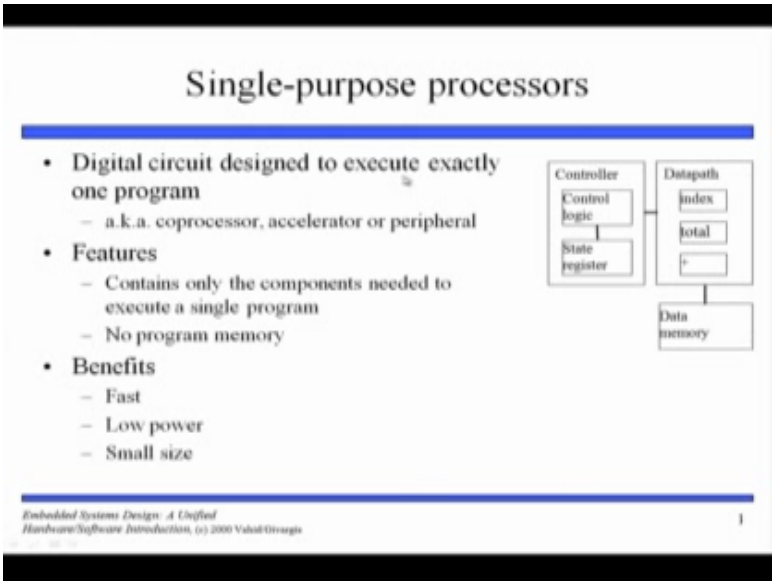
Embedded Systems Design
Prof. Anupam Basu
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 04
Designing a Single Purpose Processor

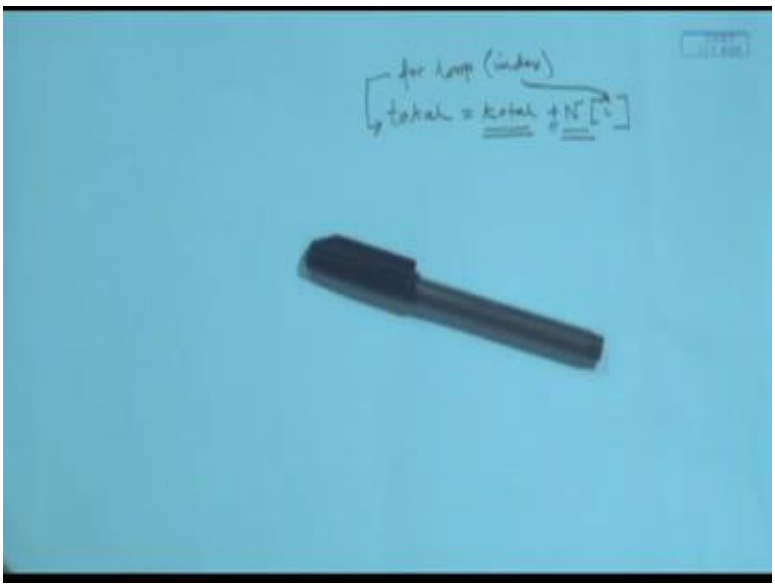
In the last class, we have discussed about application specific instruction processors and gave DSP as an example for that microcontroller to be another example for that.

(Refer Slide Time: 00:41)



Now, today we will be discussing about another extreme that is single purpose processor. Now the single purpose processors are essentially designed for only one purpose and this is not expected that it will be applied to other applications.

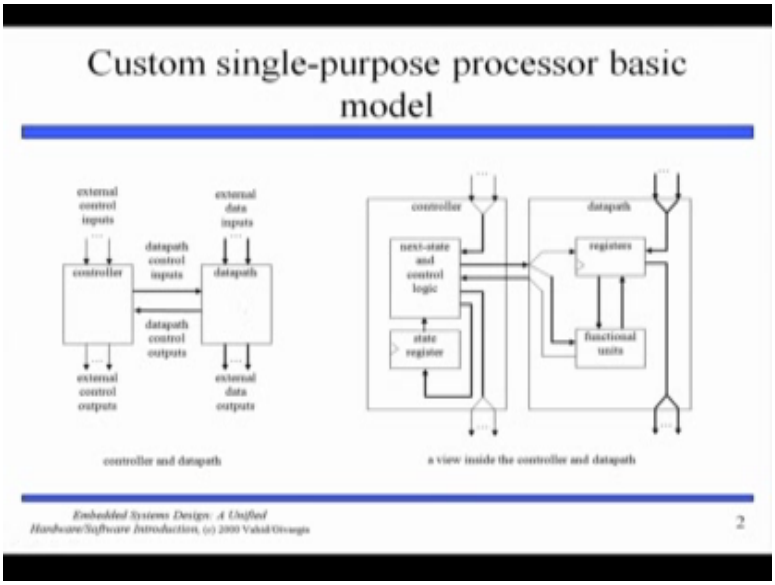
(Refer Slide Time: 01:16)



If we look at this our earlier example of just adding an array of numbers like say total is total class N i and this is something that I am doing in a for loop, if I go on doing this then for this for loop I need an index right I need some index for the for loop and I need a variable total and some data memory and this index is the same as this index, are all of you with me?

