



CSC 220: Computer Organization

Unit 10 **Arithmetic-Logic Units**

Prepared by:

Md Saiful Islam, PhD

Updated by:

Isra Al-Turaiki, PhD

Department of Computer Science
College of Computer and Information Sciences

Overview

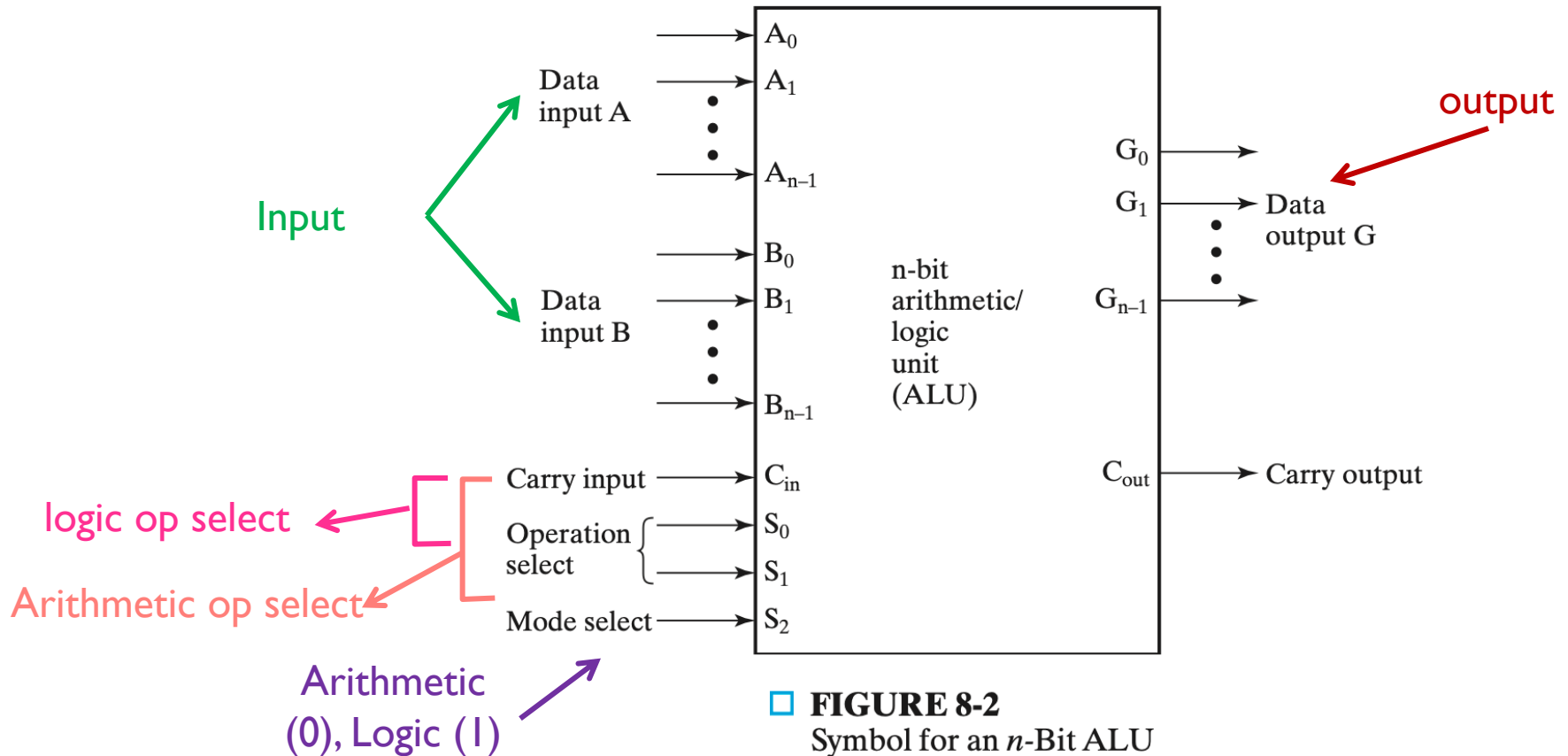
- Arithmetic Unit Design
 - MUX-based implementation
 - Primitive gates base implementation
- Logic Unit Design
- Arithmetic-logic Unit Design
- Combinational Shifter
- Function Unit Design

Chapter-8

M. Morris Mano, Charles R. Kime and Tom Martin, **Logic and Computer Design Fundamentals**, Global (5th) Edition, Pearson Education Limited, 2016. ISBN: 9781292096124

The Arithmetic/Logic Unit

The *ALU* is a **combinational** circuit that performs a set of basic **arithmetic** and **logic** microoperations.

