# CMSC 313
# COMPUTER ORGANIZATION
# &
# ASSEMBLY LANGUAGE
# PROGRAMMING

Lecture 21, Fall 2014

# TOPICS TODAY

- Circuits for Addition
- Standard Logic Components
- Logisim Demo

# CIRCUITS FOR ADDITION

# 3.5 Combinational Circuits

- Combinational logic circuits give us many useful devices.
- One of the simplest is the *half adder*, which finds the sum of two bits.
- We can gain some insight as to the construction of a half adder by looking at its truth table, shown at the right.

| Inputs | | Outputs | |
|---|---|---|---|
| X | Y | Sum | Carry |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |



© 2012 Jones & Bartlett Learning, LLC
www.jblearning.com

# Half Adder

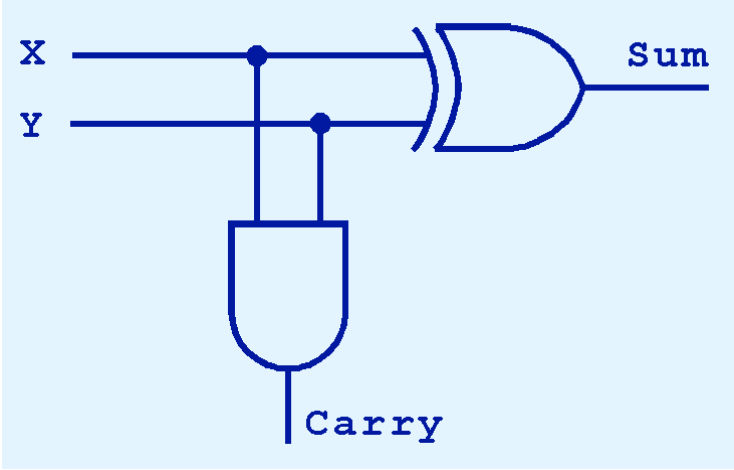
- Inputs: $A$ and $B$
- Outputs: $S$ = lower bit of $A + B$, $c_{\text{out}}$ = carry bit

| $A$ | $B$ | $S$ | $c_{\text{out}}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

- Using Sum-of-Products: $S = \overline{A}B + A\overline{B}$, $c_{\text{out}} = AB$.
- Alternatively, we could use XOR: $S = A \oplus B$.

# 3.5 Combinational Circuits

- As we see, the sum can be found using the XOR operation and the carry using the AND operation.




| Inputs | | Outputs | |
|---|---|---|---|
| X | Y | Sum | Carry |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |



© 2012 Jones & Bartlett Learning, LLC
www.jblearning.com

# 3.5 Combinational Circuits

- We can change our half adder into to a full adder by including gates for processing the carry bit.
- The truth table for a full adder is shown at the right.

| Inputs | | | Outputs | |
|---|---|---|---|---|
| X | Y | 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 |



© 2012 Jones & Bartlett Learning, LLC
www.jblearning.com

# Full Adder

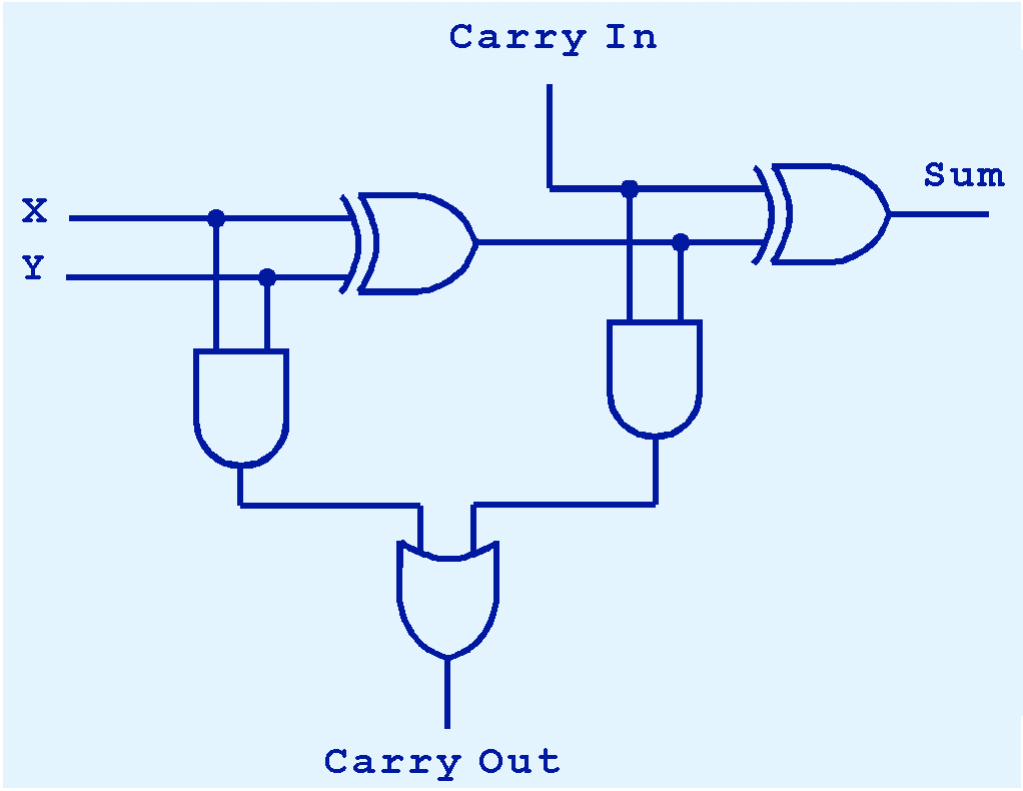
- Inputs: $A$, $B$ and $c_{\text{in}}$
- Outputs: $S =$ lower bit of $A + B$, $c_{\text{out}} =$ carry bit

| $A$ | $B$ | $c_{\text{in}}$ | $S$ | $c_{\text{out}}$ |
|---|---|---|---|---|
| 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 |

- $S = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC = A \oplus B \oplus C.$
- $c_{\text{out}} = \text{MAJ3} = AB + BC + AC.$

# 3.5 Combinational Circuits

- Here's our completed full adder.



