Digital VLSI System Design

Dr. S. Srinivasan

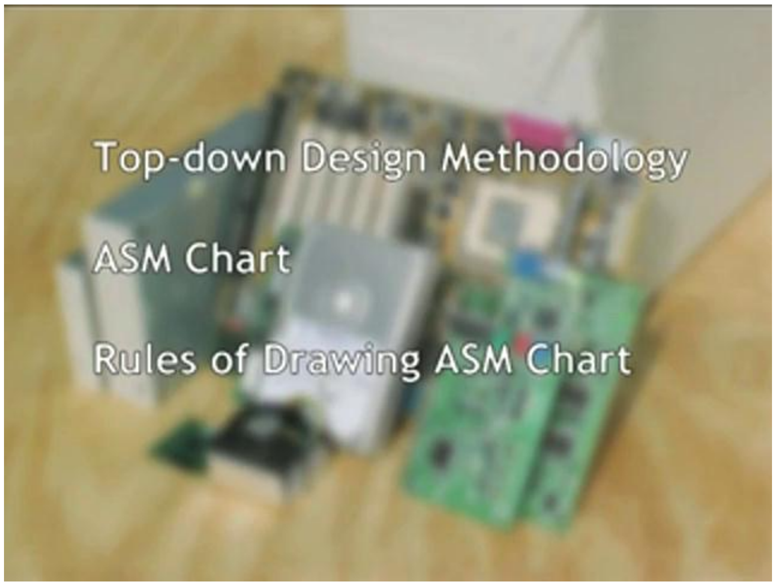
Dept of Electrical Engineering

Lecture - 18

Example of System Design using ASM Chart

Slide – Summary of contents covered in previous lecture.

(Refer Slide Time: 01:08)



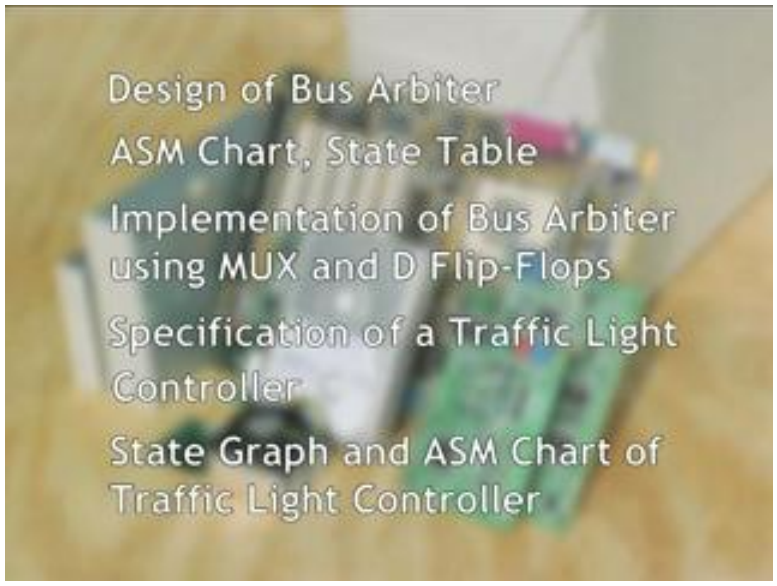
Top-down Design Methodology

ASM Chart

Rules of Drawing ASM Chart

Slide – Summary of contents covered in this lecture.

(Refer Slide Time: 01:32)



Design of Bus Arbiter

ASM Chart, State Table

Implementation of Bus Arbiter using MUX and D Flip-Flops

Specification of a Traffic Light Controller

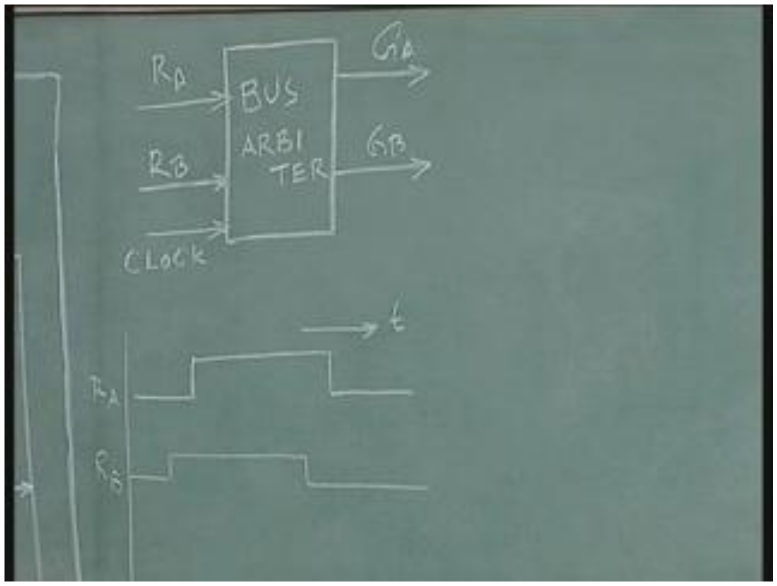
State Graph and ASM Chart of Traffic Light Controller

In the last lecture, we saw the concept of digital system design using top down methodology, where the design was partitioned into an architectural, functional or data processor block. Functional blocks are generally available as components, either as individual IC parts or in the case of IC design, as verilog codes. The controller is the one we develop, based on the specification of the problem specific to the customer's needs or specific to the system design problem which needs to be solved. The best way to go about designing a controller list is to draw an ASM chart. ASM stands for Algorithmic State Machine and in the last lecture, we started the design process with a simple example with a bus arbiter wherein, there is an available bus which will be requested by 2 units called A and B.

When unit C requires the bus it gives R_A as signal to the bus arbiter and when unit B requires the bus it gives signal R_B, the arbiter then takes the decision on which of these 2 units controls the bus. This is indicated by the grant signal, G_A being the grant signal for unit A and G_B being the grant signal for unit B. This is only a simple design so I will not be showing how signal G_A enables the bus for unit A and how G_B enables the bus for unit B. Normally only 1 of these requests come in and the unit which requires the bus gets it, the problem of contention comes in only when both the units require the bus at the same time.