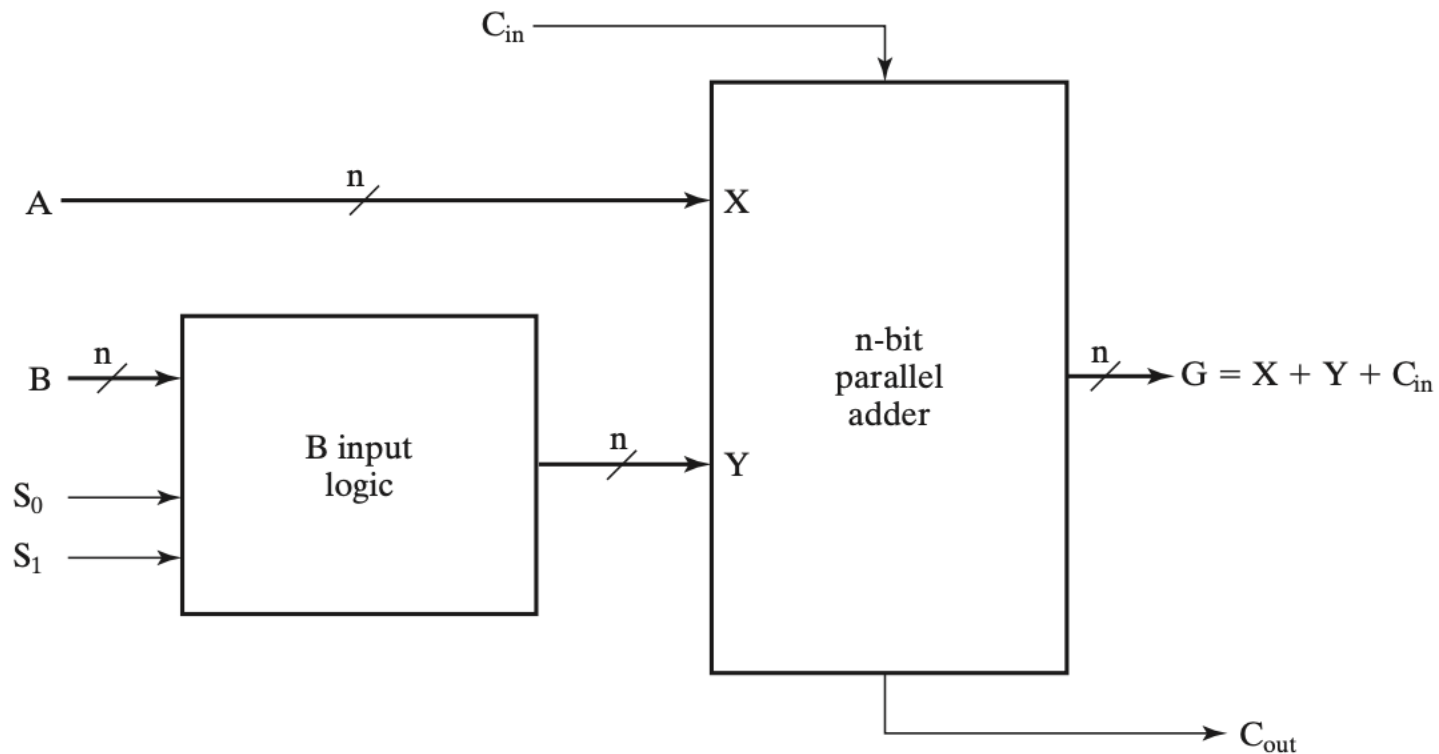


The Arithmetic/Logic Unit

- Entire register transfer operation (source registers → ALU → destination register) is performed during **one clock cycle**.
- ALU is decomposed into:
 - **An arithmetic circuit:**
 - An n-bit parallel adder.
 - A block of logic that selects four choices for the B input to the adder.
 - **A logic circuit:**
 - Gates to implement the basic logic operations.
 - A selector to select which logic operation to perform.
 - **A selector** to pick between the two circuits.

Arithmetic Unit

- The basic component of an arithmetic circuit is a **parallel adder**.
- Different arithmetic operations are obtained by **controlling the data inputs to the parallel adder**.



