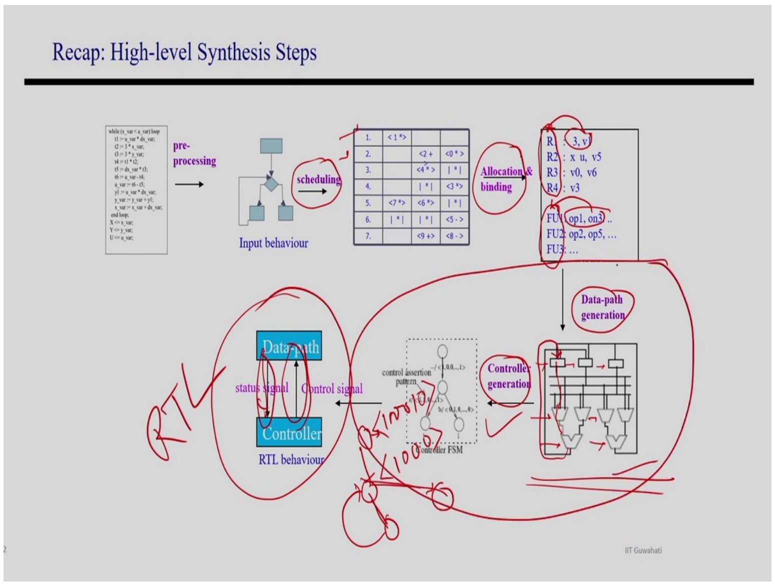
C-Based VLSI Design

Dr. Chandan Karfa
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Module - 06
C-Based VLSI Design: Allocation, Binding, Data-path and Controller Generation
Lecture - 22
Data-path and Controller Synthesis in High-level Synthesis

Welcome everyone to this course C-Based VLSI Design. In today's class, we are going to learn about Data Path and Controller Synthesis in High-level Synthesis. So, let us again look into this high-level synthesis flow.

(Refer Slide Time: 01:07)



Recap: High-level Synthesis Steps

So, if we have already seen that high-level synthesis consists of pre-processing, scheduling, allocation binding, data path, and controller generation then finally, we will get the RTL right. So, we have already have discussed about this scheduling and also allocation binding. So, scheduling assign the operations to time steps.

So, we understand what operation is going to execute in which time step and then, allocation and binding actually identify the number of registers needed and how these variables get mapped to registers, how many FU's needed and how the operations are getting mapped to FU's right. So, after this allocation and binding, we already have that information with us. So, we know how many registers are there and how many FU's are there and exactly, how the operations and variables are mapped to them that is also note to know.

So, the next phase is data path and controller generation. So, in this step so, we have to make the connection from this register to FUs right because I know this operation is going to happen in FU and the data should come from registers so, we have to make the connections from the register to FU’s and also once the operation is done in the FU, that data has to go back and to be returned to the registers. So, this is what is the inter-connections and that is what is the data path synthesis.

So, based on this inter-connection requirement, you try to then make the connections right. So, obviously, wants to try to make the connections, your objective will be to optimize the inter-connection cost. So, we will come to that point what is the inter-connection cost and how to optimize it, but the overall idea is that ok.

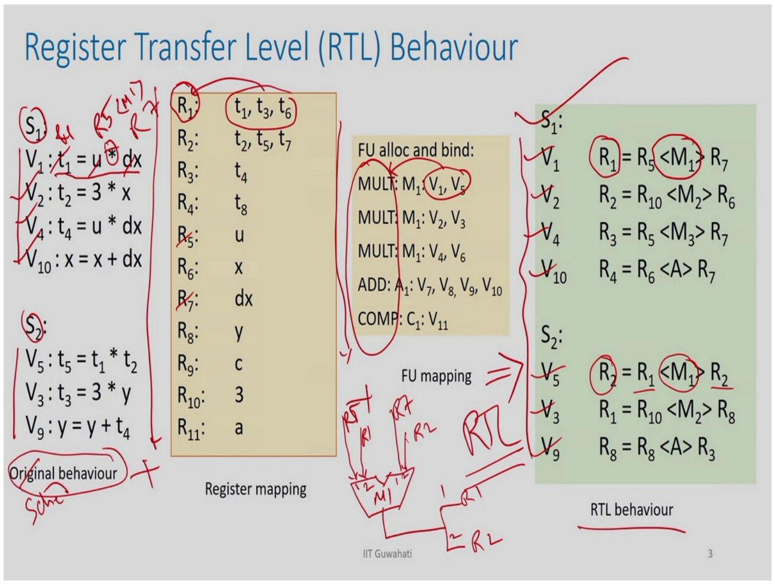
So, once the connection is also made right so, then the controller generation comes. So, we already know how many states are needed to execute the behavior. So, we know the FSM states and the transitions right. So, this is also known. So, what are the states needed and what is the transitions so, these are all known to me.

So, what is this controller FSM does? Because this data path is going to execute this behavior now and this data path inter-connections are basically time-division multiplexed. So, we have to make sure that in every clock so, suppose you are in clock 1, the operation to be executed in operation time step 1 is going to happen in the data path right. So, that is something you have to make by the controller.

So, based on that, it has to assign some proper control signals to this inter-connection units and also to the registers (Refer Time: 03:33) right. So, these are the control signals. You have to assign appropriate the operation that is scheduled in that state right. So, based on that, this controller will identify how many minimum control signals is needed and what is the value of those control signals in every time step right so, that is what is the controller FSM.

So, basically here, it will just identify the value of these control signals and so on right. So, this is what is the controller synthesis. So, once you get both, your final RTL is ready right so, this is your final RTL. It consists of a data path and a controller and they actually do handshaking, this controller generates the control signals for the data path to perform operations and the data path gives some status signal based on which the step transition actually happening in the controller FSM ok. So, this is what is the high-level synthesis flow. In today's class, we are going to learn about this phase ok.

(Refer Slide Time: 04:34)



Register Transfer Level (RTL) Behaviour

S_1:
$V_1 : t_1 = u * dx$
$V_2 : t_2 = 3 * x$
$V_4 : t_4 = u * dx$
$V_{10} : x = x + dx$

S_2:
$V_5 : t_5 = t_1 * t_2$
$V_3 : t_3 = 3 * y$
$V_9 : y = y + t_4$

Original behaviour

R_1:	t_1, t_3, t_6
R_2:	t_2, t_5, t_7
R_3:	t_4
R_4:	t_8
R_5:	u
R_6:	x
R_7:	dx
R_8:	y
R_9:	c
R_{10}:	3
R_{11}:	a

Register mapping

FU alloc and bind:
MULT: M_1: V_1, V_5
MULT: M_1: V_2, V_3
MULT: M_1: V_4, V_6
ADD: A_1: V_7, V_8, V_9, V_{10}
COMP: C_1: V_{11}

FU mapping

S_1:
V_1 $R_1 = R_5 <M_1> R_7$
V_2 $R_2 = R_{10} <M_2> R_6$
V_4 $R_3 = R_5 <M_3> R_7$
V_{10} $R_4 = R_6 <A> R_7$

S_2:
V_5 $R_2 = R_1 <M_1> R_2$
V_3 $R_1 = R_{10} <M_2> R_8$
V_9 $R_8 = R_8 <A> R_3$

RTL behaviour

IIT Guwahati 3

So, let us move on. So, as I mentioned in the previous slide that after scheduling, we know in every time step what operation is going to happen. So, this is my schedule information, after register allocation and binding we will know how many registers are needed and what is the mapping of the variable to the registers and after FU allocation and binding, we know how many FUs we need and what are the operation is going to map to each FU right. So, this information is it there.

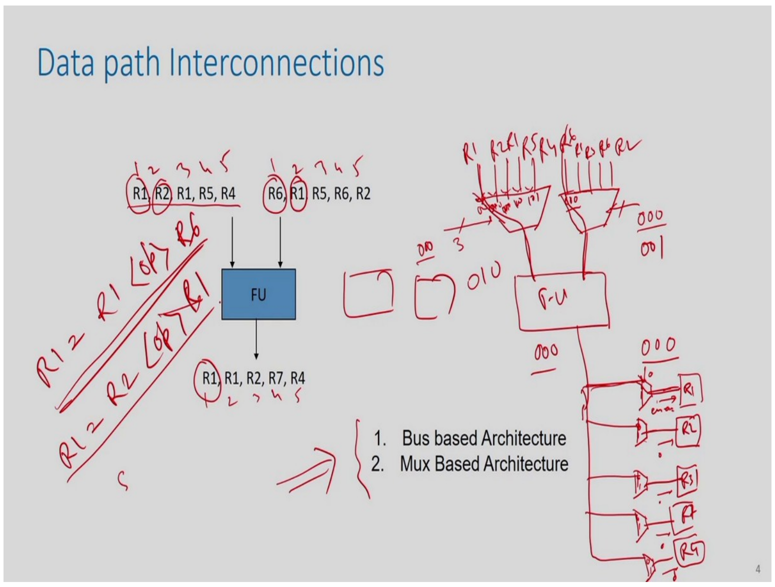
So, now, we can combine this and this and we can get the RTL behavior ok. What I can do? I can rewrite my schedule behavior which is schedule behavior which I can combine this allocation and binding information and then, I will get the register transfer level behavior how, what is there?

So, in every time step so, in the time step 1, there is an operation $t_1 = u * dx$ right. So, how t1 is stored in the hardware? It is R_1. So, I can replace this by R_1 this u store in R_5 so, I can replace this by R_5 and dx store in R_7 so, I can replace by this R_7 and this operation 1 is actually mapped into V_1 is mapped to operation MULT 1(M_1.

So, I can replace this multiplier with (M_1) right. So, this is my register transfer level behavior. So, this is why I can replace all the operations by the corresponding registers and the FUs and I will get this behavior right. So, this is what is called RTL behavior. So, here, it is of transfer of register values that behavior is represented using the transfer or register values, that is why it is the RTL behavior.