If we prioritize G_A as having a higher priority than G_B, R_A as unit A having higher priority than unit B which means that if both R_A and R_B come in at the same time R_A gets the higher preference and the bus is given to unit A and G_A is generated. But if R_B comes in earlier than R_A, even if R_A comes in before the unit B has completed the transaction unit A, it has to wait for the transaction of unit B to be completed. This means the unit that requires the signal can be represented as a time function if you were to draw the signals. If R_B comes in before R_A, even though R_A has higher priority than R_B, R_B gets the bus and R_A has to wait for the bus to be released by unit B before it can get to service. To make this simple, we assume that the request signal is held for as long as the period you want the bus to be available to that unit. If you remove the unit A request signal R_A unit A, is disconnected from the bus. This means that if unit A requires the bus for a particular duration of time then for that entire duration of time R_A has to be available, the R_A signal has to be held high for that period.

(Refer Slide Time: 06:17)



This was drawn for the last lecture; just for review, we have already drawn it. The state in which both R_A and R_B are monitored is called idle state. If R_A is true it gets the preference and G_A is generated, G_A continues to be generated as long as R_A is true. Only when R_A is false and the arbiter looks at unit B which means as long as R_A is true it goes through several clock cycles during which G_A continues to be generated. Clock cycle G_A

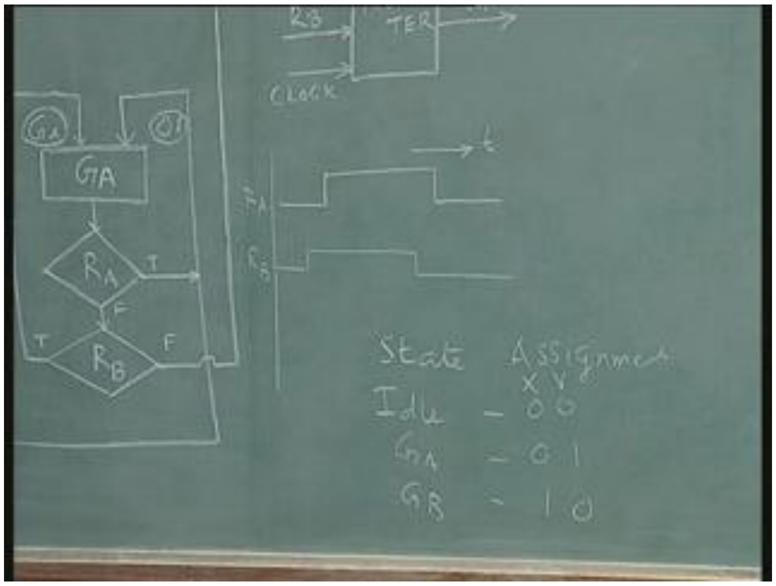
is available for entire duration for which R_A is true after which it checks if unit B requires the bus and if yes then it generates grant signal G_B. This is then held as long as R_B is held high and when it is done it goes back and checks for R_A. In the meantime unit A could have sent a R_A request signal, if there is no such request it goes back to the idle state. Similarly, when R_A is done if R_B is not requested that means unit B does not request the bus and it goes back to the idle element. I want to draw your attention to couple of points here which is convention more any concept or theory.

The ASM is drawn with rectangles and diamond boxes and what you have inside is the signal, there is no signal in the idle state so there is nothing inside. In the idle state G_B grant B state and G_A grant A state are drawn as arrows, request arrows are always drawn going to the arbiter and grant signal arrows are drawn going out. It is convention to draw true or false. Of course, there is nothing wrong in representing it as a 0 or a 1, but generally in the ASM the original concept of the design drawing is in a way a decision box. There is always two paths in the output one is called the true path and the other is called the false path, true or false is the same as 0 and 1. It is better to follow conventions when you draw a chart but the next step is to implement it as hardware. This is nothing but a state graph, you can think of this as a 3-state, state graph with R_A and R_B being inputs and G_A and G_B the outputs. The different conditions of R_A and R_B go into different states and give different outputs.

We will not redraw the ASM when the state changes because, we know how to implement a state graph into hardware and for that reason we will not draw the ASM again. As we mentioned earlier, the advantages of an ASM are ease and elegance so we can directly translate or map the ASM into hardware. Before we implement the ASM we have to assign similar logic to the states, similar in concept to the state graph. There are three states so you need two state variables, we will call them x and y because A and B have been used, we will then arbitrarily assign the state 00 for this; 01 for this and 10 for this or 11; does not matter. As a convention we put a circle around the states. The state assignment is marked outside the box on the right hand top and enclosed in a small circle, this is the convention.

There are 3 states, an idle state whose assignment is 0 0, G_A state whose assignment is 0 1 and G_B state is whose assignment is 1 0. State assignments are G_A state and G_B state and x and y are the state variables, so you draw the ASM table similar to the state graph.

(Refer Slide Time: 11:20)



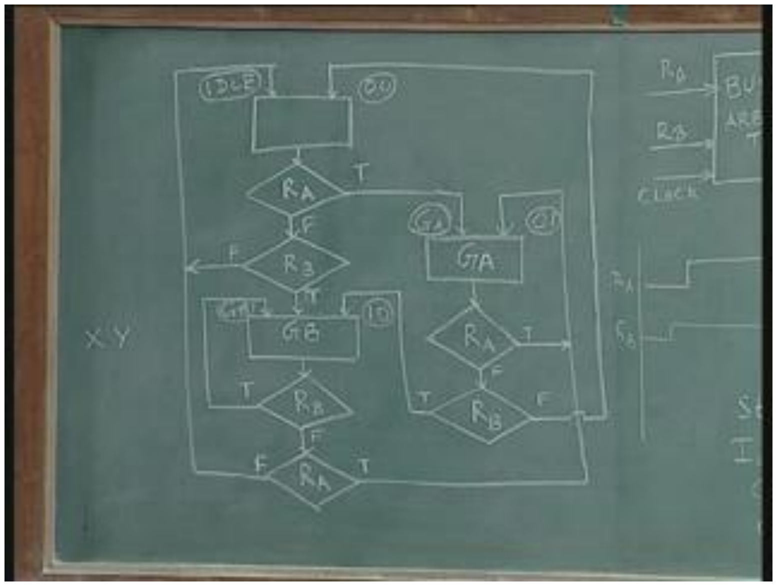
The first table you implement in the ASM chart is to assign the state variables' state assignment; once you do that then you go for the ASM table.