□ **FIGURE 8-3**
Block Diagram of an Arithmetic Circuit

Arithmetic Unit

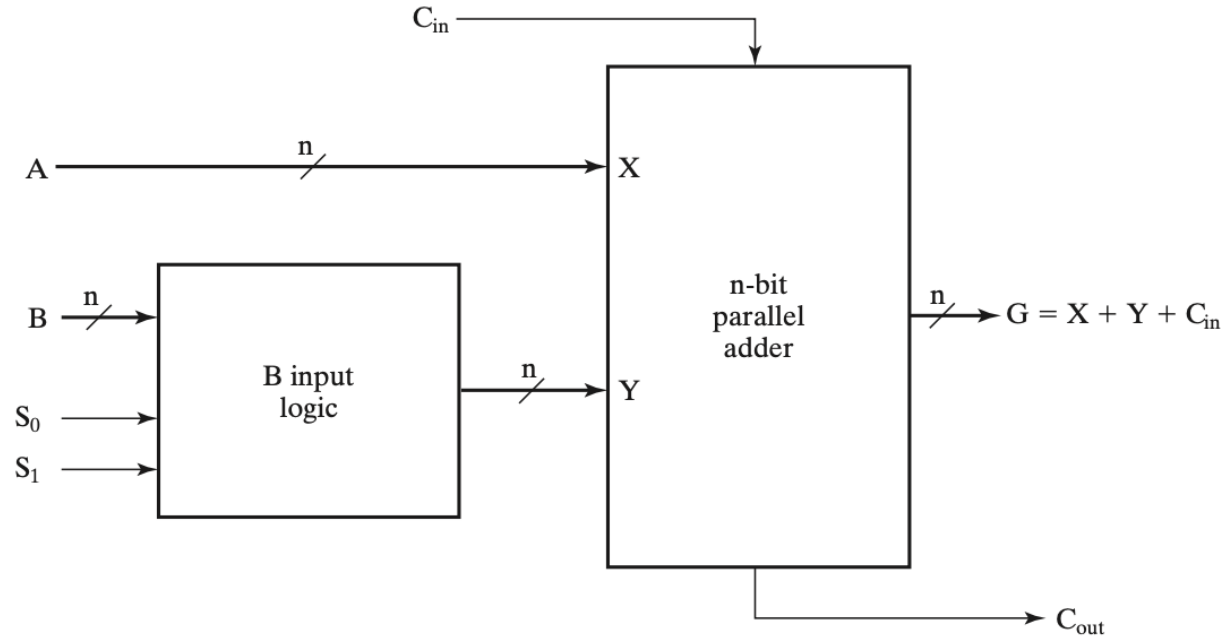
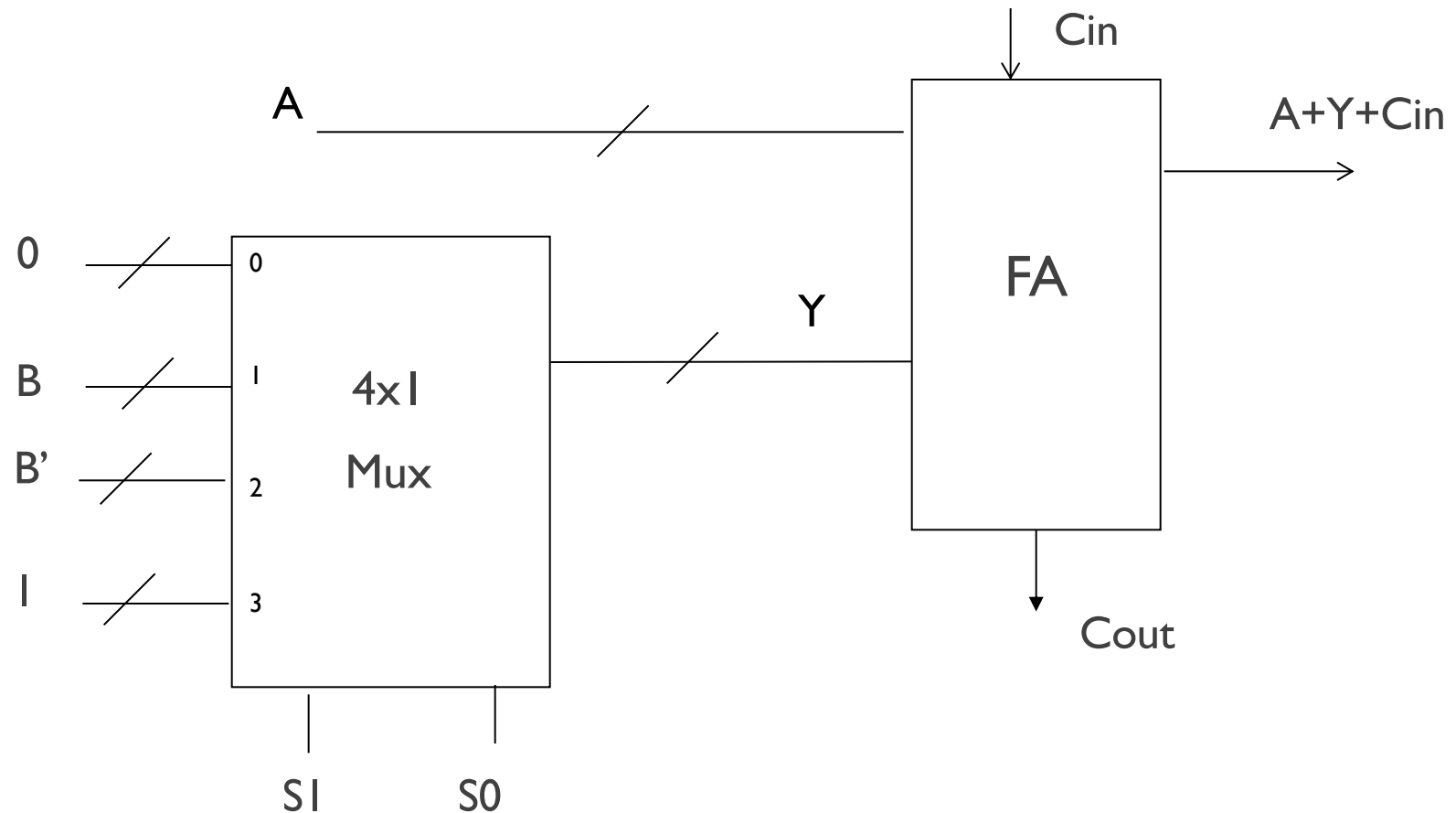


