

CPS 104
Computer Organization and Programming
Lecture-9: Boolean Algebra & gates.

Sep. 24, 1999

Dietolf (Dee) Ramm

<http://www.cs.duke.edu/~dr/cps104.html>

Administrivia

- Homework Due today.
- Look for UTAs “office hours”
- New homework-3
- **Read Appendix B!**

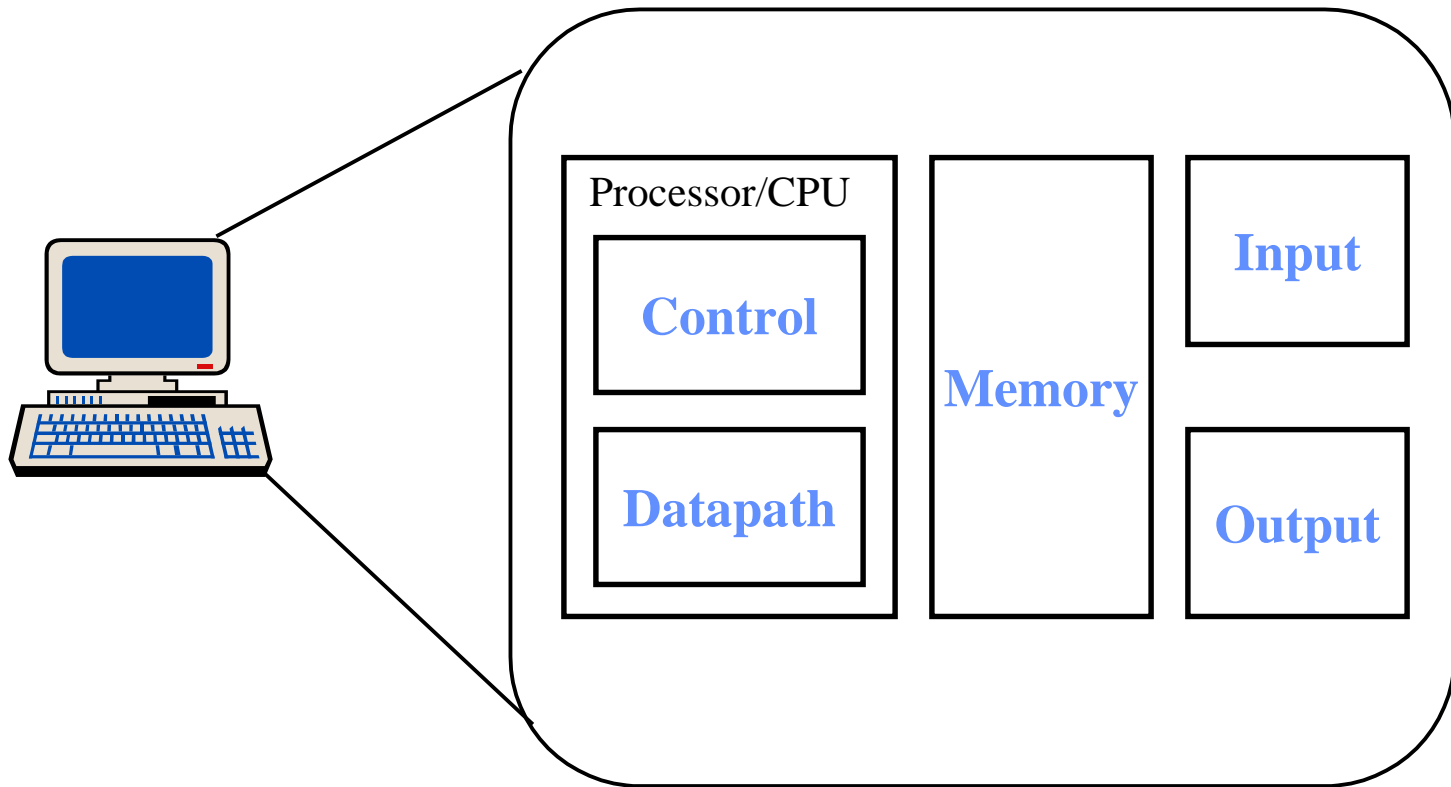
Overview of Today's Lecture:

- **Truth tables, Boolean functions, Gates and Circuits**
- **Examples: 2-1 MUX, Full Adder**