So we will have to see what the present state is and what condition leads to a next state and what outputs are generated in that state, these will be the columns of a ASM table. In the ASM table; there are the 3 states, idle state, present variables are x y, next state variables are also x y. In order to differentiate between the present value and the next value of x and y we put x plus and y plus as the convention. We will follow this in state graph implementation also.

So we do not give the conditions, we write the conditions rather than inputting them here. So for an idle state of 0 0, what are the different conditions under which it goes to next state? If your idle state R_A is true then it goes to G_A and if R_A is true and R_B is false then it goes to state G_B. If R_A is false and R_B is also false it goes back to idle, all the 3 conditions for the three paths have to be represented in the ASM table.

So the ASM table will have 1 row each corresponding to a path from a given state to the next state. So this state has 3 paths leading to three next states, 1 or 2 of them may be the same state, but as long as there is a path to another state then you have to create a separate path for it. There are 3 rows, in the first row R_A is true, in the second row R_A is false and R_B is true so R_A bar R_B and in the third state R_A and R_B are both false.

(Refer Slide Time: 13:20)

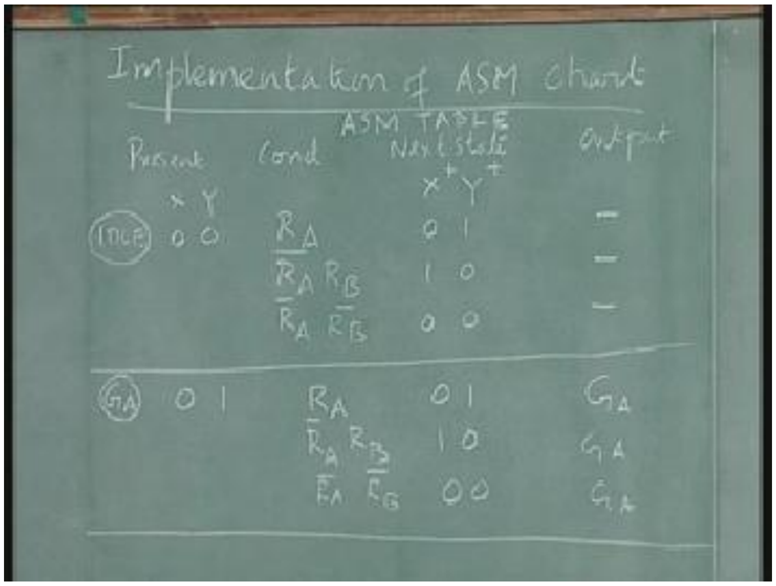


So under the first condition it goes to G_A state which is nothing but 0 and 1, under the second condition it goes to G_B which is 1 and 0 and in the third condition it goes to R_A and R_B both 0 which is the idle state. So I have finished all the transitions from the current idle state with a list for all the present states and the corresponding paths leading to the next states. What is the output here? In none of these there is an output because the output corresponds to the present state. The present state is the idle state and in the idle state there is no output, no output, no output (Refer Slide Time: 14:40).

So to quickly complete this table, the next state is G_A which is 0 1 state, there is a path if G_A, R_A is true it goes back to R_A, if G_A, R_A is false and R_B is false it goes back to idle and if R_A is false and R_B is true it goes to 1 0 state. So first is R_A true, the second is R_A false and R_B is true and the third is R_A, R_B both false. In the first instance it goes back to G_A, in the second instance it goes to G_B which is 1 0 and in the third instance it goes back to idle state which is 0 0. What are the outputs? As long as you are in the state the output is G_A because there are no conditional outputs in this ASM. All the outputs in the ASM are, yesterday I talked about conditional and non conditional outputs.

In a conditional output you will have an output; at the output of the state based on the condition being true. It is given by an elongated bubble. Whereas, in the unconditional output the output is related to the state in which it decides. When the output is marked within the state, it is unconditional removed output. So when the output is G_A all the 3 states, all 3 paths will have G_A and finally in G_B state the 1 0 all 3 paths will again go back here.

(Refer Slide Time: 16:25)



Implementation of ASM Chart

ASM TABLE

Present		Cond	Next State	Output
	$X\ Y$		$X^+\ Y^+$	
IDLE	0 0	R_A	0 1	–
		$\overline{R_A}\ R_B$	1 0	–
		$\overline{R_A}\ \overline{R_B}$	0 0	–
G_A	0 1	R_A	0 1	G_A
		$\overline{R_A}\ R_B$	1 0	G_A
		$\overline{R_A}\ \overline{R_B}$	0 0	G_A

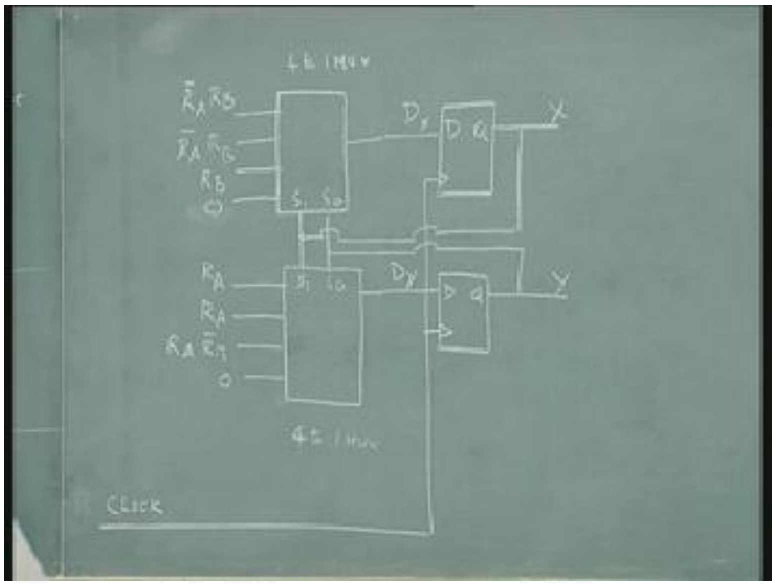
So G_B, if R_B is true it goes back to the same state; if R_B is false or R_A is true it goes to 0 1 state. If both are false 1 0 state, 0 0 state and in each of these paths each of these rows output is G_B. Conceptually, the state graph ASM is not from state graph. This table is not conceptually different from state table but, the way in which you write this is slightly different so that it can directly be implemented. This is drawn this way so you can have a direct correlation of the hardware that you have in mind. With the signals coming in, activating and giving outputs signals; the transitions occur based on the signals outputs. You can see that this table directly gives you the hardware implementation otherwise there is no conceptual difference. Once you have a state ASM chart you can also draw it in ASM graph and finish the design, there is going to be no more change in the hardware. Since we know several methods of implementing state graph, we will use the most the simplest and the most elegant method, namely the multiplexer based approach.

