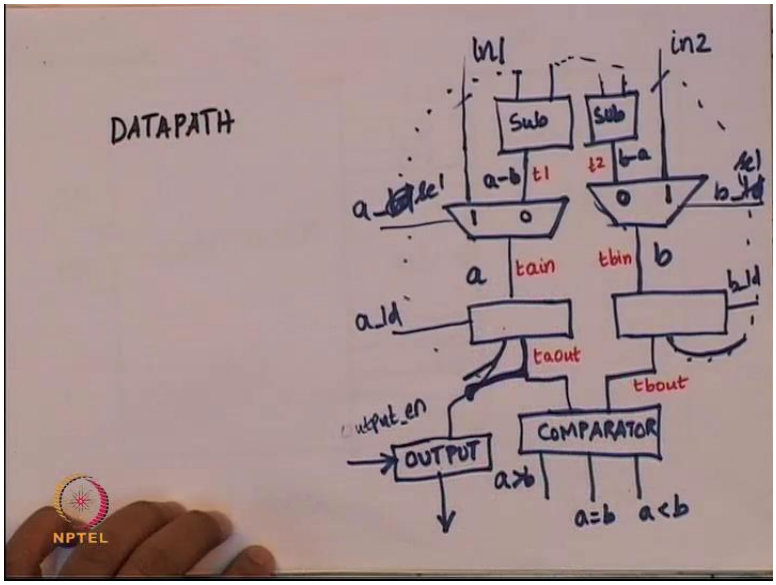
Digital Circuits and Systems
Prof. Shankar Balachandran
Department of Electrical Engineering
Indian Institute of Technology, Bombay
And
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module - 39
Datapath Greybox view

Hi, in the last module I did the data path diagram and I started writing the variable of description for the data path. So, in this module what will see is as a grey box, which means we know the connections, but we still do not know the internal implementation of the various modules for inside the data path. So, what I am going to do is, I am going to do a description of the data path itself as though it is picking various modules, and connecting them together. Just like what we did for the top most design I am going to do something very similar for the data path itself.

(Refer Slide Time: 00:52)

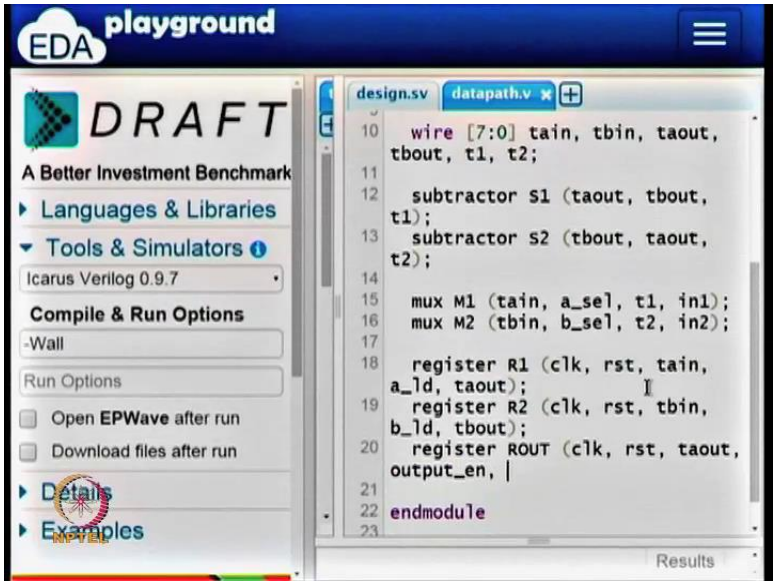


So, let us look at the picture, I have now name the wires. So, I am going to call this wire t 1 and this wire t 2, this is ta in tb in, this is ta out and tb out, this is register a and that this register b, this is register called output. So, we had all of these earlier.

(Refer Slide Time: 01:08)

So, let us go and write the descriptions for it. So, I left it at this point in the last class. So, let us see the picture. From the picture of the first thing we need to do is, the subtractor is suppose to take one of the subtractor is suppose to take ta out as the one of the inputs, tb out as the second input, and it is suppose to produce t 1 as the output. So, I will write a subtractor s 1 which takes ta out as the first input, t b out as the second input and its suppose to produce t 1 as the output. So, that the first subtarctor.