So, here, it is just given as the operations, but you have to realize this RTL behavior using a data path and that is what you are going to get in this data path synthesis diagram. So, if you want to get a data path that way we are going to realize this RTL behavior which is written as a text file ok.

(Refer Slide Time: 06:34)



Data path Interconnections

1. Bus based Architecture
2. Mux Based Architecture

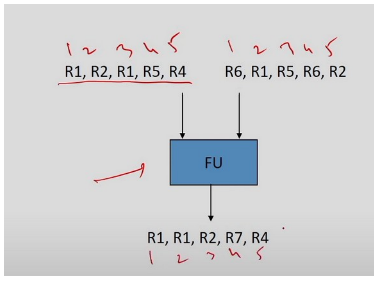
So, now, let us understand what is this data path interconnection synthesis problem. So, as I mentioned so, we already know that these are the operation that is going to happen in time step 1 in the register transfer level, and these are the operation that is going to happen in the operation time step 2 in the register transfer level so on. So, you may; you might have many states, I am just taking two states for explanation purposes ok.

So, what we can do for each FU, you can identify what are the inputs possible inputs to this FU in the overall all-time step right. So, if you just think about M_1, for M_1 the so, suppose this is your FU M_1 which is a multiplier 1 so, in time step 1, R_5 will come here right. So, R_5 will come here, and R_7 will come here right so, this is in time step 1

(R_1=R_5 <M_1> R_7). In time step 2, this is again M_1 and R_1 will come here right so, this is R_1 will come here in time step 2 and here, it will R_2 right (R_2=R_1 <M_1> R_2).

So, this how if here I have taken only two states, but if suppose there are 20 states so, every time step, you can identify the inputs of that FU over all time step right. So, if you just do this for each FU, we will get such behavior right. So, you will understand these are the possible values of inputs. So, this may be in time step 1, 2 so, 3, 4, 5, 1, 2, 3, 4, 5 and this is the output right so, 1, 2, 3, 4, 5.

(Refer Slide Time:08:04)

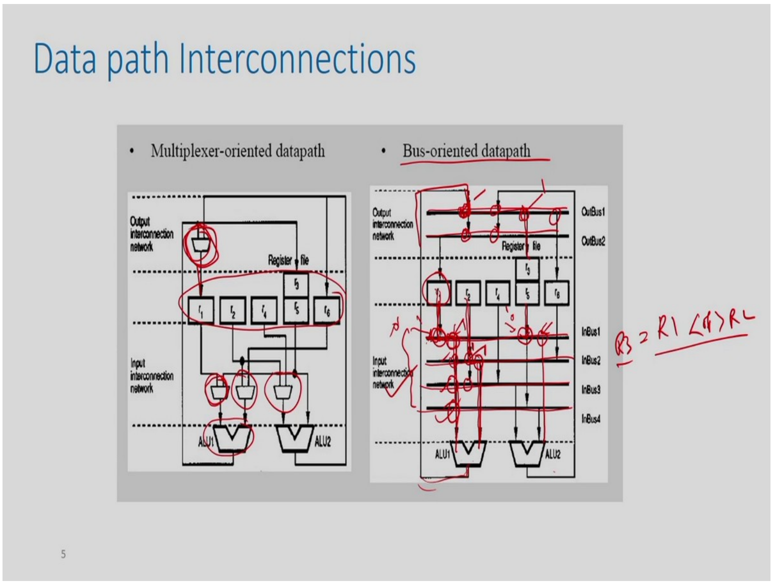


In the previous behavior, what I have not done is basically this will go to multiple places right. So, this is updating R_1 in the time step 1, this is updating in time step 1, and this is updating R_2 in time step 2 right. So, you will and identify all the possible inputs for the FUs and what are the possible outputs of these FUs and their actual time division multiplex. It is not that all are coming in the same clock, they will come in different time step right.

So, this information we can gather from this RTL behavior, and this is what does it been, I have to make connections from this R_1, R_2 and this register to the left input of this right and similarly, I have to make connections from these registers to the right input of this FU.

I have to make connection from this FU output to those registers such that every clock I have to give proper control signal so that that particular operation can happen ok. So, the make the connections, there are two possibilities. So, the possible data path interconnections are bus-based architecture and mux-based architecture. Let us understand that.

(Refer Slide Time: 09:26)



Data path Interconnections

- Multiplexer-oriented datapath
- Bus-oriented datapath

So, in the mux-based architecture which is multiplexer-based architecture, see you have the FU right and you have the set of registers, you know these things right so, this is given, and I know what are the input should come to a ALU. So, you add a multiplexer here just to multiplex the input.

So, in the previous example if you take, for this FU right, you put a multiplexer at the left input. So, you put a multiplexer at each input of the FU right and then, you have R_1, R_2, R_1, R_5, R_4 right and then, you need so, there are four in; five inputs so, you need kind of 3 bits control signal right so, there are 3 bits control signal.

And similarly, here you also put five inputs basically, this is 000, this is 010, this is 011, this is 100, this is 101 say and similar to this. So, here, you are putting R_6, R_1, R_5, R_6 and R_2. So, and again you need a 3-bit control signal. And then in the first clock, you give 000 right and here also, you give 000 right.

So, you give 000 here, here you need to give 000 and similarly, this output is going to this so, basically, this will go to all this register so, there should be some enable signal to the registers and there should be a mux at the input of the register because I just talked about one FU, but it maybe that the input to the register can come from multiple FU. So, you have to put a multiplexer at the input of the registers also right.

So, that is this mux-based architecture. So, you have this multiplexer at the input of the FUs, you have multiplexer at the input of the registers, we got the register input can come from many FUs and to the input of the ALU can also come from many registers ok. So, this is why you have to put this multiplexer.

And then, you have to give appropriate, you have to assign control signals and you have to make sure that the operation that has to be executed in that particular control step, you give that corresponding control signal, then things will happen right. So, for example, suppose you have this is say R_1, let me just complete this diagram. So, R_2, R_3 sorry this is R_7 and this is R_4.

And let us assume there is a mux at the input of this registers right and this is one of the inputs to this so, since I have only taken one FU so, I am just making one connection, there may be some other FUs right, for that FU, the other connection will come and also, there will be some enable signal. It may be that in some state, you do not want to update one register at all right so, you need some enable signal as well ok.

So, now, suppose you give the answer this is so, this is a single bit and this is 01, 01, 01, 01, this is 01 ok so, just for simplicity I have assumed that. So, now, in the first control step in cs 1, in time step 1 because I want to put R_1 here, R_6 here and R_1 here so, the operation is actually $R_1 = R_1$ <op> R_6 right, this is going to happen.

So, if you put 000 here so, this R_1 will come to this place, if you put 000 control signal, this R_6 will come here and since you give a 0 here so, this will basically go to this register right. So, and for and you have to give the enable equal to 1 other enable is 0 right. So, if you do this, this particular resist person will do $R_1 = R_1$ <op> R_6 that is the operation is expected right.

In the time step 1, you should give 010 here, here sorry this will be 001, I mean you understand that and here, you have to give 0 here and others will be 0 right. So, if you give that, it will do $R_1= R_2+ R_1$ so, R_2 <op> R_1. So, this is how the connections will happen right.

This is what is a mux-based architecture that you take, you put mux at the inputs of the registers as well as the FUs and based on the connection requirement, you put the control signals and during the execution of the controller FSM, you give the assign the proper control signal so that the expected operation is going to happen in that FUs ok.

The alternative architecture is called bus-based architecture ok. In the bus-based architecture, we are not making any multiplexers so, so what we are doing? We are taking a fixed set of buses ok so, we are going to take fix set of buses and the bus we are going to connect these registers and the FUs based on the connections.

So, basically, we know one bus, if I connect say this r_1 and then, there will be a switch here ok so, so that if the switch is on, then this data, this bus will take and say I have also connected r_6 to this bus_1 ok so, that means, if I put this value 0 and this is 1, then r_5 value will come to bus_1 right.

If I put this is 0, this is 1, then this r_1 value will come to this bus ok. So, you cannot put more than 1,1 at a time step, then it is a race condition, there are two data coming to the same bus so, it will be an unstable c-section right. So, at and in a control step, only one of the inputs can give the data ok.

Similarly, if this r_1 needs to transfer to this ALU_1 so, I will make a connection from this bus to this and I will put a switch and say from this bus_1, say I want to make a connection to this an ALU_2 also, I will make a connection like this and I will put a switch here and then, what is going to happen?

So, in say time step 1, the see if I put this 1 also that means, and say this is 1 so, what is going to happen this r_1 will come here and then, from this switch it will come to this. So, the left input of this FU will be r_1. Similarly, since this r_2 is also coming to this switch and this bus so, r_2 can come here and this is also connected here. So, suppose I put this control signal, you control come to this control signal 1 so, then we r_2 will come through this line to this ALU right. So, this way so, that means, R_1 <op> R_2 is going to happen right.

And similarly, at the input of the register also need to add some buses and this FU will be connected to this and there will be switch. So, every connection is basically a switch and this register also switch and say suppose I want to put this data into say r_3 so, what I am going to do? I am going to put this equal to 1 and this equal to 1 so, that means, this output of this FU_1 will come through this line, it will come here and through this you will go to r_3 right. So, then it is basically $R_3=R_1<op>R_2$.