If I want to implement this then it is sufficient to have a data path which need not have many registers or many variables many locations, it will have one register to hold index, another register to hold the total; the total amount, one arithmetic operator that is addition I do not need an ALU for that and data memory which is holding that array N that is all. And my controller will be a very simple controller which will have control logic and a state register which will execute this look. So, that is much simpler and dedicated.

(Refer Slide Time: 02:43)

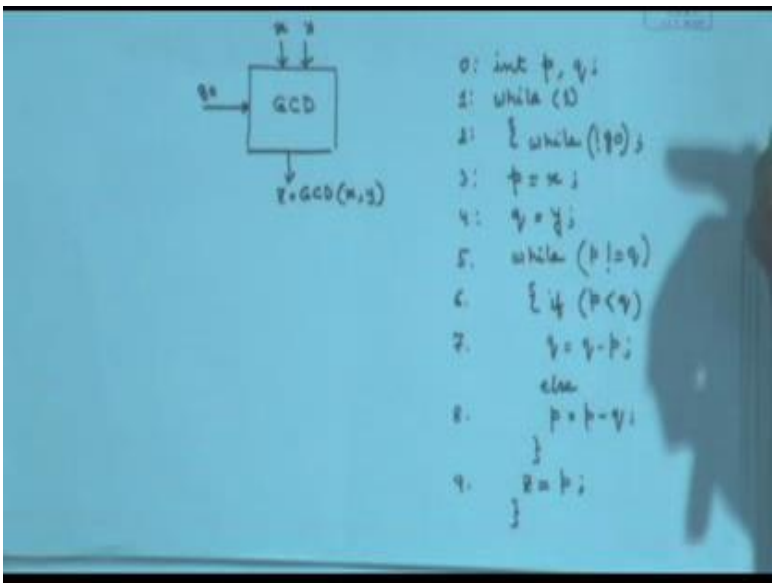


And if going to the next 1, you can see that I can also look at a custom single purpose basic model to look like this, where I have got a controller and a data path for any circuit or any system that we develop there will be 2 parts; control path and data path.

The control path will have external control inputs and external control outputs will go on. Now the data path; there will be data inputs and external data outputs and the data path and control path will be interacting with each other. For example, if you look at this diagram, the controller will consist of a next state and control logic all of you have done state machine design, where you have got the next state table. So, the system is in a particular state and given a particular input you translate to another state. So, there will be inputs coming to this next state and on the data path will have registers and some functional units that is what we have seen.

Now, we will try to design a simple single purpose processor and in the process we look at the process by which it is done.

(Refer Slide Time: 04:19)

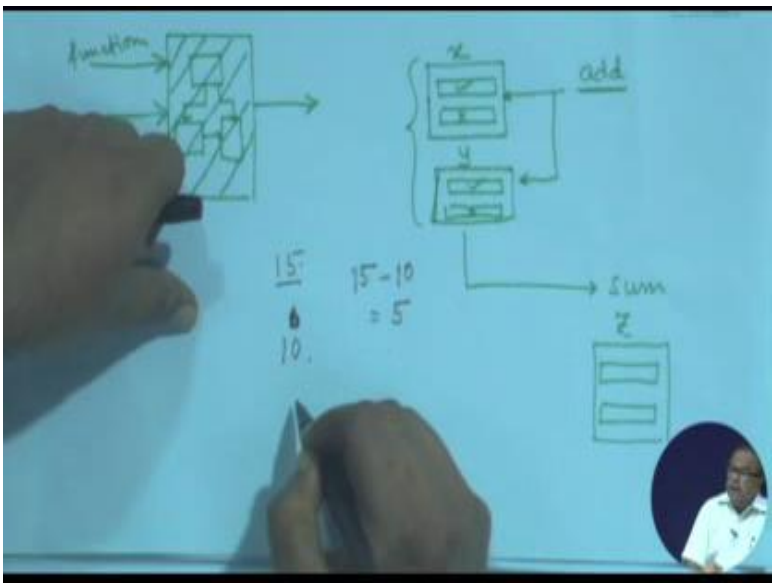


Suppose we want to design a GCD, a GCD; all of you know or HCF, greatest common divisor which will have 2 inputs x and y and will produce z which is the GCD of x and y and will start finding the GCD when there is a control command go. So, if that be that is my function that is what I want to do or you can also say that is the behavior of the whole thing that I want to achieve.

