Digital System Design

ALU

Lecture - 2

What is an ALU

- **Arithmetic logic unit (ALU)** is a digital circuit used to perform arithmetic and logic operations.
- Represents the fundamental building block of the central processing unit (CPU) of a computer

How Does an ALU work?

- Most of the operations of a CPU are performed by one or more ALUs, which load data from input registers.
- A register is a small amount of storage available as part of a CPU.
- The **control unit** tells the ALU what operation to perform on that data and the ALU stores the result in an output register.
- The control unit moves the data between these registers, the ALU, and memory.

Functions of an ALU

- Processor's core component
- Try to faster the hardware
- A combinational logic circuit
- Can perform multiple operations in runtime
- Performs **8 Arithmetic** Operations
- Performs **4 Logical** Operations

Operations in ALU

- **Arithmetic**
  - Addition ($F = A + B$)
  - Addition with Carry ($F = A + B + 1$)
  - Subtraction ($F = A – B$ or $F = A + B' + 1$)
  - Subtraction with Borrow ($F = A - B -1$ or $F = A + B'$)
  - Increment ($F = A + 1$)
  - Decrement ($F = A - 1$)
  - Transfer ($F = A$)
  - Transfer with Carry ($F = A$; $C_{out} = 1$)

Operations in ALU (Contd.)

- **Logical**
  - And (F = A . B)
  - Or (F = A + B)
  - XOR (F = A xor B)
  - NOT (F = A’)