FIGURE 8-3
Block Diagram of an Arithmetic Circuit

TABLE 8-1
Function Table for Arithmetic Circuit

| Select | | Input Y | G = (A + Y + C _{in}) | |
|----------------|----------------|----------------|--------------------------------|---------------------------------------|
| S ₁ | S ₀ | | C _{in} = 0 | C _{in} = 1 |
| 0 | 0 | all 0s | $G = A$ (transfer) | $G = A + 1$ (increment) |
| 0 | 1 | B | $G = A + B$ (add) | $G = A + B + 1$ |
| 1 | 0 | \overline{B} | $G = A + \overline{B}$ | $G = A + \overline{B} + 1$ (subtract) |
| 1 | 1 | all 1s | $G = A - 1$ (decrement) | $G = A$ (transfer) |

AU Multiplexer-based Implementation



- The B input logic can be implemented with n multiplexers.
- The arithmetic circuit can be constructed with n full adders and n 4-to-1 multiplexers.

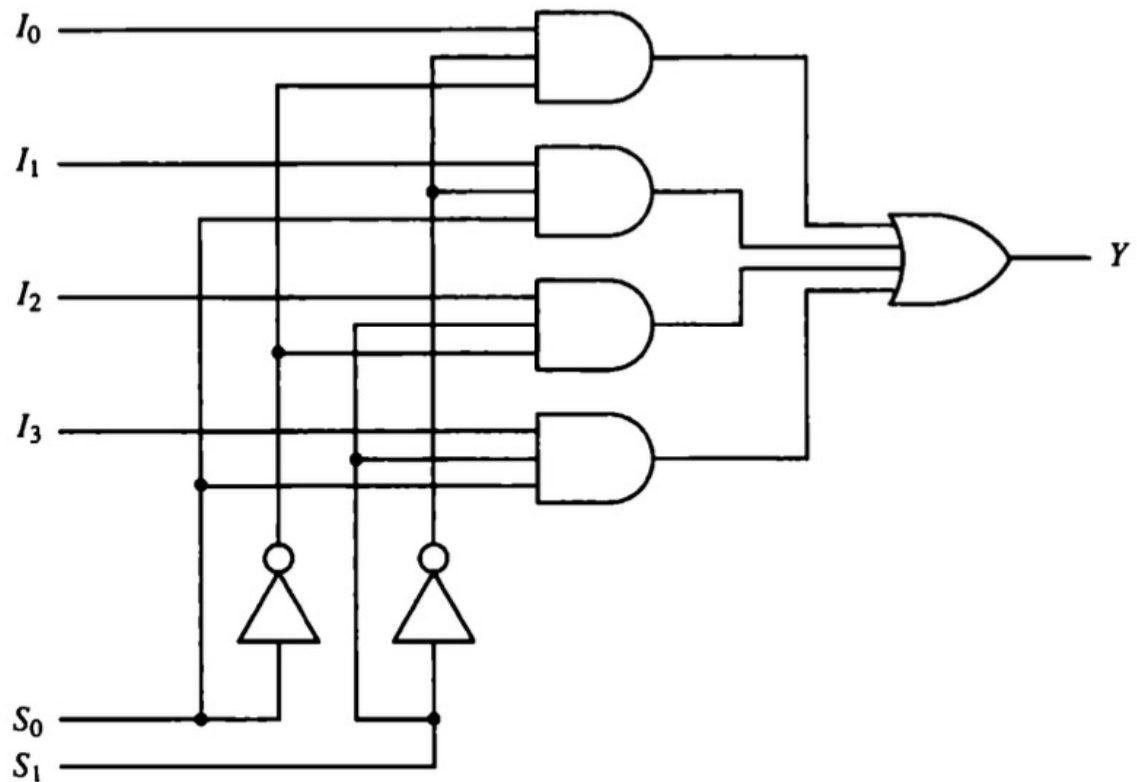
4-to-1-Line Multiplexer

Remember from Unit 6

Condensed Truth Table for 4-to-1-Line Multiplexer

| S_1 | S_0 | Y |
|-------|-------|-------|
| 0 | 0 | I_0 |
| 0 | 1 | I_1 |
| 1 | 0 | I_2 |
| 1 | 1 | I_3 |

Figure 2-4 4-to-1-line multiplexer.



