Unit 1

Register Transfer and Micro-operations

SANTOSH MEENA

Register Transfer language (RTL)

- The operations executed on data stored in registers are called “micro-operations”.
- The symbolic notation used to describe the micro-operation transfers among registers is called a “register transfer language”.
- The term “register transfer” implies the availability of hardware logic circuits that can perform a stated micro-operation and transfer the result of the operation to the same or another register.
- Information transfer from one register to another is designated in symbolic form by means of a replacement operator. The statement R2←R1 denotes a transfer of the content of register R1 into register R2.
- It designates a replacement of the content of R2 by the content of R1. By definition, the content of the source register R1 does not change after the transfer.

Bus

- A typical digital computer has many registers, and paths must be provided to transfer information from one register to another.
- The number of wires will be excessive if separate lines are used between each register and all other registers in the system.
- A more efficient scheme for transferring information between registers in a multiple-register configuration is a common bus system.
- A bus structure consists of a set of common lines, one for each bit of a register, through which binary information is transferred one at a time.

Memory Transfers

- The transfer of information from a memory word to the outside environment is called a “read operation”.
- The transfer of new information to be stored into the memory is called a “write operation”.
- A memory word will be symbolized by the letter M.
- The particular memory word among the many available is selected by the memory address during the transfer.
- It is necessary to specify the address of M when writing memory transfer operations. This will be done by enclosing the address in square brackets following the letter M.

Continue…

- Consider a memory unit that receives the address from a register, called the address register, symbolized by AR.
- The data are transferred to another register, called the data register, symbolized by DR. The read operation can be stated as follows:

Read: DR ← M[AR]

This cause a transfer of information into DR from the memory word M selected by the address in AR.

- The write operation transfers the content of a data register to a memory word M selected by the address.
- Assume that the input data are in register R1 and the address is in AR. The write operation can be stated as follows:

Write: R1 → M[AR]

This cause a transfer of information from R1 into the memory word M selected by the address in AR.

Arithmetic Micro-operations

- A micro-operation is an elementary operation performed with the data stored in registers. The micro-operations most often encountered in digital computers are classified into four categories:
- **Registers transfer micro-operations** transfer binary information from one register to another register.
- **Arithmetic micro-operations** perform arithmetic operation on numeric data stored in registers.
- **Logic micro-operations** perform bit manipulation operations on nonnumeric data stored in registers.
- **Shift micro-operations** perform shift operations on data stored in registers.
- The register transfer micro-operation does not change the information content when the binary information moves from the source register to the destination register.
- The other three types of micro-operations change the information content during the transfer.

Continue…

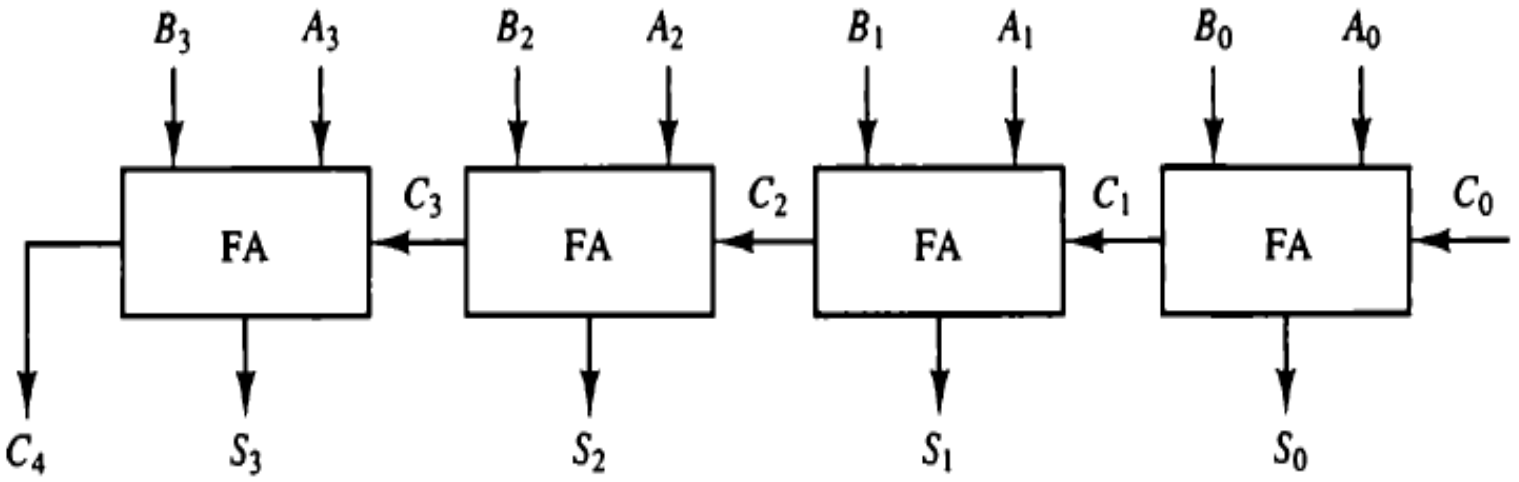
- The basic arithmetic micro-operations are addition, subtraction, increment, decrement, and shift. In most computers, the multiplication operation is implemented with a sequence of add and shift micro-operations. Division is implemented with a sequence of subtract and shift micro-operations.

