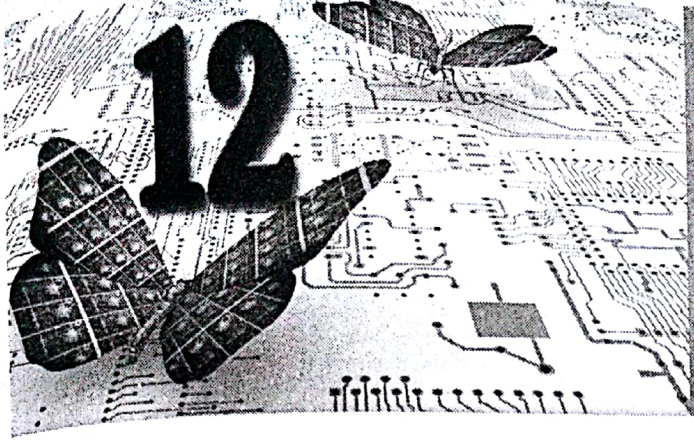


12 ALGORITHMIC STATE MACHINES

12.1. ASM CHART

ASM stands for 'Algorithmic state machine' or simply state machine is the another name given to sequential network. Sequential network is used to control a digital system which carries out a step-by-step procedure. Algorithmic state machine charts (ASM Chart) or state machine charts (SM charts) are flow chart, useful in hardware design of digital system. ASM chart resembles a conventional flow chart, but it is different from flowchart. ASM chart contain implicit timing information. It should be noted that ASM charts represent physical hardware and offers several advantages.

1. Operation of a digital system can be easily understand by inspection of the ASM chart.
2. ASM charts represent physical hardware
3. The ASM charts are equivalent to a state graph, and it leads directly to a hardware realization.
4. ASM charts can be described the operation of both combinational and sequential circuits.
5. ASM charts are easier to understand and can be converted into several equivalent form.
6. The ASM chart may be equivalently expressed as a state and output table.

An ASM chart is a description or specification of a digital system. It is an abstract description in the sense that it describes what a system does, but not how it is done.

The ASM chart can be used as the starting point of the hardware synthesis process. Any given ASM chart may be implemented in hardware in more than one way.

12.2. PRINCIPLE COMPONENT OF AN ASM CHART

(*i*) ***State box.*** The state of the system is represented by a state box. It is a rectangular box. At the top left-hand corner the name of state is shown, while at the top right hand corner the state assignment is given. Within the state box, the output signals are listed.

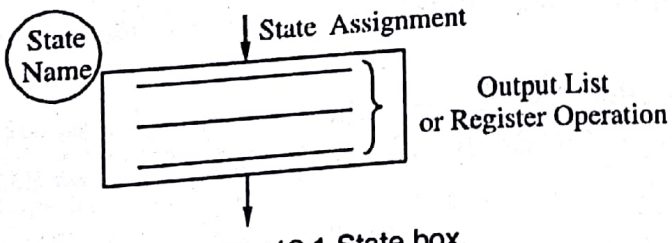


Fig. 12.1 State box.

(*ii*) ***Decision box.*** It is a diamond-shaped box with true and false branches. Boolean condition is placed in the box and the decision is made from the value of one or more input signals. The decision box must follow and be associated with a state box.

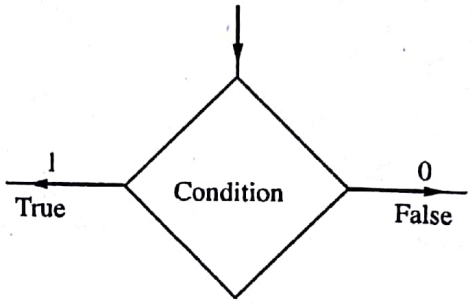


Fig. 12.2 Decision box.

(*iii*) ***Conditional output box.*** A conditional output box is shown in Fig. 12.3 is a rectangular box with curved ends. It contain conditional output list. The conditional output depends on both the state of the system and the inputs. Therefore the conditional output signals are sometimes known as Mealy outputs. A conditional output must follow a decision box.

