Digital VLSI System Design

Dr.S.Ramachandran

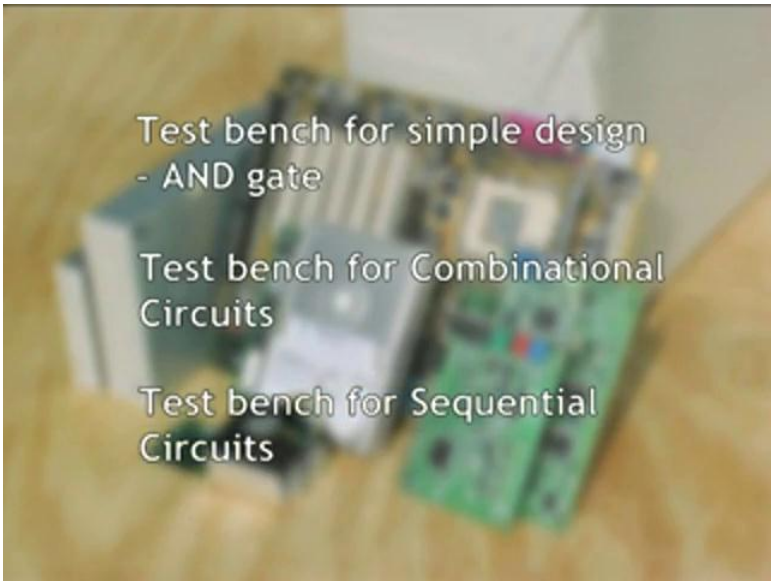
Dept of Electrical Engineering

Lecture - 17

System Design using ASM Chart

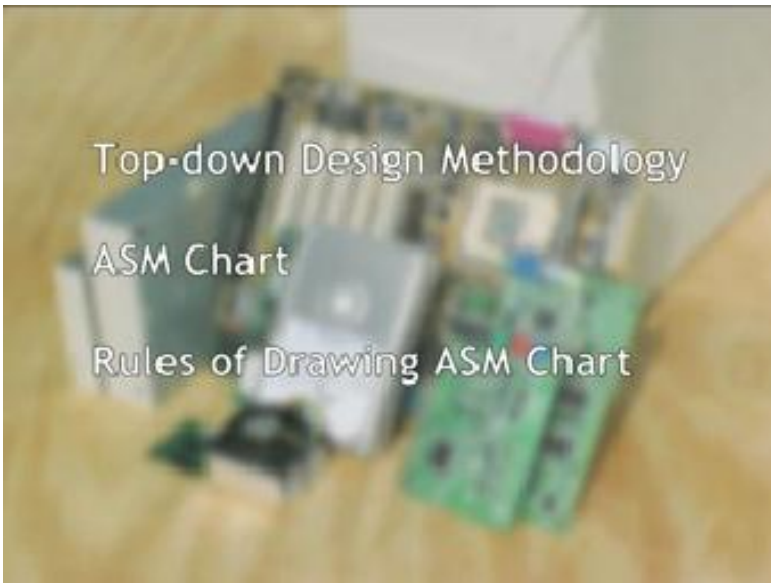
Slide – Summary of contents covered in previous lecture.

(Refer Slide Time: 01:08)



Slide – Summary of contents covered in this lecture.

(Refer Slide Time: 01:27)



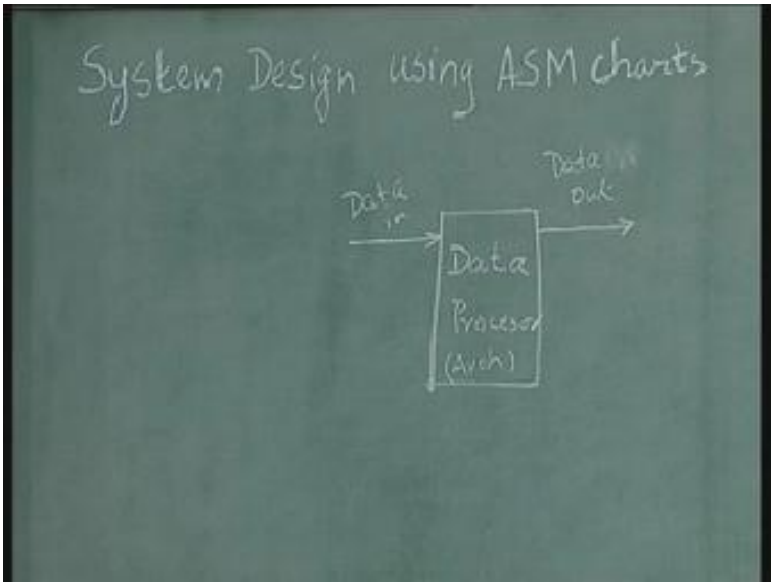
In the last lectures on design aspect, we concentrated on design of simple circuits both combinational and sequential more in the nature of the review. We introduced a few new concepts like multiplexer based design and how to fit in a design into a given PLD. Basically, simple circuits which are familiar to you and most of you might have seen the circuits in your earlier course in digital systems. We have to move on and in a real digital system design you need to think of system concepts. Given a problem, how do you approach the problem? How do you approach the design? What steps you should take to make the design possible and of course, when you reduce that problem to a set of blocks and set of specifications. Then of course you can go back and use the known techniques of both combinational and sequential design. Today we are going to start new sort of a topic where we will look at the circuit the system is a digital design as a system aspect.

We will take an approach of what is known as a top down design. That means, you are given the specifications you understand the specifications partition the specifications into realizable blocks and try to implement those blocks using the technology as well as tools available at your disposal. This is different from the circuit based approach where we wanted an adder. We know how to do it based on either the gate level Karnaugh map simplification or using an available component. In the top level design what is generally done is to partition the given problem into

two parts you have architecture of the design or the system you are designing; some people call it data processor.