So, I can describe that in the form of an algorithm as is shown here all of you try to follow when I write int p q; that means, I am talking of some internal registers p and q internal variables while 1 is always true. So, it will be an infinite loop, you remember while loop while 1 means it will go on while not go; that means, go signal has not been put. So, there is a semicolon here; that means, it is waiting here or its looping here its actually looping here right on go what is being done? The internal variable p is getting x and internal variable q is getting y clear.

Now, while p is not equal to q, how do you find out the GCD? I will go on dividing it till it divides perfectly. So, while p is not equal to q, if p is less than q then q is being made q minus p, else p is being made p minus q and this goes on is this algorithm clear to all of you clear all of you.

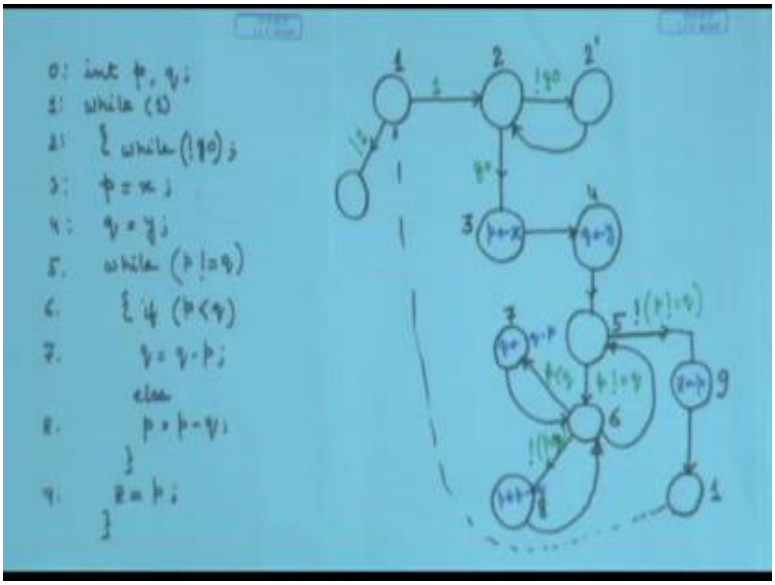
(Refer Slide Time: 06:49)



Just to say 15 and 3, if I do then 15 and 5, 15 and 10 so, these are not equal. So, I will divide, which one is more? 15 is more. So, I will divide 15 minus 10. So, I will get 5. So, what is being done here while p is not equal to q? This is while, this is p and this is q, p is not less than q. So, I make it p assign p minus q and I go on doing this while p is as long as p is not equal to q, ultimately I will come down to 5. So, that will be GCD.

Now, this is the behavior that I have got; now this behavior I have to implement in a single purpose processor that is my objective. How do I go about doing that? Now if I look at this program, this program has got an inherent or there is an embedded state flow inside this.

(Refer Slide Time: 08:19)



I can draw that state machine like this, both of these are being now being kept side by side, you can see this, now let us look at this a little bit I am starting say this declaration part you forget that just a declaration.

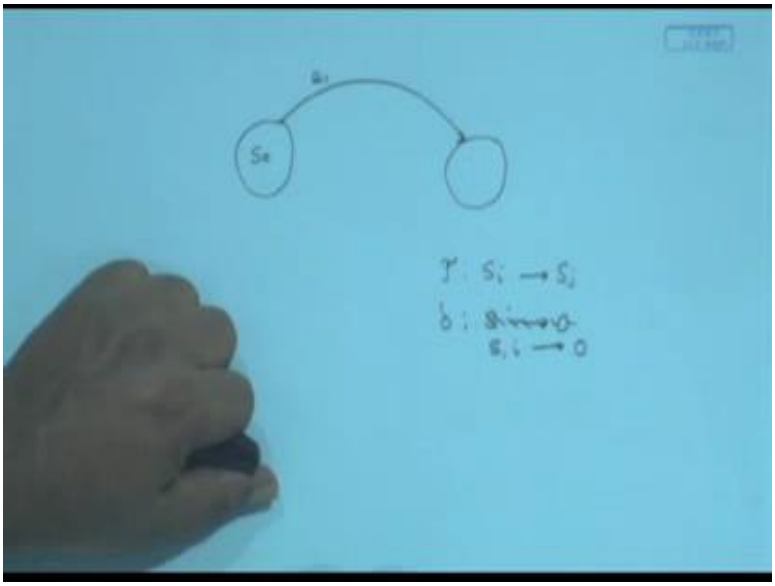
Now, this is state 1. While 1 I am entering the machine and while not 1, I am going out to some other state and then here I am checking go while not go as long as it is not go, what is happening? It is going to 2 dash and is again coming back to this, I could have done it in a self loop also, but I have just since this transition is being explicitly checked, I am creating another auxiliary state for this. So, it goes like this, if it be go then I am coming to this state 3 and what is being done in state 3?

