Computer Organization and Architecture

Lecture notes

SHAMBHUNATH
Group of Institutions
... Shaping the future

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Prepared by: Rajan Mani Tripathi

Computer Organization and Architecture (KCS302)

Course Outcome (CO) Bloom's Knowledge Level (KL)

At the end of course, the student will be able to understand

CO 1

Study of the basic structure and operation of a digital computer system.

CO 2

Analysis of the design of arithmetic & logic unit and understanding of the fixed point and floating-point arithmetic operations.

CO 3

Implementation of control unit techniques and the concept of Pipelining

CO 4

Understanding the hierarchical memory system, cache memories and virtual memory

CO 5

Understanding the different ways of communicating with I/O devices and standard I/O interfaces.

DETAILED SYLLABUS

Unit-1

Introduction: Functional units of digital system and their interconnections, buses, bus architecture, types of buses and bus arbitration. Register, bus and memory transfer. Processor organization, general registers organization, stack organization and addressing mode.

Unit-2

Arithmetic and logic unit: Look ahead carries adders. Multiplication: Signed operand multiplication, Booths algorithm and array multiplier. Division and logic operations. Floating points arithmetic operation, Arithmetic & logic unit design. IEEE Standard for Floating Point Numbers

Unit-3

Control Unit: Instruction types, formats, instruction cycles and sub cycles (fetch and execute etc.), micro operations, execution of a complete instruction. Program Control, Reduced Instruction Set Computer, Pipelining. Hardwire and micro programmed control: micro programmed sequencing, concept of horizontal and vertical microprogramming.

Unit-4

Memory: Basic concept and hierarchy, semiconductor RAM organization. ROM memories. Cache memories: concept and desi mapping and replacement Auxiliary memories: magnetic disk, Virtual memory: concept implementation.

Unit-5

Input / Output: Peripheral devices, I/O interface, I/O ports, Interrupt type of interrupt and exceptions. Modes of Data Transfer: Program Direct Memory Access., I/O channels and processors. Serial asynchronous communication, standard communication interfaces.

Computer architecture refers to those parameters of a computer system that are visible to a programmer or those parameters that have a direct impact on the logical execution of a program. Examples of architectural attributes include the instruction set, the number of bits used to represent different data types, I/O mechanisms, and techniques for addressing memory.

Computer organization refers to the operational units and their interconnections that realize the architectural specifications. Examples of organizational attributes include those hardware details transparent to the programmer, such as control signals, interfaces between the computer and peripherals, and the memory technology used.

Basic Computer Model and different units of Computer

The model of a computer can be described by four basic units in high level abstraction. These basic units are:

- ❍ Central Processor Unit
- ❍ Input Unit
- ❍ Output Unit
- ❍ Memory Unit

Basic Computer Model and different units of Computer

A. Central Processor Unit [CPU]:

Central processor unit consists of two basic blocks:

- ❍ The program control unit has a set of registers and control circuit to generate control signals.
- ❍ The execution unit or data processing unit contains a set of registers for storing data and an Arithmetic and Logic Unit (ALU) for execution of arithmetic and logical operations.

In addition, CPU may have some additional registers for temporary storage of data.

B. Input Unit:

With the help of input unit data from outside can be supplied to the computer. Program or data is read into main storage from input device or secondary storage under the control of CPU input instruction.

Example of input devices: Keyboard, Mouse, Hard disk, Floppy disk, CD-ROM drive etc.

C. Output Unit:

With the help of output unit computer results can be provided to the user or it can be stored in storage device permanently for future use. Output data from main storage go to output device under the control of CPU output instructions.

Example of output devices: Printer, Monitor, Plotter, Hard Disk, Floppy Disk etc.

D. Memory Unit :

Memory unit is used to store the data and program. CPU can work with the information stored in memory unit. This memory unit is termed as primary memory or main memory module. These are basically semi conductor memories.

There are two types of semiconductor memories –

● Volatile Memory: RAM (Random Access Memory).

● Non-Volatile Memory: ROM (Read only Memory), PROM (Programmable ROM) EPROM (Erasable PROM), EEPROM (Electrically Erasable PROM).

Secondary Memory:

There is another kind of storage device, apart from primary or main memory, which is known as secondary memory.

Secondary memories are non volatile memory and it is used for permanent storage of data and program.