(Refer Slide Time: 01:54)



Then let us look at this the second subtractor here, the second subtractor should be subtactor s 2 which suppose to take tb out as the first input, ta out as the second input and its suppose to produce t 2 as the output. So, we have taken care of connections that are coming into the subtractor and out of the subtarctor.

So, again let us look at the picture. So, now we are suppose to design the mugs. So, the mug is suppose to take t 1 as the 0 line and in one as the one line, and a select as the select, ta in suppose to be the output, this mux is suppose to take t 2 as the 0 line, t 1 as the in 2 line, in 2 line as the one line, and the select is b select, and the output is tb in. So, we need two mux is… So, I am going to assume that mux are given in this form. So, in the select line, the output that is going to come from it and the two inputs, that is supposed to be supply to it.

So, I am going to assume that the mugs. So, I will call that mux m 1, the first mux m 1 is suppose to given output. So, the output is actually ta in. So, we have the output which is ta in, I will give the select line which is a select, we have the a select, then you have the two inputs. In this case is suppose to be t 1 and in 1. So, t 1 is the in 1 input, and in 1 is the sorry t 1 is the in 0 input, and in 1 goes to the in ones input.

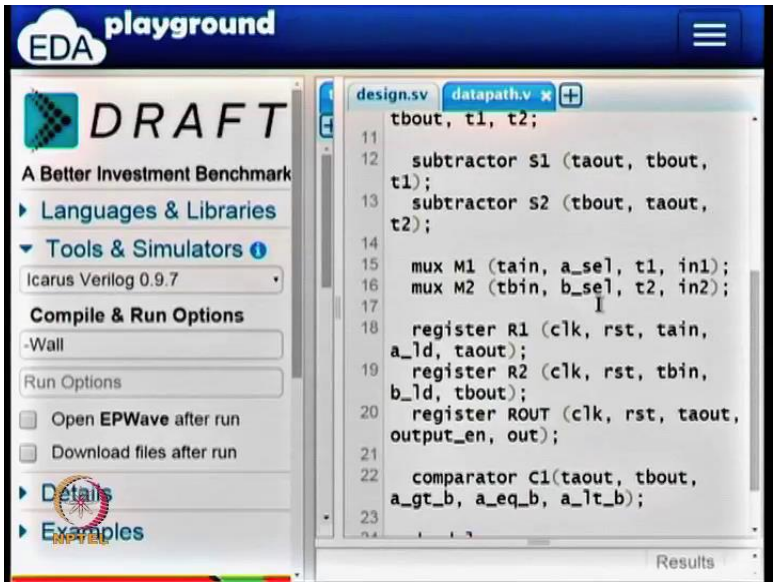
Similarly we have mux m 2 . So, again looks at the picture mux m 2, this is suppose to give tb in as the output. So, tb in as the output then we have at the 0 pin we have… So, the select line is called b select, we have b select, at the 0 line we have t 2 and at the one line we have in 2. So, I will put t 2 and in 2. So, you have taking care of four different things here, you are taking care of the subtractors, we are taking care of the multiplexer. Now we need to take care of the registers, there are three registers and for the registers what I we need is? So, I know these are the bunch of flip flops. So, they of course will take clock and reset as the set of inputs that you need, so these are two inputs that you need for them, and besides that there is an input that is coming in, there is an output and there is a load enable. So, this load enable if is on ta out is suppose to copy ta in.

So, that is the thing that we wanted. So, when I describe the verilog description when I did the data path design as I said we do not want the output changing in every cycle, we have to retain the values for one of them when the gcd is running either this will be kept constant or this will be kept constant till the next subtraction happens right. So, we need the load signals. So, I will assume that all of these load signals and other things are there

in a module called a register, and for this register what I am going to do is, I will write the design description right now. So, I will assume that somebody's going to give me a design description for a register. So, I have register R 1 which is suppose to take clock as an input, because it is a sequential circuit reset as another input. So, clock in reset are given, then its suppose to take ta in as an input, and the load enable as an input, and give ta out as a output.