In this program, you can see x is being assigned to p and then I am going to state 4 and there is no event that is driving this transition, what does it mean? That means, it is at that the next clock it is going there right or it can be that as soon as its completed its going there and then I go, I assign y to q. Now I come to state 5, in state 5 what I am doing? I am checking whether p is not equal to q, if p is not equal to q then I am going to 6, otherwise I am going out going out of this loop and coming here to state 9 where z is assigned being assigned p z is the GCD and then I am going to the final again going back to the earlier state if p is not equal to q I am coming here and then 2 conditions can occur

either p is less than q or q is less than p. Accordingly, I am going to say state 7 and state 8 or state 8 and there I am assigning q to be q minus p or p to be p minus q. So, this is a state transition finite state transition diagram for this algorithm now I just give you a minute to just understand this, I am keeping these 2 side by side it is.

Now, here what is this diagram that I have drawn? What is this? No, no, no, it not, state is a finite state machine, but what is a finite state machine? All of you know what a finite state machine is right all of you know.

(Refer Slide Time: 12:03)

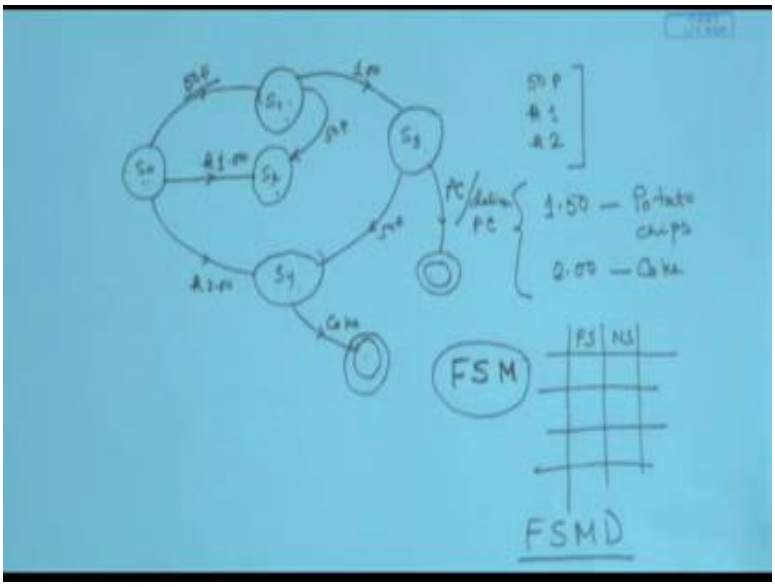


What a finite state machine is a finite state machine consists of a finite number of states and. So, there some states, there some states are there, some inputs there, some outputs and there are 2 types of functions, what are the functions? One is a state transition function say tau that is taking you from one state a psi to S j. Another function can be delta which is an output function that is given a particular state and is giving some particular out and sorry some state and some input maybe it will give some output.

I am in a particular state S 0, on a particular event I can come here. So, that event can be e 1 and there may or may not be some output, it can be say for example, you are using a

vending machine you put a coin, let us draw a quick vending machine state diagram, I mean state diagram or state machine for a vending machine.

(Refer Slide Time: 13:34)



A vending machine suppose takes 50 paise coins and 1 rupee coin or 2 rupee coin, these are 3 possibilities can accept, only these 3 and on 1 rupee, 50 paisa, you get potato chips and on getting 2 rupees, how nice it would have been and for 2 rupees you can get a coke, alright. So, if that be this that be the case in, we start from state S 0 and you have got 3 options, you can put a 50 paisa has been inserted, that is an event or you could have put a 1 rupee or you could have put a 2 rupee.