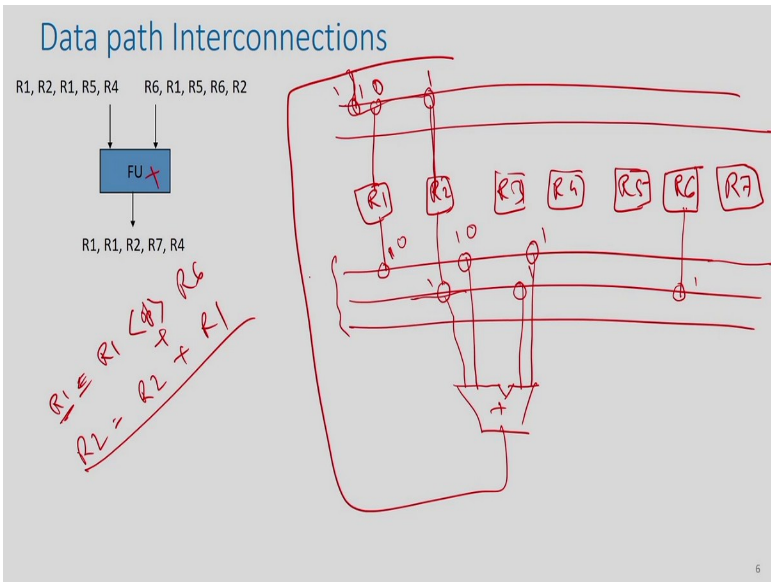
So, this is how I can make the connections and I will add the switch in every place and then, I want to make sure that in every time step, based on the transfer required, the actual RTL operations I am going to set the bus signal to 1 and as a result, the corresponding register input will come to the FU and based on the destination address, based on the (Refer Time: 17:41) register, I am going to make the switch connection accordingly so that the output of the FU will go to the actual registers right. So, this is what is the bus-based connections.

Here, obviously, idea is that you use a minimum number of switches because you cannot add all switches, then it is too much of switches and cost. So, you have to make sure that you first determine how many minimum buses is needed and for each FU input, what are the line it should be connected. So, you have to identify the minimum number of buses and a minimum number of switches for each component, for each register, and what is the bus needs to be connected. You identify the minimum such occurrences right.

Similarly, for each FU, it has to connected to how many, what is the minimum number of buses it needs to be connected? Similarly, the output of the FU, it can be connected to both or it may be connected only one based on the requirement. So, your optimization objective is to identify the minimum number of buses and minimum number of switches ok.

So, that is the optimization objective whereas, here, your optimization objective is to identify the minimum number of muxes, what is the minimum number of muxes needed that is the optimization objective. We will discuss this in detail in this class. So, we understand this. So, we have already explained this for these connections, what is this mux-based connection so, I can actually explain again for this what is the bus-based connection.

(Refer Slide Time: 19:02)



So, say suppose you have a FU here right so, this is your FU right and your register say R_1, R_2, R_3, R_4, R_5, R_6 right R_7 and suppose so, basically here you know you have to execute R_1 = R_1 <op> whatever the FU, suppose this is plus so, it will be R_1=R_1+ R_6 right. So, suppose there are three buses here right so, to make sure, I have to make this R_1 connection so, I can make a switch here so that this will come to bus_1, and I have to also make a switch here so that this R_1 will come here.

Similarly, R_6 unit so, you cannot put into this bus because in the first time step you need this data so, this R_6 must be connected to the other bus and similarly, you need a connection here as well. So, then if you put 1 and 1 here, 1 and 1 here so, this data R_6 will come here.

Suppose there are two buses here as well ok and this FU is connected here and the output source go to R_1 so, what is going to happen? This must be connected to one of the bus and from there, there will be a connection here and if you put it 1 and 1 here so, then this so, in this time step, this R_1 = R_1 + R_6 is going to happen.

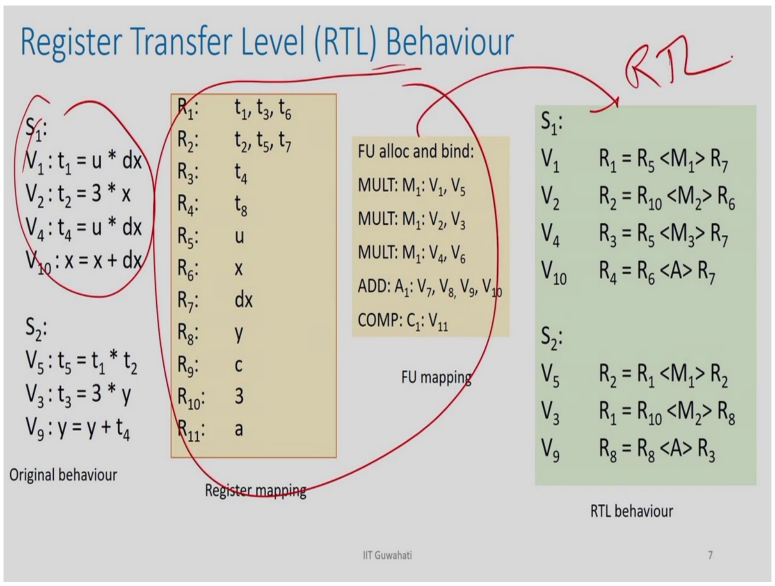
And in the next time step, the operation is R_2 = R_2 + R_1 right so, that means, here you need R_2 at this input. So, what we can do? From R_2, say suppose you make a connection here, you make a connection here and you make a 1 here and then, this will come here right so, then in the next step, you put this 0 so that only this R_2 will come here and from

R_1, you make another connection here and you put it 1 here, 1 here so, this will put come on here.

So, through this bus, R_1 will come here and through the second bus, R_2 will come here and these outputs will go into R_2 so, I can put again a switch here which is already there so, I can put 1 here and 1 here so, then this output now and this is 0 now so, then this output will come through this right. So, this is how I am going to make the connections.

As I have already explained that here the objective will be to identify what is the minimum number of buses and how to assign this variable to these buses so that there is no overlap or contradiction and the use of buses is minimum right. So, that is something is the objective during bus-based interconnection generation ok.

(Refer Slide Time: 22:01)


Register Transfer Level (RTL) Behaviour

S_1:
$V_1 : t_1 = u * dx$
$V_2 : t_2 = 3 * x$
$V_4 : t_4 = u * dx$
$V_{10} : x = x + dx$

S_2:
$V_5 : t_5 = t_1 * t_2$
$V_3 : t_3 = 3 * y$
$V_9 : y = y + t_4$

Original behaviour

R_1:	t_1, t_3, t_6
R_2:	t_2, t_5, t_7
R_3:	t_4
R_4:	t_8
R_5:	u
R_6:	x
R_7:	dx
R_8:	y
R_9:	c
R_{10}:	3
R_{11}:	a

Register mapping

FU alloc and bind:
MULT: M_1: V_1, V_5
MULT: M_1: V_2, V_3
MULT: M_1: V_4, V_6
ADD: A_1: V_7, V_8, V_9, V_{10}
COMP: C_1: V_{11}

FU mapping

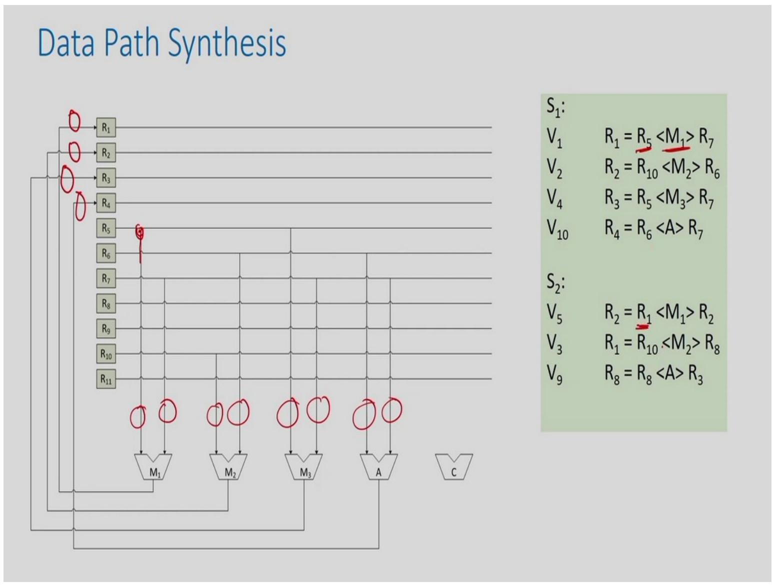
S_1:	
V_1	$R_1 = R_5 <M_1> R_7$
V_2	$R_2 = R_{10} <M_2> R_6$
V_4	$R_3 = R_5 <M_3> R_7$
V_{10}	$R_4 = R_6 <A> R_7$
S_2:	
V_5	$R_2 = R_1 <M_1> R_2$
V_3	$R_1 = R_{10} <M_2> R_8$
V_9	$R_8 = R_8 <A> R_3$

RTL behaviour

IIT Guwahati 7

So, for this example, I will just try to show you how the mux-based connections happen very quickly. So, we have already discussed that from this, if I combine this and this; if I combine this and this, I will get this RTL behavior right that I have already discussed.

(Refer Slide Time: 22:17)



Data Path Synthesis

S$_1$:

V$_1$ $R_1 = R_5 <M_1> R_7$

V$_2$ $R_2 = R_{10} <M_2> R_6$

V$_4$ $R_3 = R_5 <M_3> R_7$

V$_{10}$ $R_4 = R_6 <A> R_7$

S$_2$:

V$_5$ $R_2 = R_1 <M_1> R_2$

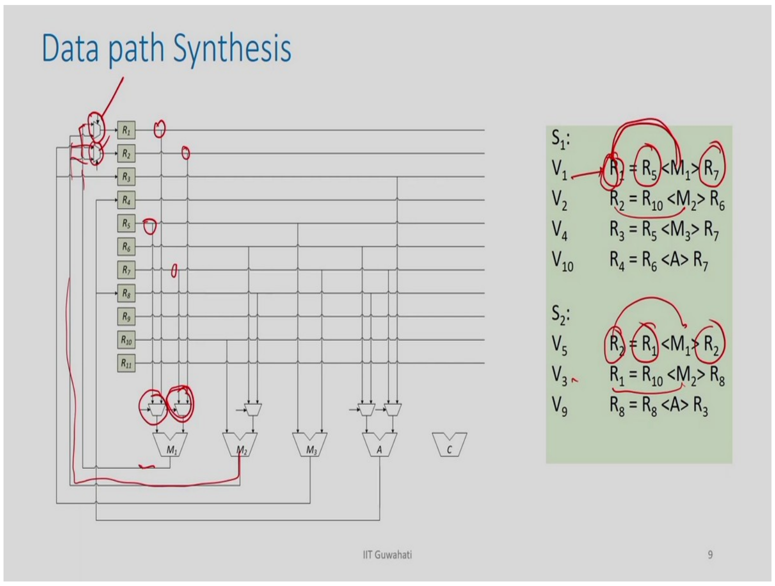
V$_3$ $R_1 = R_{10} <M_2> R_8$

V$_9$ $R_8 = R_8 <A> R_3$

And suppose I have this RTL behavior is with me now, and I have to make the connections, the mux-based connections so, here this FUs are there, there are 5 FUs and these are the 11 registers and I have to put basically mux here right so, I have to put mux here because there are many operations coming here.

