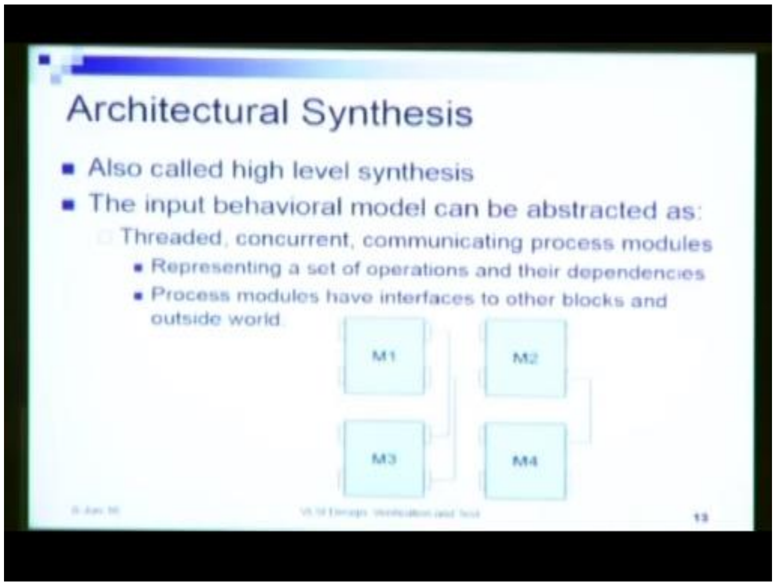
Better to module 3 of lecture 1, in the first modules we went over a discussion on while basic computer agent design is required in the VLSI design verification and testing process. And in module 2 we went through an overview of the entire design process. We understood that the design process goes from an initial specification of the behavior or functionality of an application to finalize structure of a packaged chip.

And this is an process goes through a step-by-step process and in this step-by-step process we go through a set of abstraction levels and in each abstraction levels we have different fuse. We said that the important abstraction levels are architectural level, the logical and the geometry level and the important views in each of these levels are the behavioral levels, structural level and the physical level.

Of course, in the architectural level and the logic level the physical view is not particularly explicitly defined, but when the geometric level comes we get an incisive view of the physical structure of the design. And then we also said that at inch level the design progresses through sages and the progress of from one stage to the next, from a less detailed description of the design to a more detailed description of the design this is called synthesis.

And corresponding to the levels we have three different important synthesis steps one is architectural synthesis, the second being logic synthesis, the third being geometrical synthesis or physical design. Now we will take a look at what these synthesis tells are what are the important things that happen within each of these synthesis steps.

(Refer Slide Time: 02:28)



Architectural Synthesis

- Also called high level synthesis
- The input behavioral model can be abstracted as:
  - Threaded, concurrent, communicating process modules
    - Representing a set of operations and their dependencies
    - Process modules have interfaces to other blocks and outside world

Okay, just before going through the synthesis process I would just like to iterate that the whole problem from the specification to the final implementation in the form of a packaged chip is essentially a single optimization problem, that optimization problem should take care of all issues relating to different aspects of the desire and produce a final implementation or the structural output.

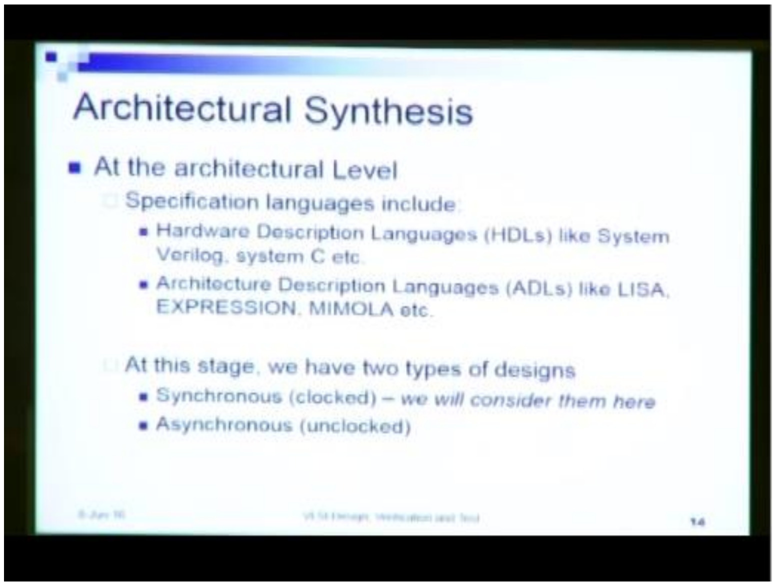
However, this whole optimization problem cannot be solved in one step due to the sheer complexity of the entire design process. And hence people have divided the entire design process into a set slipped design process into a set of steps, so that the complexity can be reduced. Obviously if you divide a single optimization problem into a set of steps some optimality will be introduced however to manage the complexity this is best creed of that we have.

