



AMERICAN INTERNATIONAL UNIVERSITY – BANGLADESH

Where leaders are created

Processor Logic Design

Processor Organization

- A processor unit is that part of a digital system that implements the operations in the system.
- It is comprised of a number of **registers** and the **digital functions** that implement **arithmetic**, **logic**, **shift** and **transfer** micro-operations.
- The processor unit when combined with a control unit that supervises the sequence of micro-operations is called a central processing unit (CPU).
- Central processing unit (CPU) consists of **MEMORY** (Register banks), ARITHMETIC LOGIC UNIT (**ALU**) and CONTROL UNIT (**CU**).
- There are **three organizations** for organizing a general-purpose processor unit:
 1. Bus organization
 2. Scratchpad memory
 3. Two-port memory

Bus Organization

- A bus organization has four registers, each register is connected to two multiplexers to form two buses A and B.
- To perform the micro-operation: **$R1 = R2 + R3$**
- The control unit must provide binary selection variables to the following selector inputs –
 1. **MUX A Selector:** to place contents of R2 onto bus A
 2. **MUX B Selector:** to place contents of R3 onto bus B
 3. **ALU Function Selector:** to provide the arithmetic operation $A + B$
 4. **Shift Selector:** for direct transfer from the output of the ALU onto output bus S (no shift)
 5. **Decoder Destination Selector:** to transfer the contents of bus S into R1

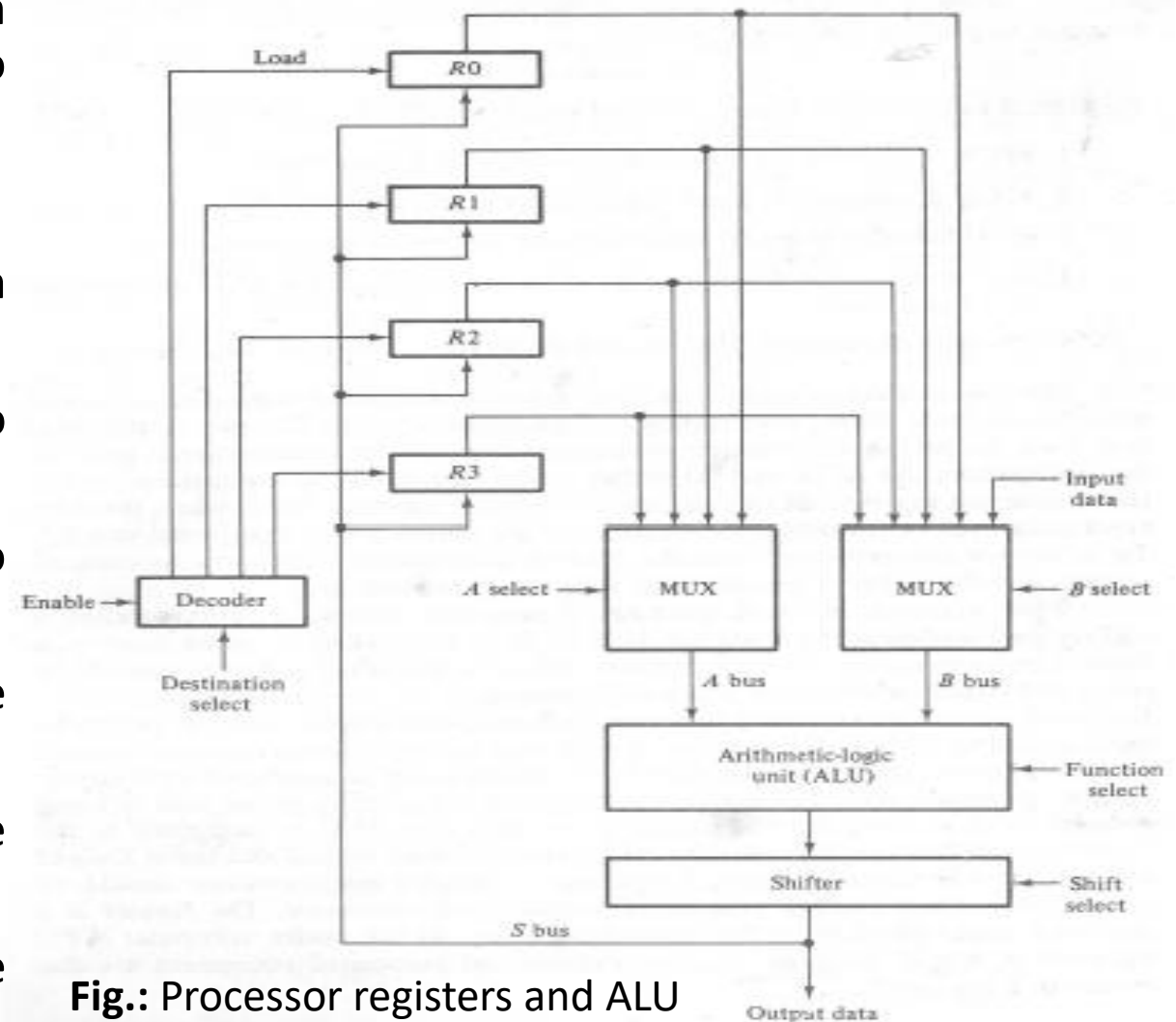


Fig.: Processor registers and ALU connected through Common Bus

Bus Organization

- The five control selection variables must be generated **simultaneously** and must **be available** during one common clock pulse interval.
- The binary information from the two source registers propagates through the combinational gates in the **multiplexers**, the **ALU**, and the **shifter**, to the **output bus** and into the **inputs of the destination register** all during **one clock pulse interval**.
- Then, when the **next clock pulse arrives**, the binary information on the **output bus** is transferred to R1.

SCRATCHPAD MEMORY

- Scratchpad memory is a kind of storage device where the registers are combined and connected together to form a memory chip.
- The registers in a processor unit can be enclosed within a small memory unit. When included in a processor unit, a small memory is sometimes called a scratchpad memory.
- The use of a small memory is a cheaper alternative to connecting processor registers through a bus system. The difference between the two systems is the manner in which information is selected for transfer into the ALU.

SCRATCHPAD MEMORY

- To perform the operation **$R1 = R2 + R3$** , the control must provide binary selection variables to perform the following sequence of **three microoperations**:

- T1: $A \leftarrow M[010]$ *Read R2 into register A*
- T2: $B \leftarrow M[011]$ *Read R3 into register B*
- T3: $M[001] \leftarrow A + B$ *Perform operation in ALU and transfer result to R1*

Ref.: Page # 363 (M. Mano)