Arithmetic Microoperations

Symbolic designation	Description
$R3 \leftarrow R1 + R2$	Contents of $R1$ plus $R2$ transferred to $R3$
$R3 \leftarrow R1 - R2$	Contents of $R1$ minus $R2$ transferred to $R3$
$R2 \leftarrow \overline{R2}$	Complement the contents of $R2$ (1's complement)
$R2 \leftarrow \overline{R2} + 1$	2's complement the contents of $R2$ (negate)
$R3 \leftarrow R1 + \overline{R2} + 1$	$R1$ plus the 2's complement of $R2$ (subtraction)
$R1 \leftarrow R1 + 1$	Increment the contents of $R1$ by one
$R1 \leftarrow R1 - 1$	Decrement the contents of $R1$ by one

Continue…

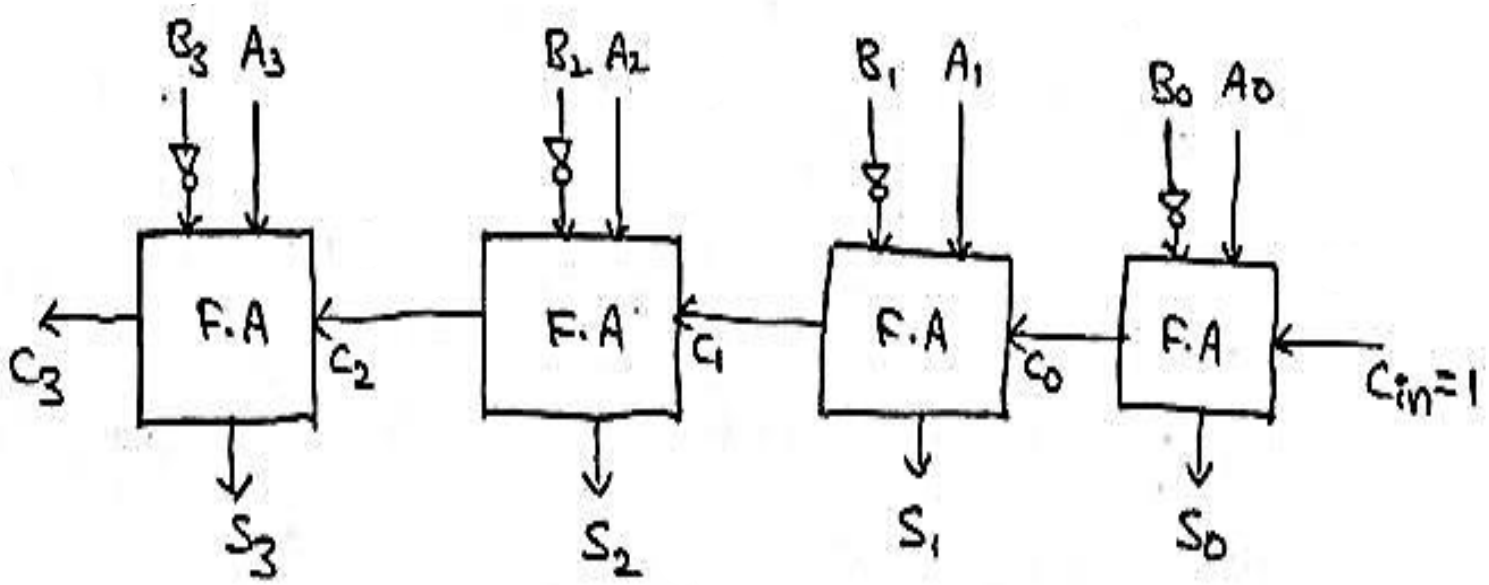
- **Binary Adder:** To implement the add micro-operation with hardware, we need the registers that hold the data and the digital component that performs the arithmetic addition.
- The digital circuit that forms the arithmetic sum of two bits and a previous carry is called a full-adder. The digital circuit that generates the arithmetic sum of two binary numbers of any length is called a binary adder.