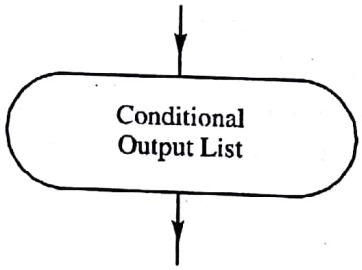


Fig. 12.3 Conditional output box.

(*iv*) ***ASM Block.*** ASM block is a structure consisting of one state box with one or more decision boxes and conditional boxes associated with that state. An ASM block has one entrance path and one or more exit paths. An ASM chart consists of one or more interconnected blocks. Execution of ASM block is sequential process whereas execution of instruction within ASM block is concurrent process.

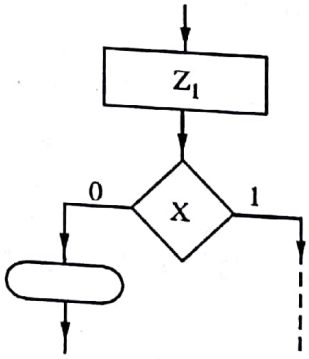


Fig. 12.4 (a) ASM block.

(*v*) ***Branch and link path.*** A path through an ASM block from entrance to exit is referred to as a link path.

Basic element for branching is decision box. After decision box we get two exit path, true or false. Depending upon condition one can branch itself upward or downward.

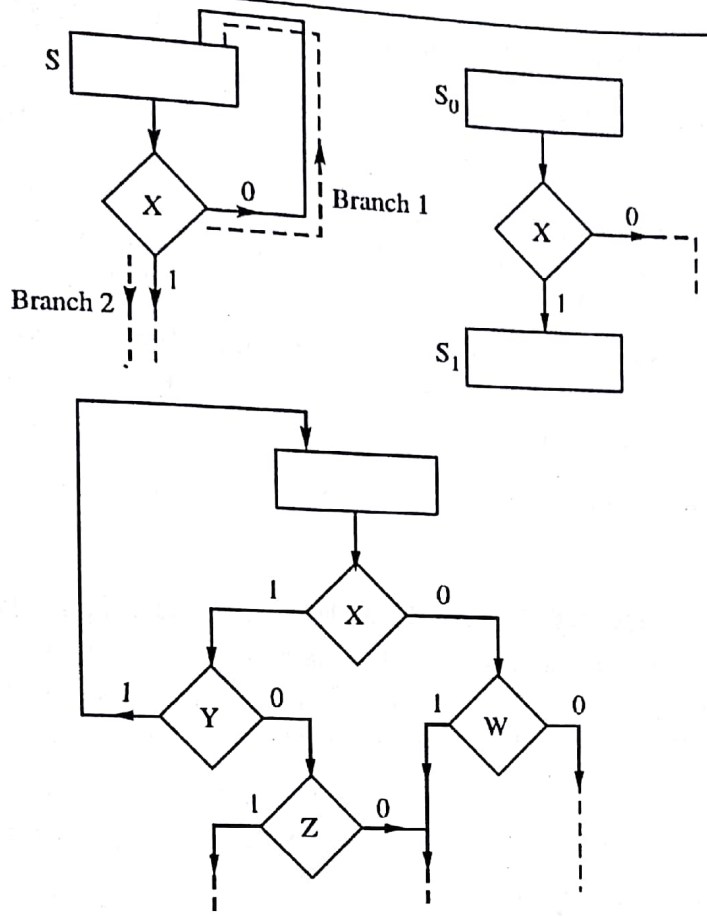


Fig. 12.4 (b)

12.3. EQUIVALENT ASM CHARTS

ASM charts are not unique, it may have more than one equivalent form Fig. 12.5 shows three equivalent ASM charts for combinational network $Z = A\,(B + C)$.