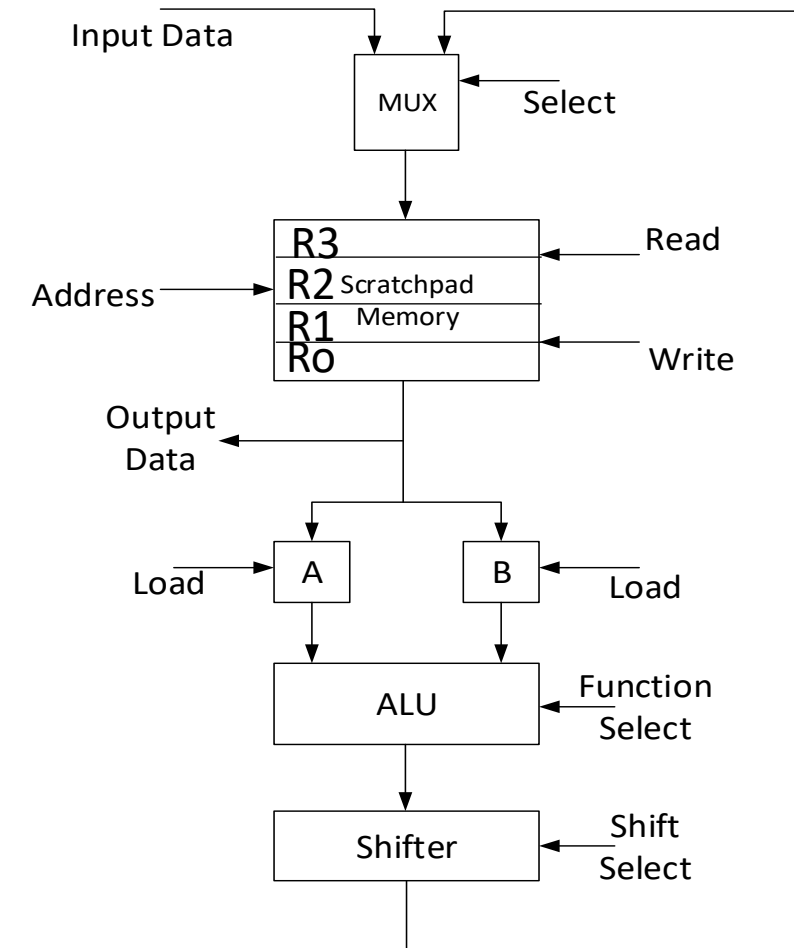


Fig.: Processor unit employing a Scratchpad Memory

Processor Unit with Two Port Memory

- Some **processors** employ a 2-port memory in order to overcome the **delay** caused **when reading two source registers**.
- A 2-port memory has **two separate address line** to select two words of memory **simultaneously**, and two source registers can be **read at the same time**.
- If the destination register is the same as one of the source register, then the entire microoperation can **be done within one clock pulse period**.

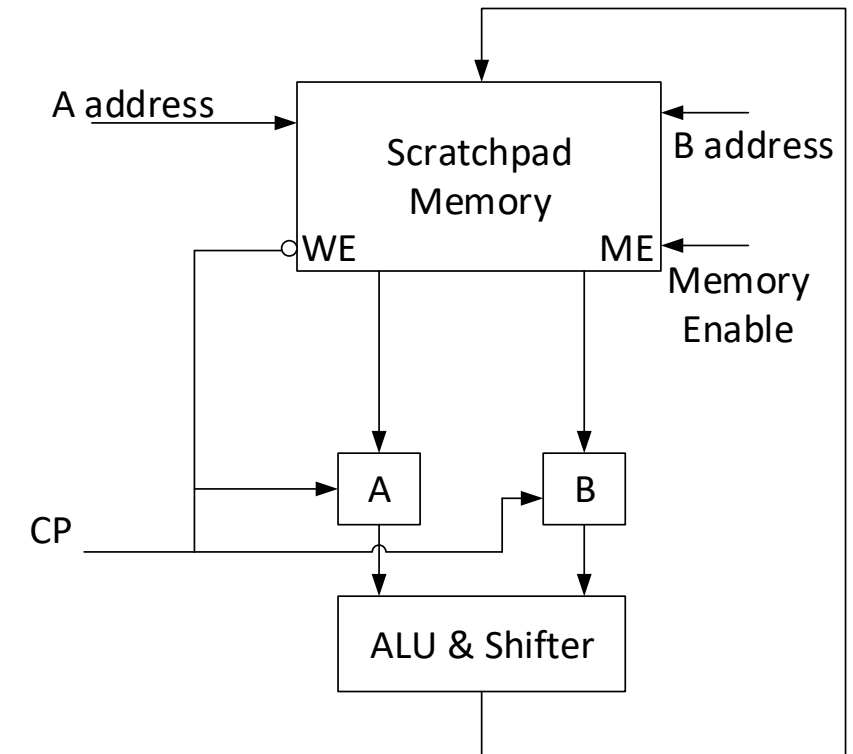


Fig.: Processor unit with 2-port memory

Processor Unit with two port memory

- When Clock (CP) = 1, A and B latches are open and accept information coming from memory. The WE is also in 0 state and disables write operation and enables read operation. Thus when Clock = 1, words selected by A and B addresses are read from memory and placed in A and B registers.
- When Clock (CP) = 0, the latches are closed and retains the last data entered. ALU operation is performed and if write is enabled since Clock = 0, and also ME input is enabled, the result of the micro-operation is written into memory word defined by B address.

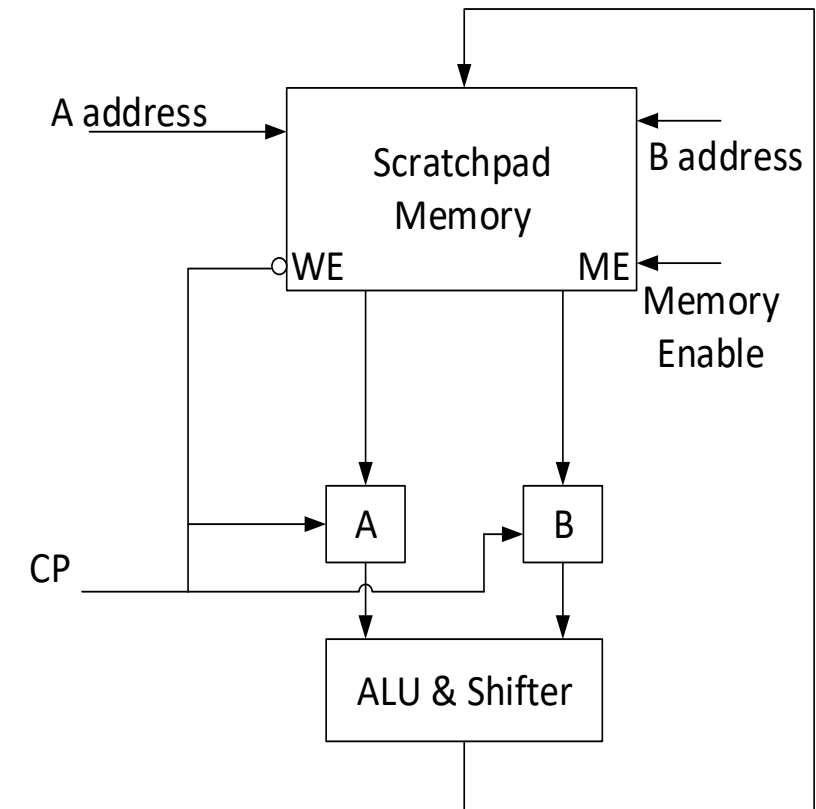


Fig.: Processor unit with 2-port memory

ARITHMETIC LOGIC UNIT (ALU)

- An ARITHMETIC LOGIC UNIT (ALU) is a **multi-operation**, combinational logic digital function.
- It can perform a set of basic **arithmetic** operations and a set of **logic** operations.
- The ALU has a number of selection lines to select a particular operation in the unit.
- The selection lines are decoded within the ALU so that k selection variables can specify upto 2^K distinct operations.