So, I am going to go through a hardware implementation of this using multiplexers, how it is done is there are 2 to 3 states; each of which requires a flip flop. I want you to recall the multiplexer design on the sequential machines. So we have x and y as two states of D flip flop, the input corresponding to x I will call it D_x flip flop input and the D flip flop corresponding to y I will call it as D_y. Both inputs are turned on by a system clock which has to be shown in the part of the block diagram.

The first time I drew the ASM I did not show this because I only gave the specification problem. R_A and R_B are the 2 input signals G_A and G_B are the 2 output signals expected. Strictly speaking you should have included the clock previously, the final implementation requires a clock and a clock has to be shown. Now how do you drive these flip flops? Using multiplexers; since there are a maximum of 4 states possible we use a 4 to 1 multiplex it here. 4 to 1 MUX, if I want D_x, another 4 to 1 multiplexer corresponding to D_y and inputs to these multiplexers. MUXs come from the conditions that we impose for changing from state to state so corresponding to each state the multiplexer selector output and selector inputs are decided by the present state and the next state is decided by the input that give at the present state.

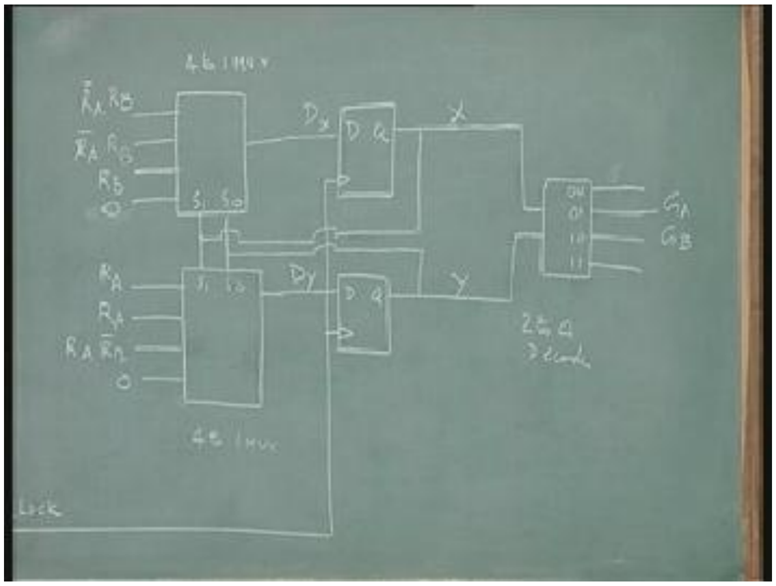
For example, here x and y are 2 variables corresponding to the state variables, now when you are in the present state, idle state which is corresponding to the state variables 0 0 assume that it is 0 0; what condition will make it to 1 0 and what condition will make it to when I remain in 0 0? This is what you have to see. What we need to do is to look at 1 flip flop at a time. First we will look at flip flop D_x this is corresponding to flip flop x and this column corresponds to flip flop y.

(Refer Slide Time: 24:04)



When the present state is 0 0, x is 0 the only condition that which x changes to 1 is R_A bar R_B, in the other 2 conditions x remains 0. So x remains 0 here and x remains 0 here but x changes from 0 to 1 here. I will have to put R_A bar R_B as the input to this which if the condition is 0 0 and if R_A bar R_B comes in as input to this, then the flip flop has to change from 0 to 1. To go over this again it is R_A bar R_B in the second state, which is 0 1 state the only case under which the output becomes 1 is when condition R_A bar R_B occurs and the last thing, the present state is 1 0. The change to next state x is 1 for R_B condition, for other 2 conditions it is 0. So I put R_B here and this is not used so it is the (NC) no connection or you can put a 0 or a 1 to be safe. If you want to be 0 0 it will be safe to put a 0 so in case something happens it goes to the reset state which is the idle state. Similarly, I can quickly do this for the second flip flop y; condition R_A leads to the next state being one for flip flop y. The design is so simple. Here again R_A leads to the next state 1, R_A R_B bar leads to the next state being one. It is again 0 so that any false inputs will land up in 0 0 state, this can be implemented using gates. This is a 2 input gate with R_B bar so, we can draw a gate for this inverter b input R_B input and you can draw a gate with a inverter R_A input and the same output can be used for these 2 inputs. So the couple of 1 gates here with an inverter 2 input and a gate as an inverter 1 2 and a gate as inverter.

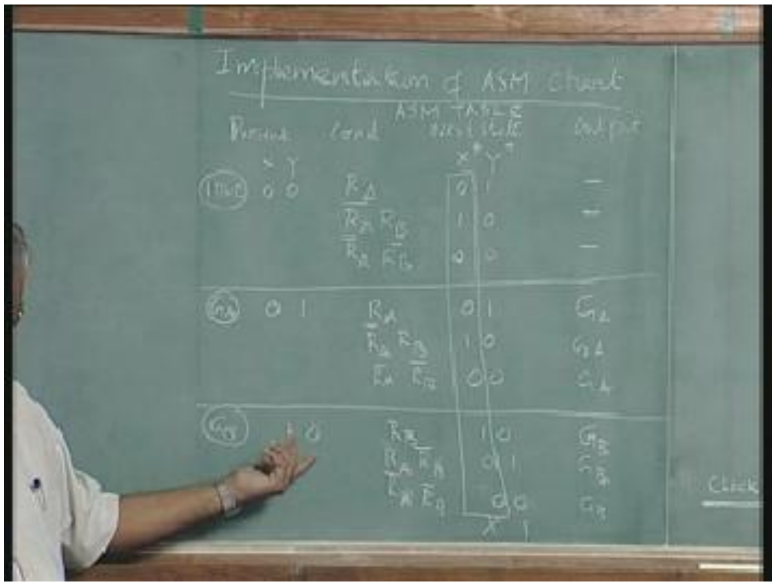
(Refer Slide Time: 26:03)



Totally 2 inputs and gates and 2 inverters will do the entire design and the output is now tied to states which makes it easy. In state 0 0 there is no output, state 0 one the output is G_A and state one 0 the output is G_B. So what we can do to make the design simple even though it is little more expensive in terms of gate count. We have talked about the reduction of number of the ICs compared to the reduction number of gates. The reduction of the number of ICs is more economical and practical solution, especially today in terms of IC design. So I will use a 2 to 4 decoder to which I will give this x and y as inputs and get 4 outputs. This is my 0 0 output, 0 1 output 1 0 output and 1 1 output. The 0 1 output corresponds to G_A and this corresponds to G_B so the design is completed with 2 MUXs, 2 D flip flops 1, 2 to 4 decoder, 2, 2 input AND gates and 2 inverters and other hardware parts. More than the hardware part list, I am not very concerned about it, these will fit into the design once you have the ASM. We almost have hardware translation and this is what makes ASM more popular.