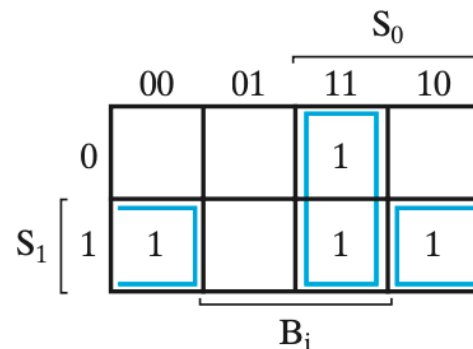
$$Y = (\bar{S}_1\bar{S}_0)I_0 + (\bar{S}_1S_0)I_1 + (S_1\bar{S}_0)I_2 + (S_1S_0)I_3$$

AU Primitive Gate-based Implementation

- The number of gates in the B input logic can be reduced.
- Instead of using 4-to-1 multiplexers, we could build this **circuit using primitive gates**.
- Remember **B** and **Y** are each 4 bits long!

| Inputs | | | Output |
|--------|-------|-------|--------------------------|
| S_1 | S_0 | B_i | Y_i |
| 0 | 0 | 0 | 0 $Y_i = 0$ |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 $Y_i = B_i$ |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 $Y_i = \overline{B_i}$ |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 $Y_i = 1$ |
| 1 | 1 | 1 | 1 |

(a) Truth table



(b) Map simplification:
 $Y_i = B_i S_0 + \overline{B_i} S_1$

□ **FIGURE 8-4**

B Input Logic for One Stage of Arithmetic Circuit

AU Primitive Gate-based Implementation

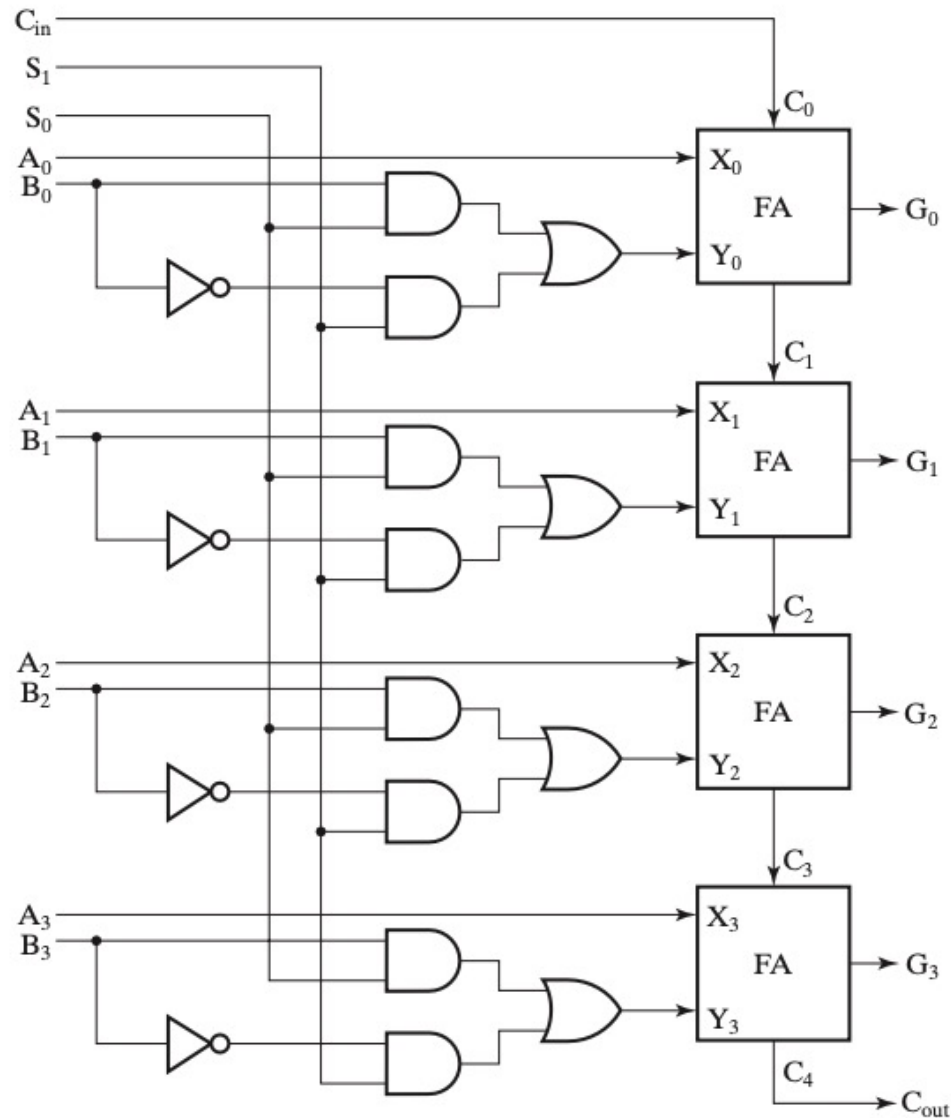
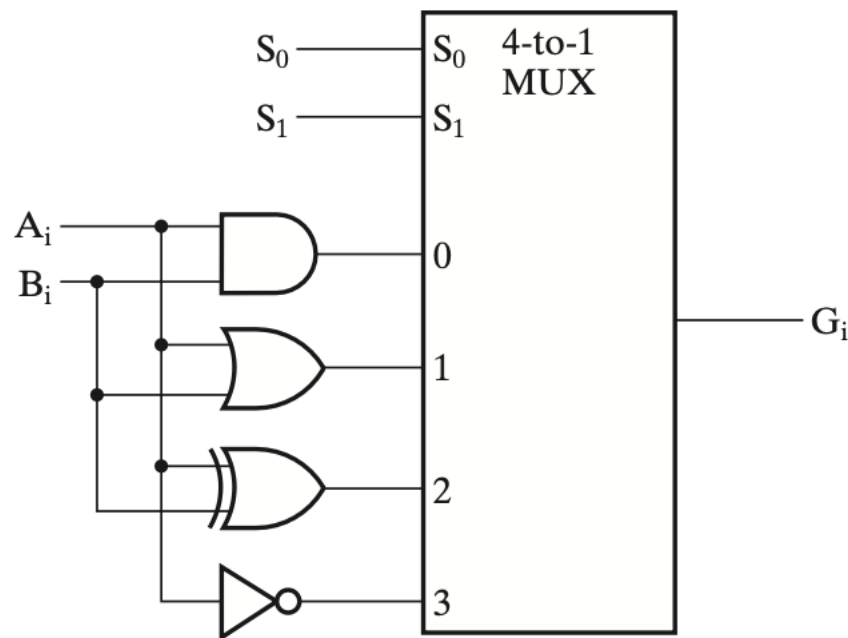


