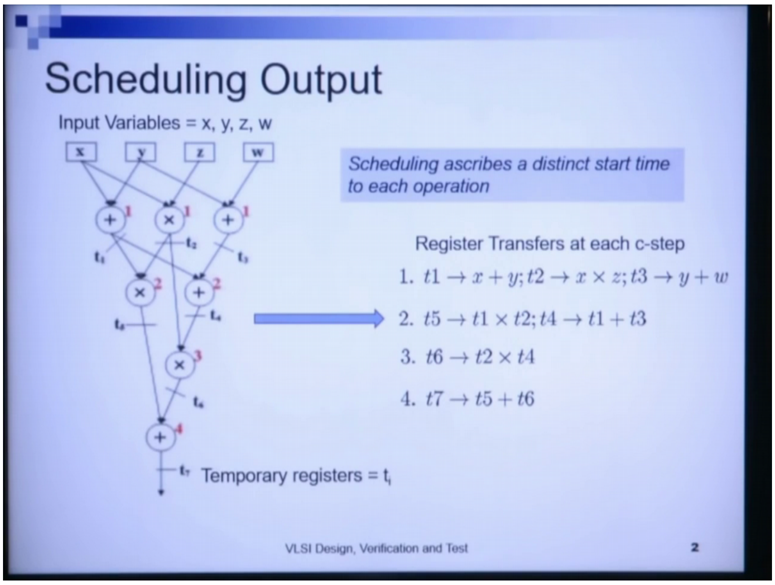
Embedded Systems – Design Verification and Test

Dr. Santosh Biswas
Prof. Jatindra Kumar Deka
Dr. Arnab Sarkar
Department of Computer Science and Engineering
National Institute of Technology, Guwahati

Lecture – 07
Hardware Architectural Synthesis – 2

Welcome to module 1 of lecture 6 of the course VLSI Design Verification and Test. In the last few lectures, we took a look at resource constrained and time constrained scheduling strategies in high level synthesis. In this lecture, we will look at an important post scheduling step, allocation and binding.

(Refer Slide Time: 00:49)



Scheduling Output

Input Variables = x, y, z, w

Temporary registers = t_i

Scheduling ascribes a distinct start time to each operation

Register Transfers at each c-step

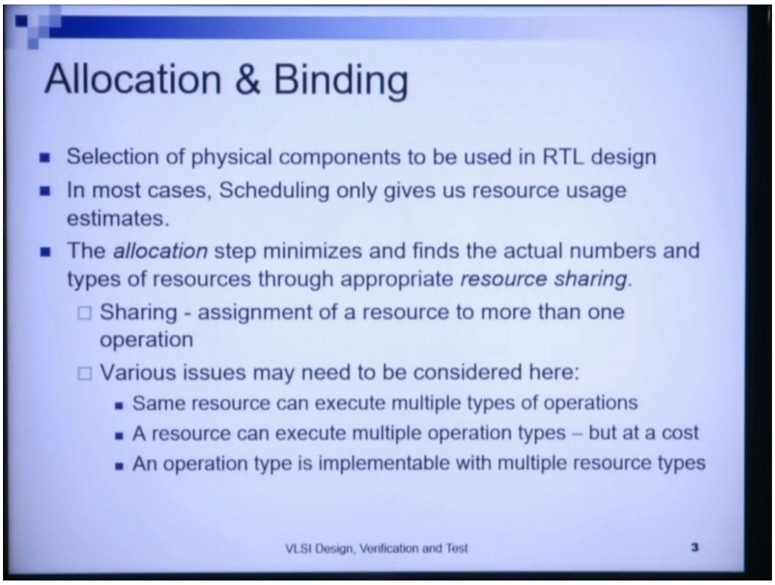
1. $t1 \rightarrow x + y; t2 \rightarrow x \times z; t3 \rightarrow y + w$
2. $t5 \rightarrow t1 \times t2; t4 \rightarrow t1 + t3$
3. $t6 \rightarrow t2 \times t4$
4. $t7 \rightarrow t5 + t6$

VLSI Design, Verification and Test 2

So, what do we have at the output of the scheduling step? At the output of the scheduling step, we have distinct time steps assigned for each operation. So, we have a scheduled operation constraints graph here, we have an arbitrary operation constraints graph, in this graph, we can see that the operation plus multiplication and adder here is performed, in time step 1 in time step 2, we perform this multiplication and addition in time step 3, we perform this multiplication and in time step 4, we perform this addition. All these are unit time operations each of these operation take input from certain input variables and produce outputs on temporary registers or temporary variables.