A 4-bit ALU

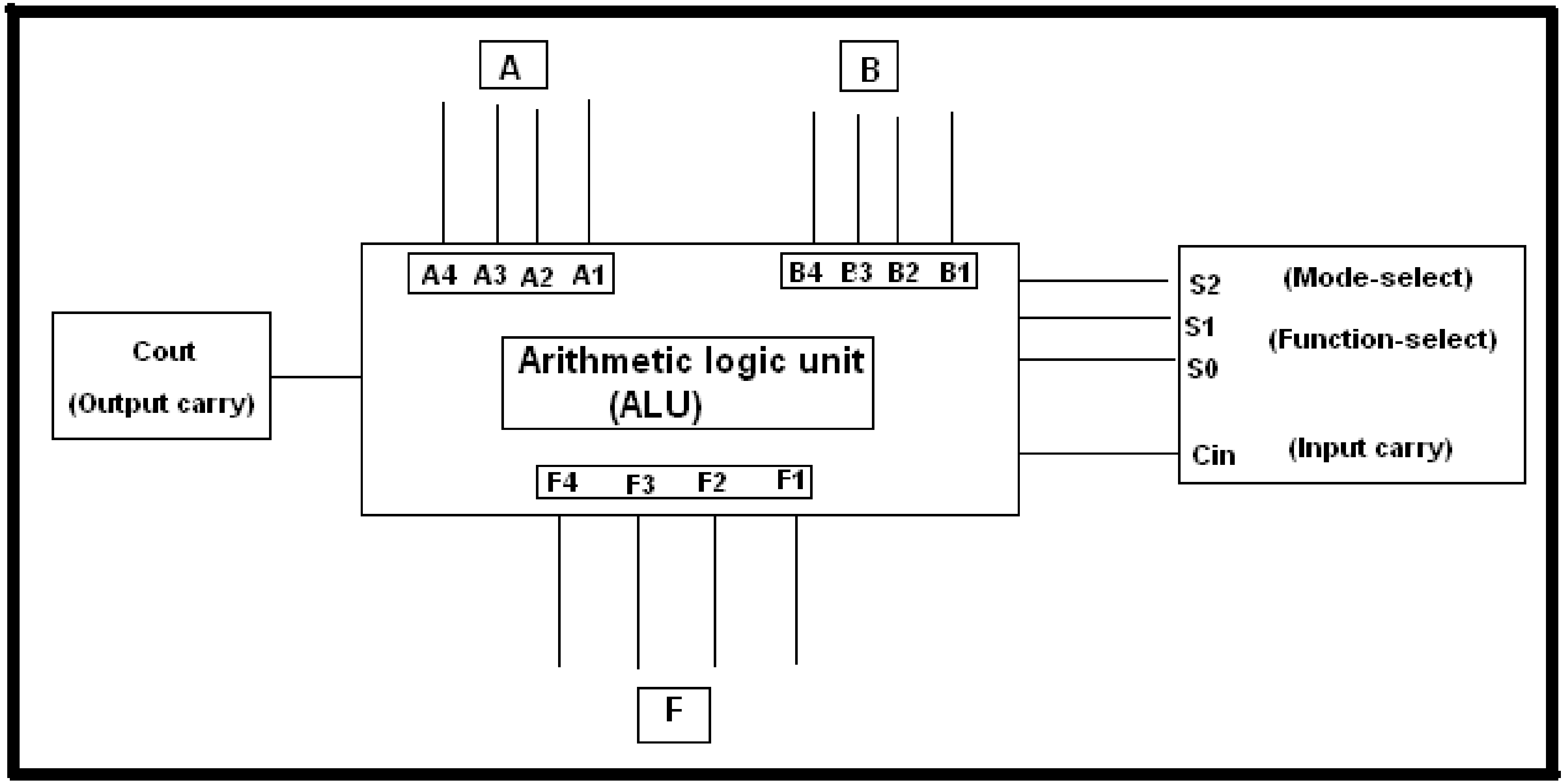


Fig: Block Diagram of a 4-bit ALU

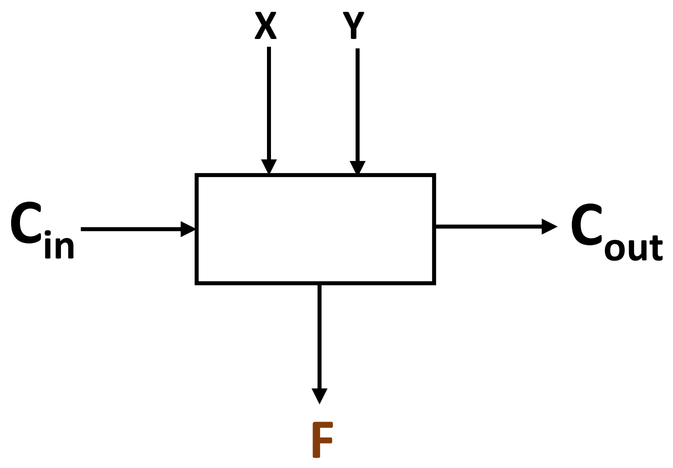
A 4-bit ALU (Contd.)

- has a number of selection lines to select a particular operation
- **S_2 distinguishes between arithmetic and logical operations**
- **S_1, S_0** are two function select inputs which specify the particular arithmetic or logic operation to be generated
- **The input and output carries have meaning only during an arithmetic operation**
- Basic component of the arithmetic section is a parallel adder
- **C_{in}** goes to the adder in the least significant bit position
- **C_{out}** comes from the adder in the most significant bit position

---For More, See Section 9.3 of **Digital Logic and Computer Design by M. Morris Mano**

ALU Design

S_2	S_1	S_0	C_{in}	X	Y	F
0	0	0	0	A	0	A
0	0	0	1	A	0	A+1
0	0	1	0	A	B	A+B
0	0	1	1	A	B	A+B+1
0	1	0	0	A	$\overline{B}$	A-B-1 $=A+\overline{B}$
0	1	0	1	A	$\overline{B}$	A-B $=A+\overline{B}+1$
0	1	1	0	A	all 1	A-1 $=2^n-1+A$
0	1	1	1	A	all 1	$2^n-1+1+A$



- X = A
- When Y = B,
  $S_1 = 0$, $S_0 = 1$
  So,
  $\mathbf{Y = \overline{S_1} S_0 B}$