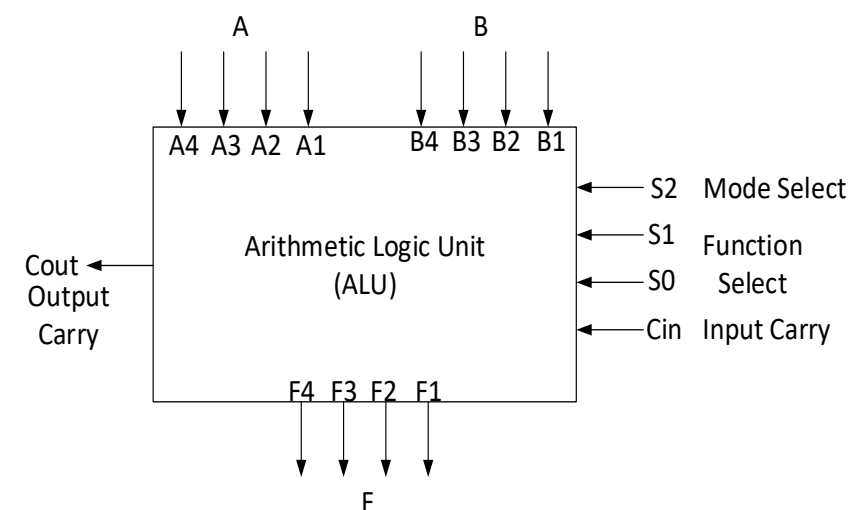


Fig.: Block diagram of a 4-bit ALU

ARITHMETIC LOGIC UNIT (ALU)

1. The mode select S_2 distinguishes between **arithmetic** and **logic** operations.
2. The two function select inputs S_1 and S_0 specify the particular **arithmetic** or **logic** operation to be generated.
3. The input and output carry (C_{in} and C_{out}) have meaning only during arithmetic operation.
4. The input carry C_{in} in the **least significant position** of an ALU is quite often used as **fourth selection variable** that can double the number of arithmetic operations. **Thus a total of 8 arithmetic operations and 4 logical operations can be performed.**

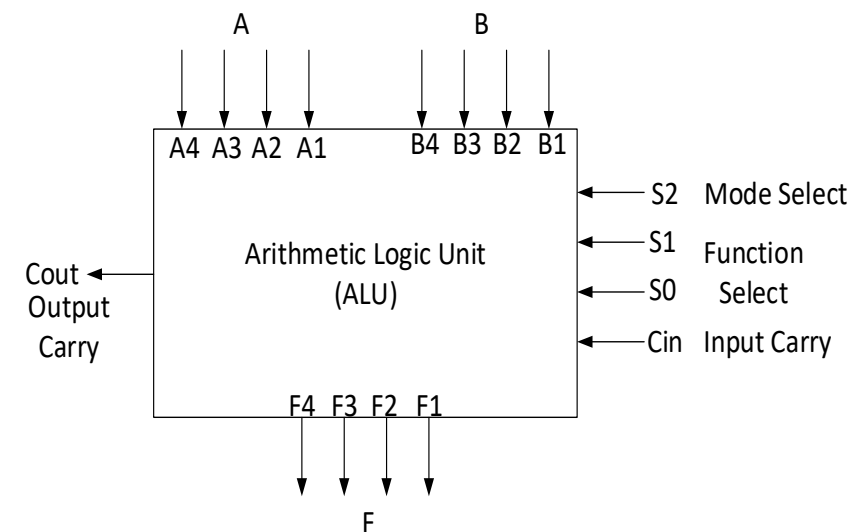
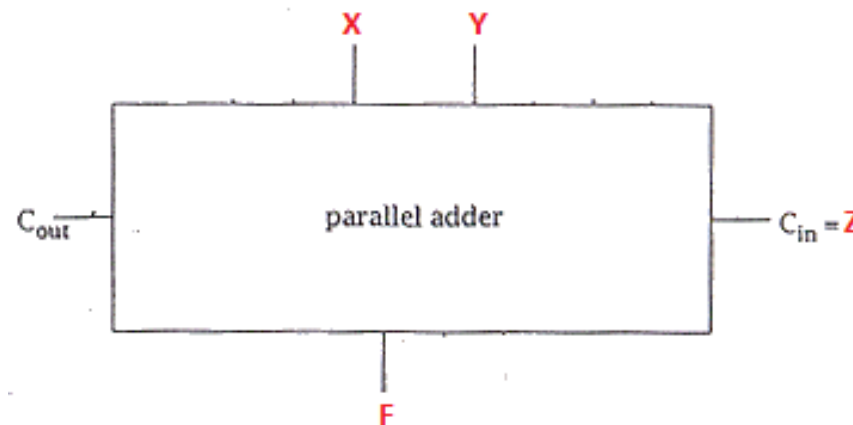


Fig.: Block diagram of a 4-bit ALU

Design of Arithmetic Circuit

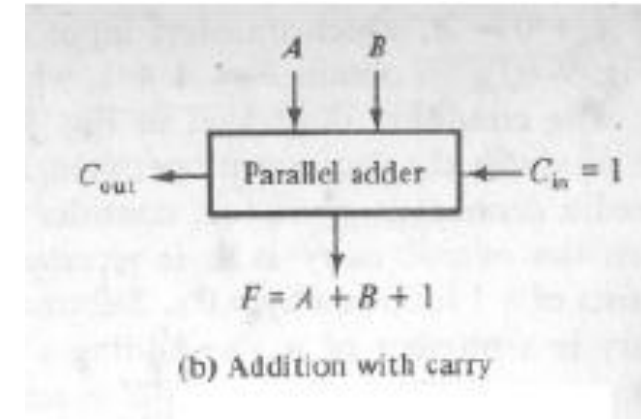
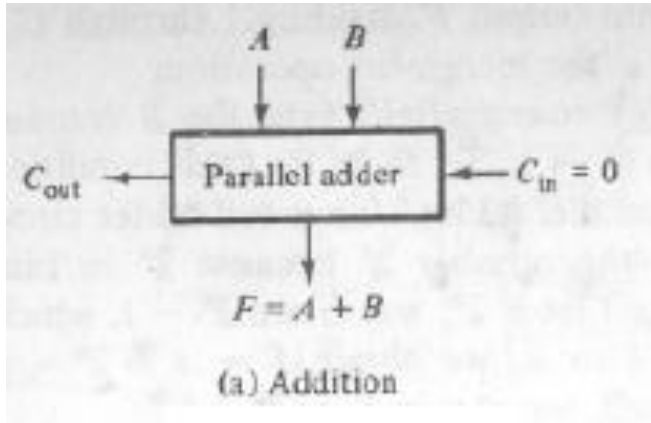
□ Arithmetic Section

- The basic component of the arithmetic section of an ALU is the **parallel adder**.
- A parallel adder is constructed with a number of **full adder circuits cascaded**.
- By **controlling the data inputs** to the parallel adder, it is possible to obtain different types of arithmetic operations.