So, these t 1, t 2, t 3 are temporary variables and what we have is a set of register transfers. The set of register transfers here are t 1 equals to x plus y. In the first time step, t 2 equals to x in to z, in the first time step similarly, we have different register transfers at each time step at the output of the scheduling step.

(Refer Slide Time: 02:13)



Allocation & Binding

- Selection of physical components to be used in RTL design
- In most cases, Scheduling only gives us resource usage estimates.
- The *allocation* step minimizes and finds the actual numbers and types of resources through appropriate *resource sharing*.
  - □ Sharing - assignment of a resource to more than one operation
  - □ Various issues may need to be considered here:
    - Same resource can execute multiple types of operations
    - A resource can execute multiple operation types – but at a cost
    - An operation type is implementable with multiple resource types

VLSI Design, Verification and Test 3

Now, after we have assigned a time step to each operation our job is to actually allocate and bind physical components to the RTL transfers to the operations, we need to assign functional units to the registers temporary registers, we need to assign actual hardware resource registers, we need to steer the outputs of the registers to the appropriate functional units and again, we have to drive the output of the functional units to appropriate registers to effect each register transfer.

So, therefore, we need to assign physical components to the register transfer level design. Now in most cases scheduling gives us an estimate of the resource usage. So, in latency constraints scheduling and time constraints and resource constraints scheduling that we had looked in scheduling every all these scheduling algorithms took can estimate of the resource usage of bound on the resource usage bound on latency and provide as a schedule now the work of the allocations step is to minimize and find the actual numbers and types of resources that will be required and this will be done via appropriate registers sharing resource sharing thing.

Now what is sharing then? Resource sharing; the assignment of a resource to more than one operation now when we do resource sharing we need to keep various issues in mind, for example, let us consider the case of functional units. So, the same functional unit can execute multiple types of operations set; that means, that in the in the scheduling step we saw that the comparison operation the plus operation and the minus operation or all being performed using the ALU.

So, I can have multiple operations being scheduled by the same resource and then I can have a resource that can execute multiple operation types. So, there we are saying that multiple or resource the same resource can execute multiple types of operations. Now here same resource can execute multiple types of operations, but comes with a cost, for example, let us say that we have we can combine multiplication and an addition as an example; however, the addition the combination of addition and multiplication comes with a cost because it requires a more complicated functional units.

So, we can have simple adders that can perform only addition simple multipliers that can perform simple multi simply multiplications and I can have combined adders and adders as well as multipliers which can do both, but possibly these combined adders are multipliers take more area than simple adders and simple multipliers now if the objective of your resource binding is to arrive at an minimum area minimum area design, then what do we have here? Now because we have a combined multiplier and adder, then these two operations become compatible meaning that if multiplication and addition is being performed at different two time steps, then multiplication and operations can be combined and implemented using a same resource instance.

However this comes at a cost; that means, that what will be the minimized area design will depend on how many combined multipliers adders you are using how many multipliers you are using and how many adders you are using let us say previously you used only multiply only simple multipliers and simple adders and therefore, previously you could not combine multiplication and addition operations in the same resource or functioning unit; however, now you are being able to combine multiplication and multiplication and addition in the same functional unit.

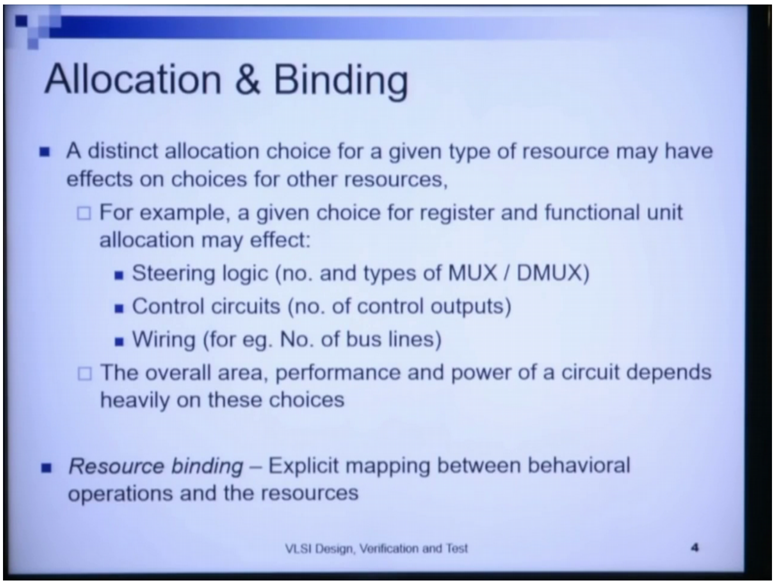
So, what you are having now is additional resource sharing. Now higher resource sharing means I am re reusing the same functional unit more; that means, I can possibly reduce the area of the circuit; however, what will be the effective improvement. Now due to higher resource sharing I am [pro/probably] probably due to higher resource sharing I have probably reduced the number of simple multipliers and simple adders let us say the total area cost due to this reduction this reduced the reduction in area due to the reduction in simple multipliers and adders are is x.