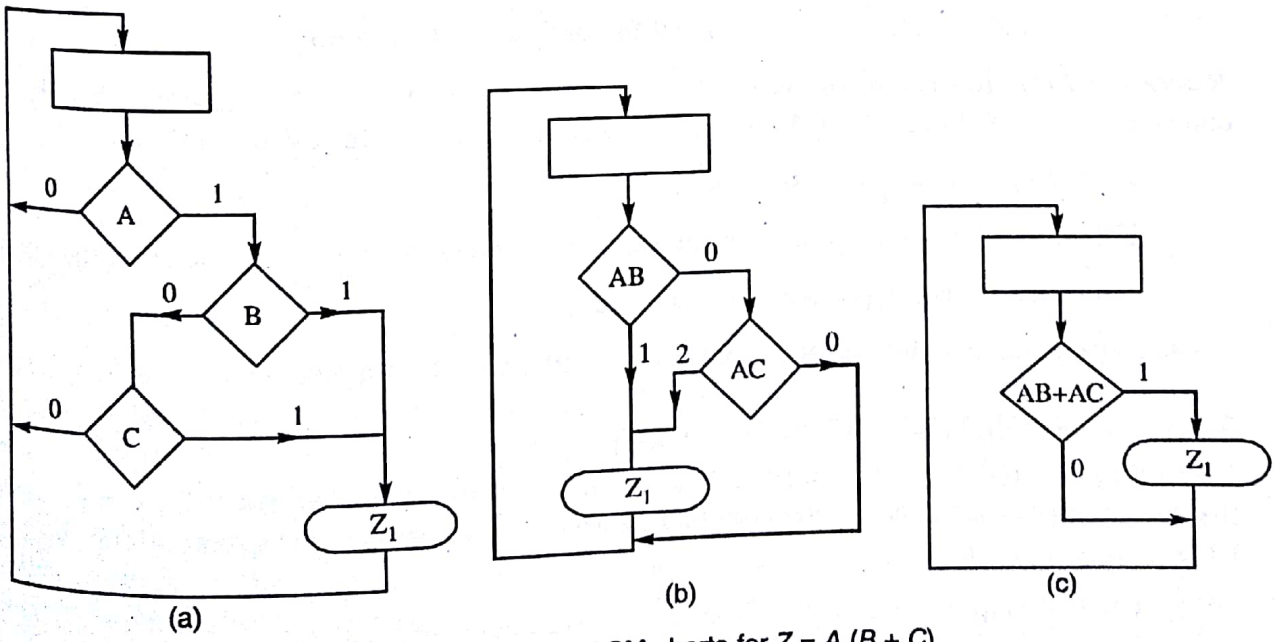


Fig. 12.5 Equivalent ASM charts for $Z = A\,(B + C)$.

ASM chart for combinational network contain only one state box. The output $Z = 1$ if $A = 1$ and either B or C equal to 1, as shown in Fig. 12.5 (a).

Fig. 12.5 (*b*) shows an equivalent ASM chart in which $Z = 1$ if either $AB = 1$ or $AC = 1$.

Similarly, Fig. 12.5 (*c*) shows that $Z = 1$ if $AB + AC = 1$. Hence, $Z = A(B + C) = AB + AC$ which is the same output function realized by the ASM chart of Fig. 12.5.

In first case (Fig. 12.5 (*a*)), we put each term in condition box, first check A if A is zero Z will be '0' regardless the value of B or C. If A is '1', then check B. If B is '1' then $Z = 1$ else check C. If $C = 0$ then output will be '0' because A is 1 and B is also 0. If C is 1, $Z = 1$ because AC will be 1.

In second case (Fig. 12.5 (*b*)), we put AB and AC in condition box. First check AB, if AB is '1' Z will be '1' regardless the value of AC. If AB is zero then check AC. If $AC = 0$ then output will be zero because both terms are zero. If $AC = 1$, Z will be 1.

In last case, we may put whole boolean expression $AB + AC$ in condition box and Z in conditional output box.