Now,

$$Y_i = \overline{S_1}S_0 B_i + S_1\overline{S_0}\,\overline{B}_i + S_1S_0 1$$

$$Y_i = \overline{S_1}S_0 B_i + S_1\overline{S_0}\,\overline{B}_i + S_1S_0 (B_i + \overline{B}_i)$$
$$= S_0 B_i (S_1 + \overline{S_1}) + S_1\overline{B}_i(S_0 + \overline{S_0})$$

$$Y_i = S_0 B_i + S_1\overline{B}_i$$

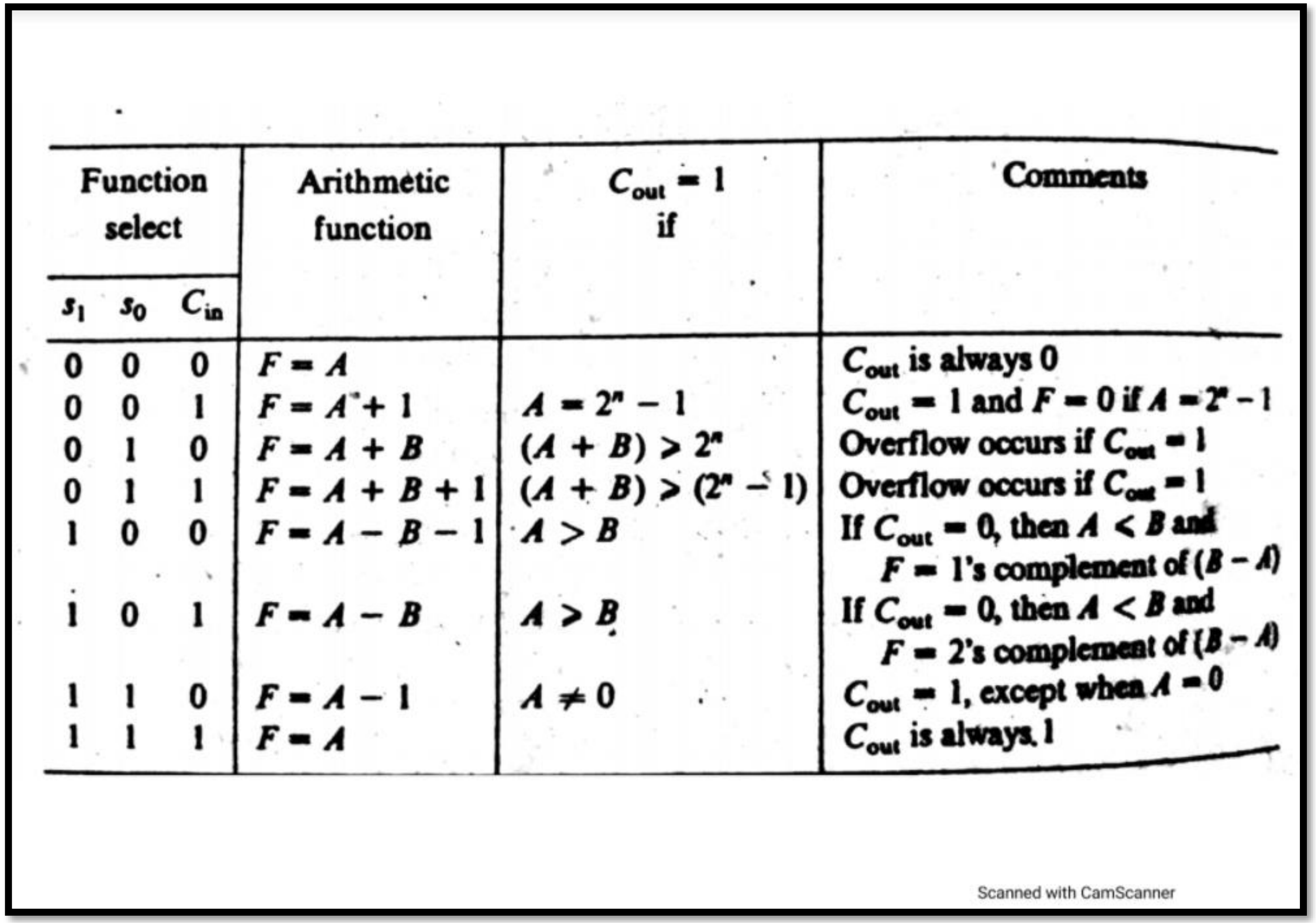
C_{in} remains unchanged i.e., $Z_i = C_{in}$

The PA works as ALU then.