This name came because most of the digital design was originally meant for computers; so you call it data processor; you can also call it data architecture. What it does is really these are the functional blocks actually the function that we want to implement finally. There is a multiplication or some computation what ever the function you are looking at you will be realized by this processor as the architecture block. How does it know what it should do and more importantly how does it know what sequence it should do? It is control signals so the data processor will have inputs and outputs which are data, data in and data out. In a case of a multiplier it could be x and y and the data in and product of x and y data out.

(Refer Slide Time: 05:00)



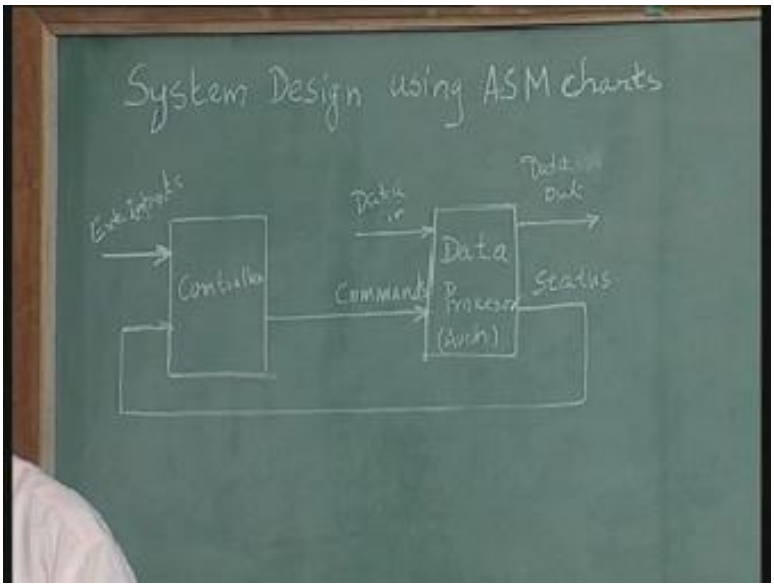
As I said how do you know when you start the multiplication? How do you know in what sequence you go over; multiply row by row? All that has to be provided with this functional block; we have another block called a controller. The job of the controller is to give signals to the data processor so that it will be done effectively; these signals are called command signals or simply commands. Based on your command the data processor initiates any action and how does the controller know when to give command it cannot keep on issuing commands regardless of what happens in the data processor. It needs some information back from the data processor so

that means we also need some signals back from data processor into the controller these signals are called status signals. The data processor is the functional block data processor. Do not attach too much importance to this phrase because these things come because of the computer jargon. Any functional block it could be traffic light controller where it will be a light which turns from red to green or green to yellow or yellow to red what ever that is also a data processor that is why I said architecture in general.

It will have its own input and output; need not have input and output sometimes but more importantly it should have commands received from the controller and status given back to the controller. Likewise the controller also may have received some signals additional to the status signal. When do you know to start the multiplication? The command controller gives the signal for step by step execution. It tells you when to load a particular register, when to load the second register. When to put the result back? These detailed sequences may come from the controller but the overall start of the whole process may depend on some external activity.

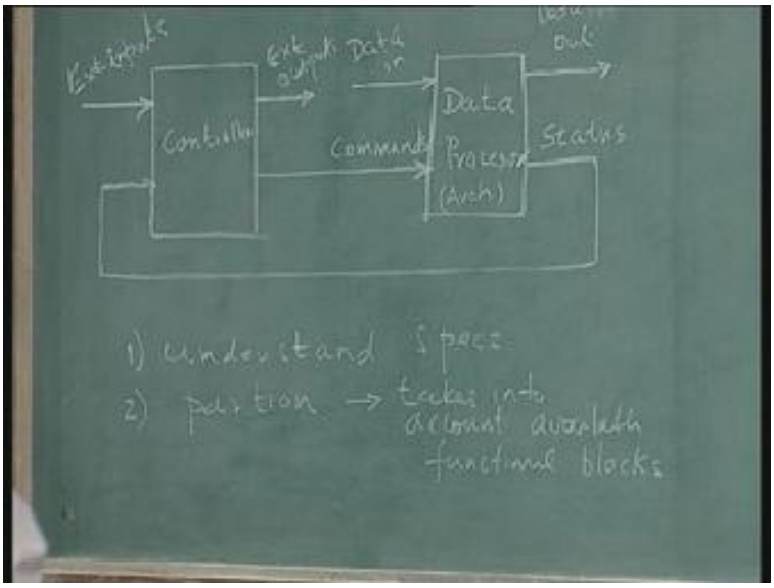
When do you want to do carry out the multiplication in your entire algorithm? If you consider the algorithm in totality, when you want to do the multiplication? This will also have to come from the controller; this has to come to the controller. These are also imports but these inputs are called external inputs. The controller has 2 sub inputs, one from external sources the other from the internal process. Likewise, the control may also have in addition to give in commands may also give you the status of what is happening the multiplication is done the result is ready the result has to be displayed.

(Refer Slide Time: 08:37)



We go on take the next so we also need external outputs. This is how you can partition a system digital system designed. Once you partition it becomes easier because data process is usually elements which are available functional blocks are available a multiplier can be both registers are available then in adder which can be purchased, so most of the time architecture functional blocks of the architecture are available of the self. Sometimes you may not have the exact specification but, you may have to work with that, on the other hand the controllers what it designed is specific to the problem that I am designing specific to the design. It is designed by the designer so the designer's job is to given a problem understand which part of it can be implemented as a functional block which is available readily and what extra things he needs to put in order to use those functional blocks in a proper way to achieve the desired goals.