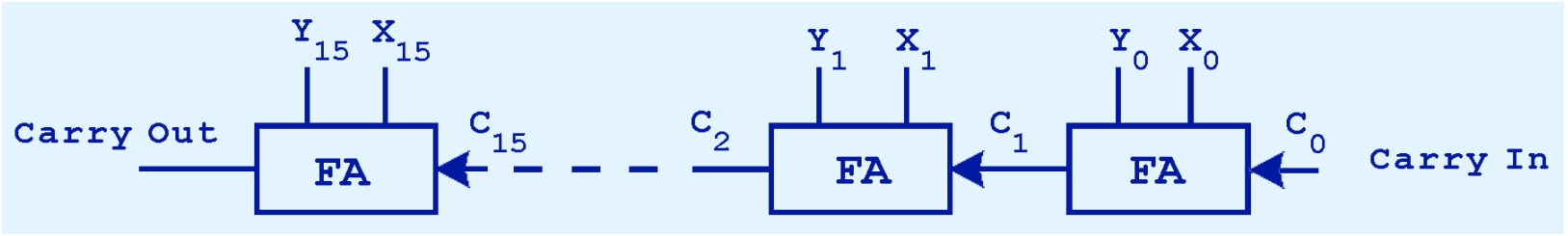

| Inputs | | | Outputs | |
|---|---|---|---|---|
| X | Y | Carry In | Sum | Carry Out |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |



© 2012 Jones & Bartlett Learning, LLC
www.jblearning.com

# 3.5 Combinational Circuits

- Just as we combined half adders to make a full adder, full adders can connected in series.
- The carry bit "ripples" from one adder to the next; hence, this configuration is called a *ripple-carry adder*.



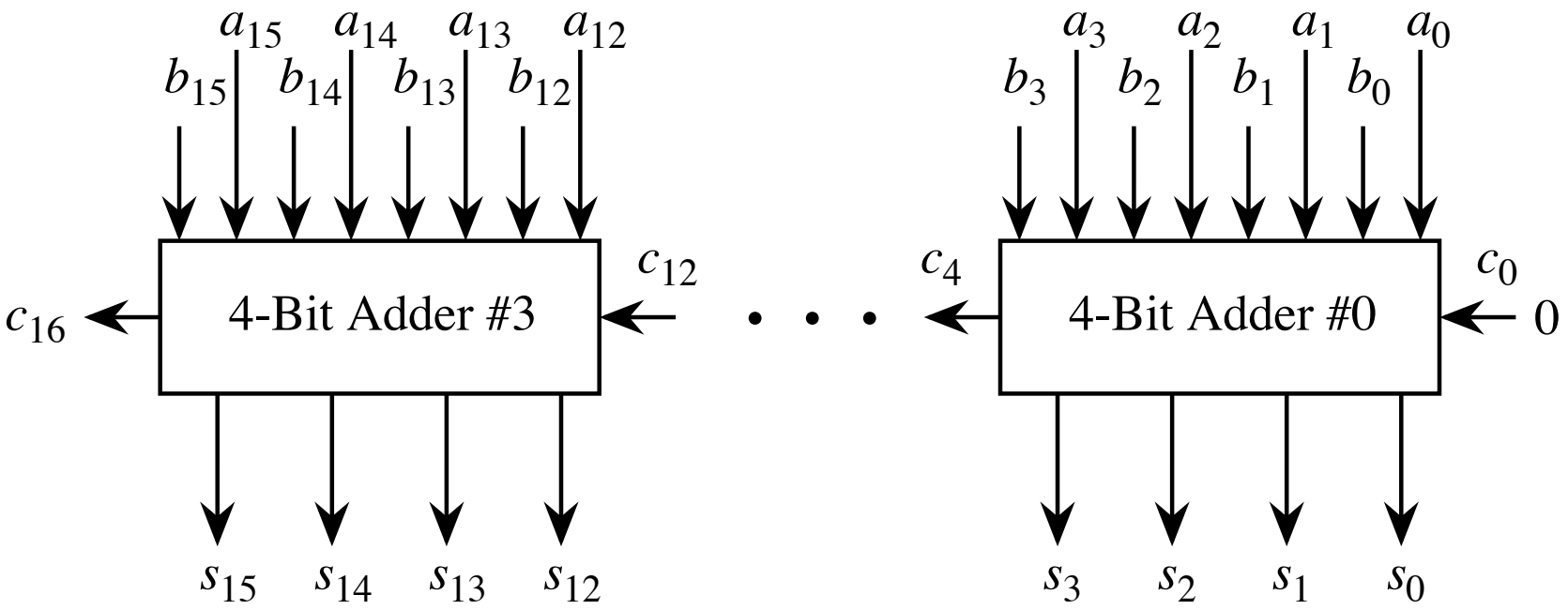

**Today's systems employ more efficient adders.**



© 2012 Jones & Bartlett Learning, LLC
www.jblearning.com

# Constructing Larger Adders

- **A 16-bit adder can be made up of a cascade of four 4-bit ripple-carry adders.**