And you have to put a mux at the input of the registers right. So, have to put mux here. So, let us see for operation 1, R$_5$ will come to the FU so, so this R$_5$ will come here so, this is a fixed line and in the next time step, R$_1$ will come right.

(Refer Slide Time: 23:03)



Data path Synthesis

S$_1$:

V$_1$ $R_1 = R_5 <M_1> R_7$

V$_2$ $R_2 = R_{10} <M_2> R_6$

V$_4$ $R_3 = R_5 <M_3> R_7$

V$_{10}$ $R_4 = R_6 <A> R_7$

S$_2$:

V$_5$ $R_2 = R_1 <M_1> R_2$

V$_3$ $R_1 = R_{10} <M_2> R_8$

V$_9$ $R_8 = R_8 <A> R_3$

IIT Guwahati

9

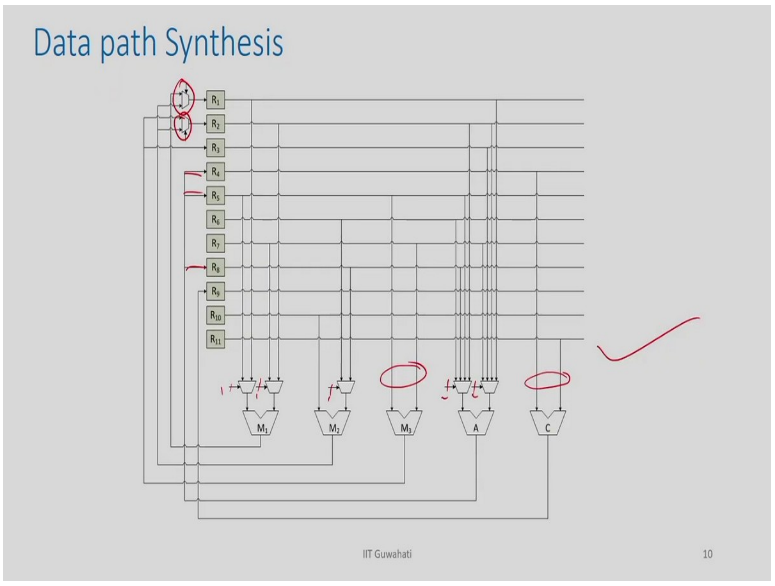
So, what you have to do is basically there is a mux, you have to put here in the first time step you need R_5 and next time step you need R_1 so, this will be a multiplex between R_5 and R_1. At the right-hand input, it is a multiplex between R_7 and R_2 so, I am putting R_7 here and R_2 here. So, this is multiplexing between R_7 and R_2. At the output, this will go to R_1 and this will go to R_2 right. So, they will be connected.

So, now, for this mux, how I will decide? I will see one R_1. So, this R_1 is coming from M_1 right and in this time step, R_1 is coming from M_2. So, there will be a mux here and one of the inputs will be from M_1 and another input is from M_2 right. So, this is how the connections will be decided.

For R_2, here it is coming from M_2 and here it is coming from M_1 right. So, again there is a mux, you have to place at the input of R_2 and there will be two lines, one is from M_1 and one is from function unit M_2 right. So, this is how you have to see for each time step, what are the possible inputs are coming and based on that, you put a mux here and for each registers input, you see how many.

So, for every time step, what are the FUs from which the inputs are coming, based on that you have to put the muxes here and if you do this for all step, you will get some connection like this. So, here I have just shown for two FUs, you can do all rest of the things, I am not completing the whole complete discussion.

(Refer Slide Time: 24:30)

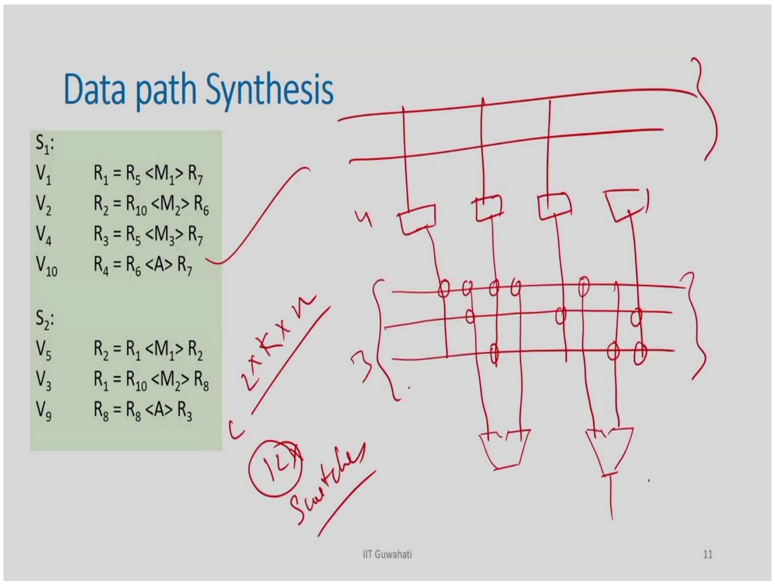


Data path Synthesis

But if you just complete this for complete behavior, you will get a connection like this, since this I got this. So, you can see here that we need a mux where the control signal will be 1 bit, it is 1 bit, it is 1 bit, but here it is 2 bit, it is 2 bit because there are 4 possible inputs ok.

Here there is no mux needed because this only one variable is coming all the time ok and similarly, here we need a mux of size two and this is a direct connection we did not need any mux ok. So, based on this connection requirement, you can identify where this interconnection and identify the mux sizes ok. So, this is what is all about.

(Refer Slide Time: 25:19)



Data path Synthesis

S_1:

V_1	$R_1 = R_5 <M_1> R_7$
V_2	$R_2 = R_{10} <M_2> R_6$
V_4	$R_3 = R_5 <M_3> R_7$
V_{10}	$R_4 = R_6 <A> R_7$

S_2:

V_5	$R_2 = R_1 <M_1> R_2$
V_3	$R_1 = R_{10} <M_2> R_8$
V_9	$R_8 = R_8 <A> R_3$

IIT Guwahati 11

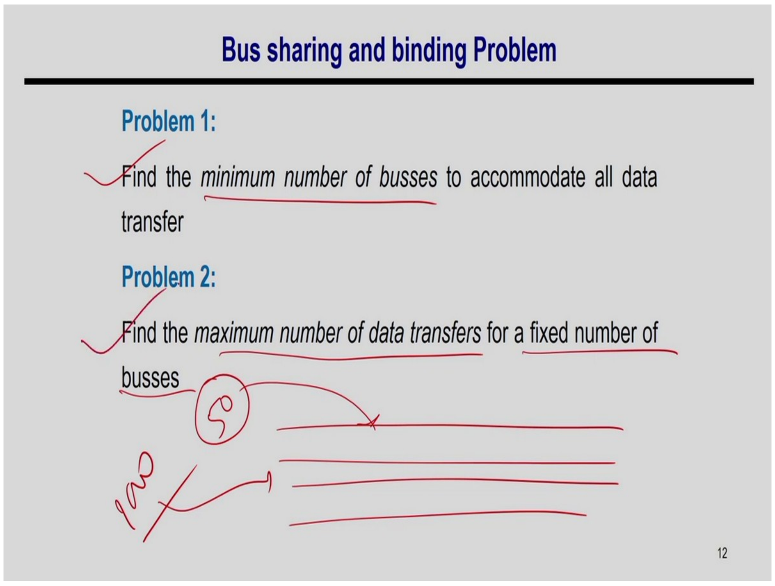
So, for this bus-based, if you just take the same behavior, I have already explained so, you have to identify the bus, the minimum number of buses that I have already explained and also the switches because there are many registers here right so, there are many registers here, I can connect this register to all switches and I can put switch there, but then I need if there are say 4 registers and 3 buses, I need 12 switches which is something may not be needed.

Because this will give you the maximum flexibility I mean any register value can go to any bus, but you may need may not need that much of flexibility. So, it is basically the maximum possible interconnections things so, I do not need that. So, probably by analyzing this behavior, what I understand say probably I do not need this switch, or I do not need this switch, I do not need this switch say probably right.

So, probably I do not need this switch and then say, I do not need this switch or say this switch right and so on. So, this is the final connection. Similarly, for FUs, you can connect this all input to all switches right. So, then, if the number of FU say k number of for FUs and say n number of buses, it is (k * n *2) right, this is the number of switches you need. Again, I mean we can show that this number of switches is too much so, you do not need that right.

So, you can actually remove some of the switches right so, you can remove say some of the switches right so, you can remove some of the switches. So, suppose you do not need this switch, you do not need this switch, you do not need this and so on. So, this can be determined by this RTL behavior right. So, what are the operations going to happen.