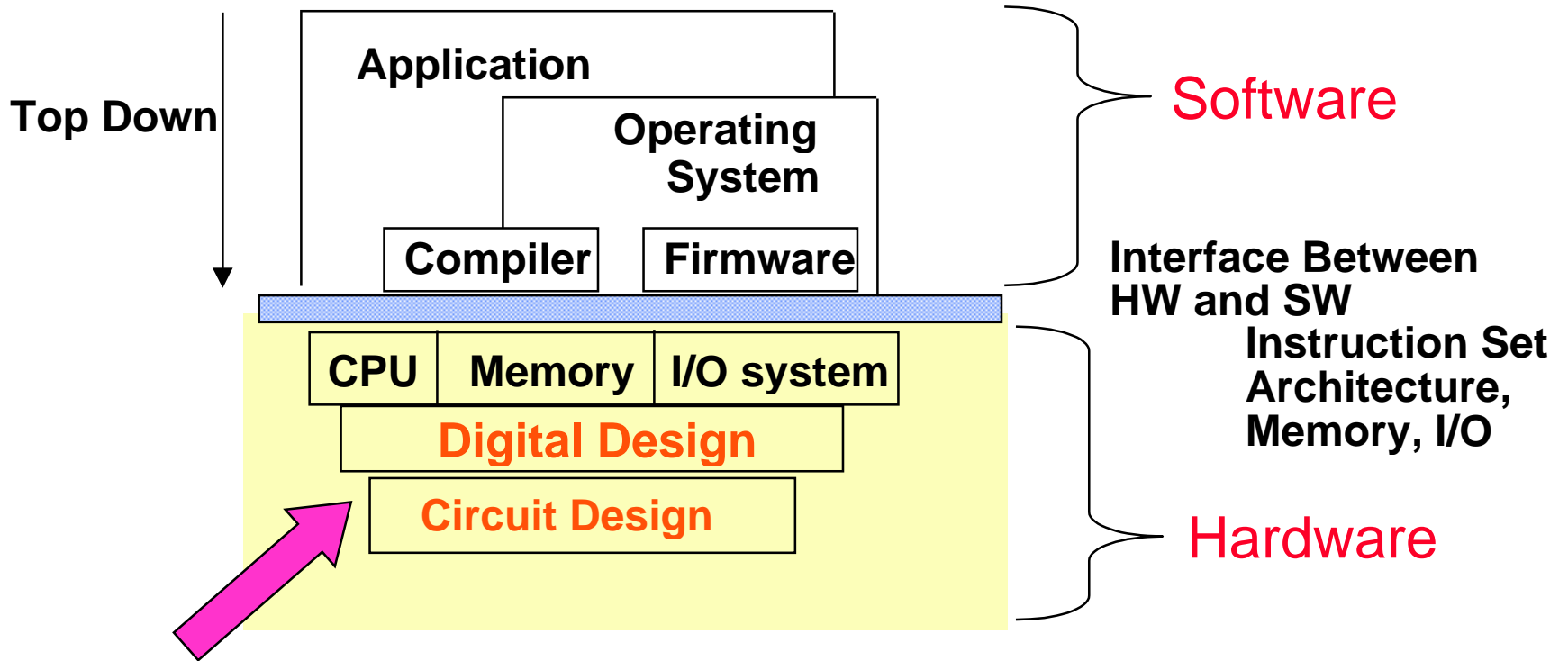
Read Appendix B

The Big Picture

- The Five Classic Components of a Computer



What We've Done, Where We're Going



Bottom UP to CPU

Boolean Functions and Gates

- Boolean functions have arguments that take two values ($\{T,F\}$ or $\{0,1\}$) and they return a single or a set of ($\{T,F\}$ or $\{0,1\}$) value(s).
- Boolean functions can always be represented by a table called: “**Truth Table**”
- Example: $F: \{0,1\}^3 \rightarrow \{0,1\}^2$

a	b	c	f_1	f_2
0	0	0	0	1
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Boolean Functions and Gates (Cont.)

- **Examples:** Boolean functions: NOT, AND, OR, XOR, . . .