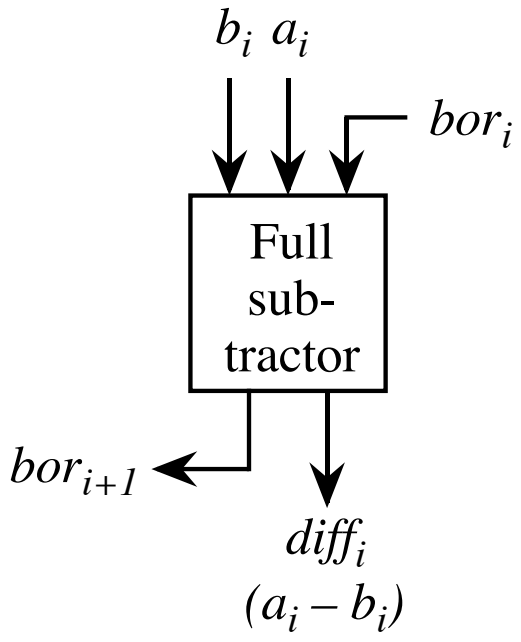


© 1999 M. Murdocca and V. Heuring

# Full Subtractor

- **Truth table and schematic symbol for a ripple-borrow subtractor:**

| $a_i$ | $b_i$ | $bor_i$ | $diff_i$ | $bor_{i+1}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

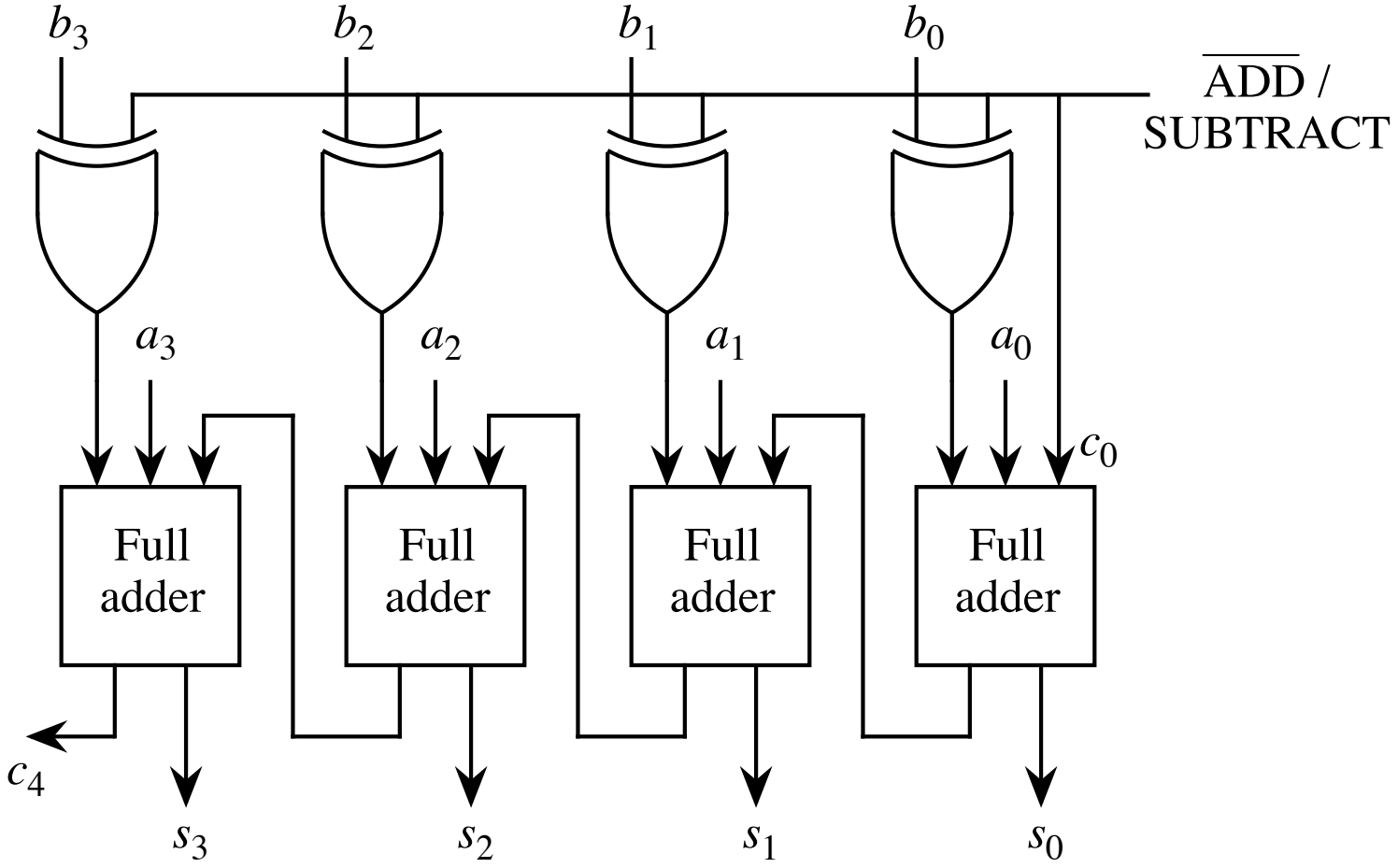




© 1999 M. Murdocca and V. Heuring

# Combined Adder/Subtractor

- **A single ripple-carry adder can perform both addition and subtraction, by forming the two's complement negative for *B* when subtracting. (Note that +1 is added at $c_0$ for two's complement.)**