However, there is an increase in area because we said that this combined multipliers and adders take a higher area then simple multipliers and adders and let us say this increase in area due to this combined functional unit is one then what your effective gain in terms of area it will be x minus y. So, we have to find out. So, there are issues what am trying to point is that there are issues choice of what allocations step what allocation choice, you will have what resource sharing you will have what resources you will choose for which operation is an complicated problem.

So, this is what I am trying to point to another example is that an operation type is implementable with multiple resource type. So, the same operation type is implementable with multiple resource types for example, let us say you have two different types of adders for doing the addition operation now adder a is higher in performance than adder b, but adder a also takes higher area than adder b now given resource assignment problem with a bound on latency as well as your area the choice of the numbers of adders adder a and adder b in your design will depend on how you optimize both area and performance area and latency.

So, again what I am trying to point out is that the problem of allocation that is finding the appropriate and correct numbers and types of resources for each operations for the different operations. So, that my design objectives are met in the complicated problem to search there are a lot of choices to make.

(Refer Slide Time: 09:47)



Allocation & Binding

- A distinct allocation choice for a given type of resource may have effects on choices for other resources,
  - □ For example, a given choice for register and functional unit allocation may effect:
    - Steering logic (no. and types of MUX / DMUX)
    - Control circuits (no. of control outputs)
    - Wiring (for eg. No. of bus lines)
  - □ The overall area, performance and power of a circuit depends heavily on these choices
- *Resource binding* – Explicit mapping between behavioral operations and the resources

VLSI Design, Verification and Test 4

Now a distinct allocation choice for a given type of a resource we have fx on choices of other resources. Now I have chosen let us say for example, I have chosen a a certain number of registers for implementing for allocating the variables in my schedules the temporary registers in my schedule and I have also decided on the number and times of functional units that I will use to implement the operations in my schedule now given a choice a particular number of solutions and particular [allo/allocation] allocation of the registers and a particular type and number of functional units.

So, for a given decision your steering logic that is the numbers and types of MUXs and DMUXs will depend for ex what will these MUXs and DMUXs, do MUXs arbitrate write accesses to fun functional units and registers and the other hand, DMUXs arbitrate read accesses for from functional units and registers.

So, the choice of which registers which variables you have put on variable registers which operation you have mapped to which functional units will that will decide on what will be the number of MUXs and what will be the types of MUXs a 4 is to 1 MUX or 8 is to 1 MUX the choice of types and number of MUXs will depend on the choice of registers and functional units that you have made, it will also make it will it will also def determine the control circuits that you have, for example, now the number of control outputs that you will that you will produce will depend on the types of numbers of

MUXs and DMUXs; let us say the selective inputs of the MUXs and DMUXs are controlled by this controllers.

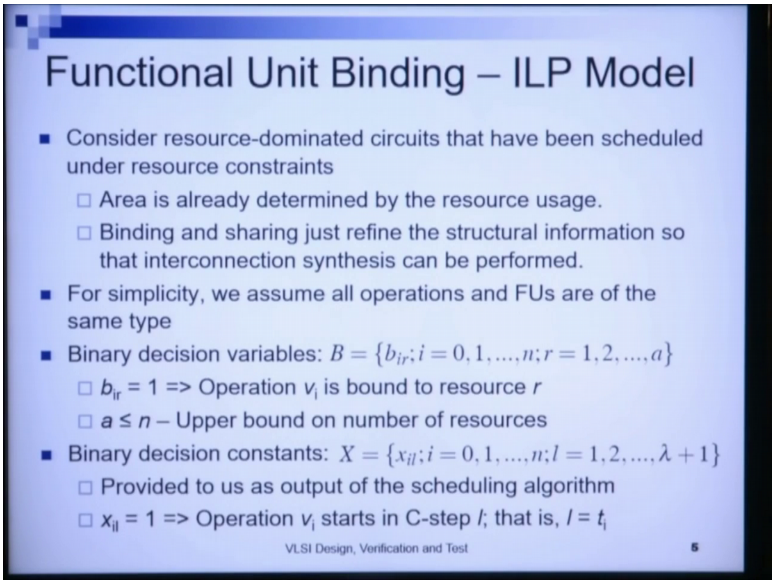
So, when the number of MUXs and DMUXs change numbers and types of MUXs and DMUXs change the control circuit that the number of outputs that the control unit will have to produce at each time steps that will also vary. So, the register and functional unit allocation step will also determine what will be the complication or in the control circuit it will also it will also affect the wiring in the circuit what will happen let us say you have buses you have buses in your circuit and your registers float their output on the buses your functional units also float their outputs on the buses.