a	NOT(a)
0	1
1	0

a	b	AND(a,b)
0	0	0
0	1	0
1	0	0
1	1	1

a	b	OR(a,b)
0	0	0
0	1	1
1	0	1
1	1	1

a	b	XOR(a,b)
0	0	0
0	1	1
1	0	1
1	1	0

a	b	XNOR(a,b)
0	0	1
0	1	0
1	0	0
1	1	1

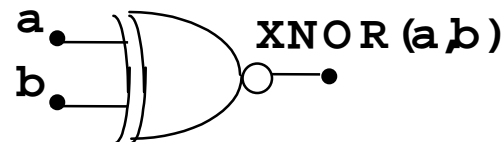
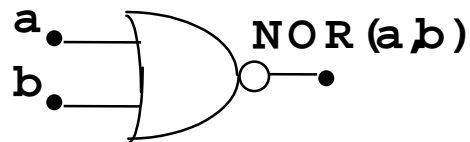
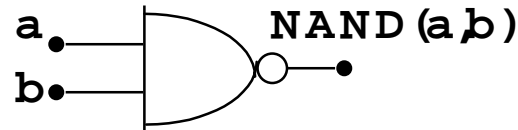
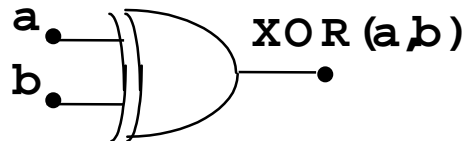
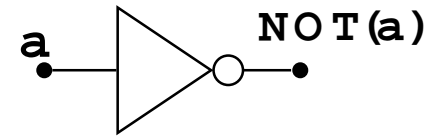
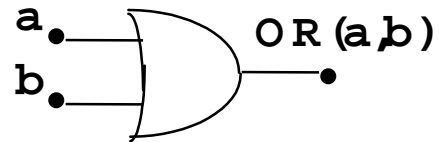
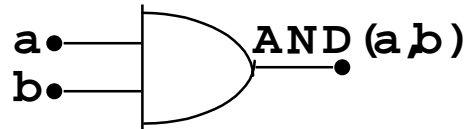
a	b	NOR(a,b)
0	0	1
0	1	0
1	0	0
1	1	0

Boolean Functions and Gates

(Cont.)

- Gates are electronics devices that implements simple Boolean functions:

- Examples:



Boolean Expressions

- **Boolean algebra notation: Use:**
* for **AND**, + for **OR**, ~ for **NOT**.
- **Using the above notation one could write boolean expressions for functions:**

Example:

$$F(A, B, C) = (A * B) + (\sim A * C)$$

Boolean Functions and Expressions

- One can evaluate the Boolean expression with all possible argument values to construct a truth table.

Example:

$$F(A, B, C) = (A * B) + (\sim A * C)$$

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Boolean functions simplification

- Boolean expressions could be simplified by using the following rules:

- * $A * A = A$

- * $A * 0 = 0$

- * $A * 1 = A$

- * $A * \sim A = 0$

- * $A + A = A$

- * $A + 0 = A$

- * $A + 1 = 1$

- * $A + \sim A = 1$

- * $A * B = B * A$

- * $A * (B + C) = (B + C) * A = A * B + A * C$

Boolean Functions simplification

○ Example:

a	b	c	f ₁	f ₂
0	0	0	0	1
0	0	1	1	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	0	1
1	1	1	1	1

$$f_1 = \sim a^* \sim b^* c + \sim a^* b^* c + a^* \sim b^* c + a^* b^* c$$

$$f_2 = \sim a^* \sim b^* \sim c + \sim a^* \sim b^* c + a^* b^* \sim c + a^* b^* c$$

Boolean Functions Simplification

$$\begin{aligned}f_1 &= \sim a^* \sim b^* c + \sim a^* b^* c + a^* \sim b^* c + a^* b^* c \\&= \sim a^* (\sim b^* c + b^* c) + a^* (\sim b^* c + b^* c) \\&= \sim a^* c^* (\sim b + b) + a^* c^* (\sim b + b) \\&= \sim a^* c + a^* c \\&= c^* (\sim a + a) \\&= c\end{aligned}$$

$$\begin{aligned}f_2 &= \sim a^* \sim b^* \sim c + \sim a^* \sim b^* c + a^* b^* \sim c + a^* b^* c \\&= \sim a^* (\sim b^* \sim c + \sim b^* c) + a^* (b^* \sim c + b^* c) \\&= \sim a^* \sim b (c + \sim c) + a^* b^* (\sim c + c) \\&= \sim a^* \sim b + a^* b\end{aligned}$$

Boolean Functions and Expressions

The Fundamental Theorem of Boolean Algebra:
*Every Boolean function can be written in disjunctive normal form as an **OR of ANDs** (Sum-of products) of its arguments or their complements.*

“Proof:” Write the truth table, construct sum-of-product from the table.

Example:

a	b	XNOR(a,b)
0	0	1
0	1	0
1	0	0
1	1	1

$$\text{XNOR} = (\sim a * \sim b) + (a * b)$$

Boolean Functions and Boolean Expressions

○ Example-2:

a	b	c	f ₁	f ₂
0	0	0	0	1
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1
1	1	1	1	1