4-bit binary adder

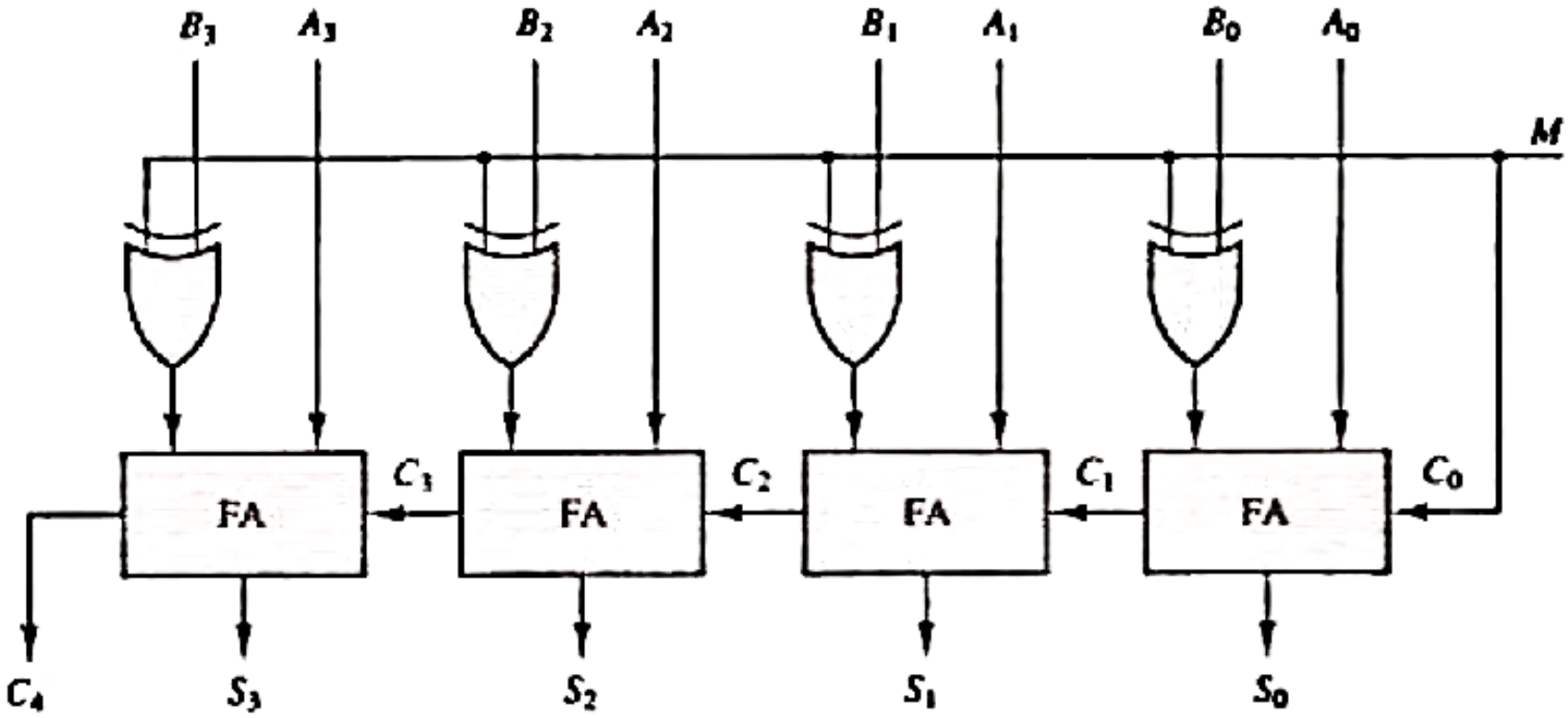
Continue…

- **Binary Subtractor:** The subtraction A-B can be done by taking the 2's complement of B and adding it to A.
- The 1's complement can be implemented with inverters and a one can be added to the sum through the input carry.



