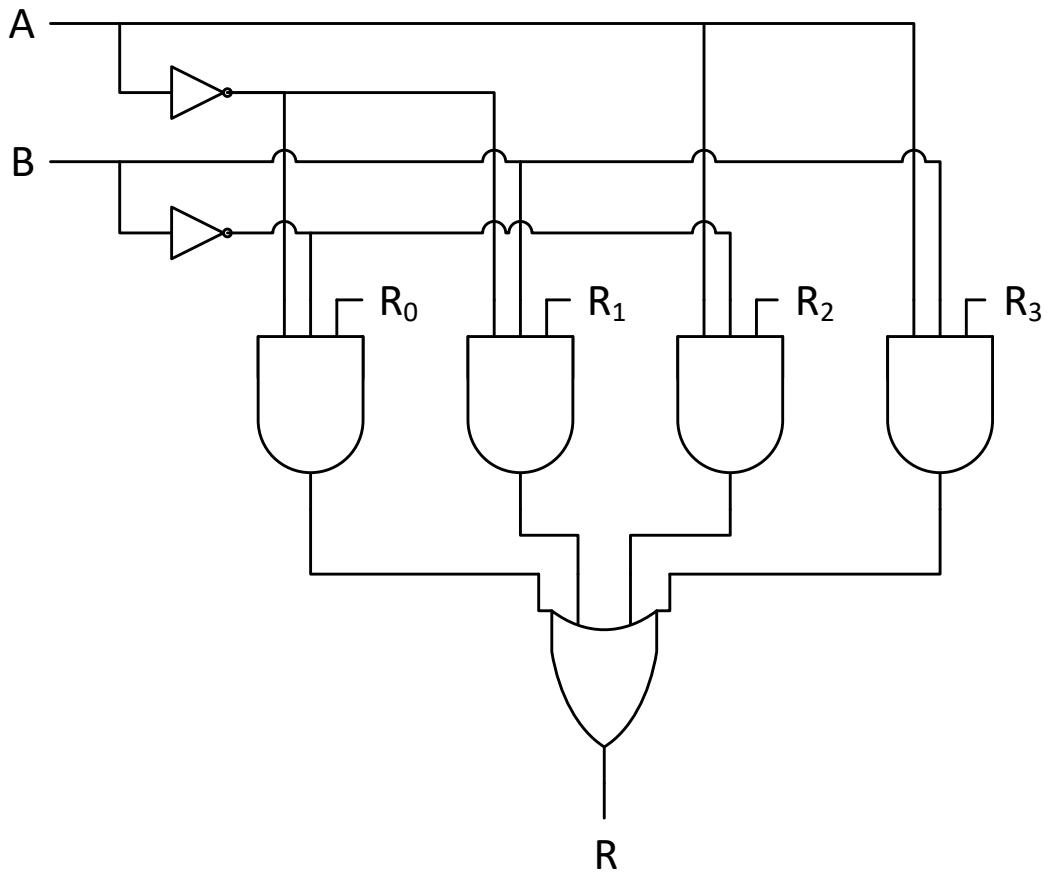
Sum-of-Products: Start with the Truth Table

A	B	R
0	0	R_0
0	1	R_1
1	0	R_2
1	1	R_3

- Generalized Formula: $R = \overline{A}\,\overline{B}R_0 + \overline{A}BR_1 + A\overline{B}R_2 + ABR_3$

Sum-of-Products: Could Draw Gates

- Generalized Formula: $R = \overline{A}\,\overline{B}R_0 + \overline{A}BR_1 + A\overline{B}R_2 + ABR_3$



Sum-of-Products: With Specific Values for R_0 through R_3, Terms are Removed and Simplified

A	B	R
0	0	$R_0 = 0$
0	1	$R_1 = 1$
1	0	$R_2 = 1$
1	1	$R_3 = 0$

- $R = \overline{A}\,\overline{B}0 + \overline{A}B1 + A\overline{B}1 + AB0$
- $R = \overline{A}B + A\overline{B}$

Sum-of-Products: Observation

- The only terms that persist are those with 1 outputs
- So, we need only include those terms