(Refer Slide Time: 10:52)

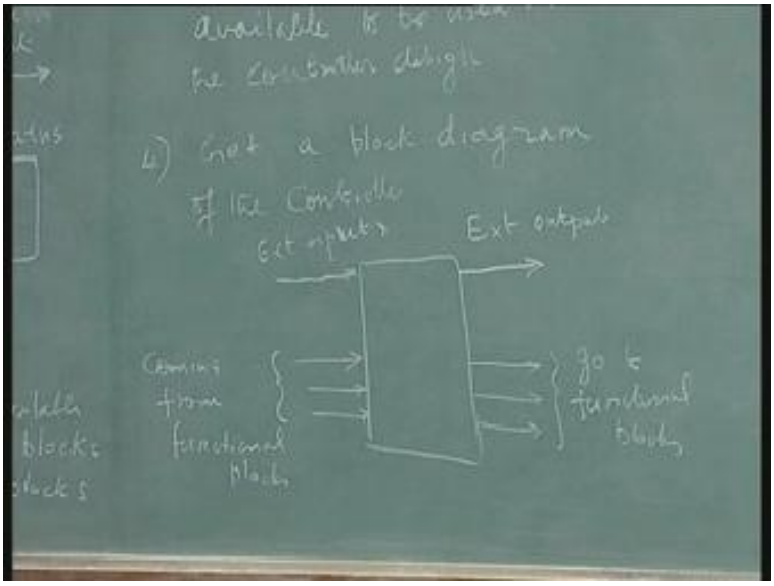


This partitioning of the problem, understand the specifications of the problem completely. It should be foolproof; there should be no vagueness about it. If it is vague, then it is difficult then you partition after understanding the specifications partition the partitioning will be done. Taking into account what is available, what is not available, so takes into account available functional blocks. Once you have these available functional blocks, then you also know what the input output specifications for this each of the functional blocks are. In additional to the data processing, take an example of register you want to use the register as a functional block one of the functional blocks of the architecture. Register is meant to store the data. There should be data in and if you want to read the data in the register data out. So, this correspond to data in data out there is no problem. What we should also know is does the register have a load feature? Does the register have a clear feature?

These are the extra things the control signals available so for each functional blocks which can be used by the controller as status signals. After having partitioned the problem to choose the proper partition takes into account functional blocks; which means basically, choose the functional blocks. After having done this I should now list let us say, list the control signals available to be used in the controller the controller design. It can be an input signal or output signal control can be output control that means I can get a control signal going back to the controller or you can have control bit signal which can come from the controller into the unit.

Both of them are identified with this I have the idea of the controller itself. That means I need to get a block diagram of the controller; these are the external inputs external outputs. These are the control signals which are available coming from functional blocks. These are the signals going to the functional blocks each one I am going to map each of the signals I should be able to map to one of the signals available in the functional block (Refer Slide Time: 14:24). Each one of this I should be able to map to the functional blocks. That means, I should be able to design very clearly which signal of the functional block is connected to which signal of the controller is the input and output. That is the clear definition. We have to do that; that way, once you design the controller you can directly connect it together it will work.

(Refer Slide Time: 14:02)



Having done this I take into account the operational requirement we have only designed the functional blocks. We have not talked about how to carry out the design operation required to be performed by the functional unit. This will be represented by this will be explained or this will be described by means of a procedure. I need to draw a procedure of control you see here for the controller operation actually.

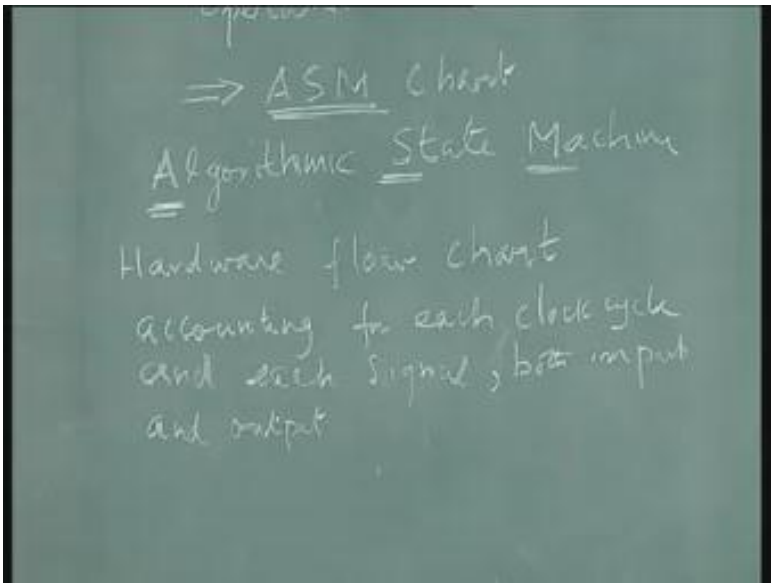
We do not use the word procedure; I just first wrote it so that I will understand what it is but really this is what is called algorithm procedure means algorithm. Actually, I will replace this word by algorithm. There is a difference between algorithm and program; the program is an

algorithm. Once you write the algorithm in software for a specific computer it becomes program whereas algorithm is the series of steps required in order to execute the particular sequence of operations in order to achieve the desired goal. This algorithm of the controller will now take into account the input specifications, the output specifications, available signals, their status and when do you give the required commands to the external functional units. All this will be taken and this is done by means of what is known as a chart called ASM chart. ASM stands for algorithmic state machine; that means it is a step-by-step clock by clock sequence of operations.