Now a bus line can only carry a single variable single data at a given time. So, depending on the numbers and types of registers MUXs functional units what will happen is that your number of bus lines to be determined and hence a choice of registers and functional units will also determine what will be your wiring how complicated will be you are your wiring, right.

So, therefore, what we conclude from this discussion is that the overall area performance and power consumed by the circuit will depend heavily on the choices of allocation that you make and hence the allocation problem is an important problem to solve as optimally as possible with this introduction we now define resource binding what is resource binding resource binding is the explicit mapping between behavioral operations and the resources.

So, you can have many instances of these resources for example, let us say we have five multipliers in a design multiplier 1, multiplier 2, multiplier 3, multiplier 4 and multiplier five each multiplier say multiplier 1 is instance of a resource multiplier 2 is an instance of a resource and each resource instance multiplier 1, let us say will be mapped to a number of behavioral operations at different times now resource binding is therefore, the explicit mapping between these behavioral operations and resource instances.

(Refer Slide Time: 14:33)



Functional Unit Binding – ILP Model

- Consider resource-dominated circuits that have been scheduled under resource constraints
  - □ Area is already determined by the resource usage.
  - □ Binding and sharing just refine the structural information so that interconnection synthesis can be performed.
- For simplicity, we assume all operations and FUs are of the same type
- Binary decision variables: $B = \{b_{ir}; i = 0, 1, \ldots, n; r = 1, 2, \ldots, a\}$
  - □ b_{ir} = 1 => Operation v_i is bound to resource r
  - □ $a \leq n$ – Upper bound on number of resources
- Binary decision constants: $X = \{x_{il}; i = 0, 1, \ldots, n; l = 1, 2, \ldots, \lambda + 1\}$
  - □ Provided to us as output of the scheduling algorithm
  - □ x_{il} = 1 => Operation v_i starts in C-step l; that is, $l = t_i$

VLSI Design, Verification and Test 5

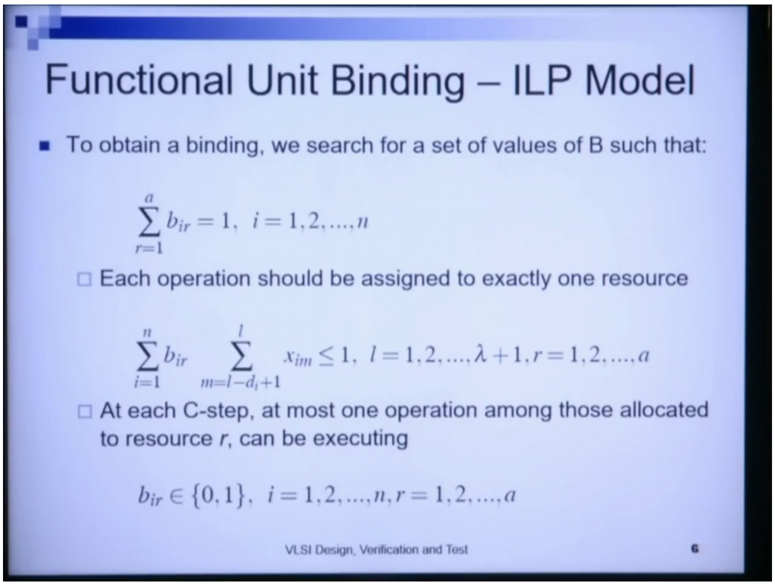
Now with this definition we first characterize resource sharing and binding problem which is simple case let us consider resource dominated circuits which has been scheduled under resource constraints. So, we have then resource constraints scheduling during my scheduling step. So, therefore, the area is already determined by the resource usage and therefore, I know the number of functional units that will be required now binding in this case binding and sharing in this case will therefore, just be to find the structural information.

So, that the interconnection synthesis can be performed by inter connection synthesis i mean let us say subsequent allocation binding of subsequent resources for example, we said MUXs buses control units etcetera right. So, when we have done an actual functional unit to behavioral operation mapping then only can we do the subsequent synthesis solve the subsequent synthesis problems eh for MUXs buses etcetera in this here for simplicity we assume that all operations and functional units are of the same type let us say that.