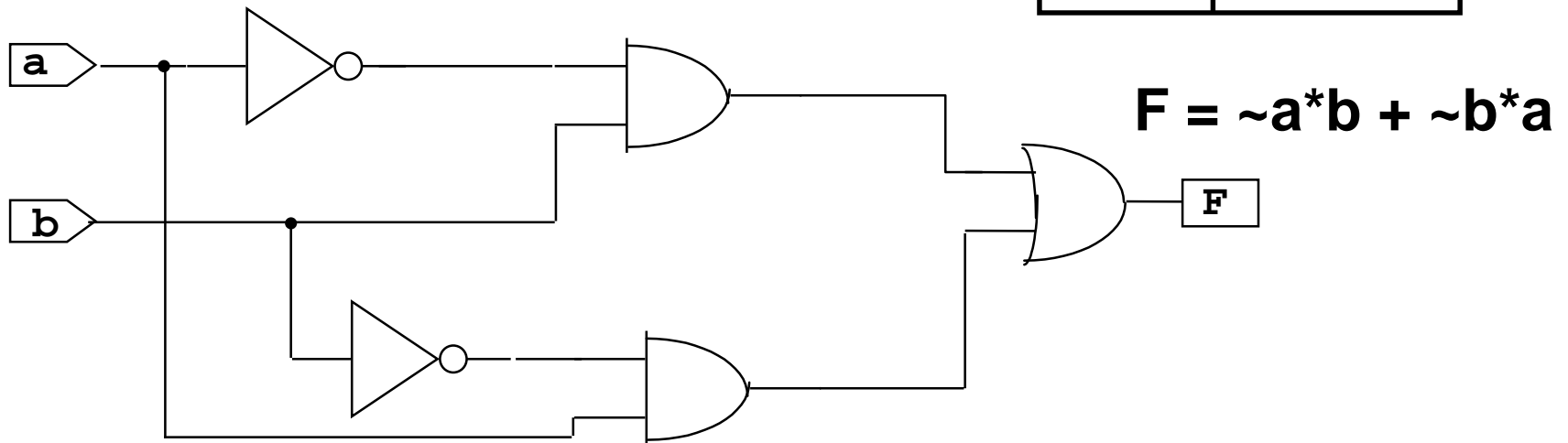
$$f_1 = \sim a^* \sim b^* c + \sim a^* b^* \sim c + a^* \sim b^* \sim c + a^* b^* c$$

$$f_2 = \sim a^* \sim b^* \sim c + \sim a^* \sim b^* c + a^* b^* \sim c + a^* b^* c$$

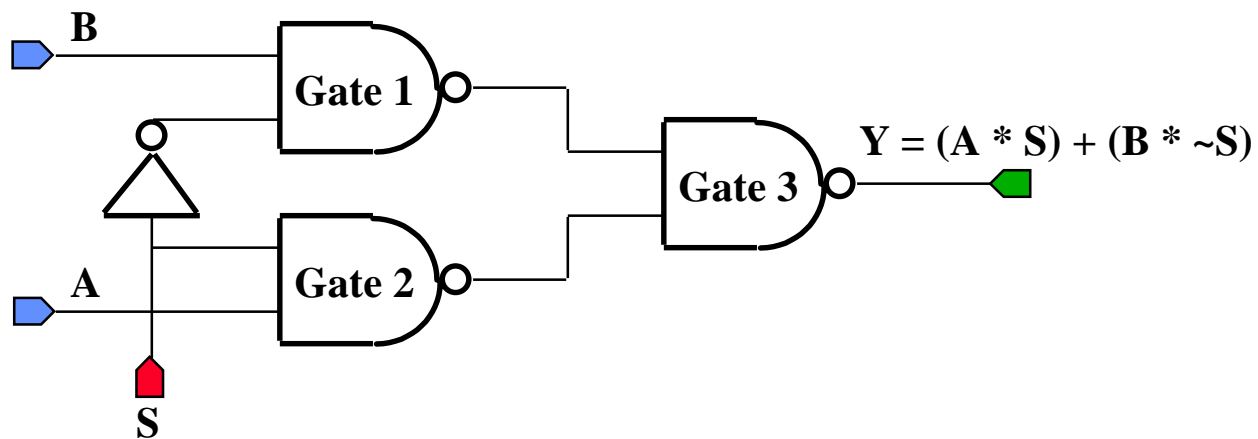
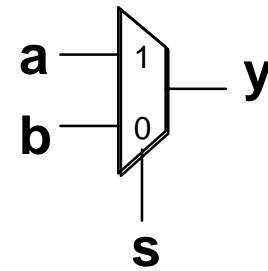
Boolean Functions, Gates and Circuits

- **Circuits** are made from a network of gates. (function compositions).
- **Example:**