And hence people over the years of people over years of research have found out these set of concrete set of steps to which the design flow should propose. So we come back to the original discussion that we were doing. So architectural synthesis, architectural synthesis is also called high level synthesis, so the input to this is a behavioral model which can be abstracted as a set of threaded concurrent, communicating process modules.

So whatever we have being saying we are said that this architectural module the behavior will be its input and it will be represented as formally as programs right. So what will these programs represent it will represent threaded concurrent, communicating processes. So these processes will run concurrently and will communicate with other processors. And what will this processors represent, these processors will represent a set of operations and the dependencies.

The process modules will have interfaces with other blocks and the outside world. So here we have four modules, so these four modules will or four different functionalities as we have found out, they are four, these are four different program codes and these codes will run independently of one another concurrently and will communicate among themselves through these interfaces and can also communicate to the outside world. So this is what this figure shows.

(Refer Slide Time: 05:26)



Architectural Synthesis

- At the architectural Level
  - Specification languages include:
    - Hardware Description Languages (HDLs) like System Verilog, system C etc.
    - Architecture Description Languages (ADLs) like LISA, EXPRESSION, MIMOLA etc.
  - At this stage, we have two types of designs
    - Synchronous (clocked) – *we will consider them here*
    - Asynchronous (unclocked)

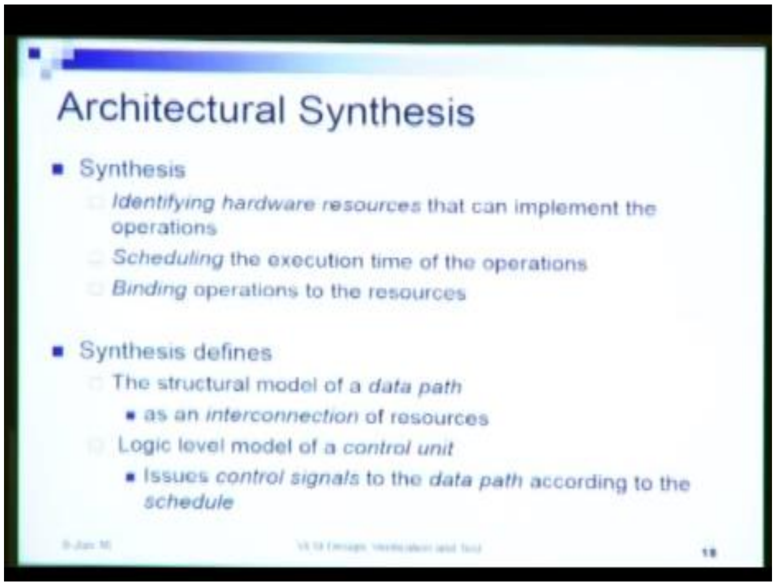
14

Now how will architectural level, this architectural level specification the input be written in so as we said it will be written in hardware description languages like Verilog, system C, HDL etc. or in architecture description languages, these architecture description languages are at a higher level, you can describe pipeline stages, you can describe the structure of the ALU at a much higher level that the Verilog.

The output of these languages will be another weight of an HDL code. So you can also describe using architectural description languages like LISA, EXPRESSION, MIMOLA etc. So at this stage or design we can have two types of designs, we can have synchronous or clock designs or asynchronous designs. So as most of today’s designs are synchronous and it is much more mature than asynchronous designs.

In the rest of these course we will only be looking, we will only be taking synchronous designs into consideration. This is to identify hardware resources that can be implement this operations, then we have to schedule the operations in different time sets. So scheduling the execution time of the operations we have to schedule the operation at distinct time sets. And then we have to bind this operations to the hardware resources that we have identified so the synthesis goes to this three major steps identify the hardware resources scheduling.