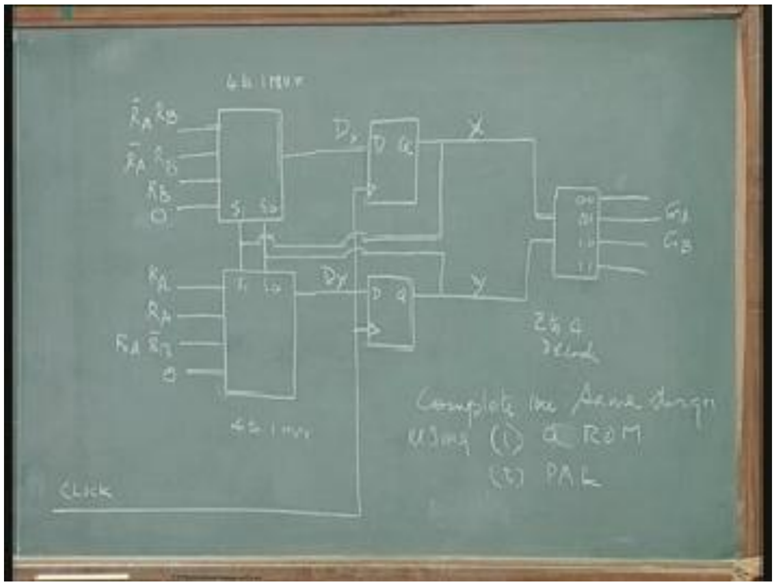
There are other ways in which you can do it; you can use a ROM once you have this table. I can implement this table; it is basically an implementation of the truth table. All I have to make sure is that the outputs that are to be given here x and y, x bar x plus and y plus are inputs given to the D flip flops which drive the flip flops to the next state.

(Refer Slide Time: 27:00)



We can think of this as combinational logic in different conditions, so that we can implement using ROM, we can also implement it using PLA or PALs. All this are familiar to you; I am not going to do it. I will leave it as an exercise for you to complete the same design using a ROM in place. ROM requires external flip flops and PAL does not require that because PAL has flip flops upwards. PAL will take the whole thing into 1 IC including the flip fops, ROM will required the combinational circuit alone which we can implement by a ROM, we will still require 2 flip flops and the output I am going to leave it as an exercise since we are familiar with this from the earlier courses and also the review of this.

(Refer Slide Time: 28:08)



So to summarize, the approach of ASM based design we have this partitioning technique where you have a top down approach. You look at the problem; identify how it can be partitioned and the functional units which are available. We do not have to design them but choose them such they are available readily and also choose them such that we have enough controls.

If you want to clear a register we should have clear register control if you want to increment a register you should have increment inputs. So these are things you should look at. Once you partition the problem you get all the input and output signals required for these functional units. Then go to the controller find out what the external inputs and outputs required are and use these inputs and outputs in the functional units. Identify a clear block diagram with the ASM or the controller, and you have that you go to the ASM chart. ASM chart is the most critical part because it has to be time wise and it has to be done properly. So you have to time it properly for example this is an important step I would say R_A will have to be kept on, G_A will have to be kept on as long as R_A is high.

Suppose I did not mention this your design would be, you have R_A true G_A and then verify R_B and then go back that is what you will design the first time. If we did it this way R_A would be kept on for a while whereas this is only 1 clock period. Every state the circuit or the system remains for 1 clock period so at the end of 1 clock period it will test R_B and it will find it is not true so it will go back. If I put the problem in a slightly different way, suppose if R_B is also true, R_A will be given and you will find R_B true so it will start giving G_B and G_A is not completed. So, there is only 1 clock period for which G_A is available, for the second clock period G_B will be available which is wrong. You want G_A for as long as R_A is available. If you do not check this, only when R_A is removed completely I will have to keep circulating. This is the cycle concept, the timing concept here which is in built into the ASM.

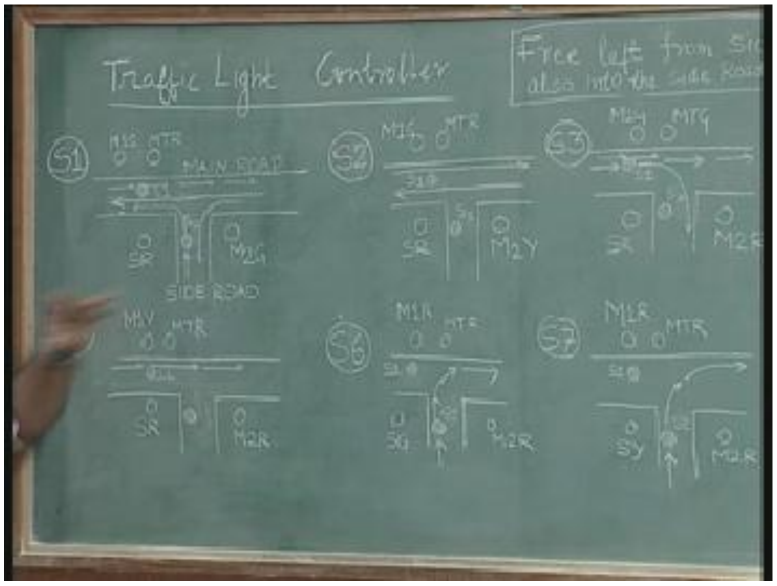
In a state graph it is not that obvious, you can interpret that if you can understand ASM and understand the state graph clearly you can find out that as long as this is in this state category you cannot go out of this state. In a flow chart you do not have the concept at all, in the flow chart you will not know what is happening. In the state graph the difficulty you can have it. Here it is clear so once the ASM is done the rest is easy. So this completes the first example and a very simple one. We will take a couple of more examples in which we will try to introduce a little bit of complexity.