FIGURE 8-5
Logic Diagram of a 4-Bit Arithmetic Circuit

Logic Circuit

- Bitwise operations:
 - AND, OR, XOR, NOT.
- Use logic gates and MUX.



(a) Logic diagram

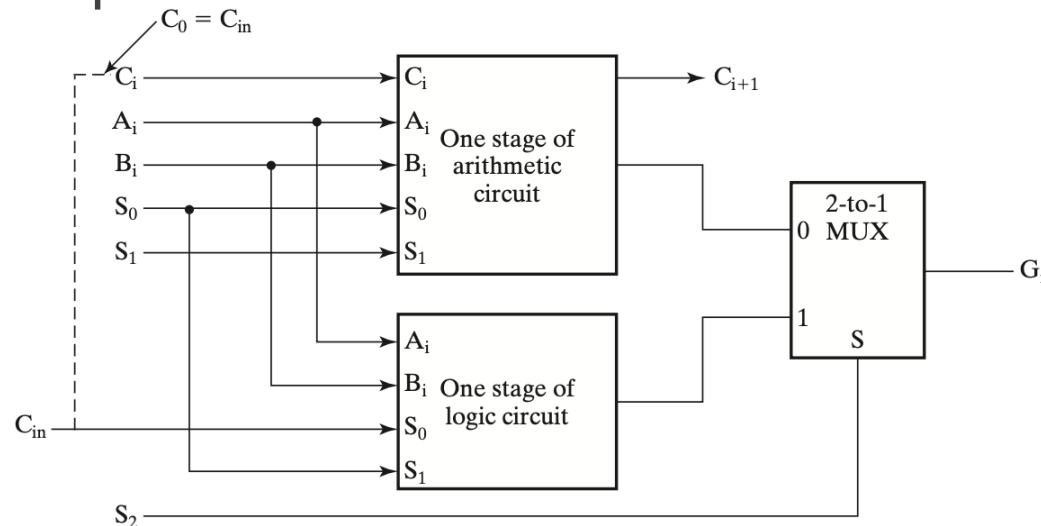
| S_1 | S_0 | Output | Operation |
|-------|-------|--------------------|-----------|
| 0 | 0 | $G = A \wedge B$ | AND |
| 0 | 1 | $G = A \vee B$ | OR |
| 1 | 0 | $G = A \oplus B$ | XOR |
| 1 | 1 | $G = \overline{A}$ | NOT |

(b) Function table

□ **FIGURE 8-6**
One Stage of Logic Circuit

Combining Arithmetic and Logic Sections

- 1 stage of arithmetic + 1 stage of logic + 2-to-1 MUX = 1 stage of ALU.
- S_2 performs the arithmetic/logic selection
- C_{in} is connected to the least significant selection input for the logic circuit, S_0 .
- C_{i+1} must be connected to the input carry C_i of the next stage in sequence.



□ FIGURE 8-7
One Stage of ALU

Function Table for ALU

□ **TABLE 8-2**
Function Table for ALU

| Operation Select | | | | Operation | Function |
|------------------|-------|-------|----------|----------------------------|-------------------------------|
| S_2 | S_1 | S_0 | C_{in} | | |
| 0 | 0 | 0 | 0 | $G = A$ | Transfer A |
| 0 | 0 | 0 | 1 | $G = A + 1$ | Increment A |
| 0 | 0 | 1 | 0 | $G = A + B$ | Addition |
| 0 | 0 | 1 | 1 | $G = A + B + 1$ | Add with carry input of 1 |
| 0 | 1 | 0 | 0 | $G = A + \overline{B}$ | A plus 1s complement of B |
| 0 | 1 | 0 | 1 | $G = A + \overline{B} + 1$ | Subtraction |
| 0 | 1 | 1 | 0 | $G = A - 1$ | Decrement A |
| 0 | 1 | 1 | 1 | $G = A$ | Transfer A |
| 1 | X | 0 | 0 | $G = A \wedge B$ | AND |
| 1 | X | 0 | 1 | $G = A \vee B$ | OR |
| 1 | X | 1 | 0 | $G = A \oplus B$ | XOR |
| 1 | X | 1 | 1 | $G = \overline{A}$ | NOT (1s complement) |