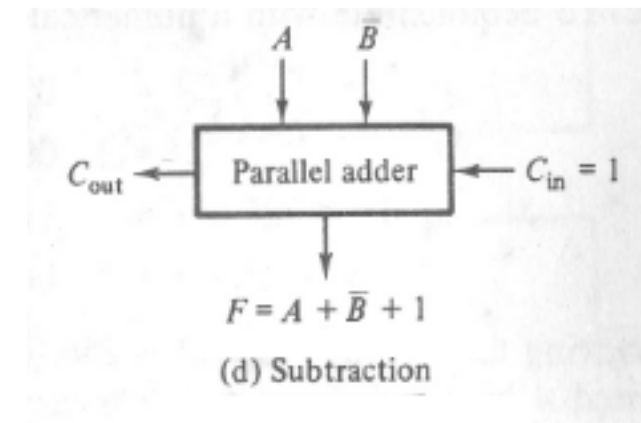
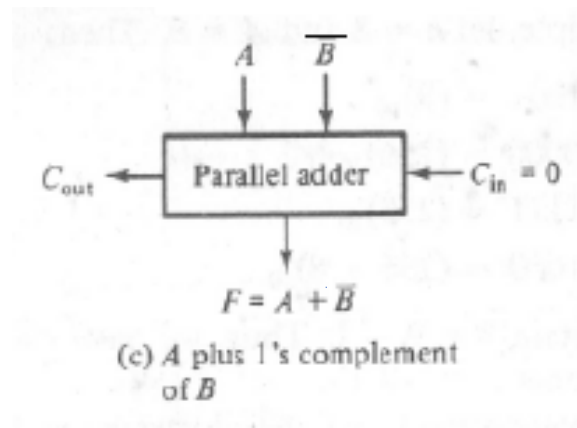


Operations obtained by controlling one set of inputs to a parallel adder

$$S_1 S_0 = 01$$

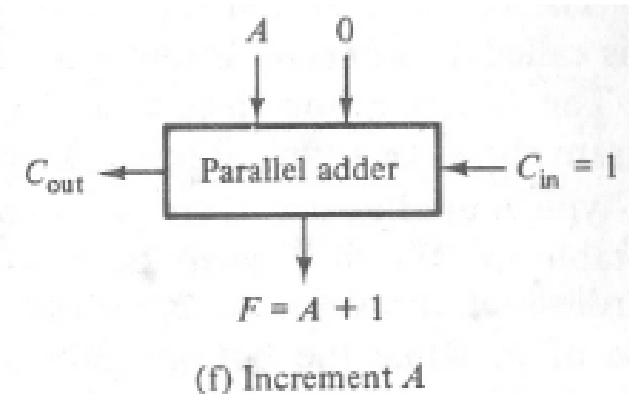
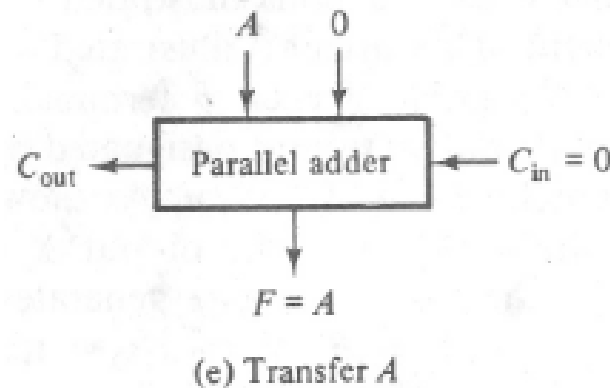


$$S_1 S_0 = 10$$

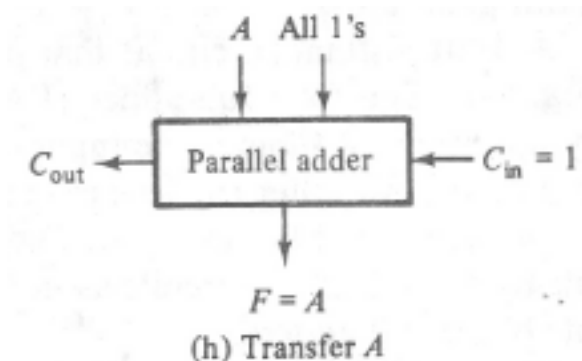
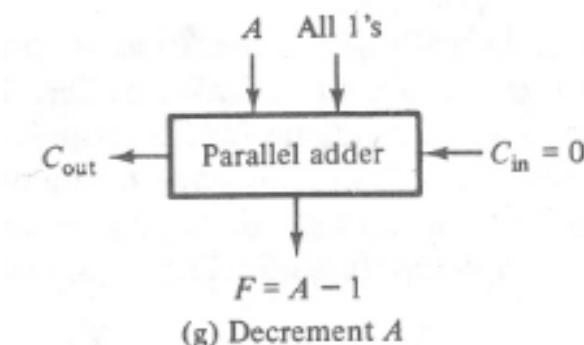


Operations obtained by controlling one set of inputs to a parallel adder

$$S_1 S_0 = 00$$



$$S_1 S_0 = 11$$



Rough

$A=3=$ **011**

$B=\text{All } 1\text{'s}$ **111**

$3-1 =$ **1010**

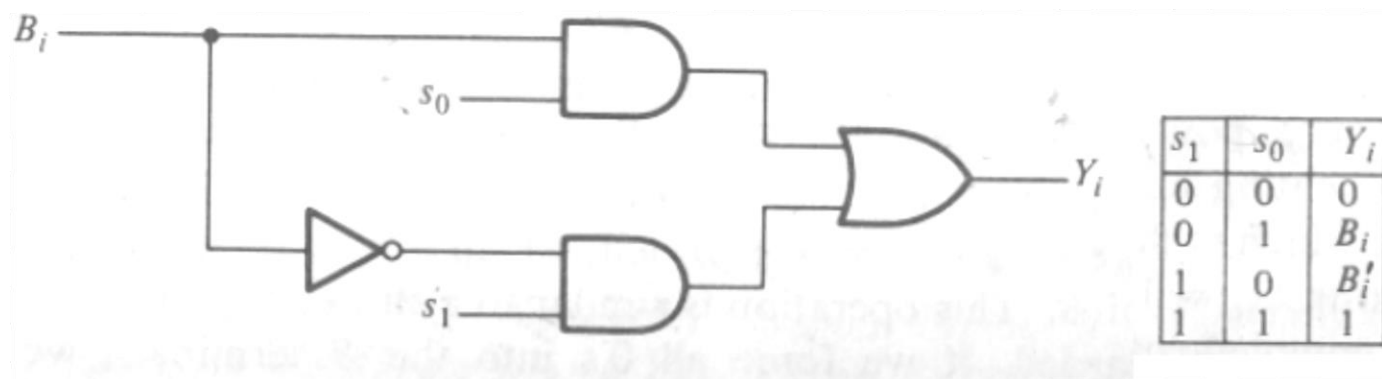
Discard 1

Function table for arithmetic Circuit

Function select			Y equals	Output equals	Function
s_1	s_0	C_{in}			
0	0	0	0	$F = A$	Transfer A
0	0	1	0	$F = A + 1$	Increment A
0	1	0	B	$F = A + B$	Add B to A
0	1	1	B	$F = A + B + 1$	Add B to A plus 1
1	0	0	\bar{B}	$F = A + \bar{B}$	Add 1's complement of B to A
1	0	1	\bar{B}	$F = A + \bar{B} + 1$	Add 2's complement of B to A
1	1	0	All 1's	$F = A - 1$	Decrement A
1	1	1	All 1's	$F = A$	Transfer A

True/Complement, one/zero Circuit

- The values of the Y depends on the inputs to the full-adder circuits that are a function of selection variables S_1 and S_0 .