The algorithm the procedure or the controller is defined by clock by clock in order to complete the given problem, given design given specifications. This is similar to flow chart in the sense; flow chart also does the same thing for a computer program in a computer program before we have to write a program you write a flow chart. The difference here we are now accounting for a clock by clock operation that is why it is called a state machine. Term algorithm is common for computer program as well as hardware design. Algorithm is generally used by computer people also but there we only talk about functions or tasks. Get numbers from memory multiply put the result back even that is detailed people will say multiply x and y.

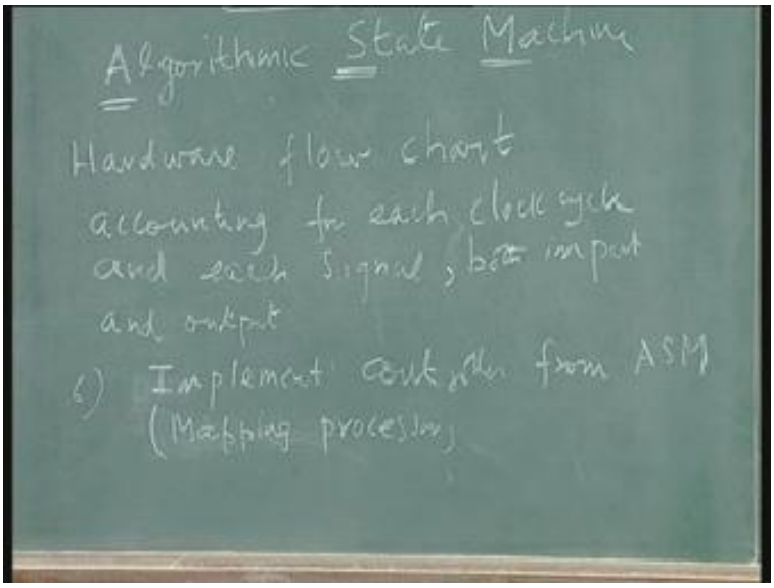
That will be one block completely in a block in a flowchart one block could be simply multiply a and b multiply x and y. I cannot afford to say for that here I have to say get x get y, multiply. The problem is even there even here even this is what is adequate I have to say to take into account how many clock cycle does it take to meet for me to get the data from x, how many clock cycles is it going to take for me to get data from y? How many clock cycles is it going to take for multiplication? What do I do till the multiplication is over? Should I wait? Should I go and do something else? All of these things are very detailed descriptions are required. This is a hardware flow chart accounting for each clock cycle and each signal both input and output.

(Refer Slide Time: 19:53)



In other words once you have this algorithmic state machine chart drawn, the hardware is realized very quickly. Once you have algorithmic states the design of the controller is only a mapping process which is a general technique, you do not need any more intelligence. From ASM the next step is, I will say, design the hardware design controller. Or, I will not even say design it is an implement controller using ASM. This implement is also a too big a word I will rather say map it; there is something which is there something like standard procedure; it is a rule based design. As I said, it is a rule based design or I will not say it is a rule based design; it is a mapping process - that is the most appropriate word for it. Process hardware availability, I mean hardware availability is not the functional level like you want to do with the gates multiplexers, ROM program logic array.

(Refer Slide Time: 21:11)



That decision you make and on that hardware chosen you map the ASM. One question is how it is different from a state machine because we have just talked about the last lectures about sequential circuits and the design of sequential circuits using state diagrams. The state diagram ASM chart are identical in one sense. There is no new information that we will get in an algorithmic state machine chart or ASM chart that is not in a state graph. ASM chart is another way of writing these state graph. You have the feel for the hardware here as you go the state graph even though it is a hardware design, possible. Once you have a state graph you have a standard technique by which you can design the hardware in terms of gates or multiplexers ROM PLA or PAL.

Here, (Refer Slide Time: 22:17) when you draw the ASM you can sort of feel the hardware. It is sort of cycle by cycle operation; there also it is cycle by cycle. Let us say you stay in a state and verify the signal - the signal input. If the input is not coming you wait in the same state, otherwise you go out of state; all that will be here also. But then, here the way it is drawn is more hardware friendly. It is more easy or easier to implement in hardware and the mapping as I said is more evident here; there also it is mapping only - the process of state graph conversion into hardware is mapping but it is a design process we call it. Whereas, here the mapping is evident as you draw the ASM looks like hardware friendly. It looks like we are already designing a machine; so that is how the ASM chart is preferred by lot of designers is more and more popular.

What we will do is in this lecture and few more lectures following lectures we will take some examples again I will go over this. We have given a problem in a global level or top level that is why it is called top down design. A top level problem is given you tried to understand what is required to be done; whether there is any functional block which can do that job. If the functional blocks are available off the shelf, do not try to design it; get them. It can be in hardware form it can be in verilog because, when I say functional block, it does not mean you have to go and get it from a shelf. It is a way of saying; half the shelf means it is available in this way of verilog design. In IC design, these are blocks which are available; block of codes available which can be borrowed and inserted; it is called IP. IP stands for Intellectual Property. If you developed a code for a particular function like multiplier, you can use them again and again whenever that function of multiplication is required.

It is IP based design they call it code base design; in an older concept we used to call it off the shelf components; that is the same thing. The concept is the same whether it is an IC design or a discrete design. You identify the processor functional blocks, clearly find out whether it is capable of delivering the goods in terms of the data processing that is required; that is the ultimate goal of course. Identify these commands which are required to initiate or activate each of those functional blocks or identify these status signals that each of these functional blocks will give at different stages. Take those into account along with external inputs and outputs to design your controller when you go to the controller identify the input and output.