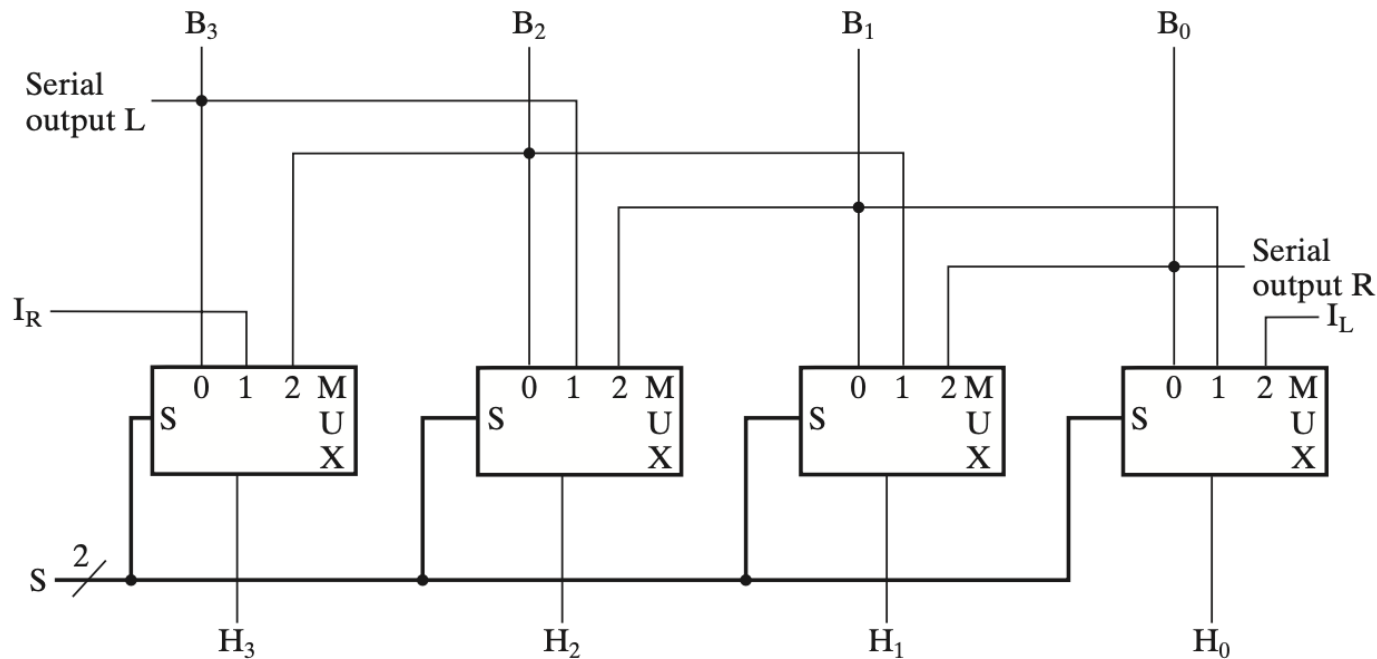
Function Table for ALU

- The table shows a sample function table for an ALU.
- All of the **arithmetic** operations have $S_2=0$, and all of the **logical** operations have $S_2=1$.
- These are the same functions we saw when we built our arithmetic and logic units a few minutes ago.
- Since our ALU only has 4 logical operations, we don't need S_1 . The operation done by the logic unit depends only on S_0 and C_{in} .

Shift Operation

- Shift right and shift left.
- Could be implemented using a **bidirectional shift register with parallel load**.
 - Requires 3 clock pulses
 - Load data from Bus to register.
 - Shift.
 - Return data to Bus.
 - Sequential.
- Alternative solution: use a combinational circuit.
 - Use MUXs.
 - Requires 1 clock pulse.