Effects of Output Carry

- The output carry of an arithmetic circuit or ALU has special significance, especially after a subtraction operation.
- To investigate the effect of output carry, we expand the arithmetic circuit to n bits so that $C_{out} = 1$, when the output of the circuit is equal to or greater than 2^n.
- An output carry of 1 after an addition operation denotes an overflow condition.
- The table in the next slide lists the conditions for having an output carry in the circuit.

Effects of Output Carry (Contd.)



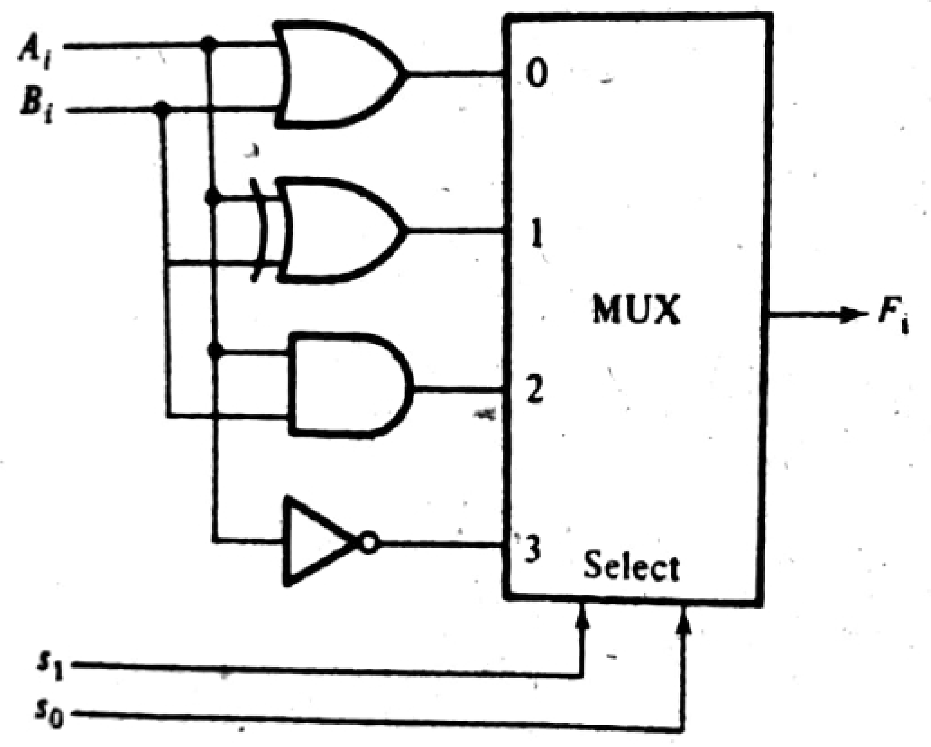
Function select			Arithmetic function	$C_{out} = 1$ if	Comments
s_1	s_0	C_{in}			
0	0	0	$F = A$		C_{out} is always 0
0	0	1	$F = A + 1$	$A = 2^n - 1$	$C_{out} = 1$ and $F = 0$ if $A = 2^n - 1$
0	1	0	$F = A + B$	$(A + B) > 2^n$	Overflow occurs if $C_{out} = 1$
0	1	1	$F = A + B + 1$	$(A + B) > (2^n - 1)$	Overflow occurs if $C_{out} = 1$
1	0	0	$F = A - B - 1$	$A > B$	If $C_{out} = 0$, then $A < B$ and $F =$ 1's complement of $(B - A)$
1	0	1	$F = A - B$	$A > B$	If $C_{out} = 0$, then $A < B$ and $F =$ 2's complement of $(B - A)$
1	1	0	$F = A - 1$	$A \neq 0$	$C_{out} = 1$, except when $A = 0$
1	1	1	$F = A$		C_{out} is always 1

---For More, Table 9.2 of **Digital Logic and Computer Design by M. Morris Mano**

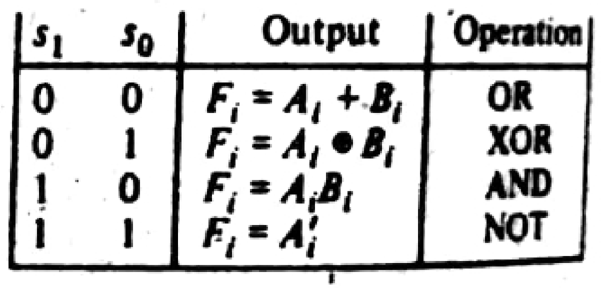
Design of Logic Circuit

- All logic operations can be done by **AND, OR and NOT**
- For **3** operations, we need **2** selection variables
- **2** selection lines can select among **4** logic operations
- So we choose the **XOR** operation also

Design of Logic Circuit (Contd.)