- The arithmetic circuit needs a **combinational circuit** in each stage specified by Boolean functions –

$$X_i = A_i$$

$$Y_i = B_i S_0 + B_i' S_1, \quad i = 1, 2, 3, \dots, n$$

		S_0	
		0	1
S_1	0	0	B_i
	1	B_i'	1

Logic Diagram of Arithmetic Circuit

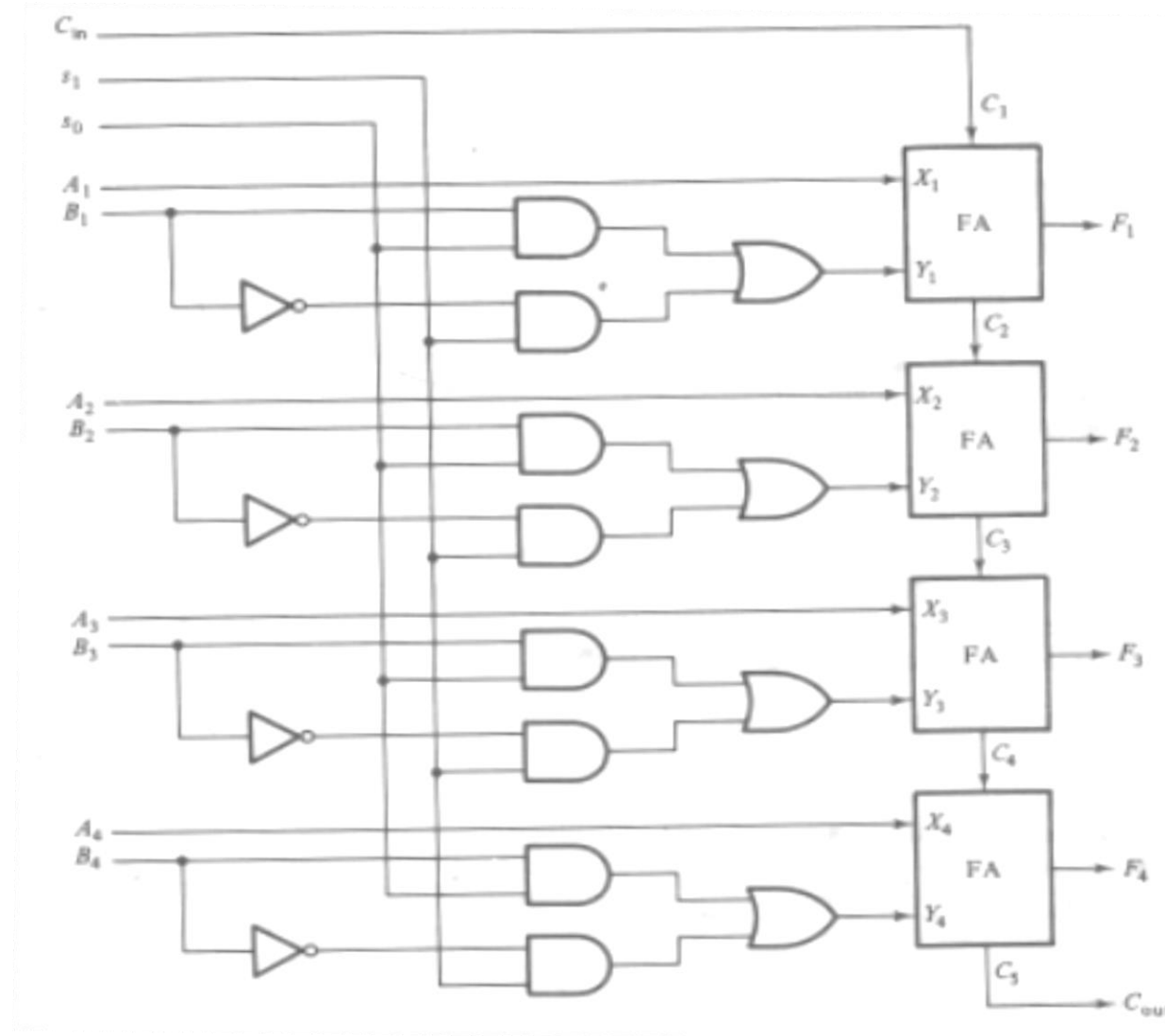


Figure : Logic diagram of arithmetic circuit (4 bit)

Ref. – Figure 9.8 (M. Mano)

Exercise (*Home Work*)

- Design an adder/subtractor circuit with a selection variable s and two inputs A and B . When $s = 0$ the circuit performs $A+B$. When $s = 1$ the circuit performs $A-B$ by taking the complement of B .

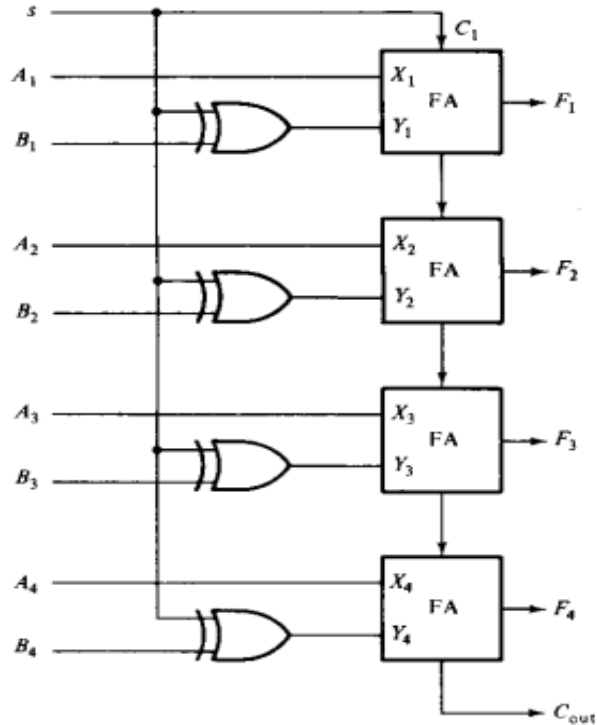
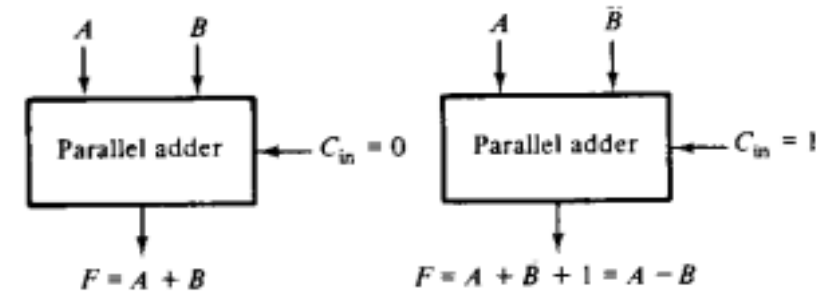
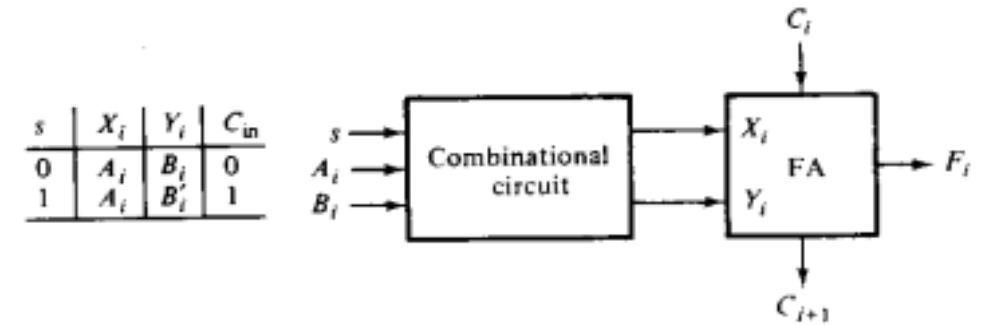