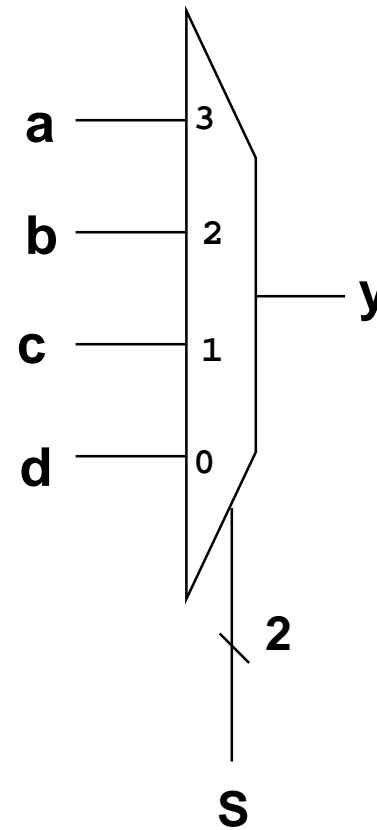
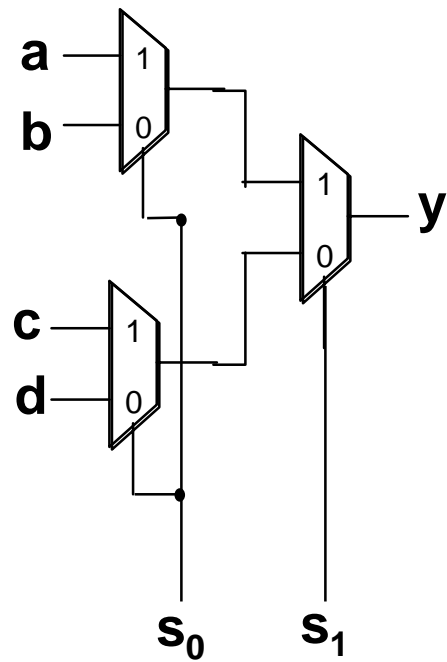
a	b	XOR(a,b)
0	0	0
0	1	1
1	0	1
1	1	0