So, I will write in that order ta in a load and ta out. Similarly, register R 2 will take log reset, will take tb in as the input, we load as the load enable signal, and tb out as the output. And finally, we need register which is for the output itself. So, there is this output register, we need a register for that. So, I am going to call that a register R out, and R out is clock reset, the input to the register is actually ta out, the enable signal is actually coming from the external things, this output enable is suppose to come from the controller. So, I am going to put that here, output enable, and the actual output is suppose to be driven to the output itself which is this one, which is called out.

(Refer Slide Time: 07:05)



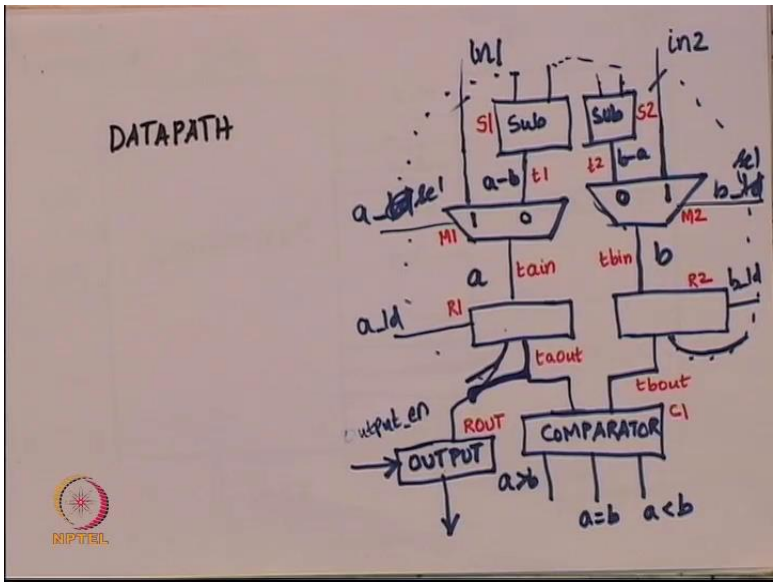
So, I am go to put that and this takes care of the registers, the only other thing that we need is the comparator, the comparator… So, let us again look at the picture. So, I will call the comparator, let us say C 1. So, this comparator is suppose to take two inputs; in 1 and in 2, and suppose to give three outputs in this order. So, I am going to put that as three different things. So, the input that is suppose to take or ta out and tb out; these are

the two data inputs, that is suppose to take and there are three outputs namely a greater than b, a equal to b, and a less than b. This actually completes the designs of the data path. So, what we have is we know that the data path is suppose to contain two subtractors, three registers, and two muxers, and one comparator. And we have already put in the design description that is required.

So, again from a top level view if you have this picture, all we have done is taken the picture that we have drawn and put that on the, put that in the verilog form, that is it. We still do not know how to we have an written that description for a subtractor, for a multiplexer, and so on, but that is we do not need to right now. So, I am coming top down and when I am coming top down have decided varies thing that I need, now as a design team right, let us say I am going to built a design team to do a chip, I am may actually tell one engineer to go and design a subtractor, one engineer to go and design a multiplexer and one not. So, usually these are very, very small things, you do not give them to different engineers, but just as an example if I want different engineers to take up different portions, I can tell them see this is a design description that I need, functionality wise this is what I want. And these are suppose to be the input, these are suppose to be the output go ahead and design.

Once they are again collect all of that back together, and already I have the design description which is suppose to put all of them together and connect them together, I will be ready to run the design.

(Refer Slide Time: 09:15)



So, let us take one last look at the picture here. So, in this picture what we do not have so far is that names. So, I had different names earlier, now I put the names in the picture itself. So, this subtractor I call s 1, this subtractor I call s 2, this multiplexer I called m 1, this multiplexer I called m 2, this register I called R 1, this one I called R 2, this one was called R out with the capital OUT, and this one comparative was called C 1. So, these are the names that were given to the different modules. And these modules are instantiated right now in the verilog description, these instantiation have to be the return up, we do not have the design descriptions yet. We only have the instantiation at the data path. We now have to go and right what a subtractor is, what a multiplexer is, what a comparator is, and then what a register is, and these are things that at least a multiplexer and register are things that you are already know. So, I am going to show you how to write a subtractor, and how to write a comparator. So, we will see that in the next module.