© 1999 M. Murdocca and V. Heuring

# Carry-Lookahead Addition

$$s_i = \bar{a}_i\bar{b}_i c_i + \bar{a}_i b_i \bar{c}_i + a_i\bar{b}_i\bar{c}_i + a_i b_i c_i$$

$$c_{i+1} = b_i c_i + a_i c_i + a_i b_i$$

$$c_{i+1} = a_i b_i + (a_i + b_i)c_i$$

$$c_{i+1} = G_i + P_i c_i$$

- **Carries are represented in terms of $G_i$ (generate) and $P_i$ (propagate) expressions.**

$$G_i = a_i b_i \quad \text{and} \quad P_i = a_i + b_i$$

$$c_0 = 0$$

$$c_1 = G_0$$

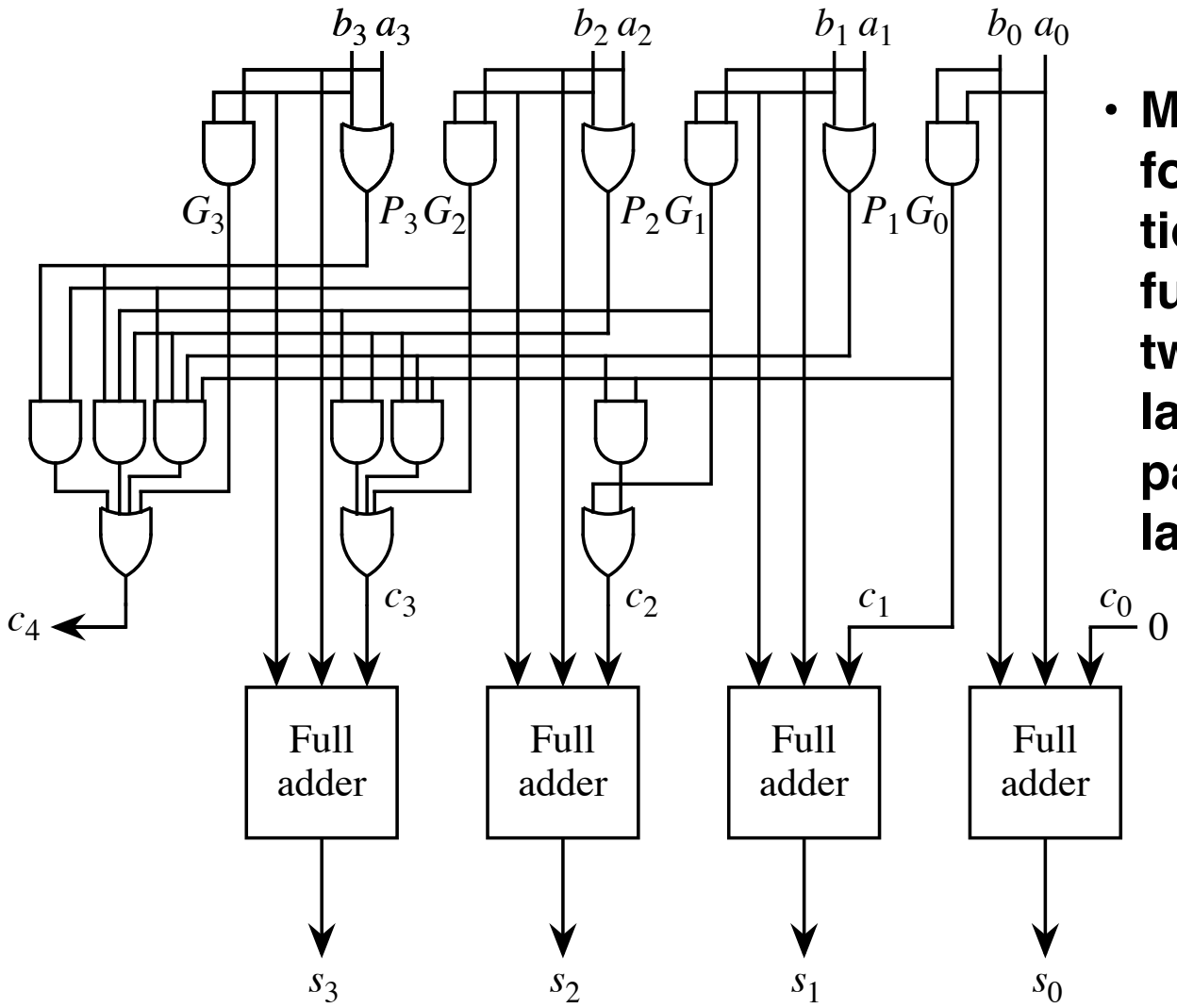
$$c_2 = G_1 + P_1 G_0$$

$$c_3 = G_2 + P_2 G_1 + P_2 P_1 G_0$$

$$c_4 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0$$


© 1999 M. Murdocca and V. Heuring

# Carry Lookahead Adder




- **Maximum gate delay for the carry generation is only 3. The full adders introduce two more gate delays. Worst case path is 5 gate delays.**



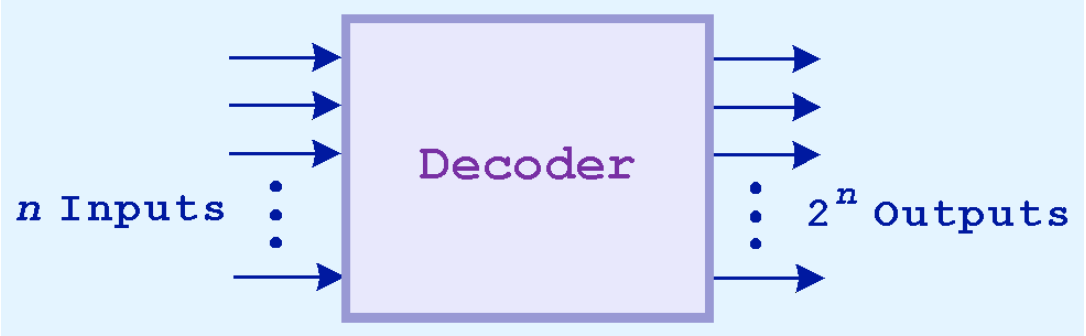
© 1999 M. Murdocca and V. Heuring

# STANDARD LOGIC COMPONENTS

# 3.5 Combinational Circuits

- Decoders are another important type of combinational circuit.
- Among other things, they are useful in selecting a memory location according a binary value placed on the address lines of a memory bus.
- Address decoders with *n* inputs can select any of $2^n$ locations.