Accordingly you are coming to another intermediate state and at this intermediate state there is an intermediate state and you are not getting anything here because nothing is there for 50 paisa. When you come to 1 rupee, you are also coming to another intermediate state may be S 1, this is S 2 and suppose you are here, you came here and you place another 50 paise then you come to this state and if from here you pay 1 rupee then you are coming to another state which can be a final state provided. So, you are coming S 1, S 2, S 2, S 3, S 2 is already here S 3 and then you can do one thing, you can say when press the button potato chips and you get the potato chips and that is the end or you have paid 1 rupee 50 paisa and you pay another 50 paisa and you come to this state

because you have paid 2 rupees and then you press coke. So, you come to another final state delivery state.

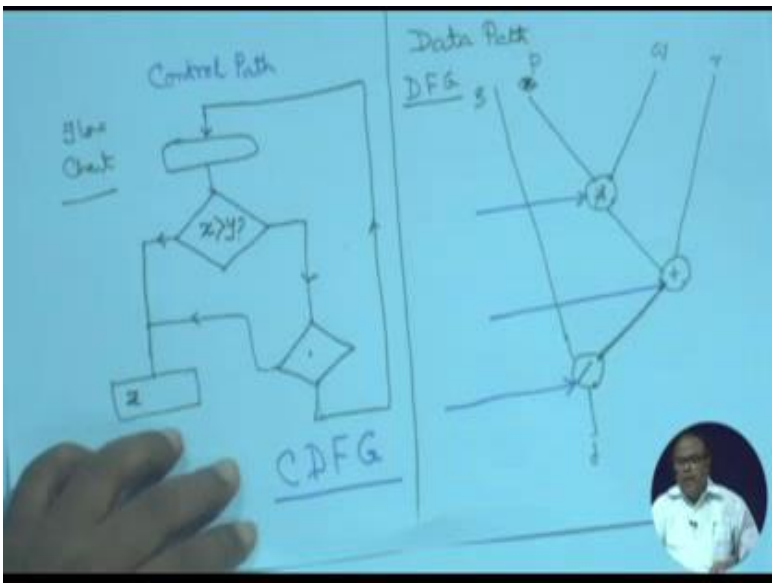
Similarly, from here 1, so from here you can put another 1 rupee or you can complete this, you can further complete this, now this machine has got the states designated the events designated here, for example, and in some cases along with the events when it is making a transition from this state to this state, it is say for example, PC is coming to the state, here there is output deliver delivery; deliver what is this potato chips? I have placed potato chips. So, there is an output for this. So, there can be an output, there can be a state transition for some event and a set of states and some inputs, but what we had? So, that is known as finite state machine.

And how do you represent this? We represent this as a table is known in digital logic class. We make it as a table; each of these states how many states? We have here 1, 2, 3, 4, 5, 6, 7 may be 8, whatever I have to encode, each of these states will do that and accordingly will have a table that if this be my present state and this is the input then this will be the next state, in that way, they will be table and from this table, I can design the controller, this is a basic digital electronics or switching circuits knowledge. The point that the point of distinction that I want to draw is that here we have got an FSM, but the diagram that I have drawn here has got something more.

If you look at this, you can see that I have not only put the states, but I have also put in the data that is being exchanged or data that is being assigned right. So, this is known as finite state machine with data this has got a separate name can you see that that these are the events fine, but here all these blue one all these blue ones are basically data typically in an FSM we do not show data here we are showing both data here, it is we are showing only states, but if we include data here like the one this one this one this is known as FSMD finite state machine with data.

In other words, whenever we design a system, as we have shown that there are 2 parts, one is a control path as a data path, now for the control path, we can have the control flow graph, for example, let me elaborate on this a little bit more.

(Refer Slide Time: 19:57)



We have got control path and data path, you know that control data path consists of the data element multiplexers connections and etcetera control path is for example, if I draw a flowchart that flow chart is nothing, but the control path say I come here and I read something then I take some decision x is greater than y say here you read x read y accordingly I take this path or I take this path right if I come here then I take another decision something and based on that I may go up here.

What is this thing doing? Actually this flowchart; it is actually dictating from which state to which state? The machine should move, if it comes here and meets with a particular condition, it will go back. If the condition is not met maybe it will come to some other computation area, may be here. So, it is actually I am not so much concerned about what is the value of x? What is the value of y? What is the value of any other variable? I am not concerned here, there is some competition on z maybe I am not concerned, I am just concerned in the call flow chart which are the parts which are explored.