(a) Logic diagram



s_1	s_0	Output	Operation
0	0	$F_i = A_i + B_i$	OR
0	1	$F_i = A_i \oplus B_i$	XOR
1	0	$F_i = A_iB_i$	AND
1	1	$F_i = A_i'$	NOT

(b) Function table

---For More, Section 9.5 of **Digital Logic and Computer Design by M. Morris Mano**

The circuit must be repeated n times for an n-bit logic circuit

Combining Logic and Arithmetic Circuits

- Can be combined with the arithmetic circuit to produce one arithmetic logic unit
- S_1 and S_0 can be made common along with a third variable, S_2
- S_2 used to differentiate between the arithmetic and logic section
- $S_2=0$ selects the arithmetic output while $S_2=1$ selects the logical output
- Still not the best design

Combining Logic and Arithmetic Circuits (Contd.)

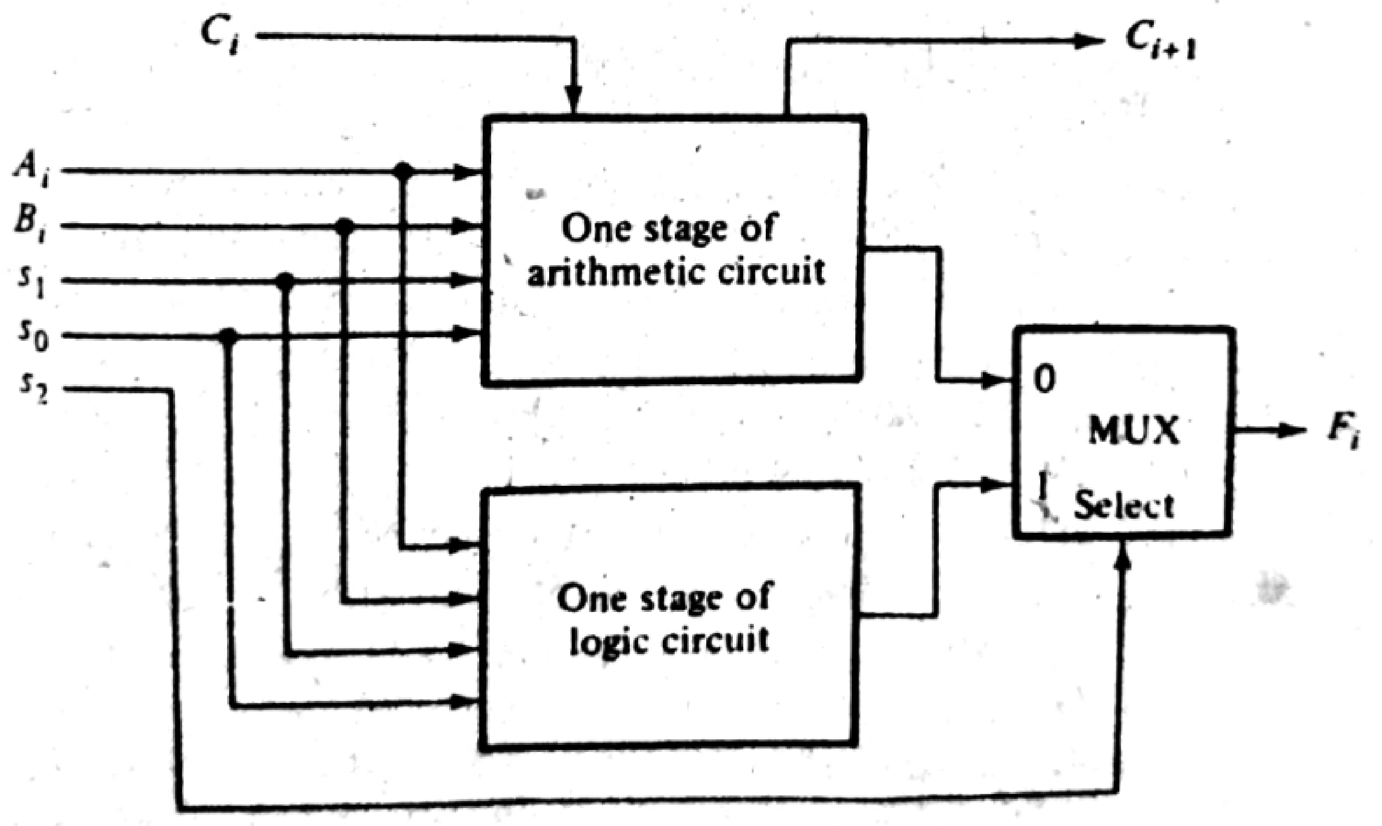


Figure 9-12 Combining logic and arithmetic circuits

Combining Logic and Arithmetic Circuits (Contd.)

Disadvantages :

- Requires a lot of ICs
- Need to build both the arithmetic and logic circuit distinctively
- More processing time with more overhead

A More Efficient ALU

- Easier to generate logic operations in an already available arithmetic circuit
- C_{in} always 0 (or, don’t care) when selection variable S_2=1