12.4. CONVERSION OF A STATE DIAGRAM TO AN ASM CHART

ASM chart can be derived from state diagram of machine, but certain rules must be followed when constructing an ASM block. First for every valid combination of input, there must be exactly one exit path defined. Second , no internal feedback within an ASM block is allowed.

Mealy machine. In case of Mealy machine, output is a function of both present state and input. For construction of ASM chart from Mealy state diagram, we should follow the following steps.

1. Represent each states by state boxes.
2. Put input in decision box after each state box.
3. The Mealy output appear in conditional output boxes since they depend on both the state and input.
4. Mealy circuit output written only when it is equal to '1' *i.e.*, true.
5. Depending on value of input connect the path to next state box.

Moore machine. In case of Moore machine, output is a function of the present state only. For construction of ASM chart from Moore state diagram we should follow the following steps.

1. Represent each states by state boxes.
2. The Moore output are placed in the state boxes since they do not depend on the input.
3. After each state box put the input in decision box.
4. Depending on value of input connect the path to next state box.

12.5. REALIZATION OF ASM CHARTS

Realization of ASM charts are similar to the methods used to realize state diagram. The following procedure can be used to derive the next state equation for a flip-flop Q, if gates and flip-flops are used.

1. The first step is to make a suitable state assignment for each state.
2. Search all the states for which Q is one.
3. For each of these states, find all of the link paths that lead into the state.

4. For each of these link paths, find a term that is 1 when the link path is followed. For example, to write the expression for next state of B, we consider link path 1 which is start in state Z_P ($AB = 01$), takes the $X = 1$ branch, and ends in state Z_Q ($AB = 11$), so the next state of B has a term $A'BX$.

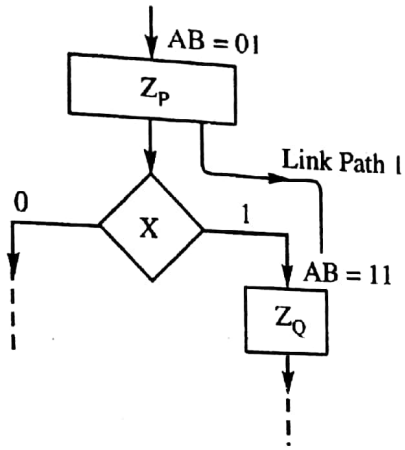


Fig. 12.6

4. The expression for the next state of Q is formed by ORing together all the terms found in step 3.
5. Similarly, the expression for output can be read directly from the ASM chart.
6. Find a simplified expression for output and next-state equations with a k-map using the unused state assignment as a don't care condition.
7. Realize the ASM chart using gates and flip-flops.

12.6. HARDWARE AND FIRMWARE ALGORITHMS

The most challenging and creative part of the design is the establishment of design objectives and the formulation of algorithms and procedures for achieving the given objectives. This task cannot be automated and require the mental reasoning of a human designer. Designer cannot develop algorithm without.

1. Thoroughly understanding the problem
2. Assuming initial configuration of equipment for implementing the procedure.

The algorithm is stated by a finite number of well-defined procedural steps. Thus an algorithm is a procedure that specifies a finite set of steps which if followed give the solution to a problem.

Hardware algorithm. Hardware algorithms are used to specify the control sequence and data processing tasks of a digital system. A hardware algorithm is a finite number of well-defined procedural steps that specifies how to implement a problem with a given piece of equipment. Generally an algorithm will contain a number of procedural steps which are result-oriented *i.e.* dependent on results of previous steps. A convenient method for presenting hardware algorithm is a flow chart. Flow charts are useful in the hardware design of digital system. It convert the given word information to an information diagram that enumerates the sequence of operations together with the conditions necessary for their execution. A special flow chart that has been developed specifically to define hardware algorithm is called an ASM chart.