For the second example of ASM based system design we will take the familiar example of a traffic controller. A traffic controller is an example which is used in different types of digital design starting from gate level using PALs for state graph implementations and even for a micro processing implementation.

We will take an example but we will introduce some features which are slightly different from what you will find in text books. It is not that they are more complex but just to tell you to drive, the point that specification is very important in your design. Unless you understand completely what is required you cannot choose the hardware and you cannot draw the ASM. Unless you draw the ASM properly, you cannot implement the circuit. So understanding the specification is the major part of the design effort and another major part of the design effort is the ASM controller.

Other things are of course usually the standard techniques. So with that in mind let us try to understand this problem very clearly. The controller for traffic lights is installed in a road junction where there is a main road and a side road. Under normal circumstances, there will be traffic from left to right as well as right to left in the main road. Traffic coming from the right hand side can turn into side road freely without any light and similarly, the traffic from the side road will turn into the main road from the right to the left without any problem.

(Refer Slide Time: 33:50)



Normally, there is a green signal on both sides of the main roads in both directions and the free left turn from side road and into the side road is also allowed (Refer Slide Time: 33:55). We are assuming that it is a 4 lane road, if there is traffic coming from the left on the main road and you turn on to the side road or if there is a traffic on the side road trying to turn into the right into the main road. Then we have to be careful about the signals so the signal is designed for this type of operation. As I said in the first case we are going to call these lights M1 and M2. M1for main road light in the direction from left to right and M2 for the main road light in the direction from right to left. Also MT for the main traffic, MT for the main road light for traffic turning into the side road and S for side road light for turning from side road into the main road and G stands for green, R stands for red, Y stands for yellow. That means the main road is green in both directions in the first diagram, M1GM2G.

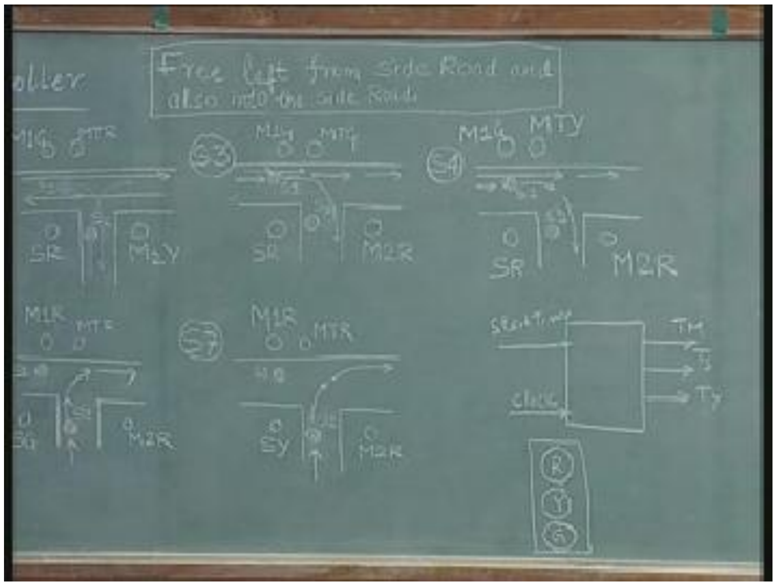
On the side road, the traffic into the turning MT is red. Similarly, on the side road here turning from here into the main road is also red. So this side road is red, this turn signal is red, the main signal on both directions is green. Traffic will flow freely as well as turn into free left. You are assuming that there are 2 sensors here s_1 and s_2 which will sense the presence of traffic which want to turn into this road or from this road into this road. S_1 will send the traffic which decides to turn into this and s_2 will send the traffic to this side. When s_1 and s_2 will be monitored one of these 2 is on, we have made the problem simpler by saying that we do not have to verify which of this is generated, a signal S1 or S2. After a certain amount of time we cannot interrupt the main traffic too frequently. So a particular amount of time has to have elapsed, after which if one of the sensors is active or generates a signal then we will assume that this is a signal requirement for turning from the main road into the side road.

This is an assumption and it has never been a justified assumption. The second sensor may be on and the first sensor may not be on but you can also turn on the second sensor later on. Otherwise it becomes more complex to make it simpler, we have made it like this so whether s_1 or s_2 is on we will assume that first the traffic flows from the main road to the side road and then from the side road to the main road in that sequence. So this has to first of all before allowing the traffic to stop, this traffic can go on and for this to go I have to stop this and this cannot be stopped abruptly. It has to first glow yellow where it was green and where it is glowing yellow this continues to be red now after a specific period of yellow time so that enough time is given for fast moving traffic to slow down on that side.

It becomes red in the third diagram here we will call it s_3; the traffic light will become red here so that it will block the traffic flow through the main road and this will become green. So our main road continues to be green and the turn will also become green near the stopped the traffic light from right to left, this traffic is always stopped, side is red. Now this will be allowed only for a certain amount of time as we cannot allow the side road to go on being green blocking the main road traffic. So at a certain amount of time has elapsed irrespective of whether the sensors are active or not we will take it yellow. This continues to be green as I said and when this is yellow again this has to keep red this will be red. This will be red, this will be green and this will be yellow. After the yellow period we have two choices, if the sensor is still active that means that a car is waiting to turn from the side road into the main road on the right hand side.

We need to accommodate that on the other hand there is no car waiting even though one of the sensors' s_1 or s_2 is on. It so happens that s_1 was the reason for that and there is no traffic waiting in s_2 so we do not have to unnecessarily go and waste our time trying to make the light green on the turn from main side road to the main road right hand side. So at this point we will only check s_2, we will not check s_1. S_1 has already been given a chance and it has to wait for some more time before it can be tested again. You want to test only s_2 now and s_2 is off so that means that no traffic is coming in this direction waiting to turn. We will go back to the original mode of this becoming green and this has already become red so now this red becomes green. So this will be green, this will be green and this will be red. This yellow will become red and this continues to be green, this continues to be red and this red will become green.