Continue…

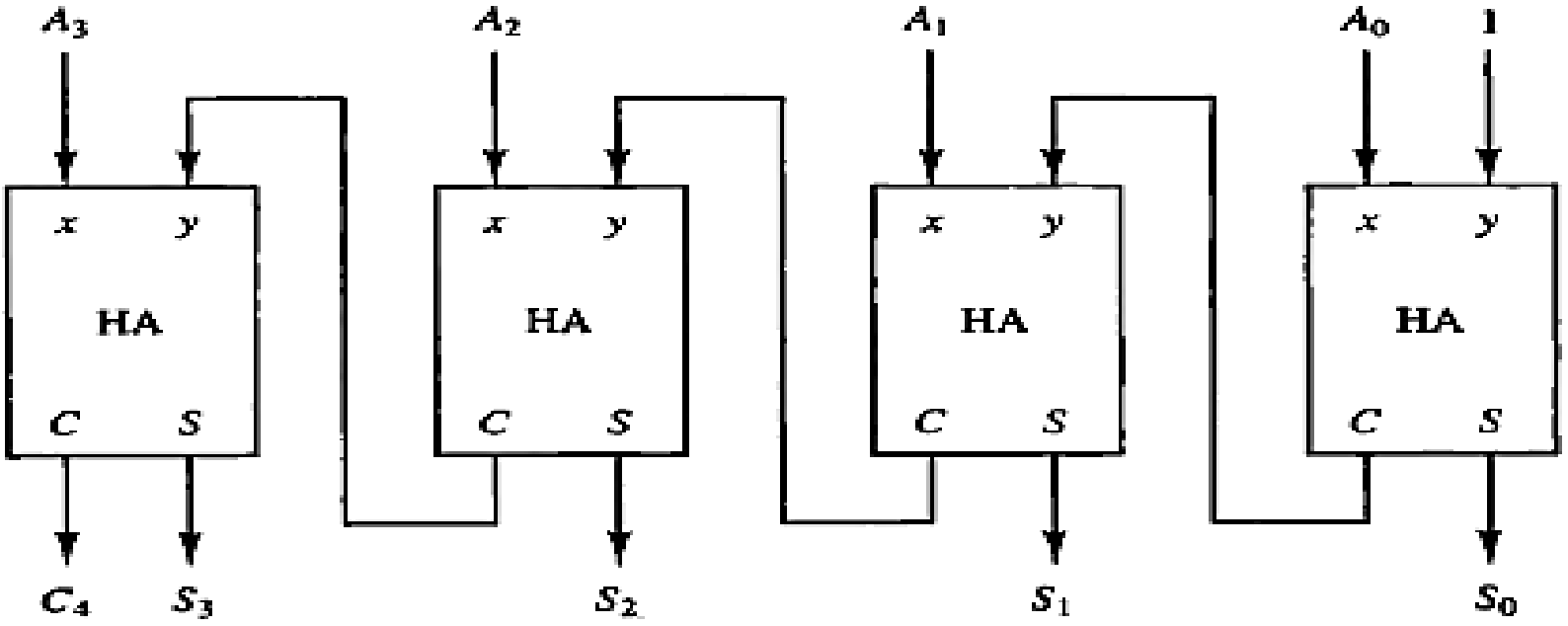
- **Binary Adder-Subtractor:** The mode input M controls the operation. When M=0, B⊕0=B, C_0=0 and the circuit is an adder (A plus B) and when M=1, B⊕1=B', C_0=1 and the circuit becomes a subtractor (A plus the 2's complement of B).



4-bit adder-subtractor.

Continue…

- **Binary Incrementer:** The increment micro-operation adds one to a number in a register. For example, if a 4-bit register has a binary value 0110, it will go to 0111 after it is incremented.



4-bit binary incrementer

Logic Micro-operations

- Logic micro-operations specify binary operations for string of bits stored in registers. These operations consider each bit of the register separately and treat them as binary variables.
- The logic micro-operations are seldom used in scientific computations, but they are very useful for manipulating individual bits or a portion of a word stored in a register.
- They can be used to change bit values, delete a group of bits, or insert new bit values into a register.
- There are 16 different logic operations (F_0 to F_{15}) that can be performed with two binary variables (x, y).

Truth Tables for 16 Functions of Two Variables

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Continue…

- The 16 different logic micro-operations are derived from these functions by replacing variable ‘x’ by the binary content of register ‘A’ and variable ‘y’ by the binary content of register ‘B’.