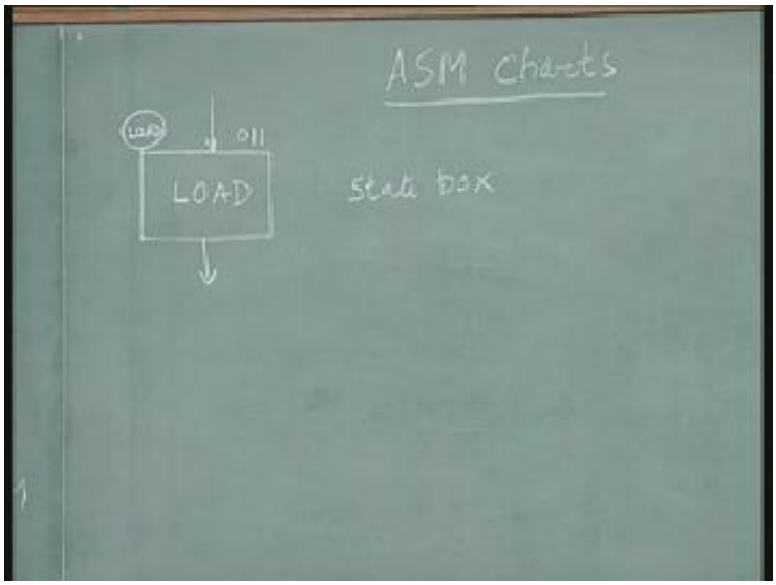
You can write an algorithm or a procedure for carrying out the control process by a chart known as ASM chart. It is very similar to flow chart but, more detailed, more cycle dependent, cycle specific, more hardware specific. With that we implement it using one of the standard techniques available for mapping and as I said state graph method in this are not different at all. This is only a different way of doing it which is more convenient and elegant; I will say elegant, convenient and hardware friendly way of doing the same job.

With this introduction, I am going to talk about a few examples before even going to the first example I am going to give you some rules about drawing state graph the ASM chart. You know how to draw state graphs the same rule can apply to ASM chart but then there are specific rules conventions adopted in drawing ASM chart. It is better to follow those conventions especially

when you are doing a design which is going to be used by somebody else. It is for trouble shooting or for later use or for later modifications; better than you conform to certain standards and norms. Let us first have a quick look at how to draw the ASM chart for a given controller. So first, we will spend some time on drawing an ASM chart then we will take a few examples. ASM chart is the simplest chart we can think of; it is not at all complicated. There are only 3 blocks in an ASM chart; the first box is a rectangle called a state box which as an input arrow, output arrow. This represents the state that we understand. In a state graph you know that at each clock cycle the circuit or the system is in a particular state and the next clock cycle the circuit or the system can change another state or continue to remain in the same state. You define the state as in a state graph or any digital system a state as period for which the circuit will not change anything, nothing will change in the circuit in terms of its behavior.

Let us also say input output nothing changes during this period between 2 states. The period between clock cycles one clock period is a state period also is possible. The second clock is the next clock period also the circuit is going to be in the same state but still it is called next state. Next state can be same as the present state but still it is in the next state because it happens in the next clock cycle.

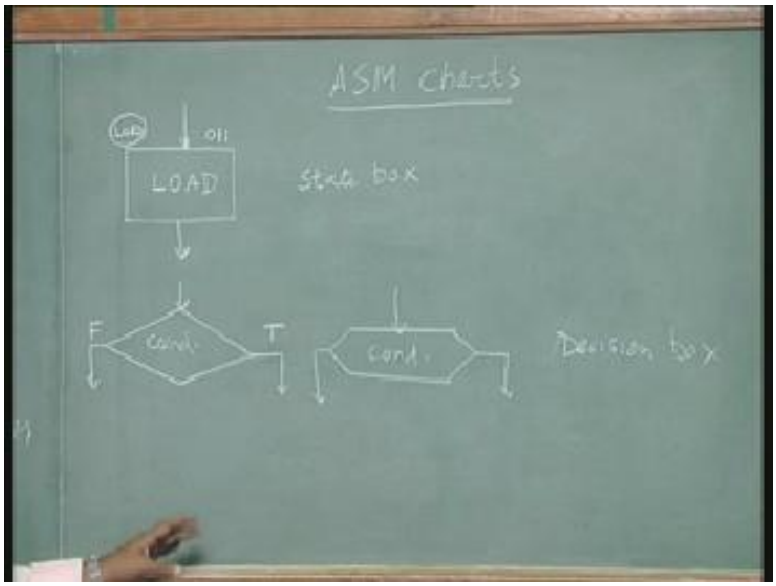
(Refer Slide Time: 28:50)



The same concept here - the state graph the state box. The state box has a name usually associated with it. The name is given here on the left hand top corner outside the box and inside the box you write the signals that will be generated as output in this state. In this case, let us say node I will say as an example, I will say LOAD. LOAD may be a signal due to a command signal I may give a register to load the input into the register that signal will be issued in this state period. We will call the state load loading state does not have to be the same. It can be s_0 s_1 s_2 or it can be a b c or it can be anything else. Usually you write something which can be easily understood here.

What is happening in the state? It will be usually understood by looking at the name and the state and the right hand top corner again outside the state you give the binary assignment state assignment. Each state in a state graph you need depending on the number of states you have; number of variables and those variables have each will have 0 or 1. Binary number representing each state and that number is given here so in this case for example I will call this 0 1 1 no specific reason; just arbitrary. 0 1 1 is the binary value of the statement for the variables binary values of the state variables of this particular system. We have a second box known as the decision box: diamond with 1 input arrow and 2 output arrows. This is where you test a condition. Suppose you want to start is the start signal available if it is available you will only start and proceed if it is not available you want to wait. This could be an example of the conditional verification or it could be memory read operation with wait state.