Sum-of-Products: Returning to the Full Adder

- Given the truth table,

A	B	$Carry_{in}$	Sum	$Carry_{out}$
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

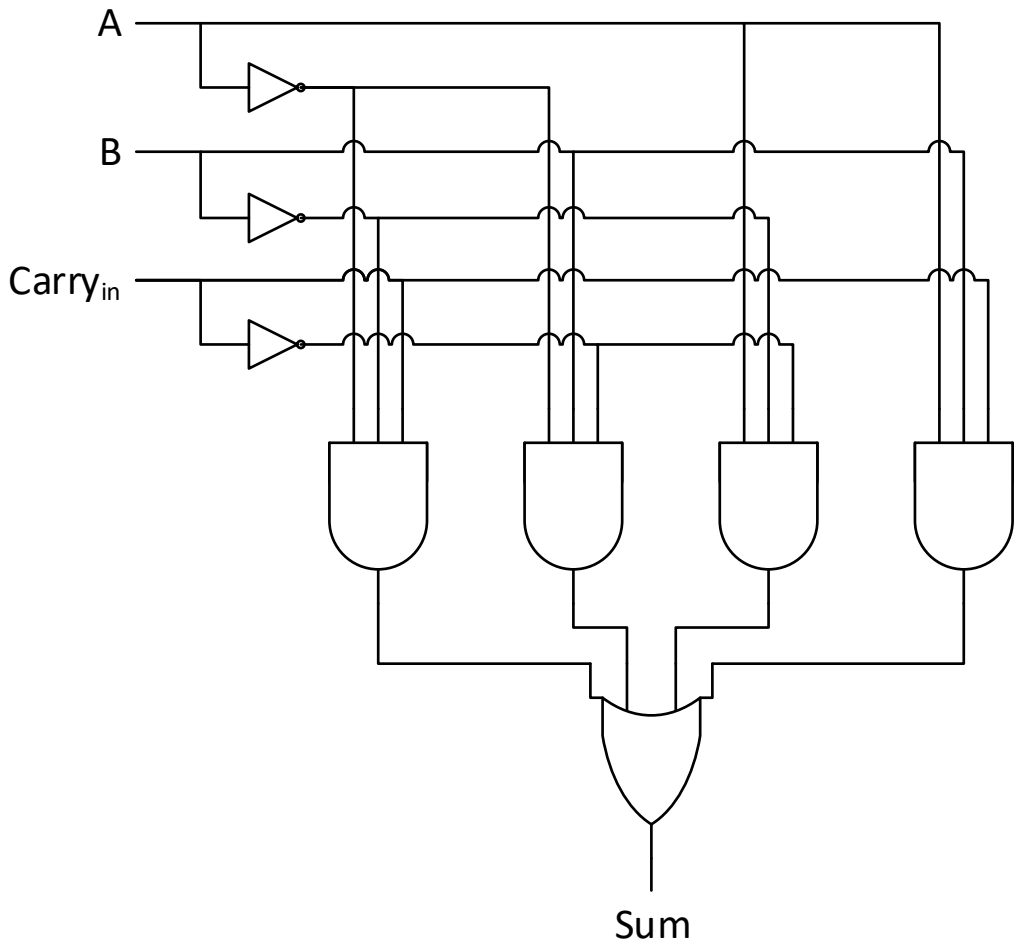
- The sum-of-products formulae are

$$Sum = \overline{A}\,\overline{B}Carry_{in} + \overline{A}B\overline{Carry_{in}} + A\overline{B}\overline{Carry_{in}} + ABCarry_{in}$$

$$Carry_{out} = \overline{A}BCarry_{in} + A\overline{B}Carry_{in} + AB\overline{Carry_{in}} + ABCarry_{in}$$