Figure 9-10 4-bit adder/subtractor circuit



(a) Function specification



(b) Specifying combinational circuit

s	A_i	B_i	X_i	Y_i
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	0
1	1	0	1	1
1	1	1	1	0

$$X_i = A_i$$

$$Y_i = B_i \oplus s$$

$$C_{in} = s$$

(c) Truth table and simplified equations

Figure 9-9 Derivation of an adder/subtractor circuit

Exercise (*Home Work*)

- Design an adder/subtractor circuit with one selection variable s and two inputs A and B : when $s=0$ the circuit performs $A+B$. When $S=1$ the circuit performs $A-B$ by taking the 2's complement of B .

S	Function	X	Y	Z
0	$A + B$	A	B	0
1	$A - B$	A	B'	1
	$= A + B' + 1$			

$$X_i = A_i$$

$$Y_i = S'B + SB' = B_i \text{ XOR } S$$

$$Z = C_{in} = S$$

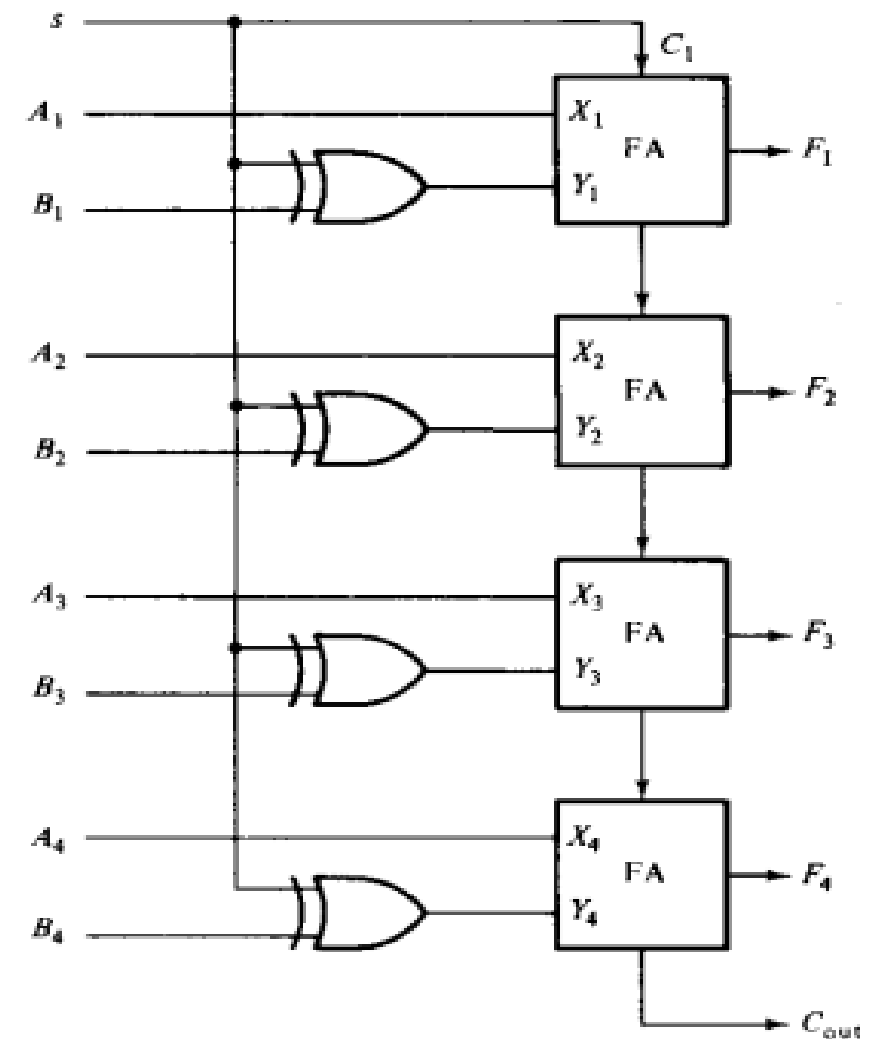
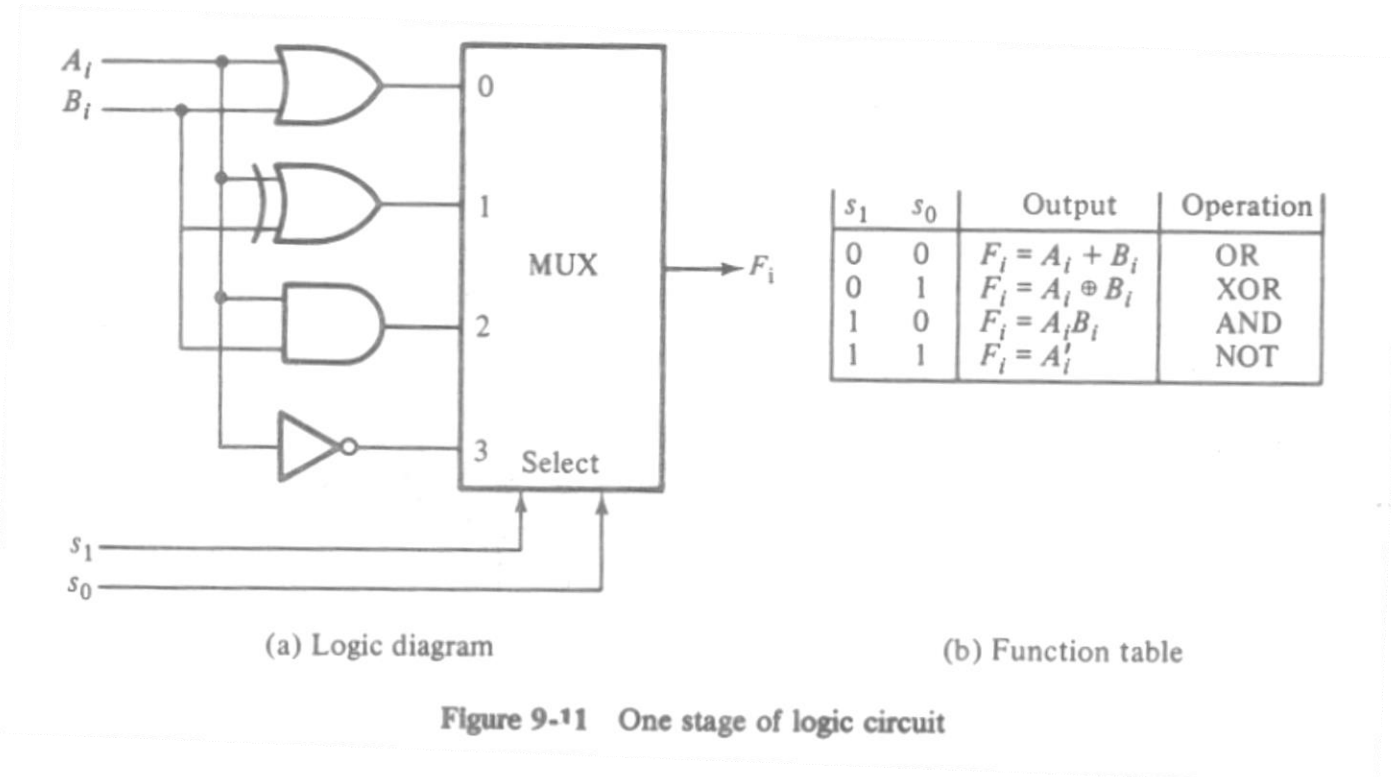


Figure 9-10 4-bit adder/subtractor circuit

Design of Logic Circuit

- The logic microoperations manipulate the bits of the operands separately and treat each bit as a binary variable.
- All logic operations can be obtained by means of **AND**, **OR** and **NOT** operations.
- For 3 operations, we need 2 selection variables but 2 selection lines can choose between 4 logic operations and hence XOR operation was also incorporated in the logical circuit.