(Refer Slide Time: 27:25)



Bus sharing and binding Problem

Problem 1:

Find the *minimum number of busses* to accommodate all data transfer

Problem 2:

Find the *maximum number of data transfers* for a fixed number of busses

12

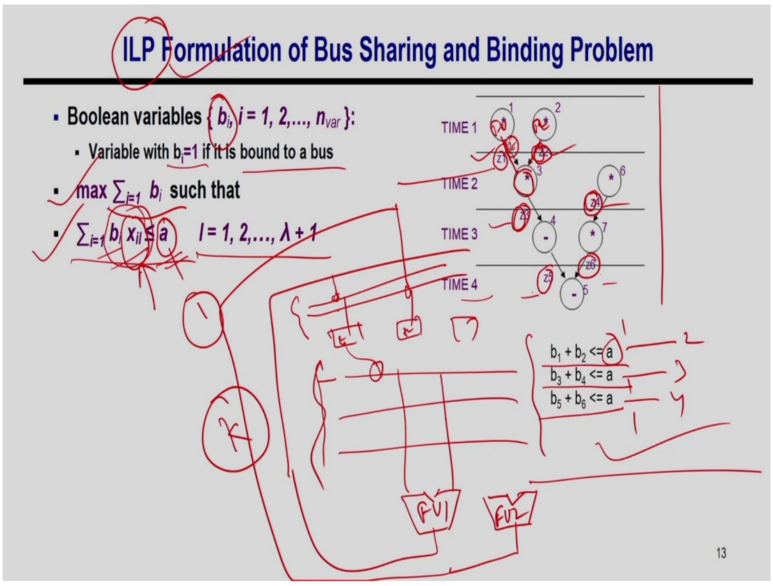
So, see if we try to summarize this data path synthesis, bus-based architecture binding problem so, your first problem is basically you identify the minimum number of buses first. What is the minimum number of buses needed to accommodate all this? So, this is the first requirement.

So, the bus to make the inputs and you need also bus here for the register inputs right. So, this is the input to the FUs, this is the register input to the registers. So, you identify the minimum number of buses to accommodate all the data transfer right. So, that is the problem 1.

And you can actually have another problem that you are given a fixed number of buses right so, say suppose you somebody tell you that there are 4 buses at the input of the FUs ok so, you have 4 buses here so, then, your objective could be that for each bus, you try to put the maximum number of data transfers right.

So, so you have given 4 buses and then so, there are maybe say 100 data transfers so, where all this 100 cannot be placed here right. So, for example, 50 can be the maximum placed here. So, you identify what is a maximum number of data transfer can be placed into these buses based on the requirement ok. So, this is what is the problem here right.

(Refer Slide Time: 28:42)



ILP Formulation of Bus Sharing and Binding Problem

- Boolean variables $\{ b_i, i = 1, 2, \ldots, n_{var} \}$:
  - Variable with $b_i=1$ if it is bound to a bus
- $\max \sum_{i=1} b_i$ such that
- $\sum_{i=1} b_i \, x_{il} \leq a \quad l = 1, 2, \ldots, \lambda + 1$

$b_1 + b_2 <= a$
$b_3 + b_4 <= a$
$b_5 + b_6 <= a$

13

TIME 1
TIME 2
TIME 3
TIME 4
1
2
3
4
5
6
7
z1
z2
z3
z4
z5
z6
FU1
FU2

So, this problem can be formulated again as a ILP formulation. So, I will talk about this ILP formulation briefly. So, it says, it is basically similar to this port assignment problem ok. So, the idea here is you basically have these variables right. So, these are the variables that has to be make transfer right. So, these are the variable, these are the input to the FU so, I am just talking about the input FUs so, similar thing will come for input to the registers right.

So, for input to FUs, these are the transfer you have to make, and you have to make sure that this transfer is possible for a fixed set of say buses right. So, what I can do either formula formulation is that I can actually identify a variable b_i if a particular variable b_i actually map to buses, then it will be 1 other it will be 0 right. So, I can actually assume a Boolean variable b_i if variable b_i is actually mapped to bus ok otherwise it will be 0 ok.

And then, what is your objective? So, what I want to do? It is similar to the port binding problem that I want to identify; I want to identify maximum number of such variables mapped to a fixed number of bus right. So, I have given a fixed number of bus a, I want to maximize this value of b_i such that maximum of such variable can be already mapped by to the fixed number of buses ok.

So, I want to maximize my score that b_i so, if($b_i = 1$) that means, that variable is already minded to this buses so, that means, if this $\sum b_i$ is maximum that means, I want to maximize the mapping of this variable to this fixed number of buses, but there should be some constraint right.

So, as I mentioned that if you in a particular state only in a time step, one bus can transfer only one data right. So, the total data transfer requirement of every state must be less than equal to of this particular fixed number of buses right ($b_i+b_j<=a$). So, how can I find out? So, this X_{il} say which operation is mapped to which time step.

See if I multiply this $\{b_i *X_{il} =1\}$ if this bus transfer is happening in time step l or not right.

So, for every time step, the number of variables that get mapped and their transfers should be less than equal to the total bus requirement right. So, that is what is given by this constraint right, this is for each time step ok. So, this actually formulates the problems. So, what it will do? If you give this to the ILP solver, what it will do?

It will give you the total number of variables that can be mapped to this, the fixed set of a right so, it will just tell that and then, you can actually if you just put that a = 1 so that what you have to do? It will try to find out all the variable transfers needed that can be mapped to one bus right.

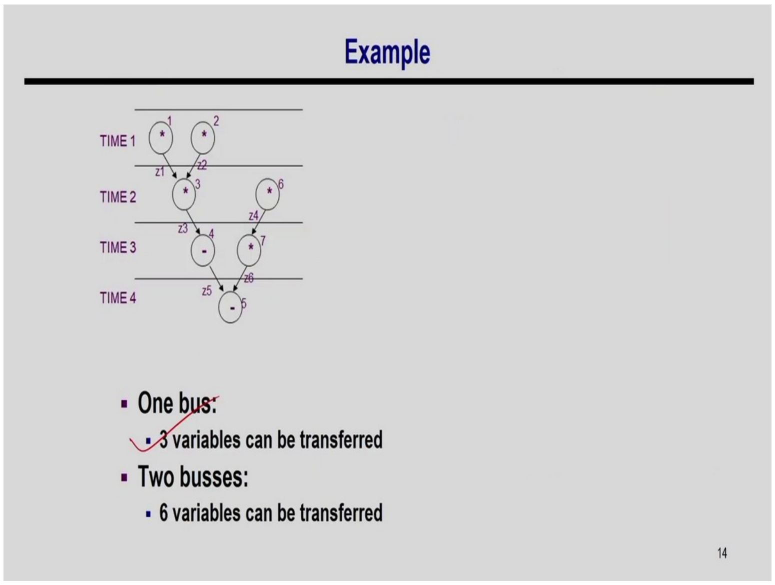
So, this is how if you just run it for say k iterations, you identify the minimum number of bus required also right. So, you have you identify the bus 1 for which you identify what are the variable can be mapped to this bus and then, you remove those transfers, then again you try to solve the problem for bus 2, you again identify the maximum number of variable transfer that we can be mapped to those bus 2 and so on right.

If you just keep doing this so, you will identify the k which is the minimum number of buses needed right. So, that is something you can actually identify the solution to the problem 1 and for problem 2, what is already the problem ah it shows here ok. So, let me take the example here.

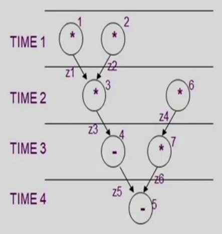
So, here the variable is z_1, z_2, z_3, z_4, z_5, z_6 and I am actually discussing the mapping of this to the FU input ok. So, there are and the buses at the FU input right. So, what is here? So, this z_1 and z_2 need to transfer to this FU input in time step 2. In time step 2, the z_3 and z_4 need to be transferred to this to the FUs, some FU s and this z_5 and z_6 is needed in time step 4 ok.

So, that means, if the total number of buses is a, the transfer of this z1 and z2 must be <= a, transfer of z3 and z4 <= a and transfer of z5 and z6 <= a in time step 4, this is in time step 3, this is in time step 2. So, for every time step, we will have such constant right.

(Refer Slide Time: 33:27)



Example



- One bus:
  - 3 variables can be transferred
- Two busses:
  - 6 variables can be transferred

14

And now, if we know this value say 1 right so, a = 1; then we can actually transfer maximum so, we have only one bus right. So, every time step you can actually transfer only one data so, then maximum of 3 variable can be transferred. If there are two buses so, I can actually transfer all of them because you can maximum transfer 6 data and there are 6 transfers so, I can transfer all 6 variables right. So, this is what this ILP formulation gives.

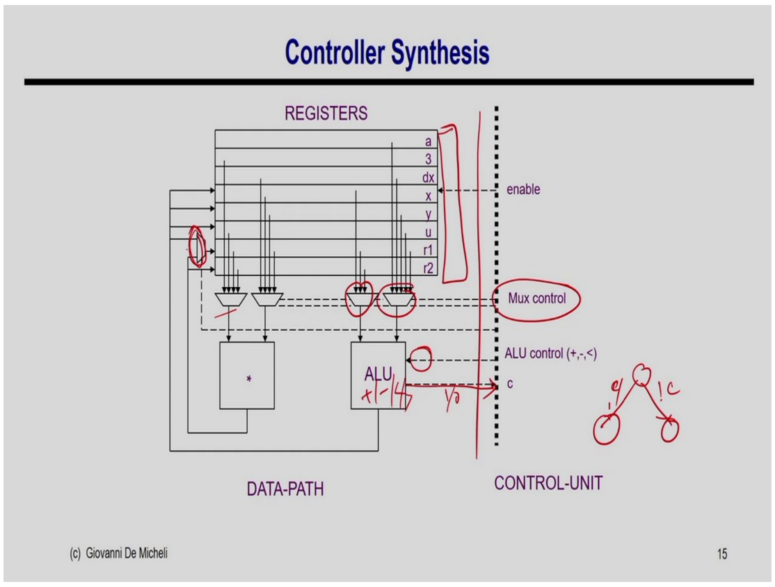
So, I can utilize this ILP formulation to identify the minimum number of buses and also mapping of this transfer from this variable to the buses as well. So, this is one part. So, which is basically the input or the FUs, the similar problem you have to solve it for the input of the registers as well because there the problem is that this see whenever you are writing this z_1 at the time step 1, this value z_1 need to be transferred to the register z_1 right.