**This is a block diagram for a decoder.**

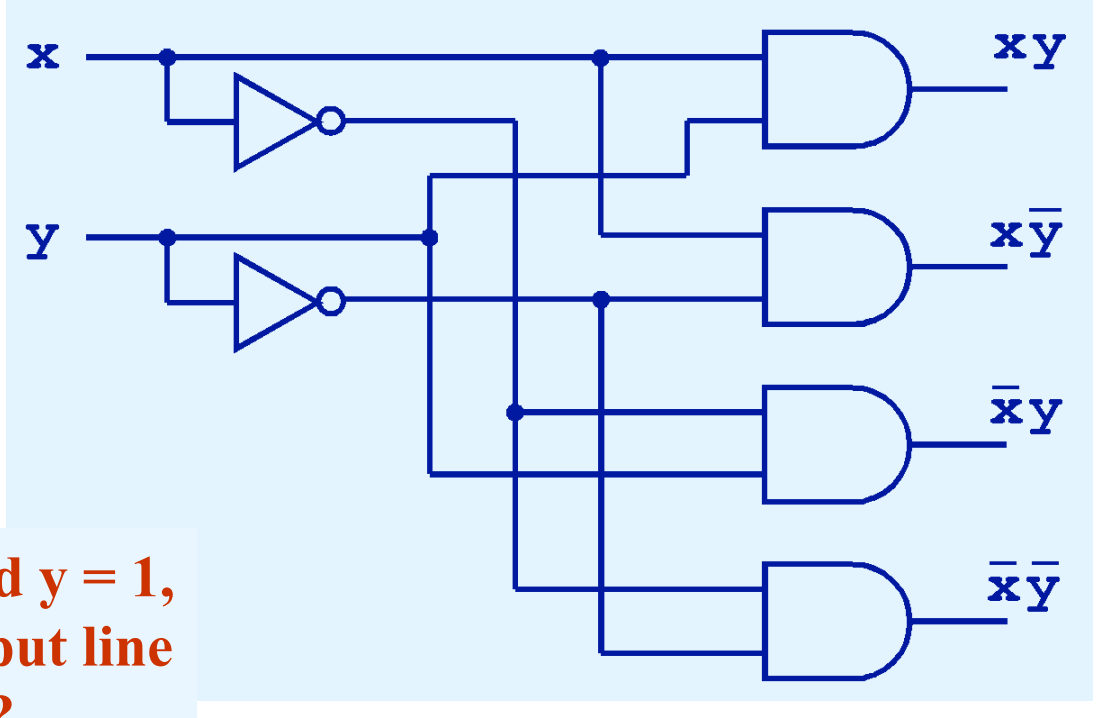





© 2012 Jones & Bartlett Learning, LLC
www.jblearning.com

# 3.5 Combinational Circuits

- This is what a 2-to-4 decoder looks like on the inside.



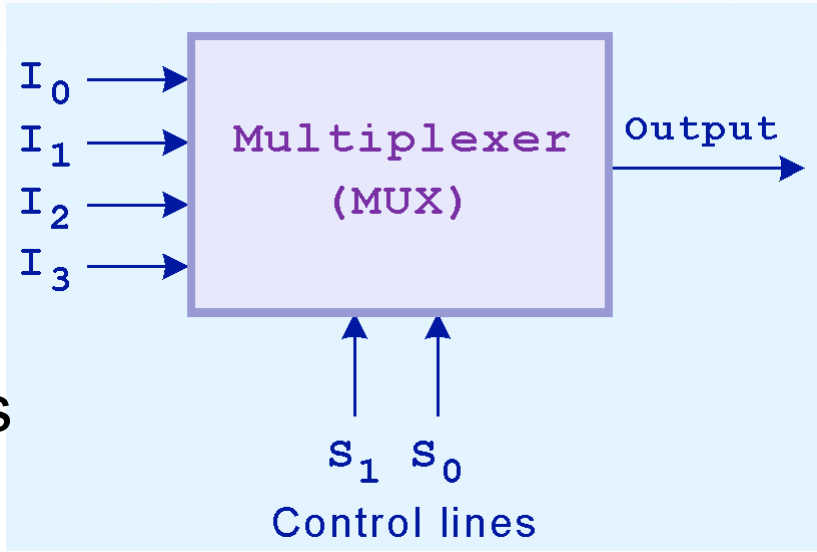

**If x = 0 and y = 1, which output line is enabled?**



© 2012 Jones & Bartlett Learning, LLC
www.jblearning.com

# 3.5 Combinational Circuits

- A multiplexer does just the opposite of a decoder.
- It selects a single output from several inputs.
- The particular input chosen for output is determined by the value of the multiplexer's control lines.
- To be able to select among *n* inputs, $\log_2 n$ control lines are needed.