Ref. – Figure 9.11 (M. Mano)

Combining logic and arithmetic Circuits

- The logic circuit can be combined with the arithmetic circuit to produce one **Arithmetic Logic Unit (ALU)**.
- Selection variables S_1 and S_0 can be made common to both sections, provided we use a third variable S_2 to differentiate between the two.

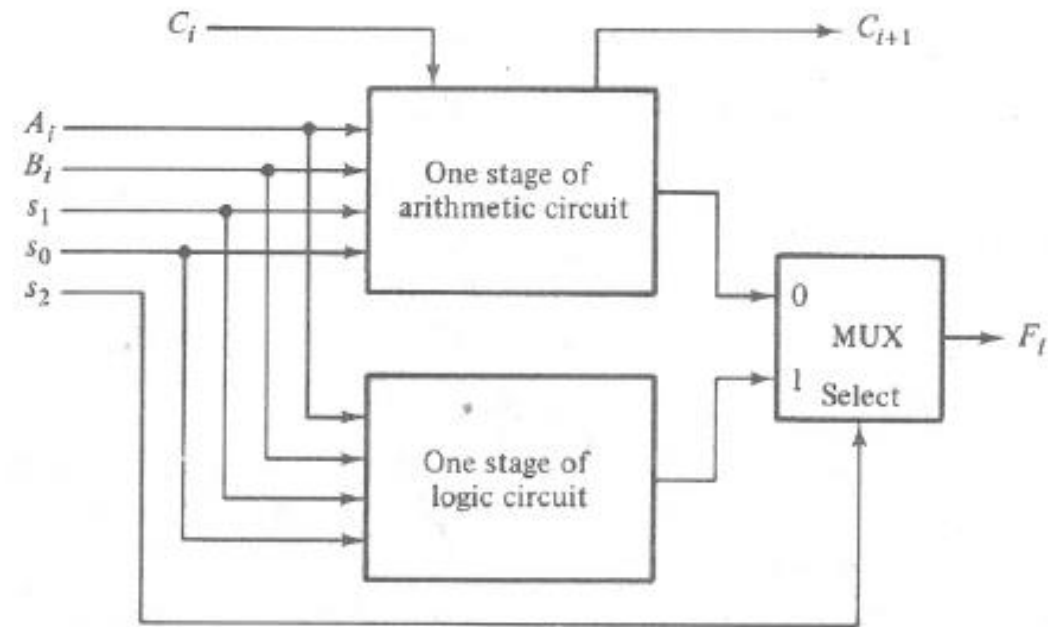


Fig : Combining Logic and arithmetic circuit

Logic Operations in one stage of Arithmetic Circuit



Table 1

S_2	S_1	S_0	X	Y	$Z=C_{in}$	$F = x \oplus y$	Operation
1	0	0	A+B	0	0	A OR B	OR
1	0	1	A	B	0	$A \oplus B$	XOR
1	1	0	$A + \bar{B}$	\bar{B}	0	A AND B	AND
1	1	1	A	1	0	\bar{A}	NOT

To perform **arithmetic operations** by the help of parallel adder, just perform **plain addition** among the input.

To perform **logical operations** by the help of a parallel adder just perform **XOR** operations among the inputs.

Rule

$$A \oplus 1 = \bar{A}$$

$$A \oplus 0 = A$$

$$\overline{A + B} = \bar{A} \cdot \bar{B}$$

$$\overline{A \cdot B} = \bar{A} + \bar{B}$$

Design of Arithmetic Logic Unit













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.

Functional Table for ALU

Binary Code	Function of selection variables					
	A	B	F with $C_{in} = 0$	F with $C_{in} = 1$	D	H
0 0 0	Input Data	Input Data	A	A+1	None	No Shift
0 0 1	R1	R1	A+B	A+B+1	R1	Shift Left with $I_L=0$
0 1 0	R2	R2	A-B-1	A-B	R2	Shift Right with $I_R=0$
0 1 1	R3	R3	A-1	A	R3	0's to the output Bus
1 0 0	R4	R4	A OR B	A OR B	R4	1's to the output Bus
1 0 1	R5	R5	A XOR B	A XOR B	R5	Circulate-Left with Carry
1 1 0	R6	R6	A AND B	A AND B	R6	Circulate-Right with Carry
1 1 1	R7	R7	A'	A'	R7	Circulate Left/ROL

Arithmetic Logic Circuit

TABLE 9-4 Function table for the ALU of Fig. 9-13

Selection				Output	Function		Xi	Yi	Zi
s_2	s_1	s_0	C_{in}						
0	0	0	0	$F = A$	Transfer A		A	0	0
0	0	0	1	$F = A + 1$	Increment A		A	0	1
0	0	1	0	$F = A + B$	Addition		A	B	0
0	0	1	1	$F = A + B + 1$	Add with carry		A	B	1
0	1	0	0	$F = A - B - 1$	Subtract with borrow		A	B'	0
0	1	0	1	$F = A - B$	Subtraction		A	B'	1
0	1	1	0	$F = A - 1$	Decrement A		A	1	0
0	1	1	1	$F = A$	Transfer A		A	1	1
1	0	0	X	$F = A \vee B$	OR		A+B	0	0
1	0	1	X	$F = A \oplus B$	XOR		A	B	0
1	1	0	X	$F = A \wedge B$	AND		A+B'	B'	0
1	1	1	X	$F = \bar{A}$	Complement A		A	1	0