Example of secondary memories: Hard Disk, Floppy Disk,

Magnetic Tape ------ These are magnetic devices,

CD-ROM ------ is optical device

Thumb drive (or pen drive) ------ is semiconductor memory.

BUS:- A typical digital computer has many registers, and paths must be provided to transfer information from one register to another. 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. Control signals

Type of Bus: There are three types of buses

1.Data bus:

It carries the data from one system module to other. Data bus may consist of 32,64, 128 or even more numbers of separate lines. This number of lines decides the width of the data bus. Each line can carry one bit at a time. So, a data bus with 32 lines can carry 32 bits of data at a time. If a processor needs to read 64 bits of data from memory, the n the processor must access the memory twice. it is a bidirectional bus.

2. Address Bus:

It is used to carry the address of source or destination of the data on the data bus. Address bus may consist of 16,20, 24 or even more numbers of separate lines.it is a unidirectional bus.

3. Control Bus:

It is used to control the access, processing and information transferring. As either the exclusive data and address lines are shared by all components or common paths may create congestion in traffic, there must be a separate and dedicated path for control signal transfer.

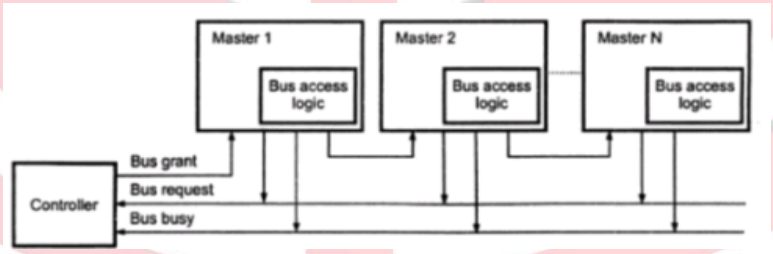
Bus Arbitration?

- A device that initiates data transfers on the bus at any given time is called a bus master.
- In a computer system, there may be more than one bus master such as a DMA controller or a processor etc.
- These devices share the system bus and when a current master bus relinquishes another bus can acquire the control of the processor.
- Bus arbitration is a process by which next device becomes the bus controller by transferring bus mastership to another bus.

There are three arbitration schemes which run on centralized arbitration.

Daisy Chaining − It is a simple and cheaper method where all the masters use the same line for making bus requests.

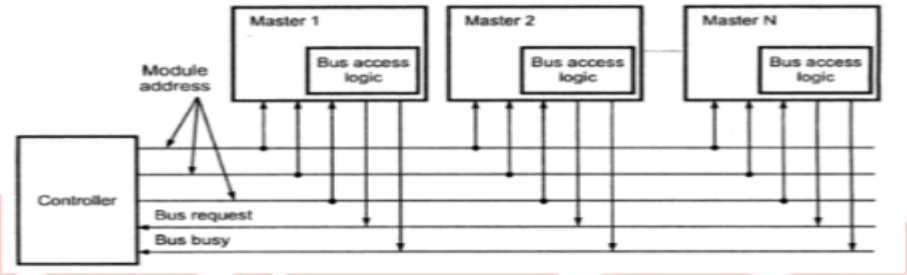
- It is simple and cheaper method. All masters make use of the same line for bus request.
- In response to the bus request the controller sends a bus grant if the bus is free.
- The bus grant signal serially propagates through each master until it encounters the first one that is requesting access to the bus. This master blocks the propagation of the bus grant signal, activities the busy line and gains control of the bus.
- Therefore any other requesting module will not receive the grant signal and hence cannot get the bus access.



b) Polling Method − In this method, the controller is used to generate address lines for the master. For example, if there are 8 masters connected in a system at least 3 address lines are required.

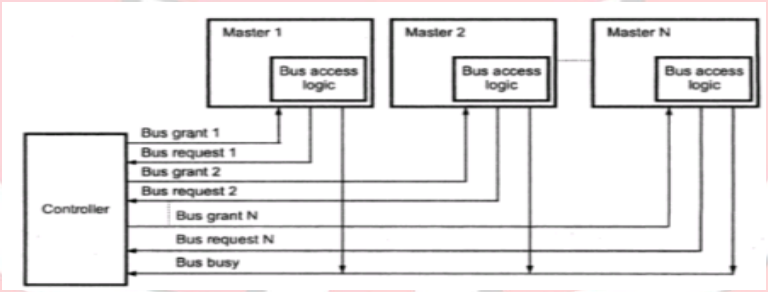
- In this the controller is used to generate the addresses for the master. Number of address line required depends on the number of masters connected in the system.
- For example, if there are 8 masters connected in the system, at least three address lines are required.