(Refer Slide Time: 07:14)



Architectural Synthesis

- Synthesis
  - *Identifying hardware resources* that can implement the operations
  - *Scheduling* the execution time of the operations
  - *Binding* operations to the resources
- Synthesis defines
  - The structural model of a *data path*
    - as an *interconnection* of resources
  - Logic level model of a *control unit*
    - Issues *control signals* to the *data path* according to the *schedule*

[illegible] VLSI Design, Verification and Test 18

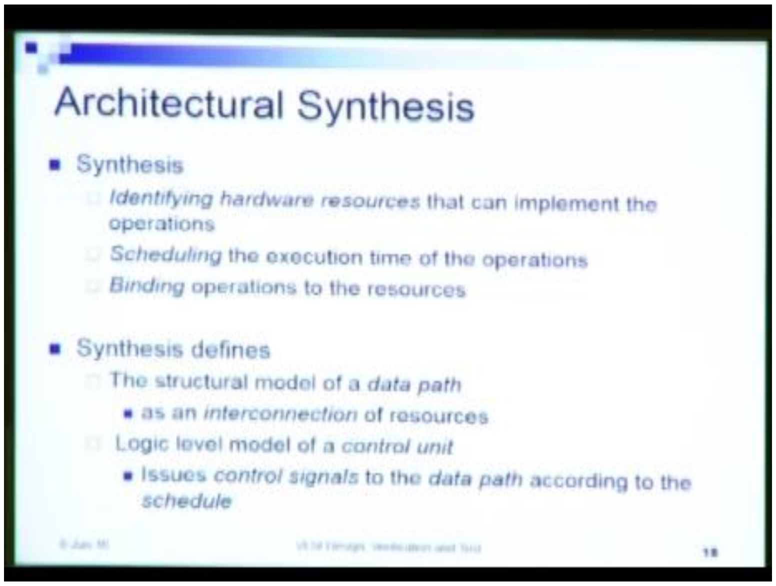
It into time steps the operations and then binding the operations in the hardware resources so what does synthesis defined so then synthesis defines the structural model of a data path so why are we saying it a data path as we said the program in will take some data as input it will transform that data and the produce this transformed data as output after a set of transformations through the operations that I have stated so this constitute a data path how the data flows initial input to the final output right.

So what it what this process will do basically will produce a structural implementation as structural model corresponding to this data path and how will this data path be realized as an interconnection over resources over the resource what are resources are we are just coming and the logic it will also produce not only this data path but a logic level model of the control unit so what will this control unit do it will issue control signals to the data path according to the schedule.

So what did we say we said that we have set of operations this operations will be scheduled in a set of time steps that data path will be realized so these operations will be realized through a data path that data path will have these operations and an interconnection among these operations so these operations will be implemented through resources like implementing these operations on recourse and they will be interconnections between these resources now when we have a schedule of which operations to be excited at a particular times steps.

How will control which operations to be scheduled at which time steps because these operations are on recourse these particular resources which contain particular data at a certain registers at a given time as to be activated as those instance of time and therefore as to be control so that the correct data flows from the register to a functional unit and a correct output is produced and goes back to a register I will take this in more detail in the next few studies.

(Refer Slide Time: 09:43)