Arithmetic Logic Circuit

KMAP for Xi

s_0 s_2s_1	0	1
00	A	A
01	A	A
11	$A+B'$	A
10	$A+B$	A

For Yi

s_0 s_2s_1	0	1
00	0	B
01	B'	1
11	B'	1
10	0	B

For Zi

s_0 s_2s_1	0	1
00	C_i	C_i
01	C_i	C_i
11	0	0
10	0	0

$$X_i = A_i + s_2s'_1s'_0B_i + s_2s_1s'_0B'_i$$

$$Y_i = s_0B_i + s_1B'_i$$

$$Z_i = s'_2C_i$$

Logic Diagram of ALU (2 bit) (Only first two stage are drawn)

$$X_i = A_i + s_2s'_1s'_0B_i + s_2s_1s'_0B'_i$$

$$Y_i = s_0B_i + s_1B'_i$$

$$Z_i = s'_2C_i$$

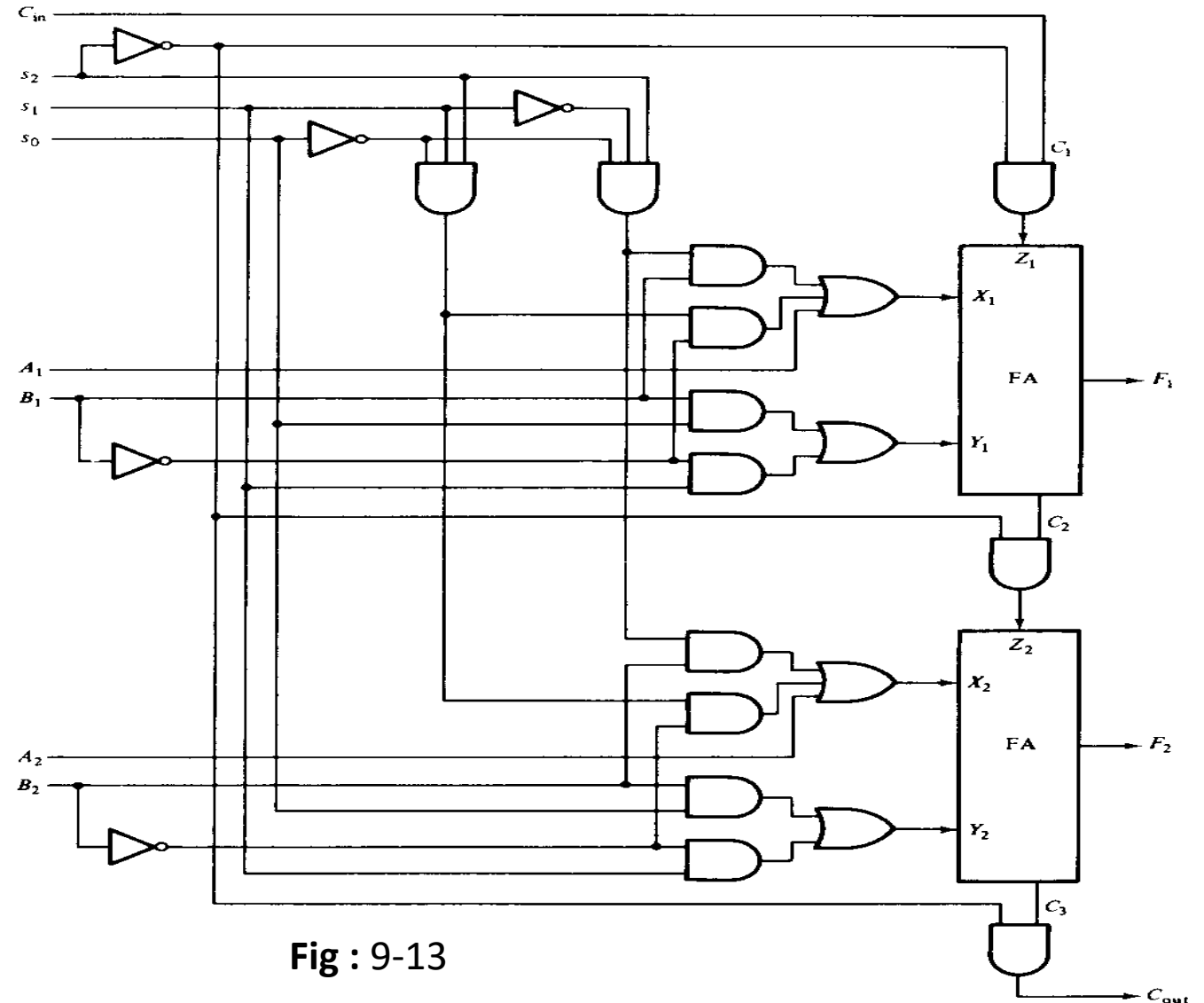


Fig : 9-13

The final ALU is shown in Fig. 9-13. Only the first two stages are drawn, but the diagram can be easily extended to more stages. The inputs to each full-adder circuit are specified by the Boolean functions:

$$X_i = A_i + s_2s'_1s'_0B_i + s_2s_1s'_0B'_i$$

$$Y_i = s_0B_i + s_1B'_i$$

$$Z_i = s'_2C_i$$

When $s_2 = 0$, the three functions reduce to:

$$X_i = A_i$$

$$Y_i = s_0B_i + s_1B'_i$$

$$Z_i = C_i$$

which are the functions for the arithmetic circuit of Fig. 9-8. The logic operations are generated when $s_2 = 1$. For $s_2s_1s_0 = 101$ or 111 , the functions reduce to:

$$X_i = A_i$$

$$Y_i = s_0B_i + s_1B'_i$$

$$C_i = 0$$

Output F_i is then equal to $X_i \oplus Y_i$ and produces the exclusive-OR and complement operations as specified in Table 9-3. When $s_2s_1s_0 = 100$, each A_i is ORed with B_i to provide the OR operation as discussed above. When $s_2s_1s_0 = 110$, each A_i is ORed with B'_i to provide the AND operation as explained previously.

Homework:

1. Design a 2-bit ALU for the operations listed in Table

Binary Code	Function of selection variables					
	A	B	F with $C_{in} = 0$	F with $C_{in} = 1$	D	H
0 0 0	Input Data	Input Data	A	A+1	None	No Shift
0 0 1	R1	R1	A+B	A+B+1	R1	Shift Left with $I_L=0$
0 1 0	R2	R2	A-B-1	A-B	R2	Shift Right with $I_R=0$
0 1 1	R3	R3	A-1	A	R3	0's to the output Bus
1 0 0	R4	R4	A'	A'	R4	1's to the output Bus
1 0 1	R5	R5	A XOR B	A XOR B	R5	Circulate-Left with Carry
1 1 0	R6	R6	A OR B	A OR B	R6	Circulate-Right with Carry
1 1 1	R7	R7	A AND B	A AND B	R7	Circulate Left/ROL

* [In Table 1, consider 'A' & 'B' as ALU source selection, 'F' as the ALU operation selection, 'D' as the destination selection and 'H' as shift operation selection variables]

HomeWork:

2. Design a 4-bit ALU for the operations listed in Table

Binary Code	Function of selection variables					
	A	B	F with $C_{in} = 0$	F with $C_{in} = 1$	D	H
0 0 0	Input Data	Input Data	$A+B$	$A+B+1$	None	High impedance to the output Bus
0 0 1	R1	R1	$A-B-1$	$A-B$	R1	Shift Right with $I_L=1$
0 1 0	R2	R2	A	$A+1$	R2	Shift Left with $I_R=0$
0 1 1	R3	R3	$A-1$	A	R3	-
1 0 0	R4	R4	$A \text{ XOR } 1$	$A \text{ XOR } 1$	R4	0's to the output Bus
1 0 1	R5	R5	$A \text{ OR } 0$	$A \text{ OR } 0$	R5	No Shift
1 1 0	R6	R6	$A \text{ AND } B$	$A \text{ AND } B$	R6	Circulate-Right with Carry
1 1 1	R7	R7	A'	A'	R7	1's to the output Bus

* [In Table 1, consider 'A' & 'B' as ALU source selection, 'F' as the ALU operation selection, 'D' as the destination selection and 'H' as shift operation selection variables]

Next

- Status Registers
- Shifter
- Design of processor Unit with control variables
- Micro operations for processor

End