On the other hand, the data path is actually showing the data operations, for example, a typical, this is a flow chart and here I can say it is a data flow graph, DFG typically can be like this, suppose there is a some x, do not bother, do not confuse it with this x, let me give it some other name p q r comes, suppose this is a data flow; that means, I have got

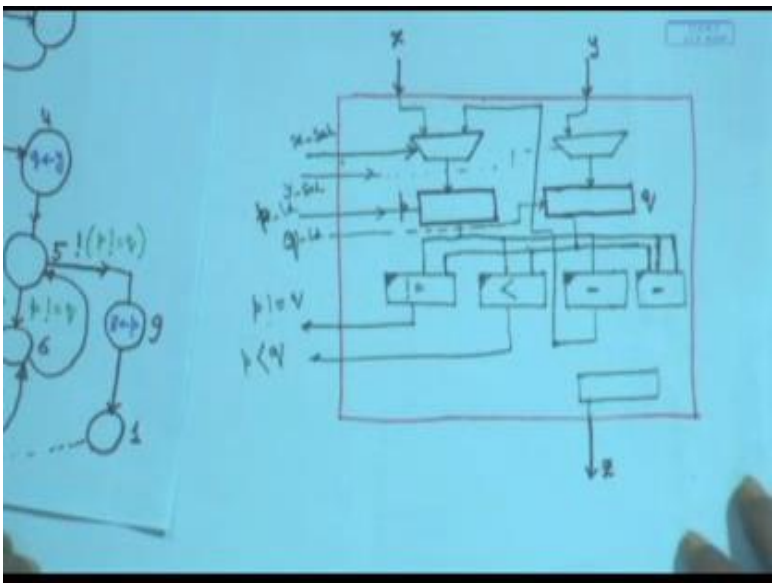
what are there I have got some operators multiplier adder divider and the data flows like this when will device when will this be activated this division I do not know as yet whenever this will come now there are there was now we do not hear about that so much, there was a particular type of architecture that was proposed and became very popular that was known as data flow architecture where they said that we really do not mean know do not need the controller as an when the data is available it will fire.

This operator as soon as it gets p q p times q plus r available here and S available here, this division will fire as soon as p is available q is available then this will fire, but normally those of you have done some design you know that this p or q will be coming on some registers and those registers will be will have some path to the inputs of the multiplier and I am using some other colour they have not given me the red colour. So, we may write in the blue colour this here something will come what is that what is coming here bolo control signal. So, it is the control signal coming saying that they do the multiplication the registers will be then taken here also maybe they else find later that also multiplexes need controls.

Similarly we will need some control signal, here you will need some control signal here like that now these blue lines are all coming from the control path and this data path is being activated by the control path depending on the control logic. So, what is there in our FSMD; in our FSMD in a way, we can say that the control path and data path have both being marched. So, in the parlance of this control path is known as control path and data path combined is known as CDFG control and data flow graph.

Now so this FSMD so we have got an FSMD here, now given this FSMD. So, once again, let us revise, we started from this diagram that we have to design this GCD, we had the detail behavior of the GCD, fine from there, we have developed this FSMD, now we have to design the data path and we will have to design the control path.

(Refer Slide Time: 26:35)



Let us keep this here, while we design the data path. So, the data path of the whole GCD will be something like this x and y will be the inputs and z will be the output. So, we can very well first of all for the different variables that we have can you identify the variable here p and q are the internal variables x y and z are outside. So, we will allocate 2 register for that next thing we will allocate 2 register next we will allocate operators.

The first thing that we allocate is the register then the operator what are the operators here you can see what are the operators not equal to is one let me mark in this way the operators not equal to is 1, what else before that less than is there whether we will do it to the comparator that will decide, but it is a function and the function is a less than function right what else is there? There are 2 substractor; 2 places. So, I am not optimizing, I am just keeping one substractor here, another substractor here. So, we start with this, next what should we do?

Now, the ports will have to be connected; now if I want to connect the ports, how do I connect the ports here? Where will the; say this x, x will have to come to the input of this p because I have to have x being assigned to p. So, should I directly put that x to p, why not because this p can sometimes have p as a right hand side, you can see here, can have p minus q can have x. So, I am very good, I need a multiplexer here since I do not have

different colors, let me put different shape here, I will need 2 multiplexes, these are the 2 multiplexers for this. So, this x will come to this multiplexer, this y will come to this multiplexer and the output of this will come to this p and the output of this will come here.