(Refer Slide Time: 40:15)



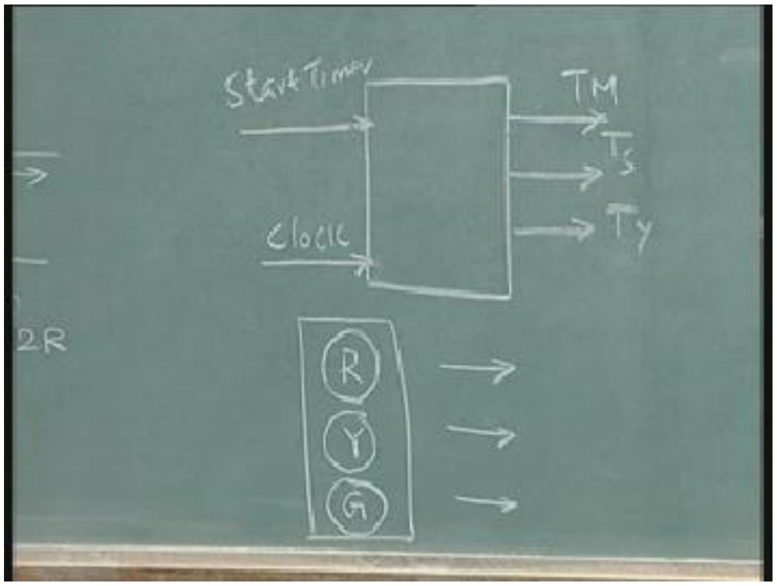
On the other hand it has 2, is on means the traffic is being sensed, we will have to give a chance for this traffic to move into this right hand side. So we will now go through a sequence where we will stop both sides of traffic. Even though when you are turning from the main road into the side road we will let the main traffic pass without any stopping. When you are trying to make the same flow from the side road to the main road you want to be careful because, you do not want the traffic going carelessly and hitting the main road traffic.

We want to stop the main road traffic in both directions and that is again a specification. You have made both here, for the left turn and for the turn from the main road to the side road right turn, we will allow the main road traffic on this direction only, this direction will stop. Whereas, when turning from side road to main road in this direction we will stop traffic in both directions, this one specification you have to be careful to remember. So we will make this yellow and this continues to be red (Refer Slide Time: 41:13), there is no problem and then this is red. This was red and now after this yellow this becomes red and this continues to be red and now we will make this green so that this traffic can go.

This continues to be red until a certain amount of time has elapsed. For this traffic to clear, make this yellow. This again continues to be red for a sufficient amount of time for which yellow is on it will go back to the original mode this being green, this being green, this being red and this being red. Basically, there are 7 different conditions that I have explained. For each of them you will also go through the state graph, the hardware requirements, the functional requirements and the partitioning requirements. How we are going to make it functionally is not a big thing, functionally all we need are lights which are controlled. Lights cannot be directly turned on by a digital circuit with very limited current drive. We will have signals which will operate power devices which will give sufficient current for these lights to be turned on. So the light will be in this fashion or y green for each one of this there will be a light fixture.

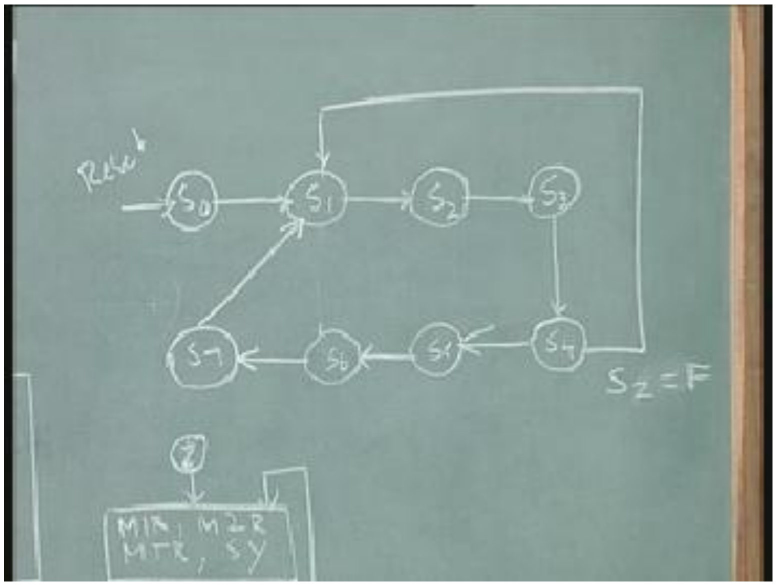
There will be a light fixture here for through traffic, a light fixture for turning traffic a light fixture for side turning traffic and a light fixture for main road traffic. For the turn we will put an arrow and in addition there will be red, yellow and green; but these will be having an arrow in it. These lights will have an arrow in it so that we know there is a turning signal and the timer is the one which is going to control the lights. Whenever we want to turn, how much time the traffic will be on the main road and how much time the traffic should be on the side road or how much time yellow should be there will be controlled by a timer. A single timer we can start every time we want the timer which is of course clocking. So it will be a simple counter and that you can set the time you need. These are exclusive events that are not mutually exclusive events. So there is no overlap between each event. Between any of these 2 intervals or any of these intervals as long as there is no overlap between any of these intervals I can have the same counter which can repeatedly reset and started.

(Refer Slide Time: 44:08)



Every time I start the timer it will start from 0 and after a particular amount of time has elapsed it will become yellow. After a particular amount of time has elapsed this becomes Ts after which there is side road duration. Side road green duration, main road green duration, yellow duration and these are as I said mutually exclusive. A quick look at the graph model just for us to understand what is happening. This is the reset state, where we are waiting for where this timer has to be turned on and then we have all this 7 states we have discussed above. All the 7 different conditions s_1 s_2, s_3, s_4, s_5, s_6, s_7 correspond to that, from s_4 there is a way out to s_1. If the sensor s_2 is false then we do not go through s_5, s_6 and s_7, we can directly go to s_1 otherwise we will go through full sequence to s_7 and come back to s_1. This is the ASM chart, you wait in the state called reset state, there is a start in the beginning and you start the timer.