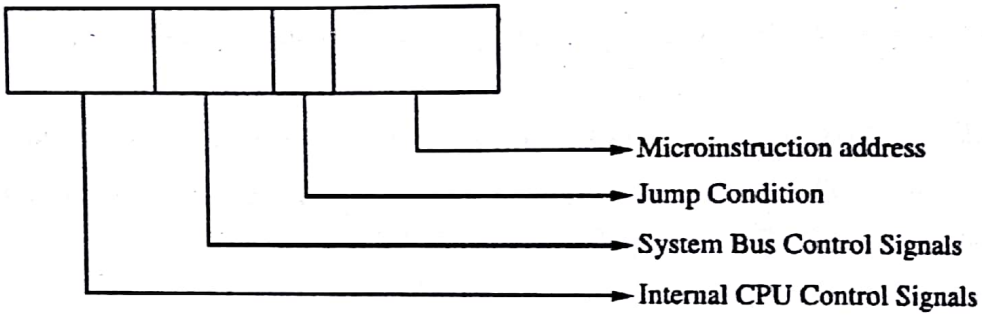
Firmware algorithm. Firmware consists of programs that are included in digital circuits during their manufacture. Firmware is used when the programs must be retained in case of power failure and when programs are not expected to be changed. The microprogram facilitates the

implementation of hardware and is therefore, called firmware to distinguish it from software.

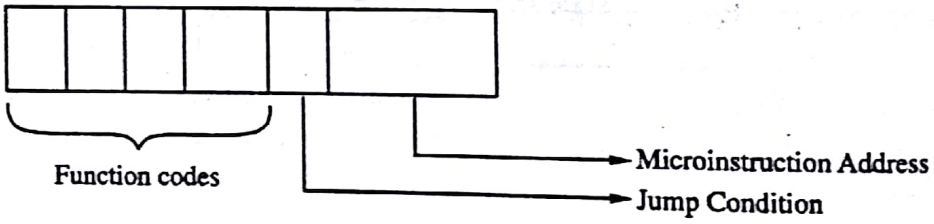
A microprogrammed computer has its central processing unit implemented primarily by a ROM rather than by discrete logic. Each memory location in this ROM, called a control memory, holds a microinstruction, the lowest level instruction a computer can execute: A sequence of microinstructions is known as a microprogram or firmware. A firmware is midway between hardware and software. We can consider microprogramming as software that can implement functions not present in the hardware. Microprograms can be changed in two ways. If the control memory is not ROM, the user can write a new sequence of microinstructions into it. The computer may have a program that can alter the microprogram stored in the control memory. Usually, however, microprograms are changed by changing the control memory.

In a horizontal microinstruction (Fig. 12.7), each control signal within the CPU is represented by one bit in the control field. In another form of microprogramming, called vertical or packed microprogramming several control signals are encoded into a smaller number of bits.

Vertical microinstruction thus has the advantage of decreasing the size of control memory needed, with the accompanying disadvantage of decreased flexibility, since it is no longer possible to have individual access to each control point.



(a) Horizontal Microinstruction



(b) Vertical Microinstruction

Fig. 12.7

12.7. CONTROLLER DESIGN

System controller is a special sequential circuit that receives input from each components of the system and outer interface and generate the control signals that control the whole system. The subsequent sections of this chapter deal with specific example that demonstrate the detailed design of control unit.

12.7.1. Using JK flip-flops. The following procedure can be used for controller designing using JK flip-flops :

1. The first step is to assign binary values to each state in the ASM chart.
2. Obtain the state table for a controller. A state table for a controller is a list of present states, inputs, their corresponding next states and outputs. The inputs are taken from the conditions in the decision boxes of the ASM chart. The outputs are equivalent to the present state of the control.
3. Obtain the excitation table of the flip-flop inputs.

4. Find a simplified expression for output and the flip-flop inputs.
5. Realize the control unit using gates and flip-flops.

12.7.2. Using D flip-flops and decoder. The main advantage of this method is that we can directly obtain the input functions from the state table without the need of an excitation table. This is because the next state is the same as the input requirement for the *D* flip-flops. Then, instead of using the flip-flop outputs as the present state condition, we might as well use the output of the decoder to supply this information, for that insert a decoder at the output of the flip-flops to obtain the necessary outputs. Now realize the control unit using gates, flip-flops and decoder.