- In response to the bus request controller generates a sequence of master address. When the requesting master recognizes its address, it activated the busy line ad begins to use the bus.



c) Independent Request − In this scheme, each bus has its own bus request and a grant. The built-in priority decoder selects the highest priority requests and asserts the system.

- The system connections for polling method are shown in figure above.
- In this the controller is used to generate the addresses for the master. Number of address line required depends on the number of masters connected in the system.
- For example, if there are 8 masters connected in the system, at least three address lines are required.
- In response to the bus request controller generates a sequence of master address. When the requesting master recognizes its address, it activated the busy line ad begins to use the bus.



Microoperation: - The operations executed on data stored in register & are called microoperations. A microoperations is an elementary operation performed on the information stored in one or more registers. The result of the operation may replace the previous binary information of a register or may be transferred to another register. Examples of microoperations are shift, count, clear, and load.

The internal hardware organization of a digital computer is best defined by specifying

1.The set of registers it contains and their function.

2.The sequence of microoperations performed on the binary information stored in the registers.

3.The control that initiates the sequence of microoperations.

Register Transfer Language: -

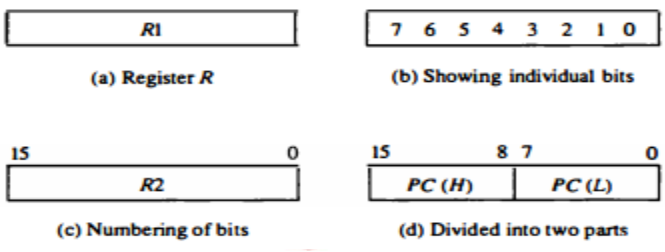
- The symbolic notation used to describe the microoperation transfers among registers is called a register transfer language.
- A register transfer language is a system for expressing in symbolic form the microoperation sequences among the registers of a digital module.
- It is a convenient tool for describing the internal organization of digital computers in concise and precise manner. It can also be used to facilitate the design process of digital systems.

Register: -

- Computer registers are designated by capital letters (sometimes followed by numerals) to denote the function of the register.
- For example: The register that holds an address for the memory unit is usually called a memory address register and is designated by the name MAR.
- Other designations for registers are PC (for program counter), IR (for instruction register, and R1 for processor register).

Register Transfer: -

- 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 Rl. By definition, the content of the source register R1 does not change after the transfer.



- Normally, we want the transfer to occur only under a predetermined control condition. This can be shown by means of an if-then statement.

If (P = 1) then (R2 <--R1)

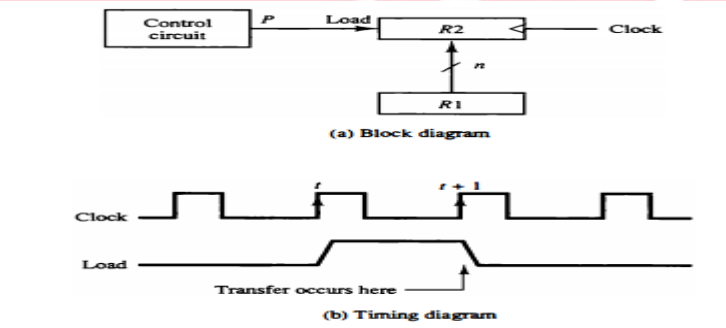
- where P is a control signal generated in the control section. It is sometimes convenient to separate the control variables from the register transfer operation by specifying a control function.

Control Function

- A control function is a Boolean variable that is equal to 1 or 0. The control function is included in the statement as follows:

P: R2 <--R1

- The control condition is terminated with a colon. It symbolizes the requirement that the transfer operation be executed by the hardware only if P = 1. Every statement written in a register transfer notation implies a hardware construction for implementing the transfer.