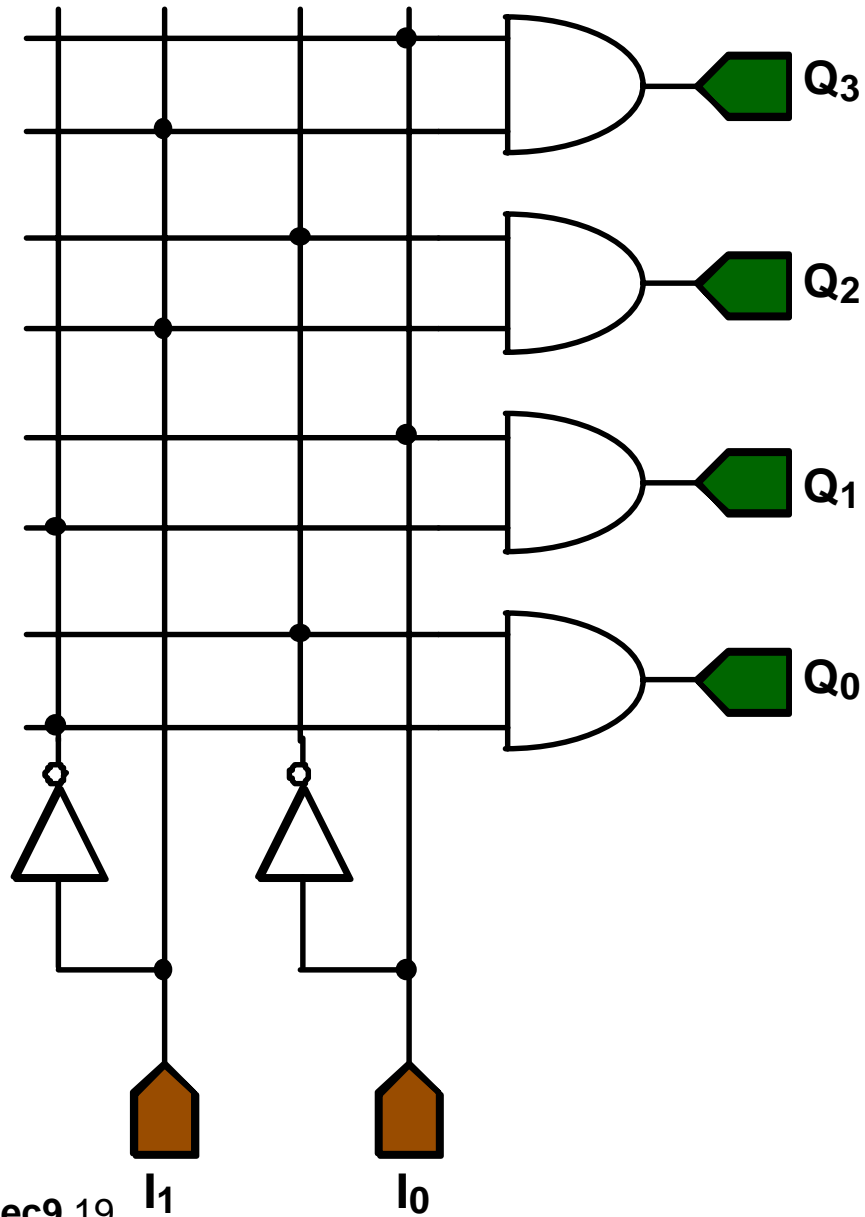
Circuit Example: 2x1 MUX



Example 4x1 MUX

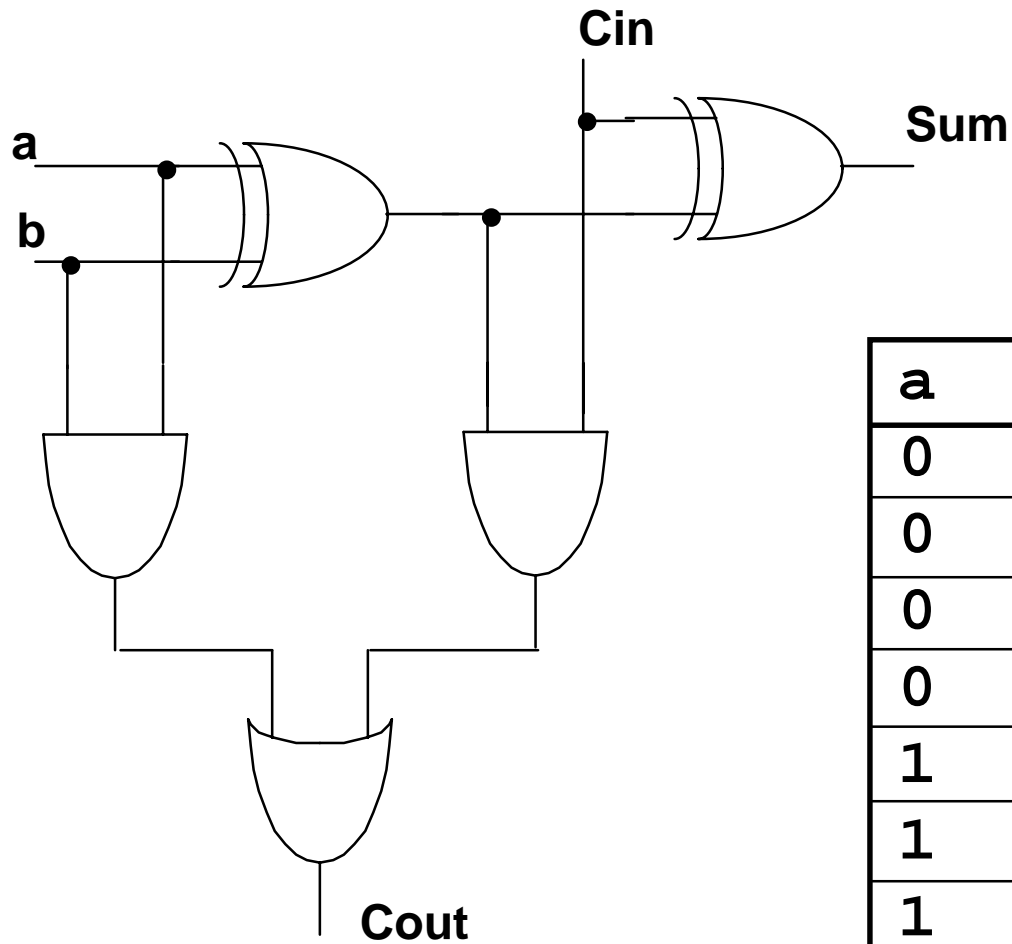


Circuit Example: Selector



I_1	I_0	Q_0	Q_1	Q_2	Q_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

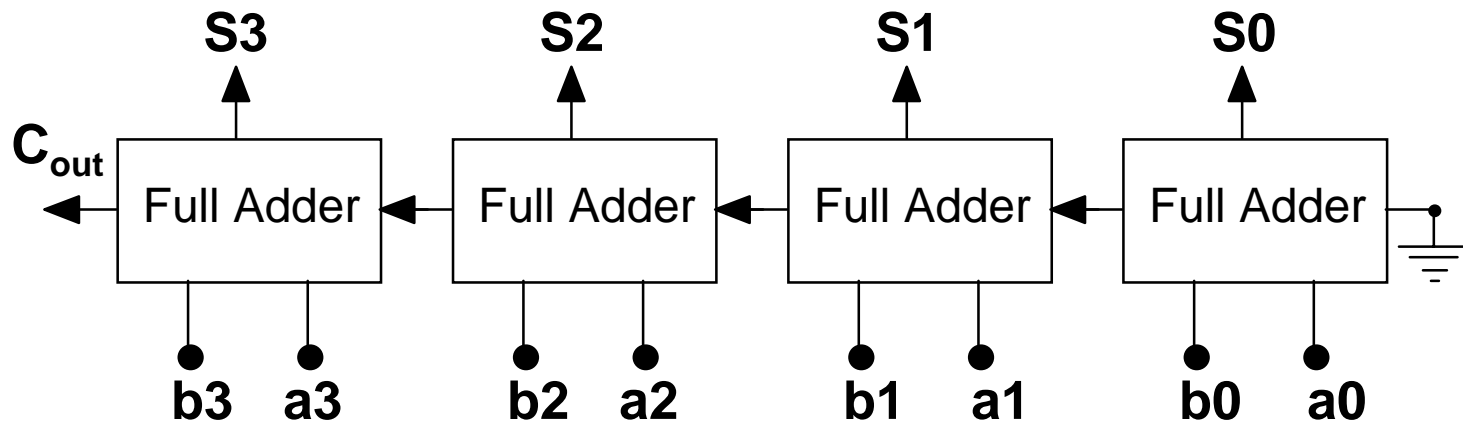
Full Adder



$$\begin{array}{r} 01101100 \\ +00101100 \\ \hline 10011001 \end{array}$$

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

Example: 4-bit adder



Overflow

Example1:

$$\begin{array}{r} 01000000 \\ \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \\ 0110101_2 \quad (= 53_{10}) \\ + 0101010_2 \quad (= 42_{10}) \\ \hline 1011111_2 \quad (= -33_{10}) \end{array}$$

Example2:

$$\begin{array}{r} 10000000 \\ \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \\ 1010101_2 \quad (= -43_{10}) \\ + 1001010_2 \quad (= -54_{10}) \\ \hline 0011111_2 \quad (= 31_{10}) \end{array}$$

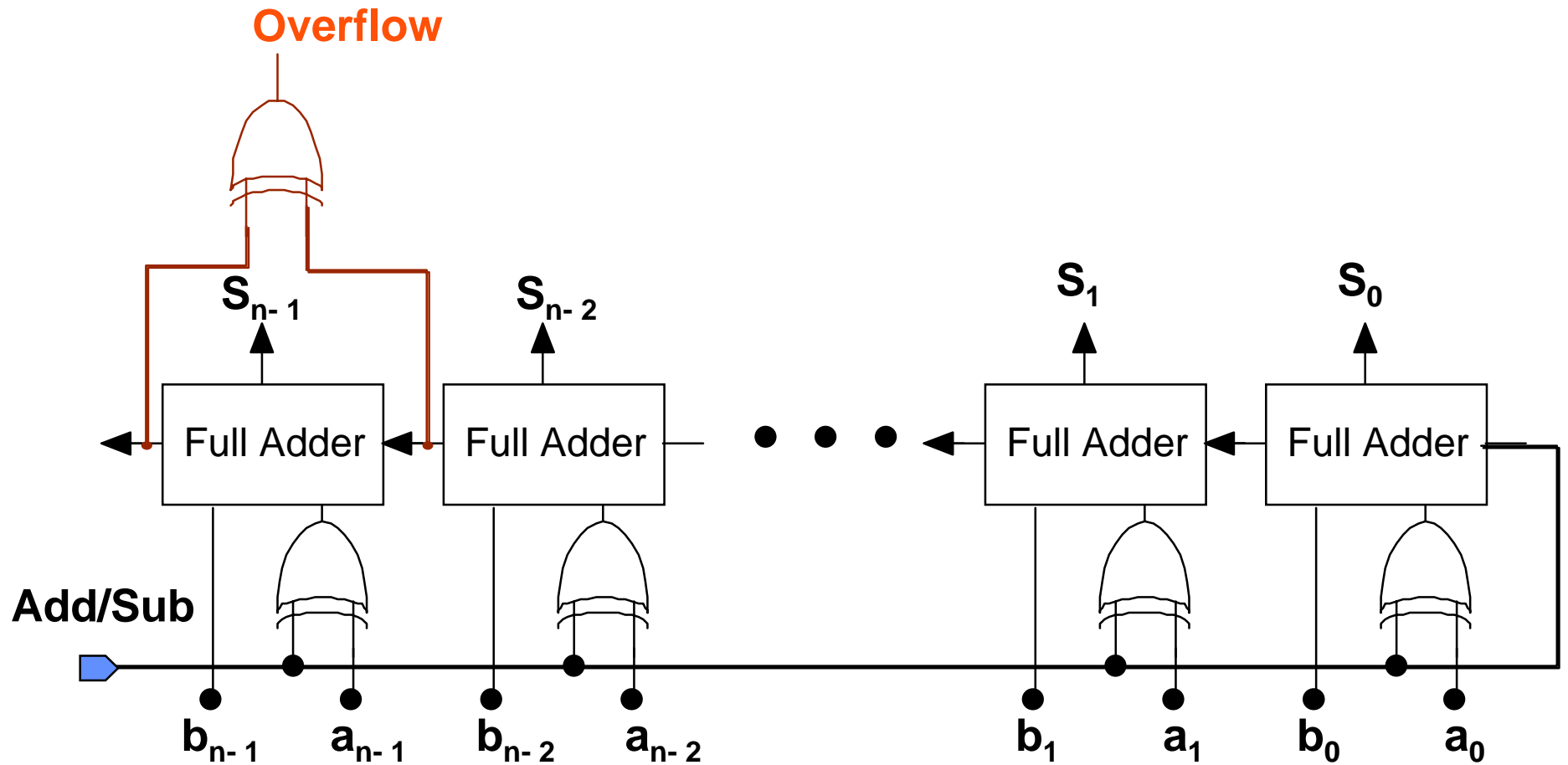
Example3:

$$\begin{array}{r} 11000000 \\ \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \\ 0110101_2 \quad (= 53_{10}) \\ + 1101010_2 \quad (= -22_{10}) \\ \hline 0011111_2 \quad (= 31_{10}) \end{array}$$