Now, this multiplex can have will must be say this multiplexer will have another input that is what? Can you tell me what? How will you interconnect this multiplexer? The output of the substractor will have to go there right why p minus q is going to p. So, the output of this substractor will have to go here and accordingly, I will have to put in some control signals here and that control signal could be select x, I call it x select where go is coming at the beginning I am not showing go here.

Now, from here, it can come to this p is coming to this p is coming to this p is coming to the substractor and p is also coming to this substractor this here, sorry, this should go to the right one because this is doing q minus p, this doing q minus p and this q is coming to here as well as; sorry also coming here, it will come here also and here also which one should come this entire thing this thing will also come everything is coming everywhere alright we get the interconnection and we will have an element that is for the output that will come out that will come at the end right now.

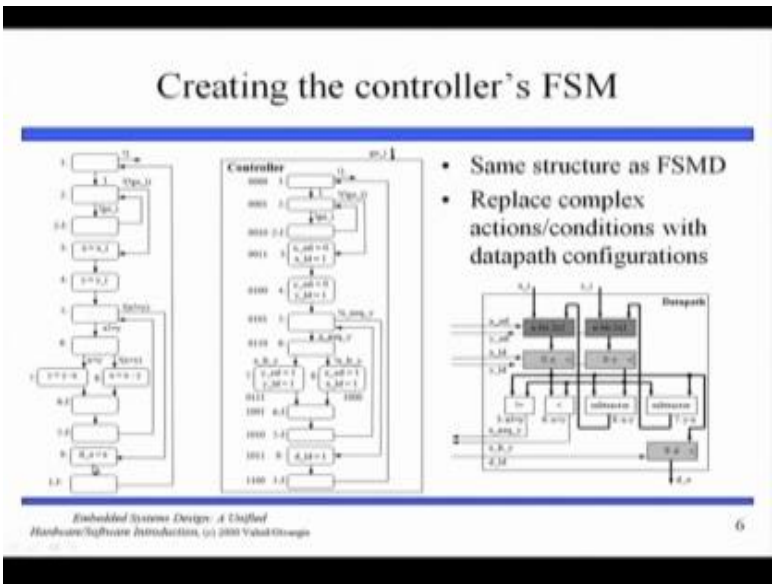
From here I have just shown 1, this thing control signal similarly, it could be y, select this path and go to y select and that will connect to this multiplexer, here what will be the control signals coming here? What will come? What can come here? This p is input can be what either this x or this y. So, it will have either load x or load y is not it this p is either having this or having the subtraction result. So, it will also have some controller control signal coming here 1 is maybe x load another is y load x load coming here and y load y load it is actually going there here, this is going there this one is going there that 1.

And from here, I am getting different information, what are those information? X not equal to y p is not equal to q like that actually I have done some mistake, you see it is this is p and q, p load and q load, I am sorry, p load and q load and here we will find, what this is not equal to? So, this will be p not equal to q, this signal is coming up, similarly from here p less than q will come up, in that way, all these data parts will come,

alright. So, how, what happens is gradually becoming more and more clumsy, but this is a simple circuit we will not of course, do it manually, but what is happening here I have created the data path the elements and I have created the interconnections and on this side I am putting in the control.

Now, when I get the control path then my initial controller specification was here this will be further modified this will be further modified, now 2 things will happen I will quickly show that keep it here.

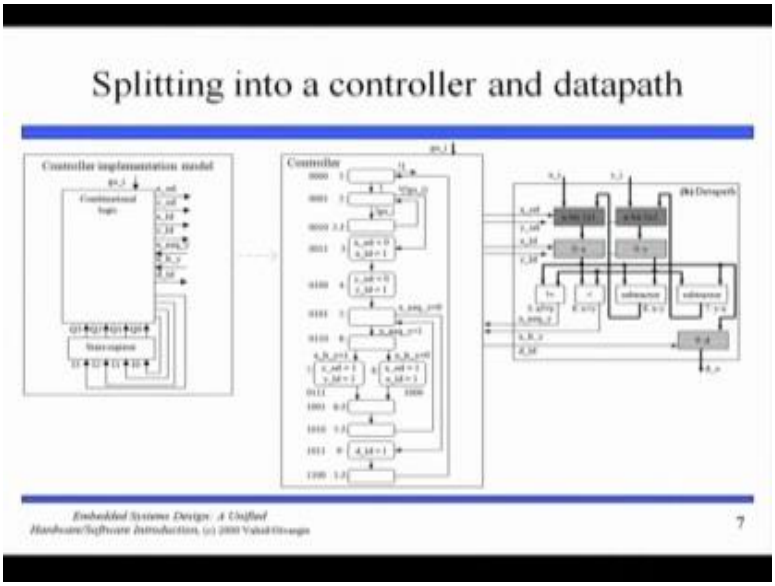
(Refer Slide Time: 35:41)