So, so what we are going here we will characterize the basic resource sharing and binding problem by understanding what its ILP model is like we did for the scheduling problem now to start with the ILP model, again, here we need to define a set of decision variables. So, what are the decision variables here the decision variables are a set B where it contains any decision variables of type b i r. Now what are this indices i and r i goes from one to n the number of operation and r goes from one to dot up to the number of resources, let us say a is the bound on the number of resources and i are the operations in my operation constraints graph. Now b i r equals to one that implies that operation vi in my operation constraint graph is bound to resource instance r.

Now, a less than n is an upper bound on the number of resources. Now because we have already conducted scheduling my decision variables that I had for scheduling, they have now become decision constants; these X is now have become decision constraints. What do these x's tell me? These x i l gives me the start times of resource i. When this x i l equals to one it means that operation vi starts at c step l and that step l is given by ti right. So, we saw these decision variables and how these start times are obtained in the scheduling step. So, now, they are decision constraints meaning these values of these constraints are known to me when each operation will start at what time step is now known to me at the allocation and binding step.

(Refer Slide Time: 18:05)



Functional Unit Binding – ILP Model

- To obtain a binding, we search for a set of values of B such that:

$$\sum_{r=1}^{a} b_{ir} = 1, \ i = 1,2,\dots,n$$

□ Each operation should be assigned to exactly one resource

$$\sum_{i=1}^{n} b_{ir} \sum_{m=l-d_i+1}^{l} x_{im} \le 1, \ l = 1,2,\dots,\lambda+1, r = 1,2,\dots,a$$

□ At each C-step, at most one operation among those allocated to resource *r*, can be executing

$$b_{ir} \in \{0,1\}, \ i = 1,2,\dots,n, r = 1,2,\dots,a$$

VLSI Design, Verification and Test 6

So, with this we con we come to the formulation of the constraints to obtain a binding we search for a set of values of b such that; that means, we need to obtain a binding; that means, we need to allocate the behavioral operations to the resources such that a set of constraints are met and what are these constraints the first constraint is that each operation should be assigned to exactly one resource how do we write this constraints for r equals to 1 to a summation b i r; that means, for each operation i goes from 1 to n this has to be true for each operation now let us say for the ith operation let us say for the first operation 1 b 1 r should be 1 where r goes from 1 to n; that means, operation one can be scheduled in only 1 and 1; exactly one of these a resources that I have only then will this summation b equal to 1.

So, operation one because b 1, I am considering can be can be assigned to at most an exactly one operator type or resource type and hence for over all the resources summation over all the resources this value b i r value will be equal to one the next constraint is that e at each c step at most one operation among those allocated to resource r can be executive. So, what does this tell me that for each resource r? So, we will take one resource at a time and for that resource we will consider for all time steps for each resource, we will consider all time steps and we want to constraint, we want to provide this constraint that a single resource will should execute at most one operation 0 or one at most one operation at each time step the resource can either be idle at that time step or can execute at most one operation, it cannot execute multiple operations at the same type or single resource cannot execute multiple operations at the same time.

So, this is what this is constraint restricts and how do we write this constraints the second part of this constraint this term here this term should be familiar to you from the scheduling step what does this term tell me it tells me whether at the current time step this operation is currently under execution. So, summation m equal to l minus di plus one to l x i m will be equal to one if operation i is currently executing at time step l and when will operation i be executing at time step l if it has started at most di time steps earlier to l; that means, at l it will be executing if it has started at most di time steps earlier than this if it has started even earlier than di time steps from l.

Then it has already finished if it has not started after di time step then it has not started at all. So, and it is not executing. So, the at the current time step operation i is executing only if it has started at most di time steps earlier than the current time step and this value will be one when operation i is executing at the current time step now we come to the first term here what does this first term tell me i equals to one to n b i r; that means, for each resource r what do I want to determine? I want to find out what this term if you see in seclusion this term will give me all operations that are bound to resource r because we are doing this whole constraints for each resources each time step then if we take this operation this term is seclusion this will give me all the operations that are bound to resource r and when I when we multiply this term with this term it gives me all operations that are bound to resource r and is actively executing at time step l.

So, this gives me all operation this gives me whether an operation is active or executing at time step l and this term gives me all operations that are bound to resource r. So, the multiplication of these two terms will give me all those operations bound to resource r and are currently executing now there cannot be two operations at a given time that are bound to resource r and are both executing and hence, this cons this value the multiplication of this two terms has to be less than equal to one therefore, we get what we want at each c step at most one operation among those allocated to resource r can be executing and the third constraint says that these decision variable are binary it can either take 0 or 1 using this ILP model, we will be able to get an appropriate mapping of the resources to the operations thus we can understanding of the basic ILP model for resource sharing and binding we come to the end of module 1 of lecture 6.