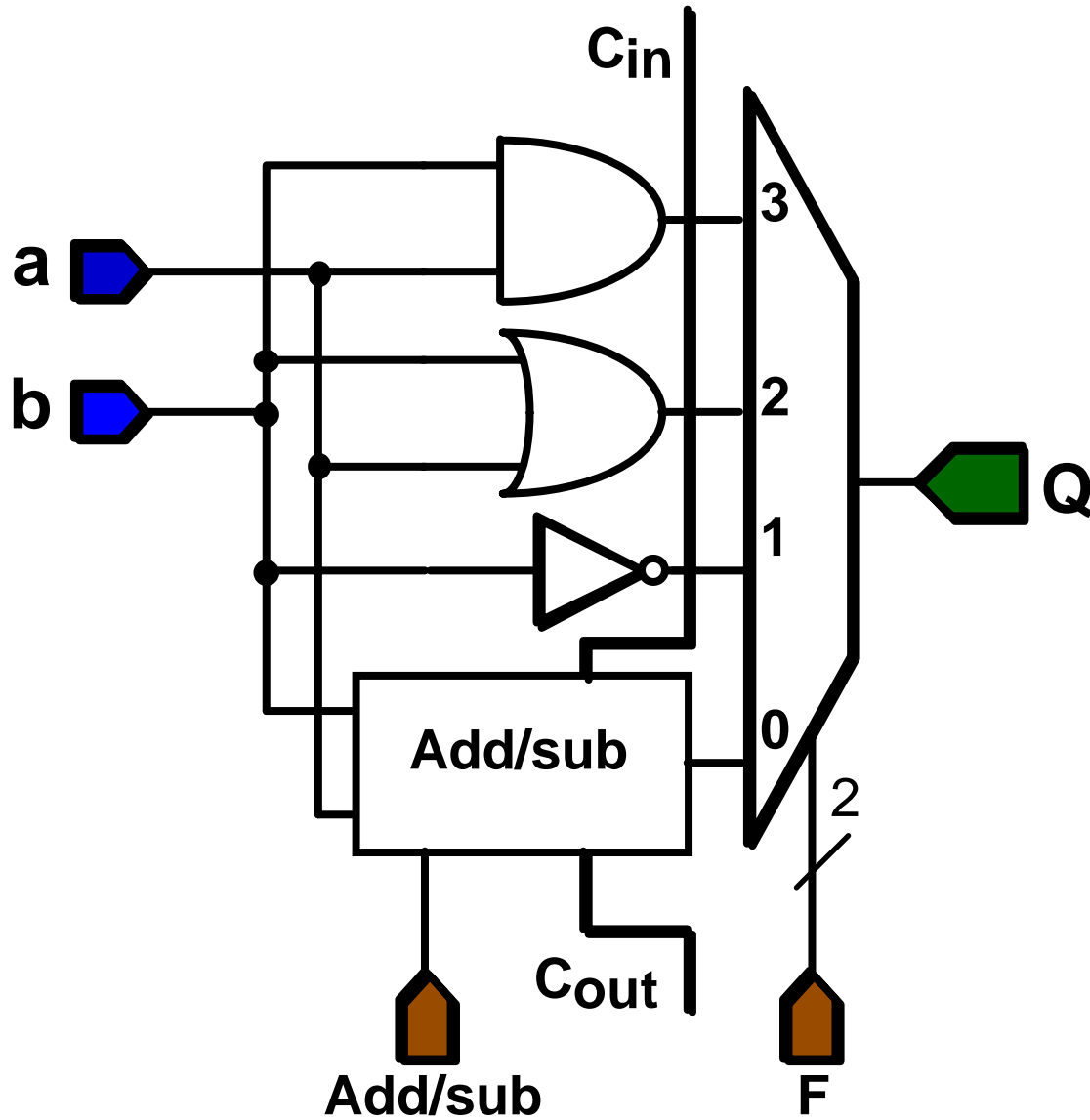
Example4:

$$\begin{array}{r} 00000000 \\ \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \\ 0010101_2 \quad (= 21_{10}) \\ + 0101010_2 \quad (= 42_{10}) \\ \hline 0111111_2 \quad (= 63_{10}) \end{array}$$

Add/Subtract With Overflow detection



ALU Slice



A	F	Q
0	0	$a + b$
1	0	$a - b$
-	1	NOT b
-	2	a OR b
-	3	a AND b

The ALU