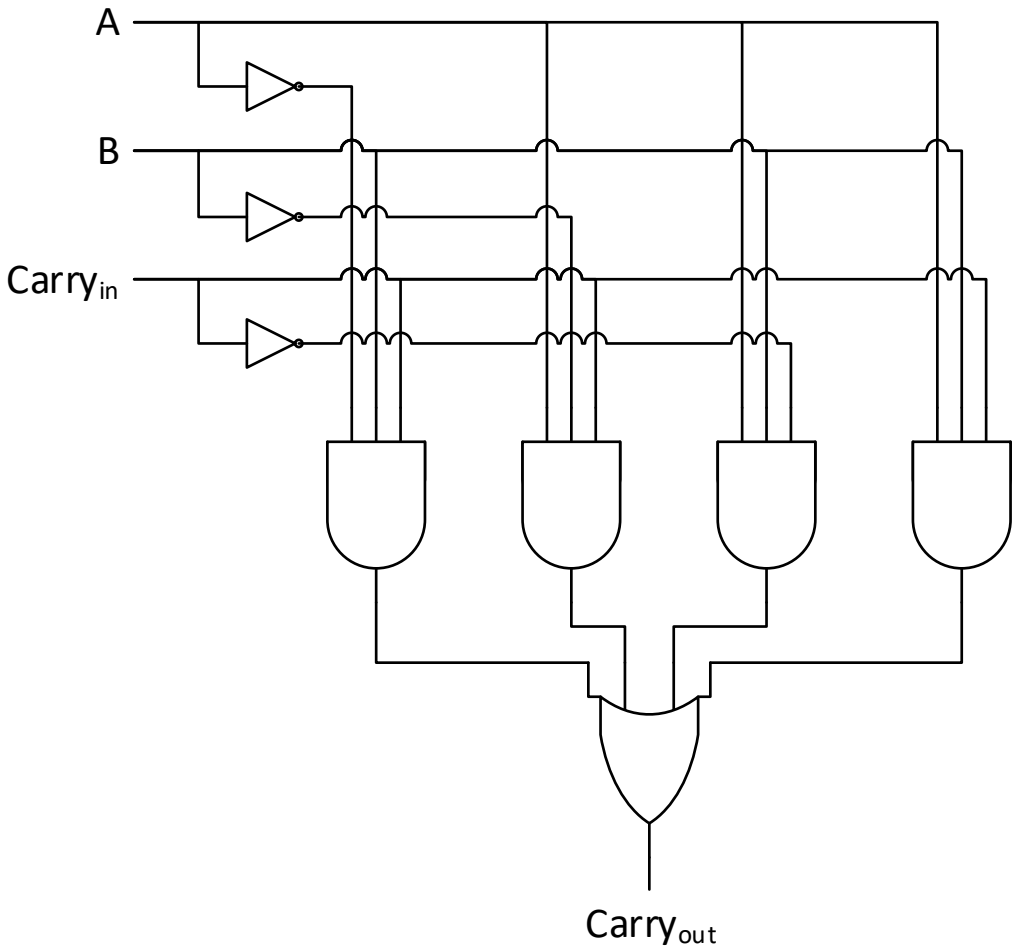
Sum-of-Products: Full Adder Sum

$$Sum = \overline{A}\,\overline{B}Carry_{in} + \overline{A}B\overline{Carry_{in}} + A\overline{B}\overline{Carry_{in}} + ABCarry_{in}$$



Sum-of-Products: Full Adder $Carry_{out}$

$$Carry_{out} = \overline{A}BCarry_{in} + A\overline{B}Carry_{in} + AB\overline{Carry_{in}} + ABCarry_{in}$$



ALU (Arithmetic/Logic Unit)

- The ALU is the component in a CPU that can perform arithmetic and logic operations
- Usually accepts two multi-bit (word size) operands
  - Let's refer to them as **A** and **B**
- Usually produces a single multi-bit (word size) result
  - Let's refer to it as **R**
- The ALU can perform one of many functions
  - Let's refer to the input that selects the function as the **function**
- Usually also produces flags that indicate properties of the result
  - Is the result zero?
  - Is the result negative?
  - Was a $Carry_{out}$ produced in computing the result?
  - Did overflow occur in computing the result?

ALU Operations

- Arithmetic operations might be
  - Addition
  - Subtraction
  - Addition plus 1
  - etc.
- Logic operations might be
  - NOT A
  - NOT B
  - AND
  - OR
  - NAND
  - NOR
  - XOR
  - EQV
  - etc.

ALU Encapsulation