So, from the FU output, see if this is say FU_1, this is FU_2 so, from FU_1 to z_1, you need to make connections. So, you have to identify how which bus you will connect the output of FU_1 right. So, this FU_1 go to so, this at the register say ok. So, this will go here and there is fix set of buses. So, you have to make sure that from FU_1, you have to make a connection either to this register or this register, say this bus or this bus. So, that you have to decide.

Similarly, for z_2 because this is say z_1, this is say z_2 now, so, this is your FU_2 ok so, this FU_2, this output also need to go to this z_2. So, you have to probably make a switch here and these two switches here so that this connection can be made right. So, you have to identify how many buses here are needed to make sure this writing to these registers ok and similarly, how and how to make the this connects transfer from FU to those buses.

Here, the connections have to make from a FU output to those buses. At the input of FU, the connection has to be made from the registers to the buses and from the buses to the FUs ok. I hope you understand the all problem of this bus-based interconnection synthesis and what is the optimization objective and how we can solve that problem ok.

(Refer Slide Time: 35:53)


Controller Synthesis

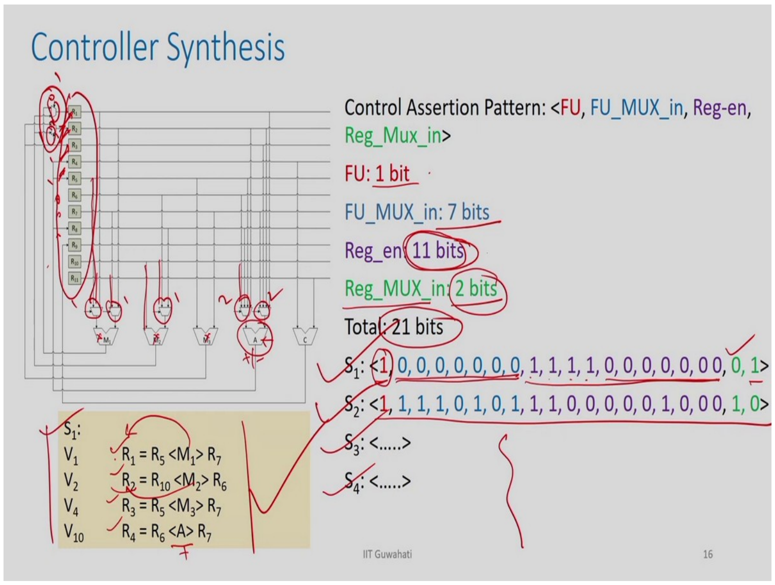
So, now, I will move on to the controller synthesis part. In the controller synthesis part so, basically, by this way, you identify the say let’s say this is a mux-based connection, you identify the muxes and the exact connections right. So, these are all identified and the registers everything is identified, the muxes needed at the input of registers, everything is identified right.

So, what are the control signals is needed? We need a control signal for the enable of the registers because it is not that all register will be updated in all cycles so, you have to stop updating the registers by a enable signal. If the enable signal is 1, then in that particular control step that register will be updated otherwise, it will not be updated right.

So, you need this control signal for this muxes so, you need control signal mux control, you need some control signal for the ALU because this ALU can do +, - , > all operations so, you have to tell this ALU which operation to be executed and you need this, and you need also control signal for this muxes right. So, these are the control signals you need to do.

And then, if this ALU is doing so, greater than, less than that greater than, less than value, 1 or 0 has to transfer to this controller right. So, based on this value, the controller test state transition will happen ok. So, if(c) it will come here, else it will come here right something like this. So, basically, once this data path is finalized, this is a very trivial task to identify the number of signals is needed and what is the value of those signals at each time step ok.

(Refer Slide Time: 37:32)



Controller Synthesis

Control Assertion Pattern: <FU, FU_MUX_in, Reg-en, Reg_Mux_in>

FU: 1 bit

FU_MUX_in: 7 bits

Reg_en: 11 bits

Reg_MUX_in: 2 bits

Total: 21 bits

S_1: <1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0 0, 0, 1>

S_2: <1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0 0, 1, 0>

S_3: <.....>

S_4: <.....>

S_1:

V_1	$R_1 = R_5 <M_1> R_7$
V_2	$R_2 = R_{10} <M_2> R_6$
V_4	$R_3 = R_5 <M_3> R_7$
V_{10}	$R_4 = R_6 <A> R_7$

IIT Guwahati 16

So, let me try to explain with this example. So, here is the mux-based connection we have obtained. So, let me understand how many signals is needed. So, first is needed the FU. So, here, I have only one ALU so, I need 1 bit right so, I need 1 bit so, this is to say suppose doing plus and minus. So, these are all doing multiplication right so, I do not need any control signals.

So, I need 1 bit for FU and then, for this I need 1 bit, for this I need 1 bit, for this also I need 1 bit, for this I need 2 bits, 2 bits, for this also I need 1 bit, 1 bit right. So, what is a total bit? 1, 2, 3, 4, 5, 9; 9 bits right. So, I need 7 bits here and I just put this is as different things so, register MUX in 2 bits. So, these are the 2 bits and there are 7 bits here ok.

If there are 11 registers, for each register I need 11 signals so, I need 11 such bits right. So, the total number of control signals is needed is 1 +7+11 + 2=21 bits so, total control signal bits is 21 bits is needed. So, I can just analyze the circuit and I can identify the total number of control signals needed to make the control the data flow in the data path ok.

So, then, each time step, you have to identify what is the value of the control signals. Let say I want to do it for say time step 1 so, in this time step 1, these are the operations happening right. So, the first bit is a FU so, here say suppose this is say + is happening here so, I have just put 1 so, plus say 1 means say + ok.

So, I am just putting 1 here and for this, these are the control bit corresponding to this MUX_in. So, since this is the first operation, all will take the left one right so, all will take the leftmost bit so, it will be all 0, 0, 0, 0 right so, this is corresponding to 0, 0, 0, 0. And then, the next eleven bits are basically corresponding to the register enable. So, in time step 1, what are the operation is getting updating? R_1, R_2, R_3, R_4.

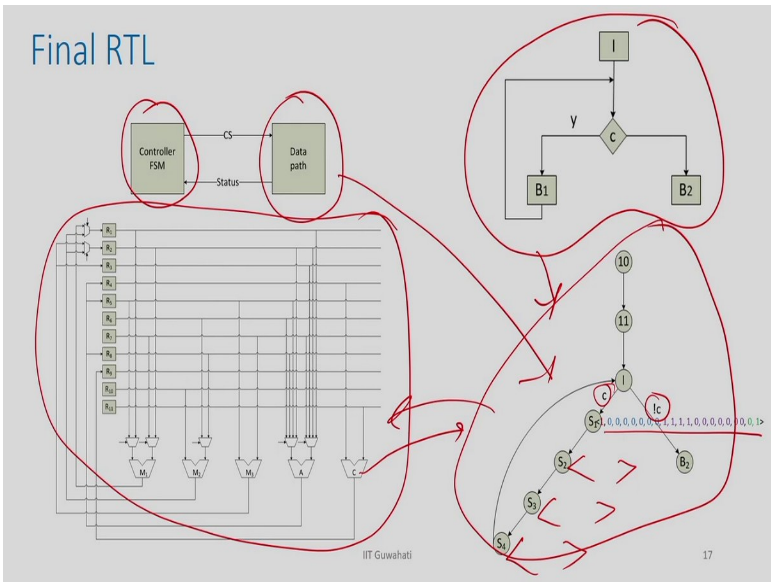
So, you have to put enable this four is 1 and rest will be 0, 0, 0, 0 right so, that is what I am done. So, R_1, R_2, R_3, R_4 is 1 and rest of the register enable is 0. So, that mean in the first control step only R_1, R_2, R_3 and R_4 data will be updated, and rest of the register would not be updated here ok.

And then here, for this muxes so, this is the MUX_in 2 bits so, for the first register, the output is coming from mux 1 right. So, this is for register R_1, it is coming from mux 1 so, I have to put 0 here so, I am putting 0 here and for this, for R_2, it is coming from M_2. So, I have to put 1 here because 0 means FU M_1 and 1 means FU M_2. So, I just put it 1. So, that means, in the control step 1, if I give this signal, it will make sure this data path is going to execute this behavior ok.

So, similarly, in control state 2, I am giving going to give this signal and corresponding to set of RTL operation is going to happen, I am to do this same thing for all other states right. This is how the controller will be synthesized ok. So, the summary of this controller synthesis is you identify the control signal first, you identify the control signal needed for FUs, you identify the control signals needed for the muxes, you identify the control signal needed for the register enables, you identify the control signals needed for the input muxes for the registers ok.

And based on that, you first identify the total control with data width, the control signal width, and then every time step, based on the operation to be executed in that time in that particular clock, you identify the value of this control signal and if you do this, you are done right, if you do this for all state, your whole circuit is ready now right.