12.7.3. Using one flip-flop per state. This method uses one flip-flop per state in the control sequential circuit. Each flip-flop represents a state and only one flip-flop is set at any particular time, while all others are cleared. The advantage of this method is simplicity with which it can be designed. The control unit logic can be derived directly from the state diagram without the need of state or excitation tables. The main disadvantage of this method is that it uses a maximum number of flip-flops.

12.7.4. Using Multiplexers. In this method logic gates are replaced by multiplexer. It is a three level design. The first level consists of MUX that determine the next state of the register. The second level contains a register that hold the present binary state value. The third level has the decoder that generates the control outputs. The outputs of register (*D* flip-flops generally) are applied to the decoder inputs and also to the selection lines of each MUX. The outputs of the MUX are then applied to the flip-flop inputs. The inputs of MUX are determined from the decision boxes and state transitions given in the ASM chart.

12.7.5. Using PLA. The design of a PLA control requires that we obtain the state table from the ASM chart of the circuit. The block diagram of the PLA control is shown in Fig. 12.8. The PLA method is superior to a ROM if the state table contains large number of don't care conditions.

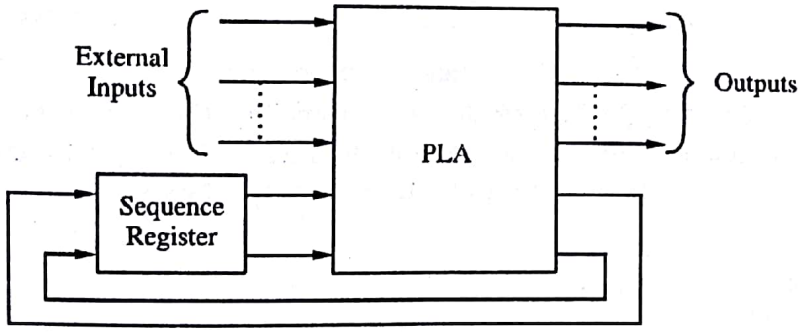


Fig. 12.8

The state table gives essentially all the information required for obtaining the PLA program table. The PLA program table is a list of product terms, one for each row in the state table, inputs and their corresponding outputs. For eacch product term, the inputs are marked with 1, 0 or –(dash). A 1 in the input column specifies a path from the corresponding input to the input of the AND gate that forms the product term. A 0 in the input column specifies a path from the corresponding complement input to the input of the AND gate. The 'X's in the state table designate don't-care conditions and imply no connection for the PLA. A '–' (dash) specifies no connection in the program table. The 0's in the output columns also indicate no connections to the OR gates within the PLA.

The conversion of the state table to a PLA program table is very easy. The don't care conditions in the input columns and the 0's in the next state and output columns are changed to '–' (dash) *i.e.*

no connection and all other entries remain the same. The inputs to the PLA are equivalent to the present state and inputs in the state table. The outputs of the PLA are equivalent to the next state and outputs in the state table. The number of states determines the number of flip-flops for the sequence register.

12.7.6. Microprogrammed controller. In a microprogrammed controller, control words are held in a separate memory called the control memory. Each control word contains signals to activate one or more microoperations when these words are retrieved in a sequence a set of microperations are activated that will complete the desired task. The set of microoperations (microinstructions) for the control of a computation sequence is called microprogram.

Structure of a microprogrammed controller. The structure of a typical morden microprogrammed control unit is shown in Fig. 12.9.