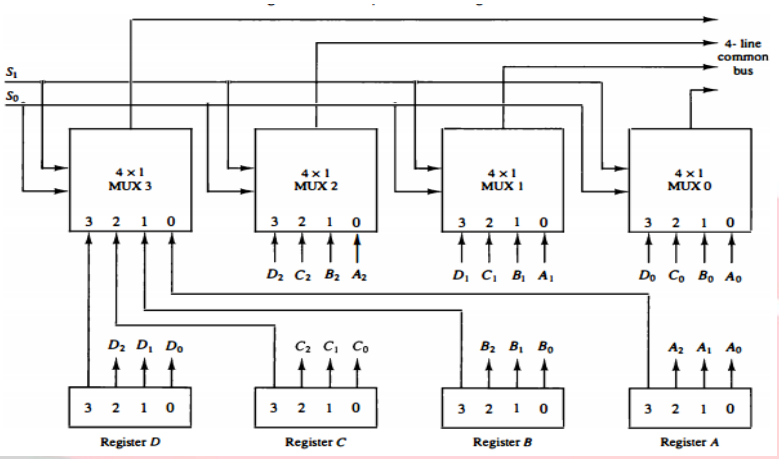


- In the timing diagram, P is activated in the control section by the rising edge of a clock pulse at time t. The next positive transition of the clock at time t + 1 finds the load input active and the data inputs of R2 are then loaded into the register in parallel. P may go back to 0 at time t + 1; otherwise, the transfer will occur with every clock pulse transition while P remains active.

Bus and Memory Transfers: -

- 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.
- One way of constructing a common bus system is with multiplexers. The multiplexers select the source register whose binary information is then placed on the bus.
- Each register has four bits, numbered 0 through 3. The bus consists of four 4 x 1 multiplexers each having four data inputs, 0 through 3, and two selection inputs, S1 and S0.
- In order not to complicate the diagram with 16 lines crossing each other, we use labels to show the connections from the outputs of the registers to the inputs of the multiplexers.

For example, output 1 of register A is connected to input 0 of MUX 1 because this input is labeled A1. The diagram shows that the bits in the same significant position in each register are connected to the data inputs of one multiplexer to form one line of the bus. Thus MUX 0 multiplexes the four 0 bits of the registers, MUX 1 multiplexes the four 1 bits of the registers, and similarly for the other two bits.



Bus Selection:

The two selection lines S1 and S0 are connected to the selection inputs of all four multiplexers. The selection lines choose the four bits of one register and transfer them into the four-line common bus. When S1S0 = 00, the 0 data inputs of all four multiplexers are selected and applied to the outputs that form the bus. This causes the bus lines to receive the content of register A since the outputs of this register are connected to the 0 data inputs of the multiplexers. Similarly, register B is selected if S1S0 = 01, and so on. Table 4-2 shows the register that is selected by the bus for each of the four possible binary value of the selection lines.

S_1	S_0	Register selected
0	0	*A*
0	1	*B*
1	0	*C*
1	1	*D*

Note: - In general, a bus system will multiplex k registers of n bits each to produce an n-line common bus. The number of multiplexers needed to construct the bus is equal ton, the number of bits in each register. The size of each multi-plexer must be k x 1 since it multiplexes k data lines. For example, a common bus for eight registers of 16 bits each requires 16 multiplexers, one for each line in the bus. Each multiplexer must have eight data input lines and three selection lines to multiplex one significant bit in the eight registers.

Three State Bus Buffers: -

Three-State Bus Buffers A bus system can be constructed with three-state gates instead of multiplexers.

A three-state gate is a digital circuit that exhibits three states. Two of the states are signals equivalent to logic 1 and 0 as in a conventional gate.

The third state is a high-impedance state. **The high-impedance** state behaves like an open circuit, which means that the output is disconnected and does not have a logic signif-icance.

Three-state gates may perform any conventional logic, such as AND or NAND. However, the one most commonly used in the design of a bus system is the buffer gate.

Bus System: - The construction of a bus system with three-state buffers is demonstrated in Fig. 4-5. The outputs of four buffers are connected together to form a single bus line. (It must be realized that this type of connection cannot be done with gates that do not have three-state outputs.) The control inputs to the buffers determine which of the four normal inputs will communicate with the bus line. No more than one buffer may be in the active state at any given time. The connected buffers must be controlled so that only one three-state buffer has access to the bus line while all other buffers are maintained in a high impedance state.

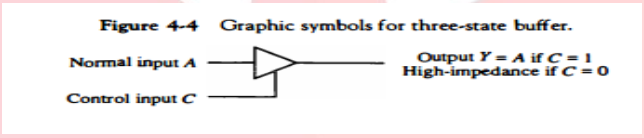


Figure 4-4 Graphic symbols for three-state buffer.