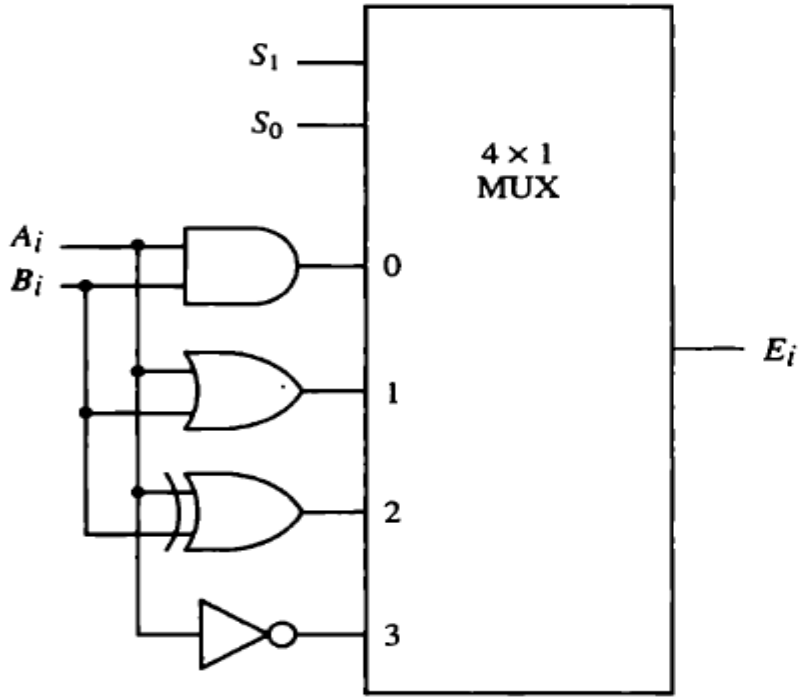
Sixteen Logic Microoperations

Boolean function	Microoperation	Name
$F_0 = 0$	$F \leftarrow 0$	Clear
$F_1 = xy$	$F \leftarrow A \wedge B$	AND
$F_2 = xy'$	$F \leftarrow A \wedge \overline{B}$	
$F_3 = x$	$F \leftarrow A$	Transfer A
$F_4 = x'y$	$F \leftarrow \overline{A} \wedge B$	
$F_5 = y$	$F \leftarrow B$	Transfer B
$F_6 = x \oplus y$	$F \leftarrow A \oplus B$	Exclusive-OR
$F_7 = x + y$	$F \leftarrow A \vee B$	OR
$F_8 = (x + y)'$	$F \leftarrow \overline{A \vee B}$	NOR
$F_9 = (x \oplus y)'$	$F \leftarrow \overline{A \oplus B}$	Exclusive-NOR
$F_{10} = y'$	$F \leftarrow \overline{B}$	Complement B
$F_{11} = x + y'$	$F \leftarrow A \vee \overline{B}$	
$F_{12} = x'$	$F \leftarrow \overline{A}$	Complement A
$F_{13} = x' + y$	$F \leftarrow \overline{A} \vee B$	
$F_{14} = (xy)'$	$F \leftarrow \overline{A \wedge B}$	NAND
$F_{15} = 1$	$F \leftarrow$ all 1’s	Set to all 1’s

Continue…

- **Hardware Implementation:** The hardware implementation of logic micro-operations requires that logic gates be inserted for each bit or pair of bits in the registers to perform the required logic function.
- Although there are 16 different logic micro-operations, most computers use only four-AND, OR, XOR, and complement from which all others can be derived.

One stage of logic circuit.



(a) Logic diagram

S_1	S_0	Output	Operation
0	0	$E = A \wedge B$	AND
0	1	$E = A \vee B$	OR
1	0	$E = A \oplus B$	XOR
1	1	$E = \overline{A}$	Complement

(b) Function table

Shift Micro-operations

- Shift micro-operations are used for serial transfer of data. They are also used in conjunction with arithmetic, logic, and other data-processing operations.
- The content of register can be shifted to the left or the right. At the same time that the bits are shifted, the first flip-flop receives its binary information from the serial input.
- There are three types of shifts: logical, circular, and arithmetic.