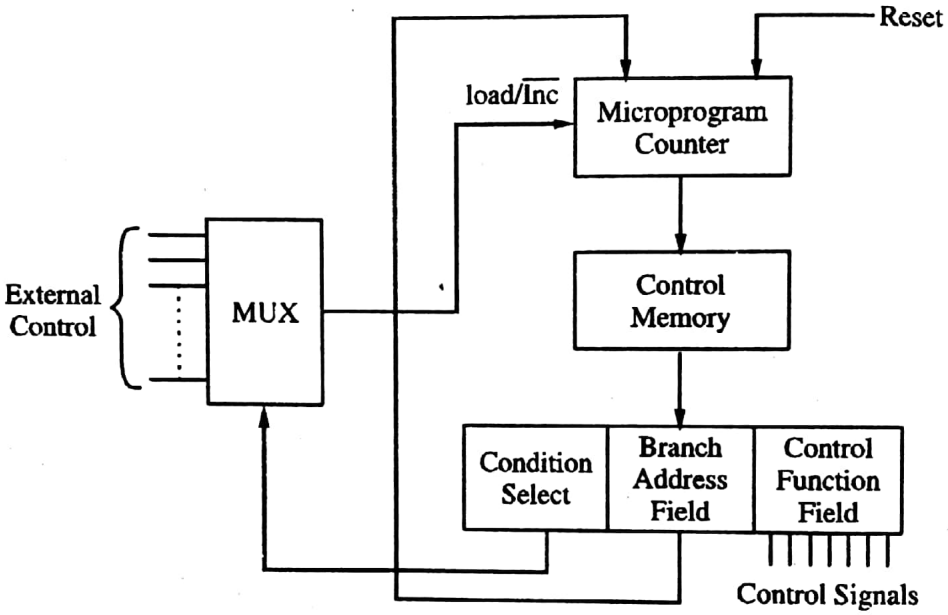


Fig. 12.9 A microprogrammed controller.

CMBR (Control Memory Buffer Register). The CMBR functions the same as the MBR of the main memory. It is basically a latch, and act as a buffer for the microinstructions retrieved from the control memory. Each microinstruction will have three distinct fields

Condition selects
Branch address field
Control function field.

The condition select field selects the external condition to be tested. If it is true, the output of the multiplexer will be 1. The output of the MUX is connected to the load / $\overline{\text{Inc}}$ input of the microprogram counter (MPC). If the output of the MUX is true, then load input will be active and the MPC will be loaded with the address specified in the branch address field of the microinstruction. If the output of the MUX is false, then increment input ($\overline{\text{Inc}}$) will be active and the MPC will point to the next microinstruction to be executed. A control function field gives the values of the control signals.

MPC (Microprogram Counter) : The typical function of the MPC include incrementing the current address by one, transmitting an externally specified address, generating a computed branch address, or specifying an initial address.

External condition select MUX : This MUX selects one of the external conditions according to

the contents of the condition select field of the microinstruction.

The comparison of a microprogrammed controller with respect to a controller implemented as a fixed network are as follows :

1. In microprogramming approach the overall speed of the system is low.
2. Microprogramming provides a well structured control organization.
3. It is more expensive than hardwired (fixed network).
4. With microprogramming any additions and changes are made by simply changing the microprogram in the control memory. A small change in the hardwired (fixed network) approach may lead to redesigning the entire system.
5. Microprogram control organization is more efficient in large, complicated systems.

12.8. DATA SYSTEM DESIGNING

Data system processes the data and produce the required output. Each data system consists of controller and data processor.

The relationship between control and the data processor in a data system is shown in Fig. 12.10.

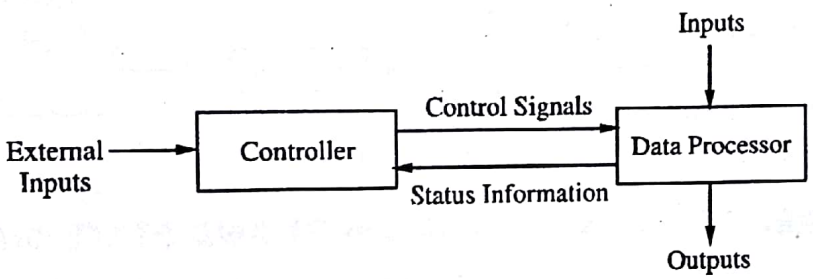


Fig. 12.10

Controller is a special sequential circuit that receives input from each components of the system and outer interface and generate the control signals that control the operation to be performed by the data processor. The data processor part may be a general-purpose processor unit, or it may consist of individual registers and associated digital functions.