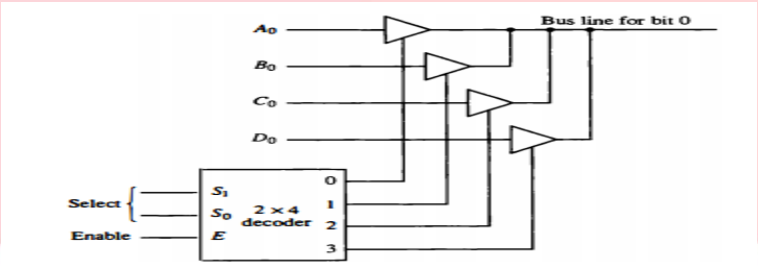


Figure 4-5 Bus line with three state-buffers.

Memory Transfer

Memory read: - The transfer of information from a memory word to the outside environment is called a read operation.

Memory Write: - The transfer of new information to be stored into the memory is called a write operation.

Arithmetic Microoperation: - A Microoperation is an elementary operation performed with the data stored in registers. The microoperations most often encountered in digital computer are classified into four categories

1.Register transfer microoperations transfer binary information from one register to another.

2.Arithmetic microoperations perform arithmetic operation on numeric data stored in registers.
3.Logic microoperations perform bit manipulation operations on nonnumeric data stored in registers.
4.Shift microoperations perform shift operations on data stored in registers.

The basic arithmetic microoperations are addition, subtraction, increment decrement, and shift. Arithmetic shifts are explained later in conjunction with the shift microoperations. The arithmetic microoperation defined by the statement

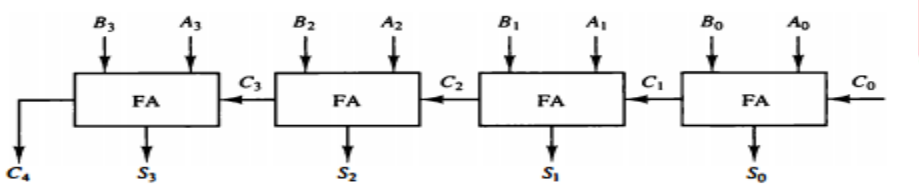
R3← R1+R2

Specifies an add microoperation. It states that the contents of register R1 are added to the contents of register R2 and the sum transferred to register R3

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 compleme
$R2 \leftarrow \overline{R2} + 1$	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

The increment and decrement microoperations are symbolized by plus -one and minus-one operations, respectively. These microoperations are imple-mented with a combinational circuit or with a binary up-down counter.

Binary Adder: - To implement the add microoperation 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. the interconnections of four full-adders (FA) to provide a 4-bit binary adder.

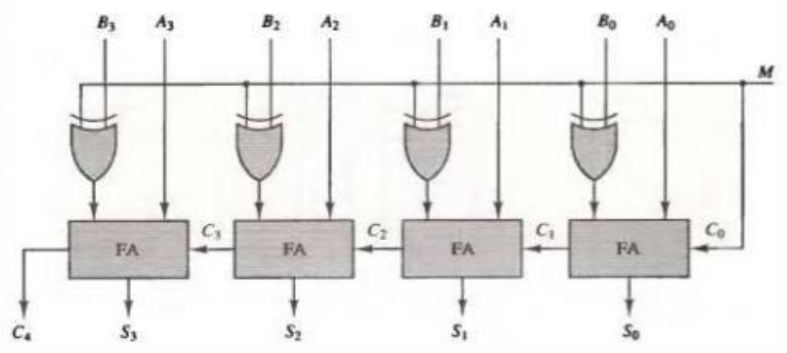


Note: - An n-bit binary adder requires n full-adders. The output carry from each full-adder is connected to the input carry of the next-high-order full-adder.