Shift Microoperations

Symbolic designation	Description
$R \leftarrow \text{shl } R$	Shift-left register R
$R \leftarrow \text{shr } R$	Shift-right register R
$R \leftarrow \text{cil } R$	Circular shift-left register R
$R \leftarrow \text{cir } R$	Circular shift-right register R
$R \leftarrow \text{ashl } R$	Arithmetic shift-left R
$R \leftarrow \text{ashr } R$	Arithmetic shift-right R

Continue…

- **Logical shift:** A logical shift is one that transfers 0 through the serial input.
- The symbols *shl* and *shr* are used for logical shift-left and logical shift-right micro-operations.
- For example:

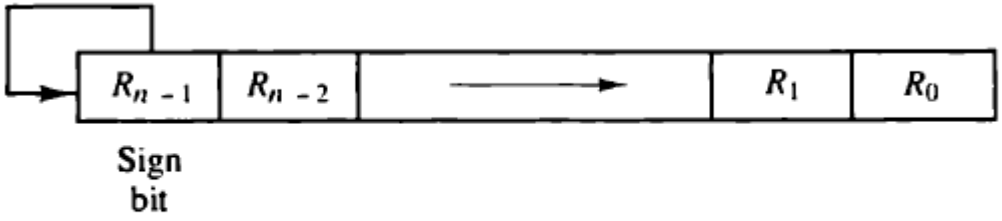
  $R1 \leftarrow shl\ R1$

  $R2 \leftarrow shr\ R2$

  are two micro-operations that specify a 1-bit shift to the left of the content of register R1 and a 1-bit shift to the right of the content of register R2.
- **Circular shift:** The circular shift (also known as a rotate operation) circulates the bits of the register around the two ends without loss of information.
- This is accomplished by connecting the serial output of the shift register to its serial input. The symbols *cil* and *cir* are used for the circular shift-left and circular shift-right, respectively.

Continue…

- **Arithmetic shift:** An arithmetic shift is a micro-operation that shifts a signed binary number to the left or right.
- An arithmetic shift-left multiplies a signed binary number by 2. An arithmetic shift-right divides the number by 2.
- Arithmetic shifts must leave the sign bit unchanged because the sign of the number remains the same when it is multiplied or divided by 2.
- The leftmost bit in a register holds the sign bit, and the remaining bits hold the number. The sign bit is 0 for positive and 1 for negative. Negative numbers are in 2's complement form.
- The arithmetic shift-right leaves the sign bit unchanged and shifts the number (including the sign bit) to the right. The bit in R_0 is lost.