And I am quickly showing you this, how do you create a controller? See this is exactly the same except for the nomenclature I was using p q and here there is x y this taken from again the book by Frank Vahid which I recommended you, now you see this was my initial thing and I have designed this data path the thing that we did so long this, the one that we have done now this is same as this and from here we have got these control paths coming out control signals. Now the controller which was just the graph has now been state encoded what is it mean how many states were there in this 10 plane 1, 2, 3, 4, 5, 6, 7, 8. So, I need 4 bits for 10 states, if I want to encode I need 4 bits. So, 0, 0, 0, 0, 0, 1, etcetera, etcetera.

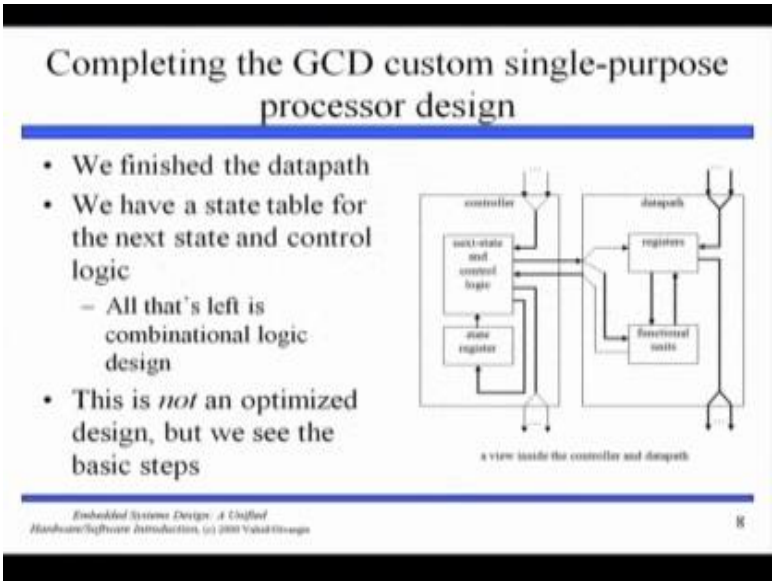
For now within each say for example, in state 3, what was happening? X i x x i was the input or in my case, it was x being assigned to p, in my case it was this one state 3 x is being assigned to p, alright. So, for that in state 3, we have to generate x select and x load that sorry, now this x is actually p, p is getting that I am selecting x and getting the p all these signals will come here. Now here everything has been shown as x i wanted to distinguish between them. So, similarly here are the control signals that are coming up as soon as x l is 0 generated from this state here a 0 comes here x load 1, x load 1 comes here; that means, what this path will be selected and it will be loaded here clear.

(Refer Slide Time: 37:59)



Similarly, when I find here at 5, I check x is not equal to y or p is not equal to q then I go somewhere else, now i p is not equal to q will drive me out, but p is sorry, p not equal to q is 1; that means, p is not equal to q, will bring me here. So, here I am putting in all the control signals and this part will next be synthesized, this diagram I can either optimize this or I can synthesize it as a finite state machine.

(Refer Slide Time: 38:49)



There by I will be getting the control path as well as the data path both. So, what are the steps that we have seen today? The steps are first we start with the functional diagram like this and then take the behavioral algorithm then convert that behavioral algorithm to an FSMD and then looking at that FSMD carefully we start designing the data path and in order to do the data path what do you do we first for each variable we allocate a register I have not optimized you can see here that there are 2 subtractors, but both the subtractions will not be simultaneously active. So, I could have optimized I have not done that avoided that complication whatever I am I have got here then I have to achieve the interconnections to the multiplexers.

And in order to activate the multiplexers and the data elements, I need some control signals then I move back to the FSMD and do this state allocation and also write down what are the signals that should be generated this signals will be just as I had shown an FSM is making state transition why making that state transition, it is generating the outputs and those will be this x l x select x load and once I have this, I will make a table out of this and then I will put it synthesize it.

Now, for complicated machines we will of course, will not do it by hand that will be error prone. So, we will have we have got several synthesizers state machine synthesizes

which we will use. So, this is the way we go up about design a custom IC custom processor; single purpose processor.