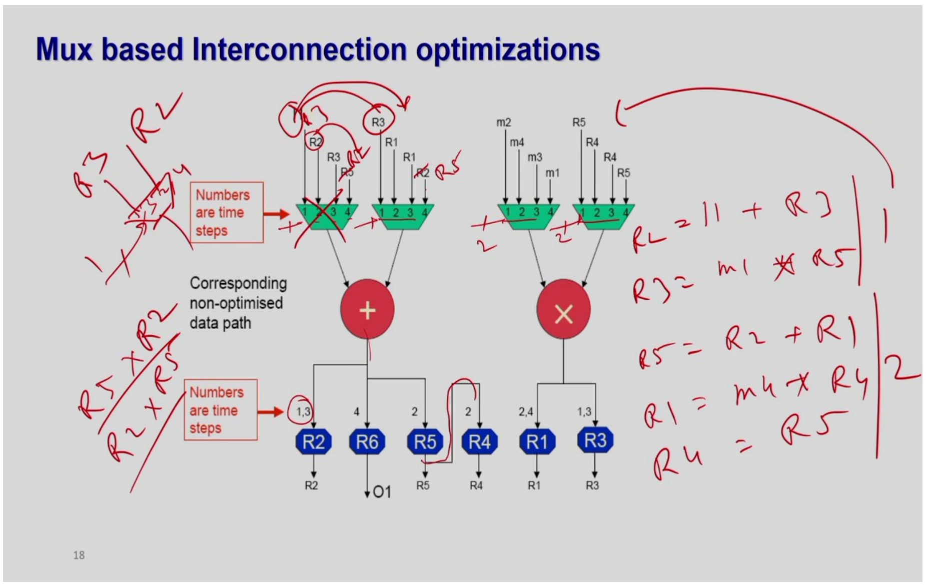
(Refer Slide Time: 41:42)



So, if you just complete, this you have a data path which is already synthesized, you have a controller which is now already synthesized so, you will get this right so, your data path is this and this is your controller, you know how many states are there from the scheduling behavior and every time step, what are the control signals is going to you have to put it here right.

So, based on that and then, this value, this comparator value will come from this comparator unit and based on that, this particular behavior is going to execute the input behavior right. So, this is what is going to happen, and this is what is high-level synthesis does and it basically finally, you have a data path, you have a controller and then if these two works together, it will execute the exact behavior that we have taken as an input to the high-level synthesis tool ok. So, this is what about this controller synthesis and this data path synthesis.

(Refer Slide Time: 42:34)


Mux based Interconnection optimizations

So, I am going to cover now a little bit on this mux-based interconnection optimization which we have not discussed so far. So, in the mux-based interconnections, looks like it is very fixed right. So, if you think here, I have already identified the multiples so, you cannot change this thing.

Looks like that based on the interconnection requirement, if there are 4 register input which is coming to one FU, you have to put a mux size of four right, we cannot do anything, but this there is an optimization possible here so, let us try to understand this. Say suppose this is the connection requirement we have identified by you, identify the operation that is going to execute in time step.

For example, here this 1, 2, this thing mentions the time in the time step. So, that means, in time step 1, I_1 should come to this FU. In time step 2, R_2 will come to here and so on. Similarly, for the right input, similarly for this FU right ok. So, you can understand here and then also this 1, 3 saying that in which time step this data will come to here. So, if I try to write the operations so, in time step 1, it is happening $R_2 = I_1 + R_3$ right, for this FU and for this FU, it is $R_3 = m_2 * R_5$. So, this is in time step 1 right.

So, similarly, I can write for time step 2. So, in time step 2, it is R_2, here it is R_1 and the output is going to R_5 so, $R_5 = R_2 + R_1$ ok and then similarly, here it is $R_1 = m_4 * R_4$ right and also, there is assignment here so, basically, $R_4 = R_5$. So, this is in time step 2 ok.

So, this is how I understand, this is a schedule behavior and based on this schedule behavior, I actually identify this mux connection. So, you will tell me that you need a 2-bit control signal here, 2-bit control signal here, 2-bit control signal here, and 2-bit control signal. So, my question here can I optimize this circuit further, ok? To do so, let us understand what can we do here because this looks like already fixed right, let us try to look into these inputs right.

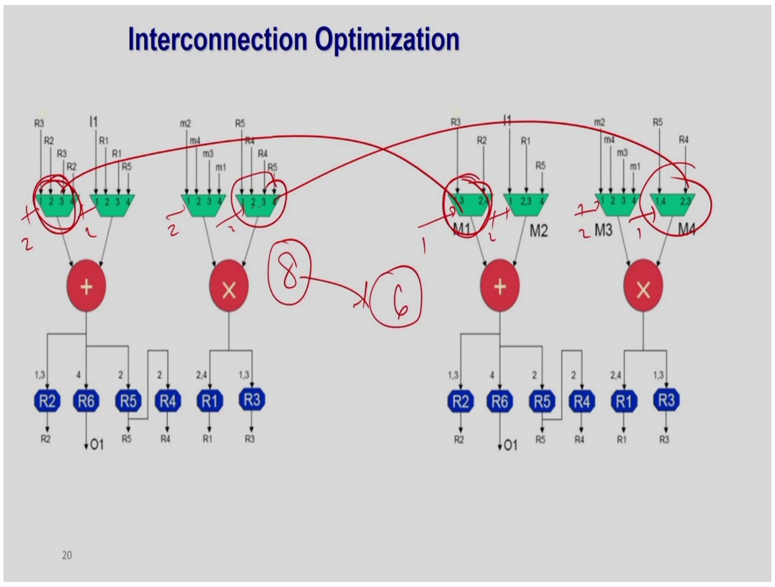
So, you can see here R_2 is already here right and here it is happening R_5* R_2. So, if you do R_5 * R_2 or you do R_2 * R_5, both are the same right because it is multiplication, but so, I can bring this R_2 here, I can make this swap this connection, I can commute it right, this is basically commutative and R_5 here, both are the same thing right.

So, I can bring R_5 here and I can bring R_2 here right because this is the same operation. Because in the time step 4, R_5 into R_2 is happening whether it is R_5 * R_2 or R_2 * R_5, they are actually the same right. So, but doing this, what is happening here? I can see that this both R_2 here so, I can reduce the one of the inputs because it is still now it becomes R_2 only right.

So, similarly, what I can do? I can bring this R_3 here and I_1 here right. So, then what is going to happen? By this way actually, there is a possibility that you might bring so, if you bring R_3 here, there are 2 R_3 right. So, now, there are 2 R_3, there are two R_2 so, you can actually reduce the size to 2 because you know only you need R_3 and R_2 and what are the time step you need R_3? You need 1 and; 1 and 3, you need R_3 and 2 and 4, you need R_4 and your control signal be now will be 1-bit ok.

So, that is what is the mux optimization that you make the connections and then, you check the commutativity of the operations and based on the commutativity, you can actually bring the common inputs of a mux in to the same side and as a result, your size of the mux might reduce ok.

(Refer Slide Time: 47:05)

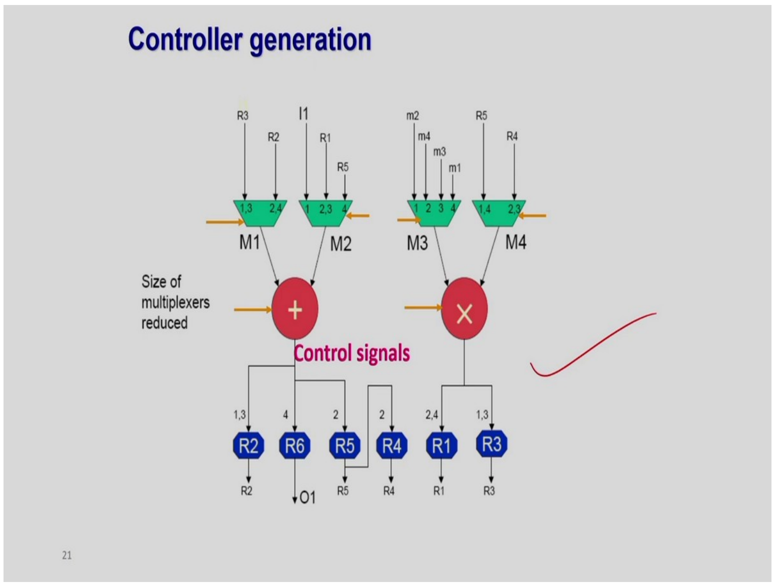


So, if you just try to do this here so, what I have done here, I just swap this I_1, R_3, then this and also, I just remove this which I have already told you this R_5 and R_2 right so, I have just swapped this R_5 and R_2. As a result, this particular mux has now only two R_3 and R_2 as the input. Similar things can be done here. So, if you just do this, you can see here that I can reduce this mux size from 4*1 to 2*1, I reduce this mux size from this to this, So, this identifying this to this conversion is a homework for you.

So, as a result, you can see earlier, you are needing the control bit 2-bit, 2-bit, 2-bit here, 2-bit here so, total 8-bits, here I need 1-bit, here still I need 2-bits, here I need 2-bits, 1-bit right so, I need 6-bits.

So, I reduce the control size, controller size as well as mux size because the mux size is also getting reduced this is as a 2 * 1 mux is a multiplexer, here it is a 4 * 1 multiplexer. So, both the size of the controller will reduce and also, the size of the data path will also reduced. So, this is how you can actually do the interconnection optimization in the mux-based architecture as well ok.

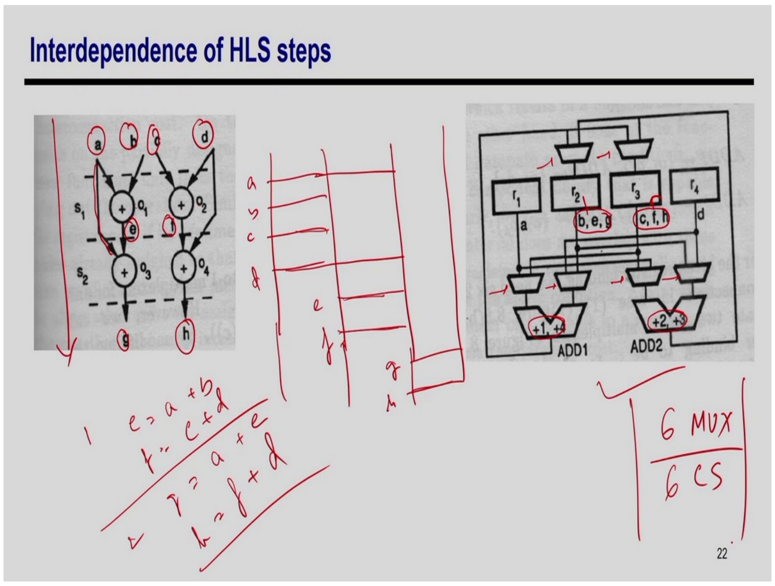
(Refer Slide Time: 48:26)



Controller generation

So, this is the final structure that I just talked about ok and this is the control signals which I already talked about ok.