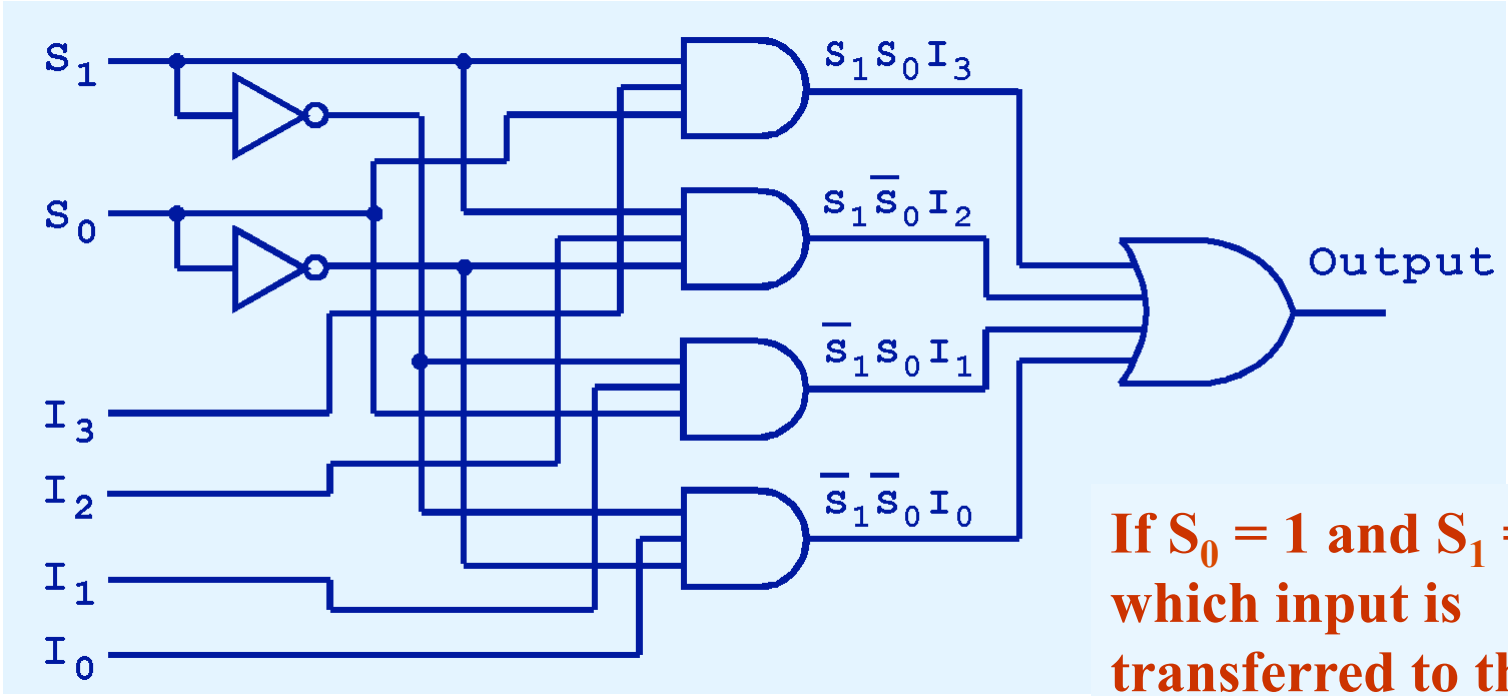

**This is a block diagram for a multiplexer.**



© 2012 Jones & Bartlett Learning, LLC
www.jblearning.com

# 3.5 Combinational Circuits

- This is what a 4-to-1 multiplexer looks like on the inside.