Arithmetic shift right

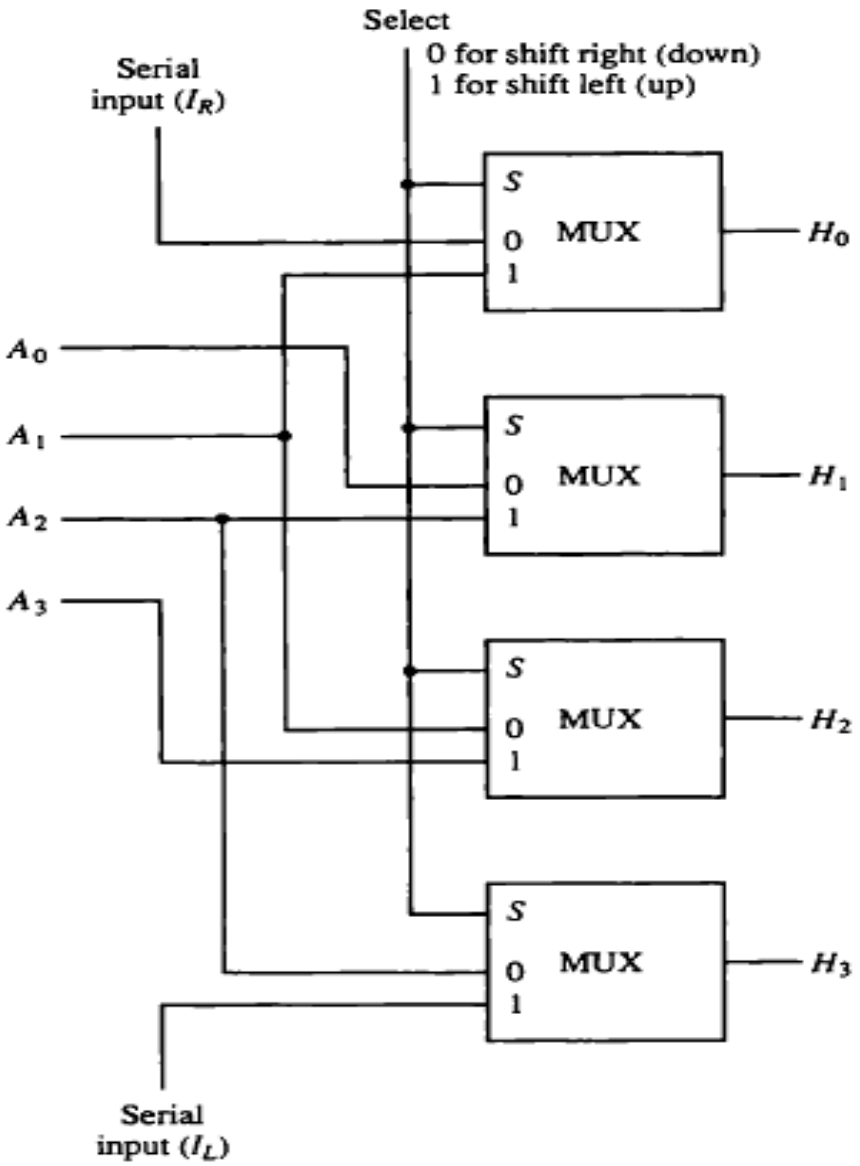
Continue…

- The arithmetic shift-left inserts a 0 into R_0, and shifts all other bits to the left. The initial bit of R_{n-1} is lost and replaced by the bit from R_{n-2}.
- A sign reversal occurs if the bit in R_{n-1} changes in value after the shift. This happens if multiplication by 2 cause an overflow.
- An overflow occurs after an arithmetic shift if initially, before the shift, R_{n-1} is not equal to R_{n-2}. An overflow flip-flop Vs can be used to detect an arithmetic shift-left overflow.

$$Vs = R_{n-1} \oplus R_{n-2}$$

- If Vs = 0 (R_{n-1} is equal to R_{n-2}), there is no overflow, but if Vs = 1 (R_{n-1} is not equal to R_{n-2}), there is an overflow and a sign reversal after the shift.
- **Hardware Implementation:** The information can be transferred to the register in parallel and then shifted to the right or left. The 4-bit shifter has four data inputs, A0 through A3, and four data outputs, H0 through H3.
- There are two serial inputs, one for shift left (IL) and the other for shift right (IR).

Continue…



Function table

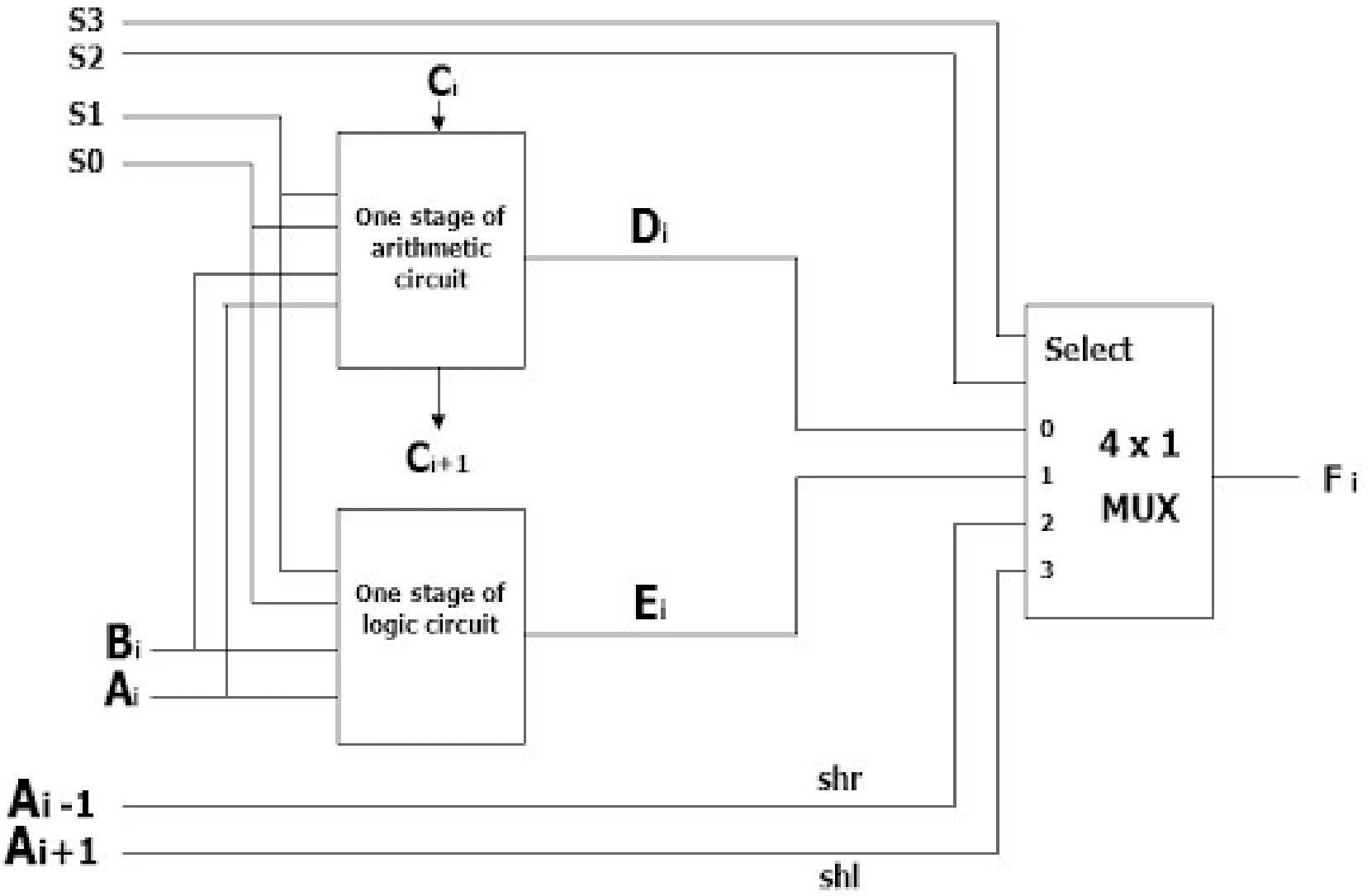
Select	Output			
S	H_0	H_1	H_2	H_3
0	I_R	A_0	A_1	A_2
1	A_1	A_2	A_3	I_L