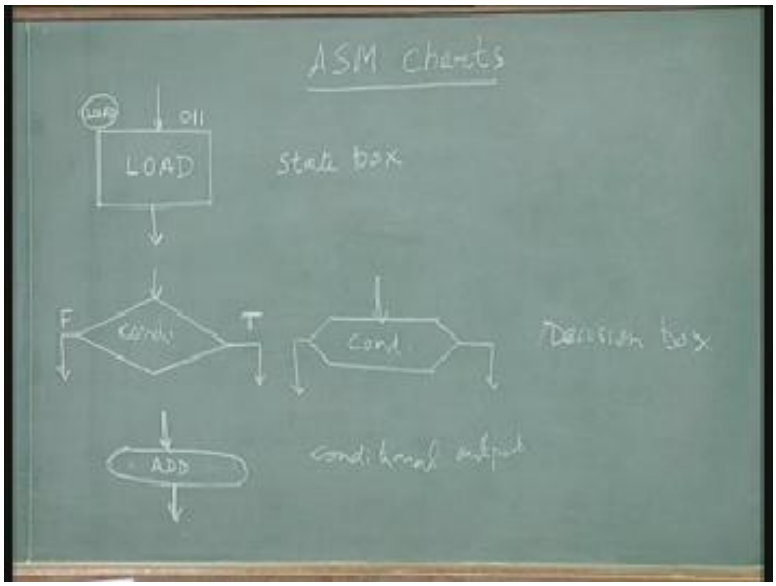
(Refer Slide Time: 31:12)



We are looking for some data ready signal coming from the memory. You can say data ready if it is no, wait; yes, proceed; like that. Usually, they put 0 1 0 1 is generally done but in real pure ASM design they call it, true or false. If the condition is true what you do if the condition is false what you do sometimes conditions will be tested big to be written in a diamond shaped box.

They also draw it like this (Refer Slide Time: 32:19) and there is lot of things to write, could not write too much in a rhombus or a diamond like this box. This as I said is a condition box, decision box and the third box, the third and the last box is called conditional output box. As an example, it could be a shift or Add. Let us say add in an algorithm like multiplication algorithm shift algorithm you know that. We do addition and shifting.

(Refer Slide Time: 33:32)



If the multiplier bit is a 1 it will only do shifting without adding. If the multiplier bit is 0, it is an example of a multiplier. When you do a multiplication of 2 numbers there is a multiplicand and a multiplier. For each multiplier bit we add the multiplicand and shift it if multiplier bit is 1; without adding the multiplier we shift if the multiplier bit is a 0. That is the condition based on the multiplier bit using 1 we want to add otherwise 0.

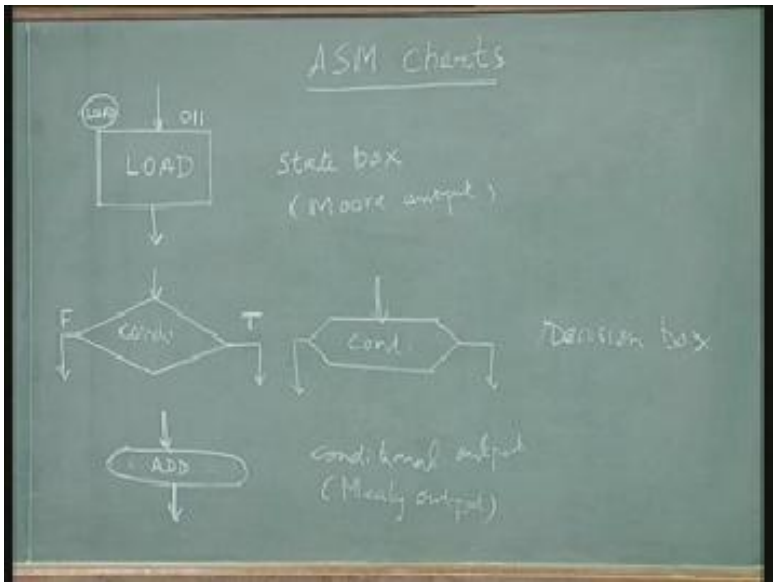
Here you put a condition whether the multiplier bit is 1 and before going to the next shift box you can say add. This is an output box similar to this; you generate an output here output generated was load here, the output generated is add. This is done only under certain circumstances of a condition being met. Whereas, here once you put a box and within that box when you write the signal whenever the circuit of the system stays in that state that signal will be generated during that clock period. During the clock period for which the circuit is in this state signal load will be generated. It is an unconditional output here. The output will be generated only when the particular condition is met. That is why it is called conditional output this is useful.

Otherwise what you have to do is to put an extra state after this it can go to this generate this add signal and go to the next state. If you do not have the conditional output I may have to put one state or add and then next state one more state shift which means I will put one extra state. If you

keep adding the number of states the mapping hardware becomes more and more complex so avoid that we have this. If you remember in a state machine the more machine we have this type of states generate the output. Outputs are associated with states in a Moore machine. in a Mealy machine. Outputs are generated with input condition from a state in Mealy. The output is defined by what is the present state. What is the present state? The circuit is in and what is the input at that state? The present state and the input decide the output in the case of Mealy whereas; the present state resembles output in the case of Moore.

This is a Moore output; this is a Mealy output (Refer Slide Time: 36:00); that is all we are familiar with this so these are 3 simple boxes all you can. You can draw any algorithmic state machine by what is the state in which the circuit is. What is the signal that is to be tested at that time? At that state what will happen if the condition is true, what will happen if the condition is false? Unless the condition is true or false it goes to the respective next state in so doing it may produce an extra output called conditional output or sometimes it may not produce an extra output. This is all there is to in any algorithmic state machine so its not a complicated affair but as I said if you draw it properly it becomes elegant and easily mappable and hardware that is why lot of people use this approach. With this introduction we will take a few examples of a state machine design. Let me first take an example of a very simple ASM.

