Computer System Architecture
COMP201Th
Unit: 2
Basic Computer Organization and Design

Lecture: 6

Logical Micro-operations

Logic micro-operations specify binary operations for strings of bits stored in registers. These operations consider each bit of the register separately and treat them as binary variables e.g. the exclusive – OR micro operation with the contents of two registers R1 and R2 is symbolized by the statement:

P: R1 $\leftarrow$ R1 $\oplus$ R2

It specifies a logic micro-operation to be executed on the individual bits of the registers provided that the control variable P=1.

e.g. let the content of R1 be 1010 and the content of R2 be 1100. The exclusive-OR micro-operation stated above symbolized the following logic computation:

1010 Content of R1

1100 Content of R2

0110 Content of R1 after P =1.

There are 16 different logic operations that can be performed with two binary variables.

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

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 logic micro-operations, most computers use only four- AND, OR, XOR (exclusive-OR) and complement ; from which all others can be derived.

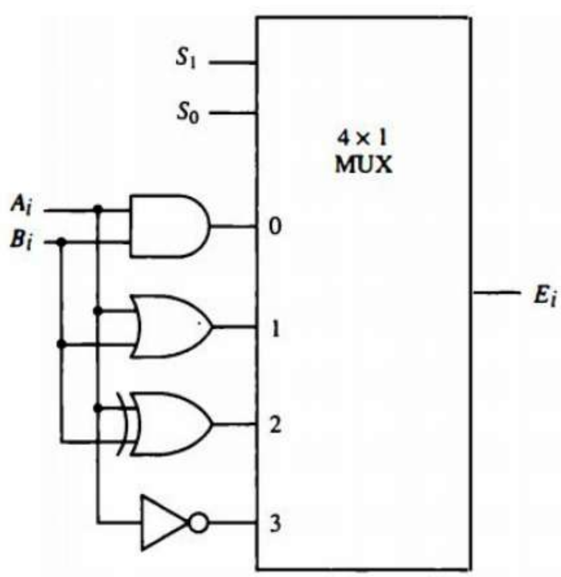
Hardware Implementation:

Fig below shows one stage of a circuit that generates the four basic logic microoperations.

- It consists of four gates and a multiplexer.
- Each of the four logic operations is generated through a gate that performs the required logic.
- The outputs of the gates are applied to the data inputs of the multiplexer.
- The two selection inputs S1 and S0 choose one of the data inputs of the multiplexer and directs its value to the output.

The diagram shows one typical stage. For a logic circuit with n bits the diagram must be repeated n times.

Figure 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

Logic Diagram of Hardware Implementation of Logic Circuit

Applications of Logic Microoperations:

1. **Selective Set Operation** - It sets 1 to the bits in register A where there are corresponding 1's in register B.It does not affect bit positions that have 0's in B. The OR microoperation can be used to selectively set bits of a register.

   Example:

   consider register A contains 1010 and register B contains 1100. The bits in A corresponding to the bit 1 in the register B will be changed to 1. Therefore, bits 1 and 0 in the register A corresponding to the bits 1 and 1 in the register B will be changed to 1 and 1.

1 0 1 0.	A before
1 1 0 0	B(logic operand)
1 1 1 0	A after

2. **Selective Complement Operation**

   The selective Complement operation complements bits in A where there are corresponding 1's in B. It does not affect bit positions that have 0's in B. The exclusive OR microoperation can be used to selectively set bits of a register.

   Example:

1 0 1 0.	A before
1 1 0 0	B(logic operand)
0 1 1 0	A after

3. **Selective Clear Operation**

   The selective clear operation clears to 0 the bits in A where there are corresponding 1's in B. It does not affect bit positions that have 0's in B. The selective clear operation can be achieved by the microoperation $A \leftarrow A \wedge B'$

   Example:

1 0 1 0.	A before
1 1 0 0	B(logic operand)
0 0 1 0	A after

4. **Mask operation** - It is similar to selective clear operation except that the bits of A are cleared only where there are corresponding 0's in B.

   The mask operation is an AND Micro Operation.

   Example:

1 0 1 0.	A before
1 1 0 0	B(logic operand)
1 0 0 0	A after