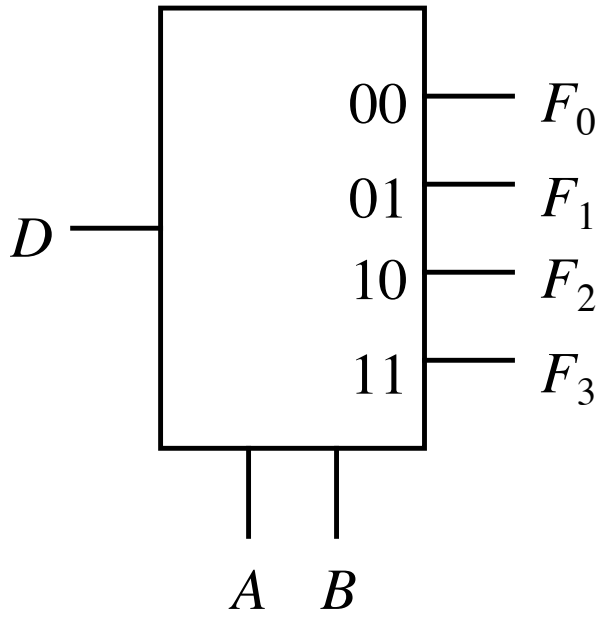



**If $S_0 = 1$ and $S_1 = 0$, which input is transferred to the output?**



© 2012 Jones & Bartlett Learning, LLC
www.jblearning.com

# Demultiplexer



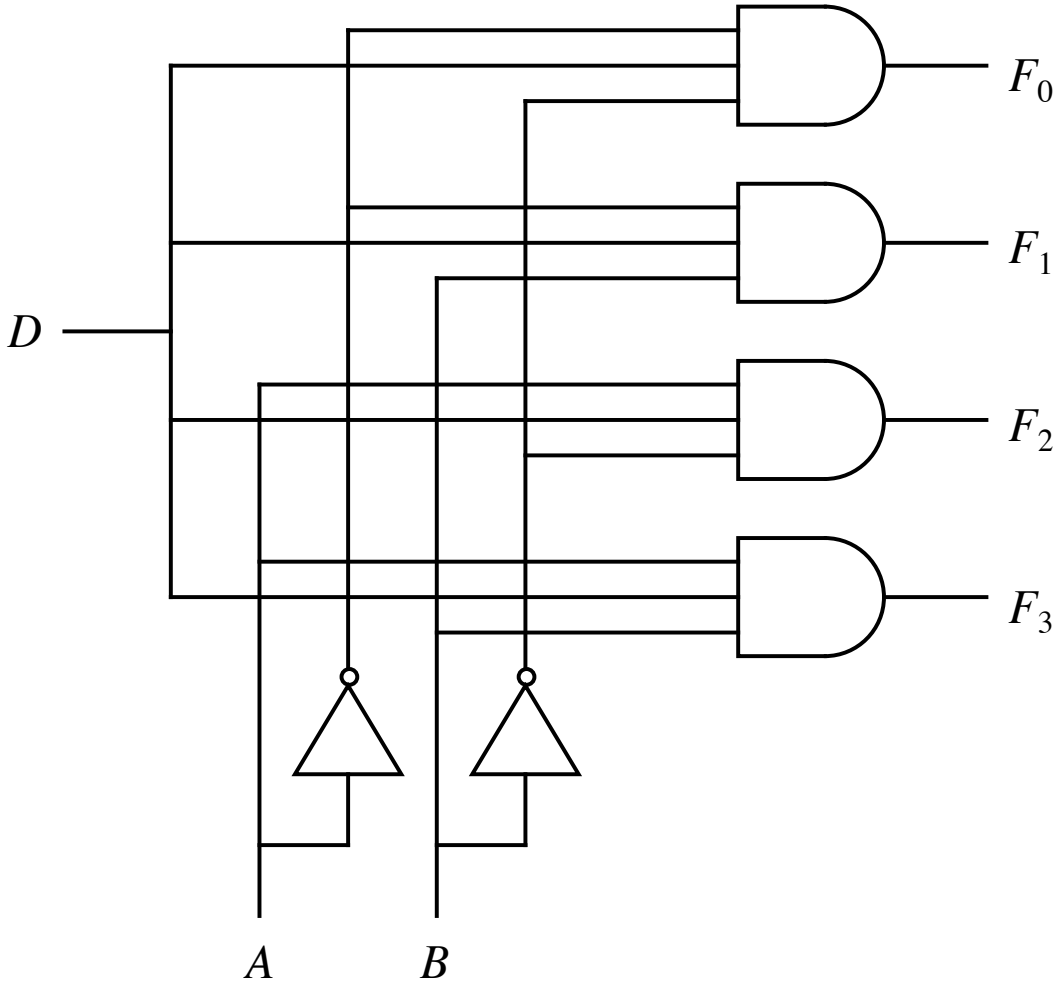

$$F_0 = D\,\overline{A}\,\overline{B} \qquad F_2 = D\,A\,\overline{B}$$

$$F_1 = D\,\overline{A}\,B \qquad F_3 = D\,A\,B$$

| $D$ | $A$ | $B$ | $F_0$ | $F_1$ | $F_2$ | $F_3$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 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 |