Design of a digital system can be divided into two categories, first part is design the data processor which stores the data in the flip-flops or register and second part is design a controller which control the correct operation of sequence to be performed by the data processor.

Before designing the data system, designer should know very well, what the system works and what is the need for system. In general design a data system involves the following steps.

1. Given the word description of the requirement obtain an algorithm necessary for solving a design problem. Algorithm may be in the form of flow chart, ASM chart, state diagram or table, or HDL.
2. From the algorithm determine the various components required for designing, such as the size and number of flip-flops needed to store the information and the kind of operational unit such as adder, comparators etc., which are necessary to perform the required processing.
3. Generate the timing diagram which will control the sequence of different steps of algorithm. A timing diagram clarifies the timing sequence and other relationships among the various control signals in the subsystem.

4. Synthesize the controller to generate the timing signals.
5. Test the system with appropriate test signals.

During the data system designing, designer should consider the following points.

1. *Cost:* depending on the nature of the system, the designer must select the most economical solution for the given system.
2. *Efficiency* : Data system should be efficient, having low propagation delay *i.e.* fast response time.
3. System should be reliable and must have some security measure to protect the sensitive data in protected location.
4. The structure (hardware configuration) is modular, regular and independent of the particular computation.

The block diagram is shown in Fig. 12.11 which indicates that how a digital system control the physical system.

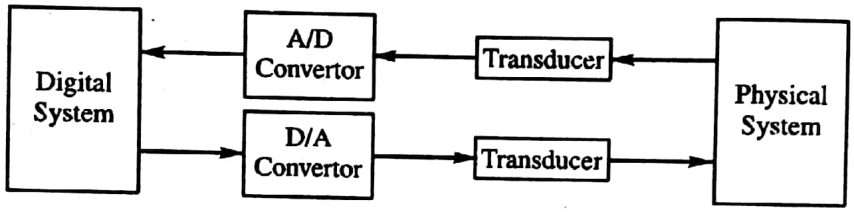


Fig. 12.11

12.9. COMPARISON BETWEEN ASM CHART AND STATE DIAGRAM

	ASM chart	State diagram
1.	ASM chart is concerned with digital system which may be combinational or sequential.	1. State diagram is concerned only with sequential system
2.	ASM charts are more voluminous than state diagrams	2. State diagrams are less voluminous than ASM charts.
3.	ASM charts may have one or more equivalent form.	3. State diagrams are unique.
4.	ASM chart contain implicit timing information.	4. State diagram is not concerned with time relationship.
5.	It is often easier to understand the operation of a digital system by inspection of the ASM chart.	5. State diagram contain less information as compared to ASM chart.
6.	It use more components than state diagram	6. It has a few components.
7.	ASM chart is more structured in nature and leads directly to a hardware realization.	7. It is often difficult to design the digital system by using state diagram as compared to ASM chart.
8.	The ASM chart specifies all the steps to be coded and helps to prevent omissions or errors.	8. State diagrams are less error prone.

Comparison between ASM chart and flow chart.

	ASM chart	Flow chart
1.	ASM charts are useful in hardware designing of digital system	1. Flow charts are useful is designing of any type of system.
2.	ASM chart contain implicit timing information	2. Flow chart is not concerned with time relationship.
3.	ASM chart is a flow chart that defines hardware algorithm	3. Flow chart describes the sequence of procedural steps and decision paths for any algorithm.
4.	ASM chart is concerned with digital system which may be combinational or sequential.	4. Flow charts provide a clear overview of the any type of problem and its algorithm for solution.
5.	The three principal components of an ASM chart are state box, decision box and conditional output box	5. It use more components of different sizes and shapes than ASM chart.
6.	No such symbols are used in ASM charts.	6. The terminal symbol in flow chart shows the beginning, end or interruption point in a program.
7.	ASM charts may have one or more equivalent form.	7. Flow charts are most probably unique in nature.