4-bit combinational circuit shifter

Arithmetic Logic Shift Unit

- Instead of having individual registers performing the micro-operation directly, computer systems employ a number of storage registers connected to a common operational unit called an arithmetic logic unit (ALU).
- To perform a micro-operation, the contents of specified registers are placed in the inputs of the common ALU. The ALU performs an operation and the result of the operation is then transferred to a destination register.
- Inputs Ai and Bi are applied to both the arithmetic and logic units. A particular micro-operation is selected with inputs S1 and S0.
- A 4x1 multiplexer at the output chooses between an arithmetic output in Ei and a logic output in Di.
- The data in the multiplexer are selected with inputs S3 and S2. The other two data inputs to the multiplexer receive inputs A_{i-1} for the shift-right operation and A_{i+1} for the shift-left operation.
- The output carry C_{i+1} of a given arithmetic stage must be connected to the input carry of the next stage in sequence. The input carry to the first stage C_{in} provides a selection variable only for the arithmetic operations.
- The circuit provides eight arithmetic operations, four logic operations, and two shift operations.

Continue…


S3
S2
S1
S0
Ci
One stage of arithmetic circuit
Di
Ci+1
One stage of logic circuit
Ei
Bi
Ai
Select
0
1
2
3
4 x 1
MUX
Fi
shr
Ai -1
Ai+1
shl

Continue…

Function Table for Arithmetic Logic Shift Unit

	Operation select					Operation	Function
	S_3	S_2	S_1	S_0	C_{in}		
AO	0	0	0	0	0	$F = A$	Transfer A
	0	0	0	0	1	$F = A + 1$	Increment A
	0	0	0	1	0	$F = A + B$	Addition
	0	0	0	1	1	$F = A + B + 1$	Add with carry
	0	0	1	0	0	$F = A + \overline{B}$	Subtract with borrow
	0	0	1	0	1	$F = A + \overline{B} + 1$	Subtraction
	0	0	1	1	0	$F = A - 1$	Decrement A
	0	0	1	1	1	$F = A$	Transfer A
LO	0	1	0	0	×	$F = A \wedge B$	AND
	0	1	0	1	×	$F = A \vee B$	OR
	0	1	1	0	×	$F = A \oplus B$	XOR
	0	1	1	1	×	$F = \overline{A}$	Complement A
SO	1	0	×	×	×	$F =$ shr A	Shift right A into F
	1	1	×	×	×	$F =$ shl A	Shift left A into F