(Refer Slide Time: 44:54)

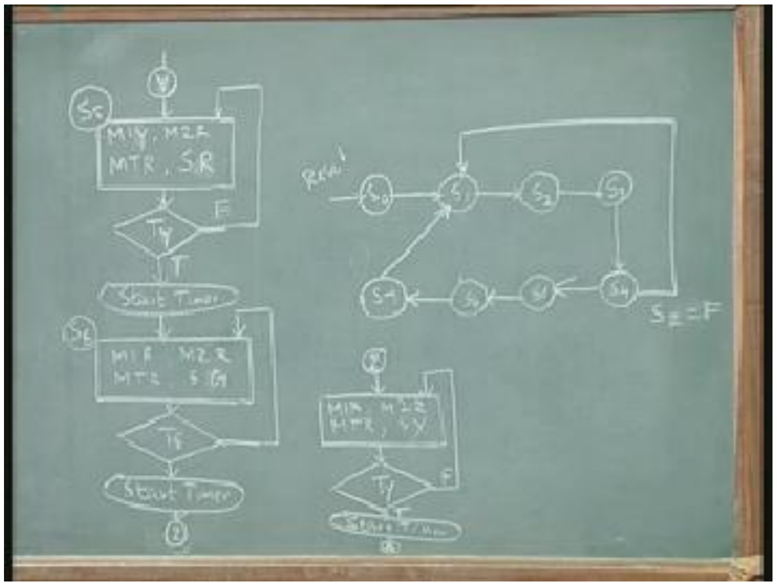


Similar nomenclature is used here; M1 for main from left to right mM2 for main right to left. So M1 and M2 are the 2 lights in the main road this is left to right and this is right to left, t is for the turn signal S is for the side signal. So all you have to look at are these indicators here, M1, M2, T and S will always be in all boxes which represent green, green, red, red and green, yellow, red and red. So first it will be green, each of these states corresponds to one of those diagrams we discussed earlier. These will be in this state until the main right duration is completed. So as long as the main light duration is not over or If one of the signals s_1 or s_2 is not on, you continue to be in the main road flow for green from left to right green from left to right.

So M1G M2G is normal mode and tm is the time elapsed in main road without interruption, if after tm one of $s_1$or s_2 becomes on then it goes to the next state where we go through the main road, green M1 green, M2 yellow turning red again. So every time we start a timer because here we have to monitor yellow here we have to monitor the tm so we start a timer. We start a timer because we need to monitor the yellow timing, until the yellow timing elapses it will continue to be in this state. If the yellow time is elapsed we will start the timer again and go to the next state where main road one is green, main road two becomes red from yellow. Now that it has become red that means turn from main to right which is the turn from main to side is now allowed with a green signal. Side to main is not allowed its red, again this period is the duration for which you want the side light to be on.

Ts stand for the side road light duration and as long as this is not over it will continue in the state. When it is over, you start the timer goes to the next state. The main 1 is green, main 2 is red, turn signal has to become yellow and side to main will be red. Again yellow signal will be monitored, if this yellow direction is not over it continues to be in this state, if it is over it goes to the next state where I introduce the concept of s_2. If you do not want to check s_2 you can go through this whole cycle without s_2.

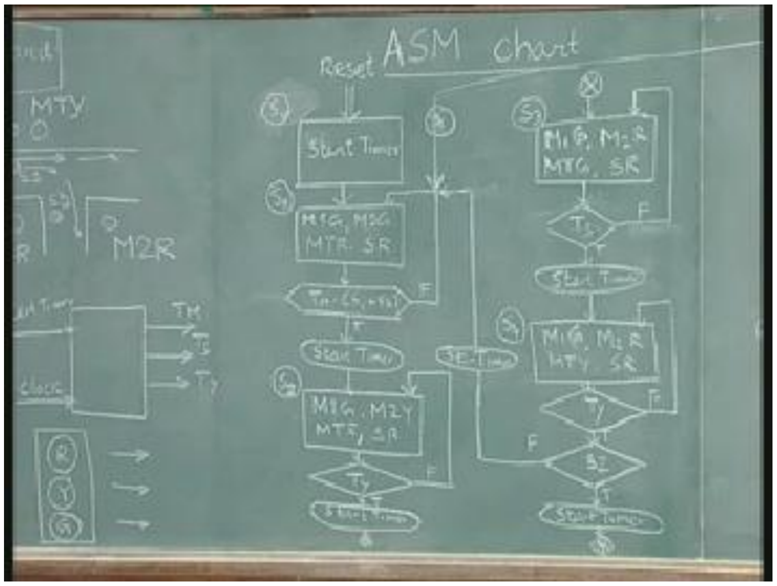
(Refer Slide Time: 48:11)



If s_2 is not on we have completed the turn from the main road to the side road right turn and if s_2 is not on then no traffic is waiting to turn from side road to the main road. There is no need to go through the other states; I can go back to the original because the main road traffic is the important traffic. So at s_2 is false we will go here and we have to start the timer. On the other hand if s_2 is true I have to go through one more light where the main road is yellow, main turn is already red M2 that is the right to left is already red and side continues to be red.

So yellow duration is monitored, during the yellow duration the main 1 remains yellow, after that duration start the timer again and main road from left to right becomes red that means both the sides main road traffic is blocked, turn and traffic from side to main is also blocked. The side road is free to turn from side road to main road, the green duration of that is Ts which is the duration for the side road turning into the main road. This Ts is monitored, once completed it goes through yellow because you cannot abruptly start from start from green to red. It goes to yellow M1 red, M2 red turn side red becomes yellow. After the yellow period elapses it goes back to the original state, this is the continuation to the original state.

(Refer Slide Time: 51:50)



This is the starting point; this has to be connected here. I can draw it like this (Refer Slide Time: 50:00). I did not draw it in the original because it will confuse you. This point is same as this point. The ASM goes like this from here to here to here; with the provision to branch off at this point as shown here (Refer Slide Time: 50:30). So the problem specification must be clearly understood if you want you draw diagrams for each one of these conditions. Clearly identify the hardware required; in this case hardware requirement is lights and the timer. That is all that you need and once you understand the problem clearly and identify what the signals are. The start time is the only input signal to this functional unit; tm and ts are the only outputs to the functional unit. The rest of the signals are the inputs to the, you monitor Tm and Ts, ty. This is the command from the controller to the functional unit, the status of the functional unit to the ASM and then the outputs are all these are outputs.

Whatever you put inside so we have followed the same procedure, the problem is not really a big problem; I intentionally made these minor things unconventionally. Only 1 side seat, free left is allowed and in this case traffic flow is allowed in one direction even when there is turn on other direction traffic is not allowed. All these unconventional flows are in here so that you will completely understand the specification before you put your design into ASM.

So understanding this specification is a very important step to just to illustrate this point. We have unnecessarily put some conditions which may not be really logical. We will see the implementation of this using multiplexer based design in the next lecture.