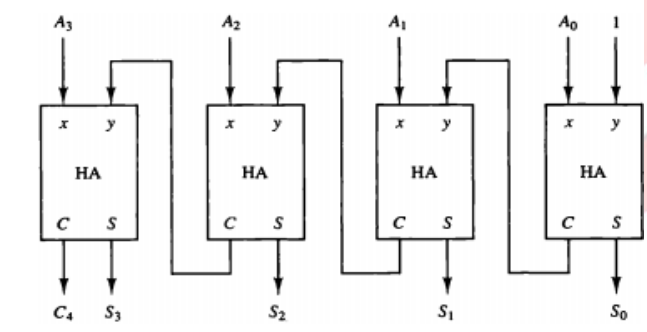
Binary Adder-Subtractor The subtraction of binary numbers can be done most conveniently by means of complements. Remember that the subtraction A -B can be done by taking the 2's complement of Band adding it to A. The 2's complement can be obtained by taking the 1' s

complement and adding one to the least significant pair of bits. The 1's complement can be implemented with inverters and a one can be added to the sum through the input carry.

Adder-Subtractor: - The addition and subtraction operations can be combined into one com-mon circuit by including an exclusive-OR gate with each full-adder. A 4-bit adder-subtractor circuit is shown the mode input M controls the operation. When M = 0 the circuit is an adder and when M = 1 the circuit becomes a subtractor. Each exclusive-OR gate receives input M and one of the inputs of B. When M = 0, we have B a1 0 = B. The full-adders receive the value of B, the input carry is O, and the circuit performs A plus B. When M = 1, we have B a1 1 = B' and C0 = 1. The B inputs are all complemented and a 1 is added through the input carry. The circuit performs the operation A plus the 2's complement of B. For unsigned numbers, this gives A -B if A '" B or the 2's complement of (8 -A) if A < B. For signed numbers, the result is A -B provided that there Is no overflow.



Binary lncrementer The increment microoperation 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. Every time the count enable is active, the clock pulse transition increments the content of the register by one. There may be occasions when the increment microoperation must be done with a combinational circuit independent of a particular register. This can be accomplished by means of half-adders

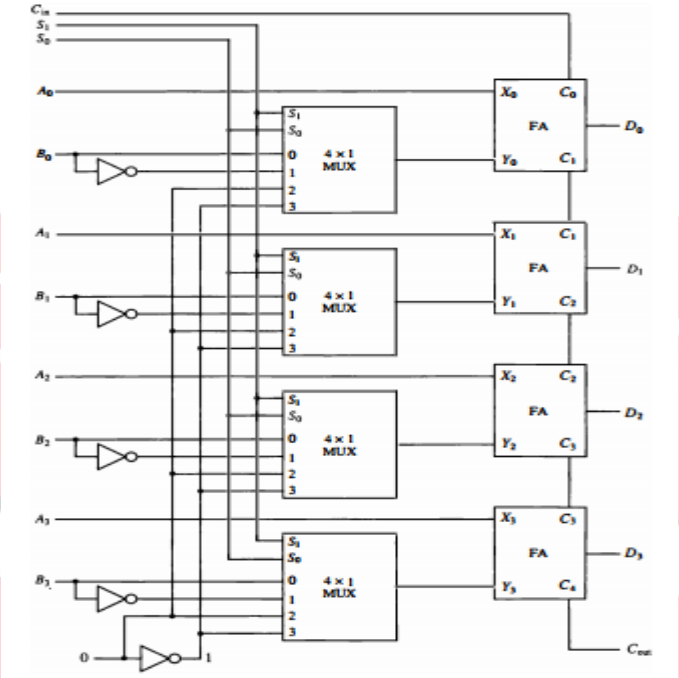


Arithmetic Circuit The arithmetic microoperations can be implemented in one composite arithmetic circuit. The basic component of an arithmetic circuit is the parallel adder. By controlling the data inputs to the adder, it is possible to obtain different types of arithmetic operations.

The output of the binary adder is calculated from the following arithmetic sum

D =A+ Y + Cin

where A is the 4-bit binary number at the X inputs and Y is the 4-bit binary number at the Y inputs of the binary adder. C'" is the input carry, which can be equal to 0 or 1.



Select			Input	Output	
S_1	S_0	C_{in}	Y	$D = A + Y + C_{in}$	Microoperation
0	0	0	B	$D = A + B$	Add
0	0	1	B	$D = A + B + 1$	Add with carry
0	1	0	$\overline{B}$	$D = A + \overline{B}$	Subtract with borrow
0	1	1	$\overline{B}$	$D = A + \overline{B} + 1$	Subtract
1	0	0	0	$D = A$	Transfer A
1	0	1	0	$D = A + 1$	Increment A
1	1	0	1	$D = A - 1$	Decrement A
1	1	1	1	$D = A$	Transfer A