4-Bit Basic Left/Right Shifter



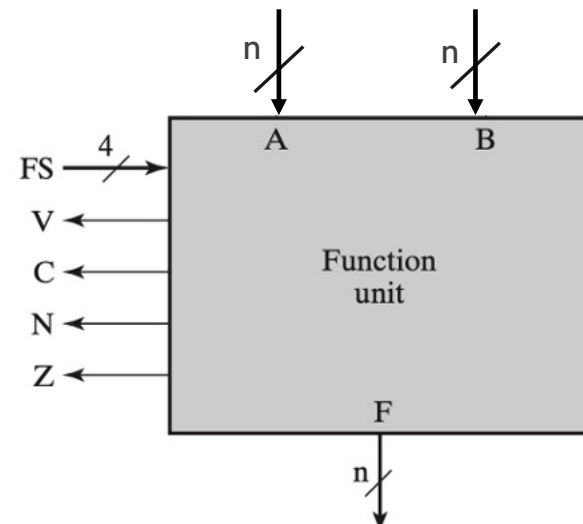
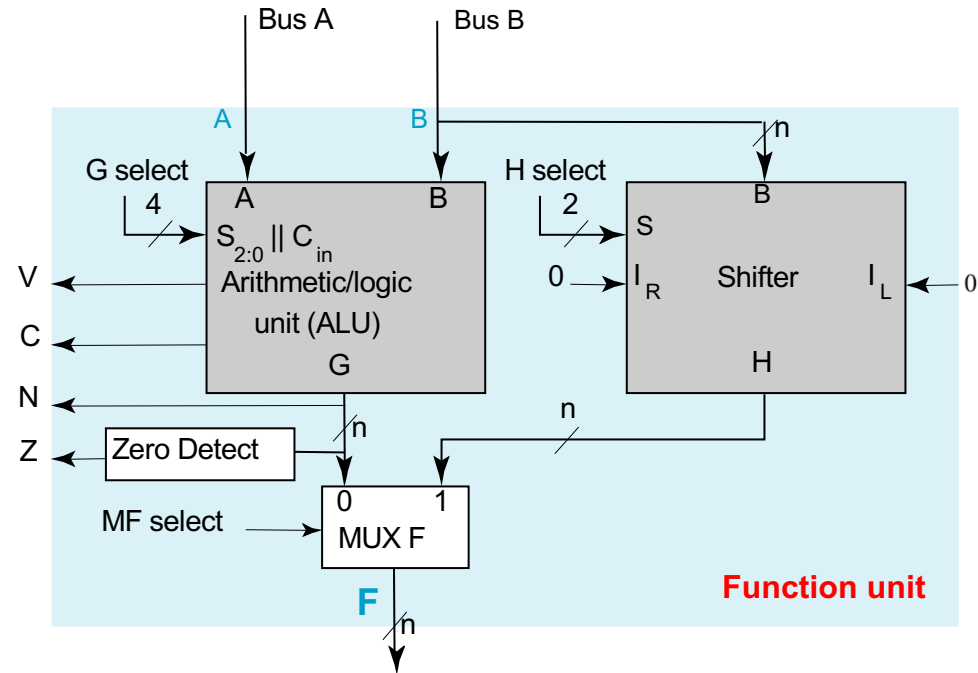
- **Serial Inputs** (value depends on shift type):
 - I_R for right shift
 - I_L for left shift
- **Serial Outputs**
 - R for right shift (Same as LSB input)
 - L for left shift (Same as MSB input)
- **Shift Functions:**

(SI, S0) = 00 Pass B unchanged
 01 Right shift
 10 Left shift
 11 Unused
- **Circuit type:**
 combinational shifter

Function Unit

Function Unit = ALU + Shifter

- **A** and **B** are two n -bit numeric inputs.
- **FS** is an m -bit function select code, which picks one of 2^m functions.
- The n -bit result is called **G**.
- Several **status bits** provide more information about the output **G**:
 - **V** = 1 in case of signed overflow.
 - **C** is the carry out.
 - **N** = 1 if the result is negative.
 - **Z** = 1 if the result is 0.



Function Unit

| FS | MF Select | G Select | H Select | Microoperation |
|------|-----------|----------|----------|-------------------------------------|
| 0000 | 0 | 0000 | XX | $F \leftarrow A$ |
| 0001 | 0 | 0001 | XX | $F \leftarrow A + 1$ |
| 0010 | 0 | 0010 | XX | $F \leftarrow A + B$ |
| 0011 | 0 | 0011 | XX | $F \leftarrow A + B + 1$ |
| 0100 | 0 | 0100 | XX | $F \leftarrow A + \overline{B}$ |
| 0101 | 0 | 0101 | XX | $F \leftarrow A + \overline{B} + 1$ |
| 0110 | 0 | 0110 | XX | $F \leftarrow A - 1$ |
| 0111 | 0 | 0111 | XX | $F \leftarrow A$ |
| 1000 | 0 | 1X00 | XX | $F \leftarrow A \wedge B$ |
| 1001 | 0 | 1X01 | XX | $F \leftarrow A \vee B$ |
| 1010 | 0 | 1X10 | XX | $F \leftarrow A \oplus B$ |
| 1011 | 0 | 1X11 | XX | $F \leftarrow \overline{A}$ |
| 1100 | 1 | XXXX | 00 | $F \leftarrow B$ |
| 1101 | 1 | XXXX | 01 | $F \leftarrow \text{sr } B$ |
| 1110 | 1 | XXXX | 10 | $F \leftarrow \text{sl } B$ |

Boolean Equations:

$$MF = FS_3 FS_2$$

$$G_3 = FS_3$$

$$G_2 = FS_2$$

$$G_1 = FS_1$$

$$G_0 = FS_0$$

$$H_1 = FS_1$$

$$H_0 = FS_0$$