(Refer Slide Time: 48:36)



Interdependence of HLS steps

So, before concluding this discussion, I will also bring one more important aspect here this high-level synthesis to steps is interdependent right. What does it mean? That you can understand that the kind of allocation binding is going to happen based on the schedule. So, if you have a defined schedule, the lifetime of the variable will be different and hence, the register sharing will be different right.

Similarly, if your schedule is different, then the FU conflict will be different right so, and then and hence, the FU binding will be different. So, the schedule interferes the allocation binding result. Similarly, in the allocation and binding, there is a multiple options of binding. So, if you just choose one, the corresponding interconnection will be one and similarly if the binding you choose is a different binding, your interconnection will be different ok and the cost of the interconnection is also changed ok.

So, I have taken a very classic example here just to explain how this FU allocation binding and register allocation binding impact the interconnection cost ok. So, let us take this is my schedule behavior ok. So, here, the variables are a, b, c, d, e, f, g and h ok, these are the eight registers ok and what is a lifetime? So, basically if I just draw it so, there are two-time step right so, you can actually show this a, b, c, d is here, a, b, c, a is needed here also it because a is needed here as well and d is also needed here right.

So, this is your d and then, e is basically here, e and f is here and g and h here, this is g and h, this is e and f. So, this is the time requirement and based on the overlapping, I can share the register. So, what I can do? I can actually map this b, e and g into one register, it is possible because b, e and g, I just map into one register and then c, e, c, f and h which is possible because they have non-overlapping right. So, suppose I put this b, g into register 2, c, f and h into register 3 ok.

And mapping of this operation you can understand that o 1 and o 2 is happening in 1 and o 3 and o 4. So, I just put o 1 and o 4 into this FU and o 2 and o 3 into this FU so, this is the binding. So, if you just do this binding and you are actually performing the operation, you can understand that in the time step 1 right so, time step 1, what are the operation is happening? E = a + b and then, f = c + d, this is what is happening time step 1 and time step 2, what is happening? G = a + e and h = f + d right, this is what is happening.

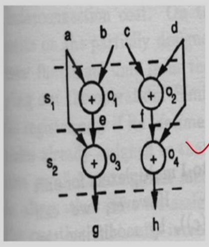
So, to perform these operations, the interconnections will be like this. So, you can actually do this as homework and identify these interconnections and you can see here the total mux is needed 6 MUX right, 6 MUX is needed and out of them, this is actually so, these are all 2-bits mux right so, and the total 6 control signals is needed because all are single 2 * 1 mux.

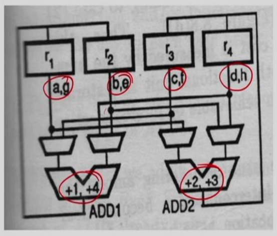
So, this is one possible mapping. So, because of this, a register allocation and binding is not a data path interconnections job, this is during the allocation and binding phase right.

So, if in the allocation and binding phase, the binding happens like this, your interconnections cost if I just assume by mux and control signal so, you need 6 control signal, 6 muxes ok.

(Refer Slide Time: 52:41)

Interdependence of HLS steps





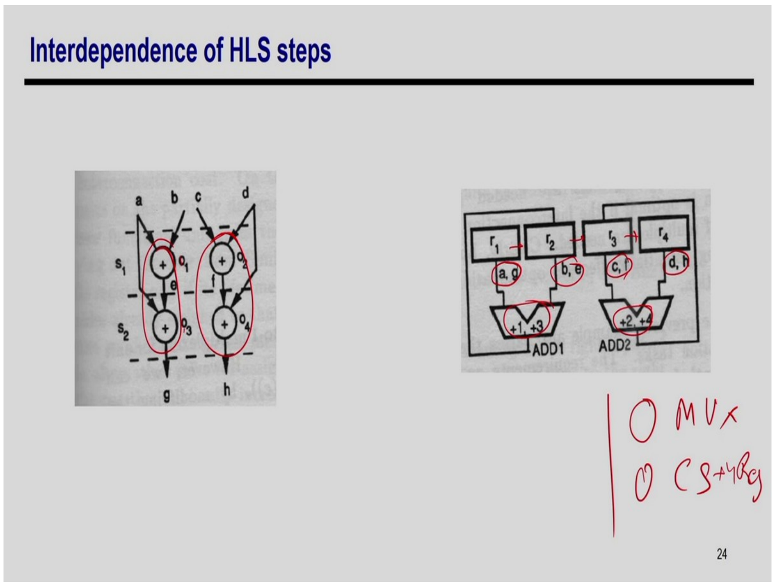
4MUX
4CS +4esReg

23

And then, say based on this so, this is one; another possible mapping is a, g, b, e, c, f. So, you can actually do this a with g which is possible, then you do this b and e so, b and e and then, c and f, this is c this is f which is non-overlapping and then, d and h right so, this is your d this is your h ok.

So, if you make these connections, these is of valid connections and you can see here if this mapping of operation is still 1, 4 and 2, 3. So, you do not need the input muxes right so, you still need the muxes at the input of the FUs. So, how many muxes is needed? 4 mux and 4 control signals. So, your control reduces from 6 to 4 right. So, some improvement is happening. So, you can understand that same schedule, but different allocation and binding cost and different interconnection cost ok.

(Refer Slide Time: 53:46)



Interdependence of HLS steps

So, now, let us move on to another form of things. So, suppose you map this a, g here, b here same whatever you have done in the previous cycle that a, g, b, e, c, f and d, h, the same mapping I am doing register allocation a, g, b, e, c, f and d, h, but what I am doing here this o 1 and o 3 I am mapping to one FU and o 2 to o 4 into another FU right. So, this is the FU allocation binding change now.

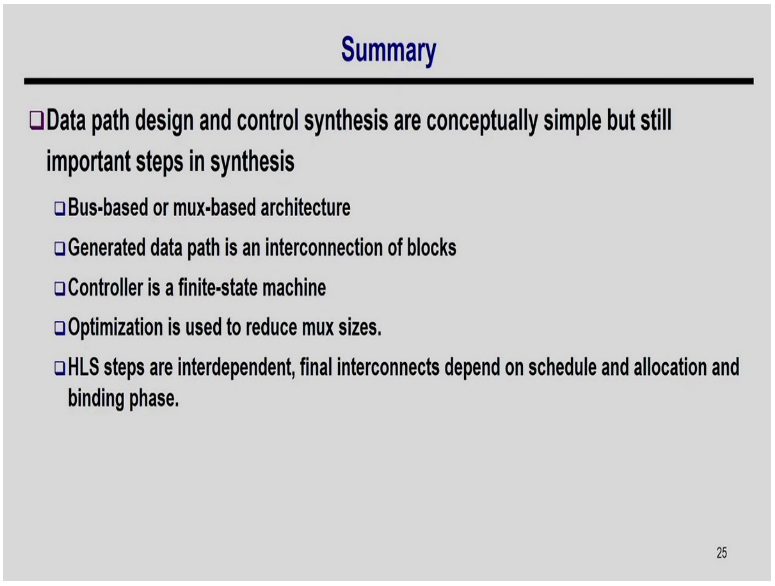
And for this, if you can realize this operation here, you do not need any FU so, your interconnection cost is 0 mux, 0 control signal right. So, basically, one thing is wrong here so, you have to have a enable. So, this control signal corresponds to the register enable, I am not considering that is fixed for all things right, there are 4 control signals for the register, here also you need to have 4 control signals for register which is something is common. So, I am not considering them right. So, here also, 4 control signals for rec because you need to enable there.

But, you can understand that from the same schedule because of the different, different register allocation and binding, my interconnections is actually changing and the cost is changing. Similarly, if the schedule changes, your allocation on binding will also change and it can also impact the interconnectional binding.

So, this is something is the interdependence of high-level synthesis, and you cannot avoid it, this is how the whole thing works ok. So, probably once you do the scheduling, you try to if you just look ahead that your allocation and how it will impact in the allocation binding that can be a good future study right. So, you do your scheduling, make sure that your allocation binding will be smooth right, your complete will be less something like this, or the way you do the allocation binding, you make sure that your overall interconnection cost will be less.

So, this can be a good study that you formulate your allocation binding problem to make sure that overall interconnection cost is minimum right. So, this is how we can actually formulate the problem, but the way we have discussed it in this course we never thought about that right. So, that can be a good future direction of research ok.

(Refer Slide Time: 60:00)



Summary

- ❑Data path design and control synthesis are conceptually simple but still important steps in synthesis
  - ❑Bus-based or mux-based architecture
  - ❑Generated data path is an interconnection of blocks
  - ❑Controller is a finite-state machine
  - ❑Optimization is used to reduce mux sizes.
  - ❑HLS steps are interdependent, final interconnects depend on schedule and allocation and binding phase.

25

So, with this I conclude today’s class. So, in this class, we basically discuss the data path and controller synthesis, although the conceptually the steps are very simple and it is kind of very conventional but is a very important step without that, we cannot realize the actual hardware right and also, we have seen that this interconnection can be bus-based and mux-based.

For bus-based, your optimization objective is to be using a minimum number of buses as well as the switch for the interconnections. For mux-based, you try to reduce the mux size and hence, the controller size ok. And then, once you have this interconnection ready, your controller will present by a controller FSM. So, in the controller synthesis, your objective is to identify the minimum number of control signals needed and then, every time step, what is the value of those control signals right. Once everything is decided, your RTL is ready and if you just run those things in a using any simulator, it will exactly simulate the behavior which was given as the c-input ok.

And also, we have also discussed one important point that this high-level synthesis steps are interdependent, and one decision of the previous step might impact hugely the synthesis result of the next steps. So, one possibility of high-level synthesis is that you make a that good allocation ever scheduling or interconnection averaged allocation binding or this data path ever allocation and binding right. So, that can be a good research areas. So, with this, I conclude today’s discussion.

Thank you.