- When S_2=1,combination of S_1 and S_0 selects the 4 logical operations
- When S_2=0,combination of $S_{1,}$ S_0 and C_{in} selects the 8 arithmetic operations

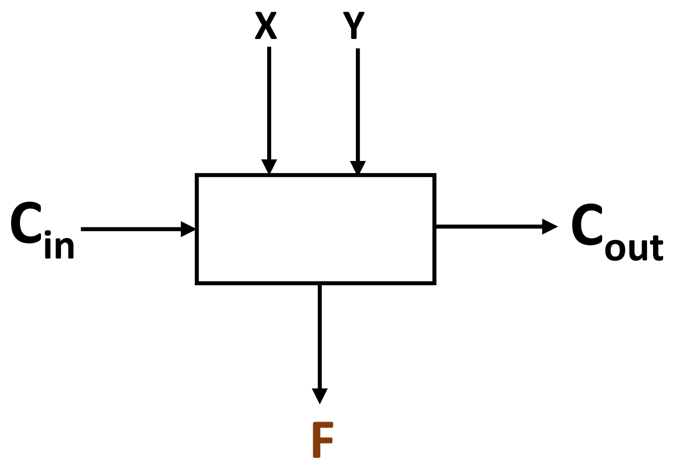
A More Efficient ALU (Contd.)

Steps:

1. Design the arithmetic section independent of the logic section
2. Determine the logic operations obtained from the arithmetic circuit in step 1, assuming that the input carries to all stages are 0
3. Modify the arithmetic circuit to obtain the required logic operations

Final Function Table of the ALU

S_2	S_1	S_0	C_{in}	X	Y	F
0	0	0	0	A	0	A
0	0	0	1	A	0	A+1
0	0	1	0	A	B	A+B
0	0	1	1	A	B	A+B+1
0	1	0	0	A	$\overline{B}$	A-B-1 $=A+\overline{B}$
0	1	0	1	A	$\overline{B}$	A-B $=A+\overline{B}+1$
0	1	1	0	A	all 1	A-1 $=2^n-1+A$
0	1	1	1	A	all 1	$2^n-1+1+A$
1	0	0	X	A + B	0	A v B
1	0	1	X	A	B	A ⊕ B
1	1	0	X	A + B’	B’	A ^ B
1	1	1	X	A	1	A’



---For More, Section 9.6 of **Digital Logic and Computer Design by M. Morris Mano**

Now,

$X_i = A_i + S_2S_1'S_0'B_i + S_2S_1S_0'B_i'$

$Y_i = S_0B_i + S_1B_i'$

$Z_i = S_2'C_{in}$ [C_{in} only works when S_2 is zero]

Practice Problems

- 9.10 - 9.14, 9.16 - 9.18