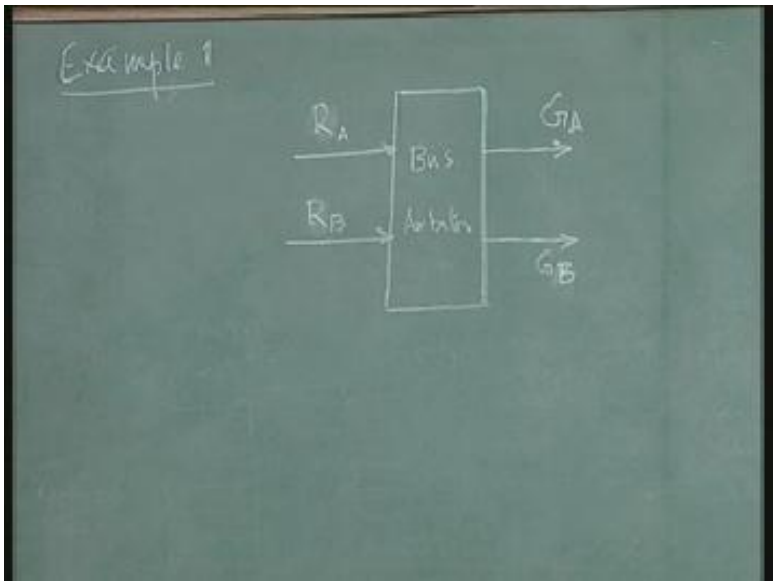
(Refer Slide Time: 37:12)



According to our rules of the design, first the problem has to be defined we have to partition it between hardware and the controller and the data processor functional units. identify the signals required to activate this functional blocks and identify these signals which are given a status in the functional blocks. Based on the signals and the external signals you design the controller its algorithmic state machine and implement hardware. We will have to take that approach. Let us now design simple, what is known as a bus arbiter. The circuit is simple but, I want always start with a simple example. What is a bus arbiter? There is a bus like a microprocessor system bus or any other bus in any system and then there are several units requiring the use of it, demanding the use.

The bus has been granted and can be granted to only to one of the users. One of the requests at a given time in order to make it simple we will say that there are 2 signals called grant A and grant B - G_B. G_A stands for request from A system and G_B request from system B. 2requests are coming and the bus arbiter has to give to actually, you call it R_A and R_B request; G_A G_B may be granting R_A and R_B. Request from system A and request from system B come and if bus is granted to system A, signal G_A is generated and the bus is granted to system B, signal G_B is generated. This is the overall system I have not partitioned it yet.

(Refer Slide Time: 40:00)



Sometimes the partition is so trivial we do not have to there is no function unit here even though the function unit will come later. How this G is taken and released all that will be functional units. We have not talked about that; we have already talked about the bus arbiter as a design. The whole concept of the design whole specification design is for the bus arbiter since only bus arbiter I wanted to design I will not talk about partitioning the system; this happens sometimes; the system is simple. This partitioning is not a very rigid partitioning; this one thing I forgot to mention that time. This partitioning is sort of a flexible partition. Some functions are very great whether you call it a functional unit or whether you call it a controller part it is difficult to say for example. I will give an example. Suppose I wanted to have shift and multiply; multiply by shift and then add; suppose there is 8 bit number I am doing. There must be 8 shifts and 8 add.

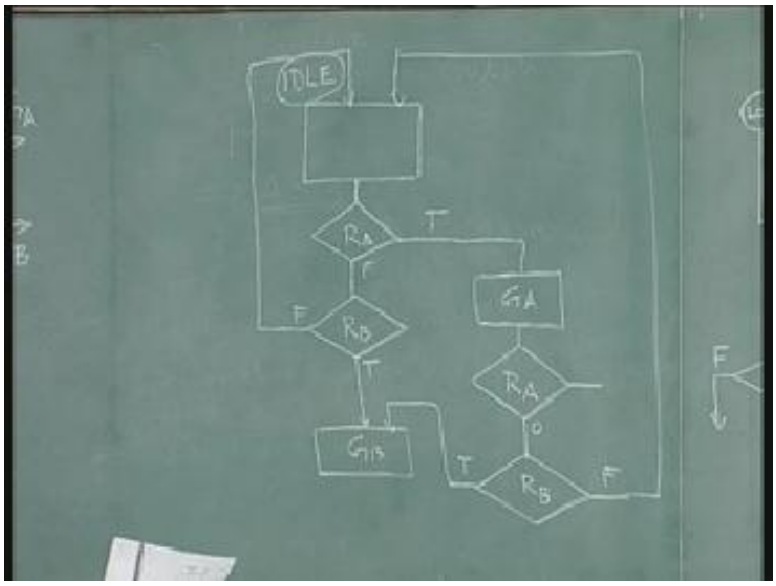
Add may be there are not depending on bits shift will always be 8 shifts. There are 2 ways of doing this. in the controller, I can keep count of the shifts. Every time the status signal says 1 shift, I can take the signal to the controller, update the controller and issue the next signal. I can keep the counter inside the controller. On the other hand, I can put the counter inside the data processor and say when 8 is reached only then I will give the status complete. Otherwise I will keep on moving shifting and adding. This is one example of how I can flexibly use the data processor or the controller for this job. When the partition becomes so vague of course it is a multiplier which requires adder for example, shift and add algorithm is clearly a function unit adder is clearly a functional unit I will not try to implement it using controller. There are certain things keeping count and all that which can be put in either this or that.

So that way this is flexible and depending on the circumstances so I would not like to do 16 times something rather than that I will put here as soon as the data it keeps going. It becomes when you put the 16 times here they might state graph is going to become extremely big 8 bit controller 8 bit multiplier 16 bit multiplier. For each multiply, I will put a state multiply, add shift, add shift each is a state. I do a big state graph. The controller in the ASM chart will be so big there are so many states; instead I can put a counter. Activate the counter, update the counter, increment the counter by 1, every time I do a shift increment the counter by 1 and the counter reaches particular value. I compare the value and then give it. The number of states required for the controller will be so I am sort of putting some of the controller function into the data processing by putting an extra counter there.