© 1999 M. Murdocca and V. Heuring

# Gate-Level Implementation of DEMUX

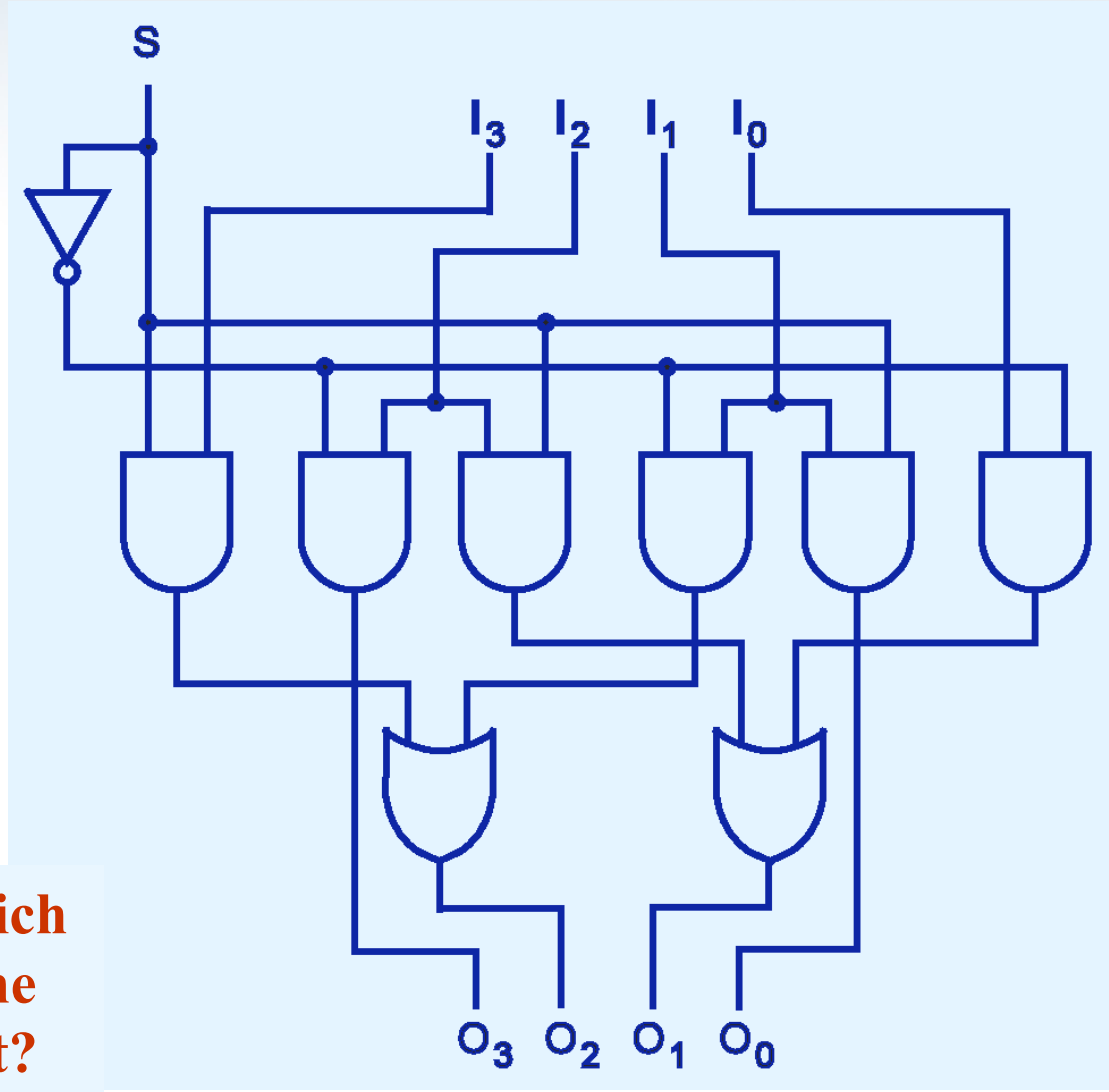




© 1999 M. Murdocca and V. Heuring

# 3.5 Combinational Circuits

- This shifter moves the bits of a nibble one position to the left or right.




**If S = 0, in which direction do the input bits shift?**



© 2012 Jones & Bartlett Learning, LLC
www.jblearning.com

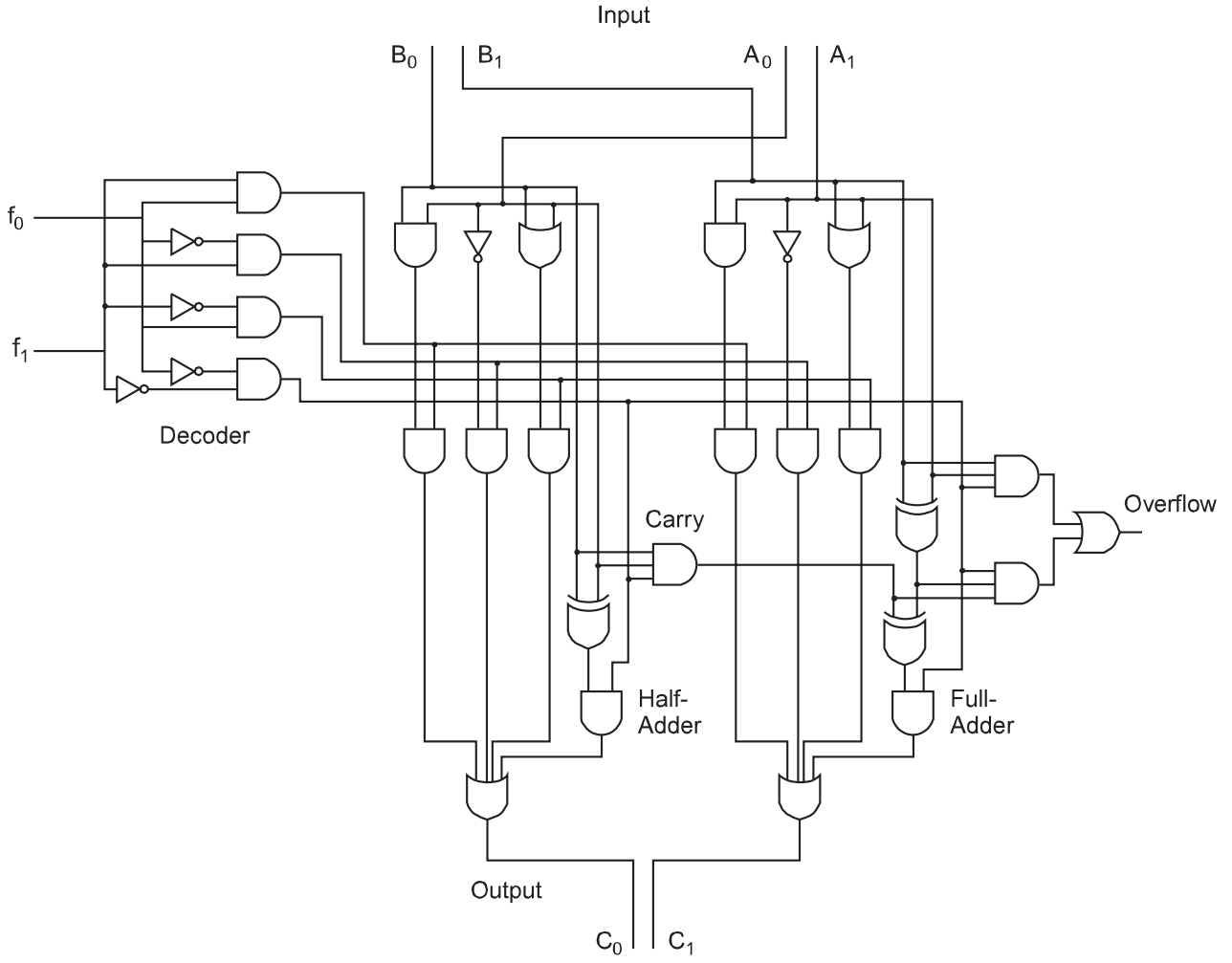



**FIGURE 3.17** A Simple Two-Bit ALU

# NEXT TIME

- **2-bit ALU**
- **Flip-flops**