Otherwise my ASM chart is going to be extremely big. When that is the approach taken here it is also very difficult to partition between the bus controller and bus arbiter and anything else. It is only a simple circuit, so I am not going to try to define or partition this circuit because the first example is simple example I want to give. I am not going to attempt to partition this into hardware or architecture functional unit and the controller I will straight away draw the ASM chart. There are only 2 signals as inputs R_A and R_B and 2 signals as outputs G_A G_B only thing I need to know is there any priority. If R_A only comes bus generally goes to, naturally goes to G_A. If R_B comes the bus goes to B.

What if R_A and R_B become together? Is there any priority we want to have? We will define the priority. We will say, if both R_A and R_B come, R_A has higher priority than R_B. This is arbitrarily I am assuming. I have no preference to R_A; just like that assumption. With this I will draw the ASM very simply I have this unit in which the circuit is waiting. I will call this idle state or wait state, nothing happens. Then you have this signal R_A coming. If R_A is yes, true, it has to process and give G_A. If R_A is false you will see if R_B is present. If R_B is also false go back to idle state or sleep some more; idle. Instead of R_A is true G_A is given and I will explain. I will put one more R_A verification here; it requires a little explanation; I am going to do it later.

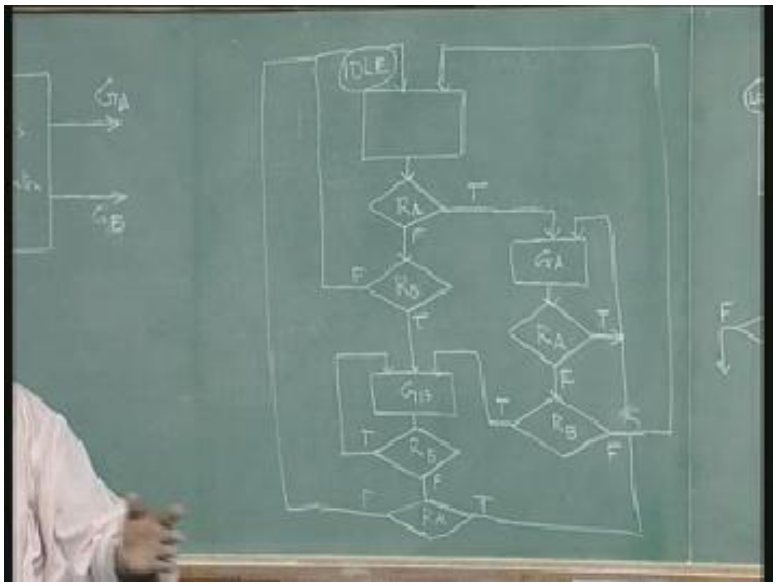
(Refer Slide Time: 46:07)



Once R_A is taken care of, I see if R_B is also waiting because R_A is priority. It is possible both R_A and R_B are present. Since R_A is taken care of you are not bothered to look at R_B. If it is true what do you do? I go and if this is true or this is true I go back to go and generate G_B. If it is false R_A has been taken care of, R_B is not there, so wait. On the other hand here after G_B - by the time I finished G_B again R_B. Check one more time I will tell you why same reason. This also will be tested one more time it is true continued to be in G_B 0 or false again. R_A is waiting; if it is not waiting I have to go back. If it is waiting I go back to this (Refer Slide Time: 46:08). This ASM is very simple there is an idle state. No output inside. You keep taking R_A and R_B in the same clock cycle. If R_A is true, the clock cycle will issue G_A and keep issuing G_A as long as R_A is kept high. That is why I put one more R_A here. I do not want to issue for just 1 clock cycle when R_A has not removed the pulse. I am assuming that means R_A will be given.

It will keep on as long as I wanted the signal. It will be held high as long as I want the bus. Once I have done with bus then I will have to remove these signals. Whenever I want G_B issue a signal and keep issuing till G_B has done its job. Once G_B is given R_B will be held. If R_B is removed only then G_B will be removed. Ga will be kept on, after they become 0 - false,…usually put arrows here (Refer Slide Time: 49:25), RB will be tested one more time because, R_B was not considered earlier. If R_B is true, G_B is issued. As long as it is true, G_B will continue to be issued. Once it is removed, G_B is required no more. R_B will be removed withdrawn the user will withdraw the request of R_B and then it will go and test R_A one more time strictly not necessary. Since already tested R_A and here but then by the time because it could have taken several clock cycles G_B would have been high.

(Refer Slide Time: 50:09)



By the time R_A would have been generated the idea is again the priority does not mean when R_B has started. R_A cannot have a priority this also is taken care of here. If R_A is there before R_B starts, R_A gets priority. R_B becomes after R_A starts, after R_B starts it has to wait till R_B is over. This is how you define the system input signals output signals the conditions specifications should be very properly written. R_A comes first. R_B comes and what will happen if R_B comes in between? All those are very clearly defined problem.

Specification is 40 percent is the problem; ASM drawing is another 50 percent problem. The rest is 10 percent or may be 20 percent; 40 percent marks for problem specification 40 percent marks for drawing ASM; 20 percent marks for mapping for identifying the functional blocks identifying the input signals output signals all that is equal to 20 percent weightage. This is important and that is why I finished it today. The next lecture what I will do is, I will see how to implement this in the hardware. There are different techniques by which you can do it. We will see those techniques in the next class.

(Refer Slide Time: 52:06)

(Refer Slide Time: 52:11)