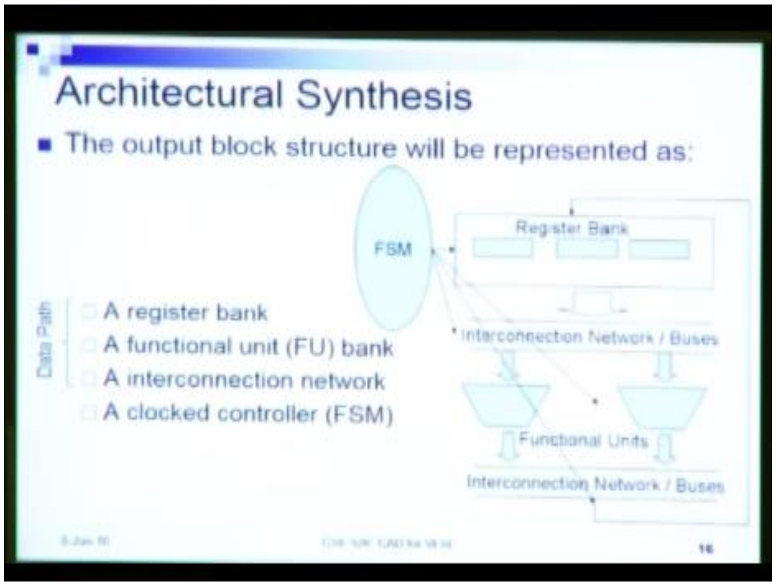


Architectural Synthesis

- Synthesis
  - *Identifying hardware resources* that can implement the operations
  - *Scheduling* the execution time of the operations
  - *Binding* operations to the resources
- Synthesis defines
  - The structural model of a *data path*
    - as an *interconnection* of resources
  - Logic level model of a *control unit*
    - Issues *control signals* to the *data path* according to the *schedule*

So what where basically trying to saying is that the control unit will issue control signals to the data paths according to the schedule so it will control which registers which functional units should be activated at a given at each time step so that the correct sequence of operations are performed according to the prescribed schedule okay.

(Refer Slide Time: 10:10)



Architectural Synthesis

- The output block structure will be represented as:

Data Path
- A register bank
- A functional unit (FU) bank
- A interconnection network

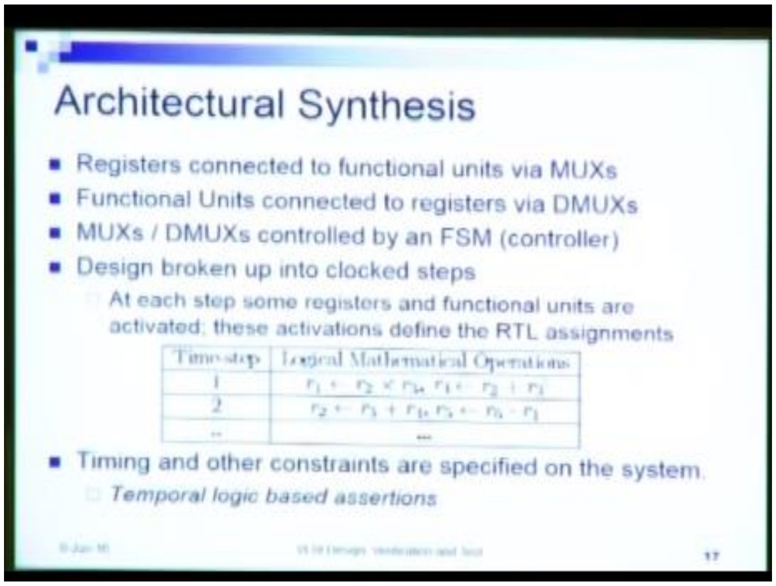
- A clocked controller (FSM)

FSM
Register Bank
Interconnection Network / Buses
Functional Units
Interconnection Network / Buses

So we said the input specification to the synthesis process we said what does syntheses process does and then what will be the output of the syntheses process the output of the syntheses process as we said is a data path and a control path the controller design and the data path the data path will be composed of what a register bank a functional unit bank and interconnection network so I will have a registers bank at the S output these will store the inputs to the operations at different time units.

So these registers at different times can store different inputs I will take an example and explain and then we these data will be these registers will be activated at approved instance of time so that correct data flows through the interconnection units to the functional unit and the correct output is produced and then it will again proceed through the buses through the register bank so the register bank as functioning unit and the interconnection network will form the data path and it will also have the controller efficient that we control which registers slash functional units to be activated when.

(Refer Slide Time: 11:37)



Architectural Synthesis

- Registers connected to functional units via MUXs
- Functional Units connected to registers via DMUXs
- MUXs / DMUXs controlled by an FSM (controller)
- Design broken up into clocked steps
  - At each step some registers and functional units are activated; these activations define the RTL assignments

Time step	Logical Mathematical Operations
1	$r_1 \leftarrow r_2 \times r_3$, $r_1 \leftarrow r_2 + r_1$
2	$r_2 \leftarrow r_3 + r_1$, $r_5 \leftarrow r_6 - r_1$
..	...

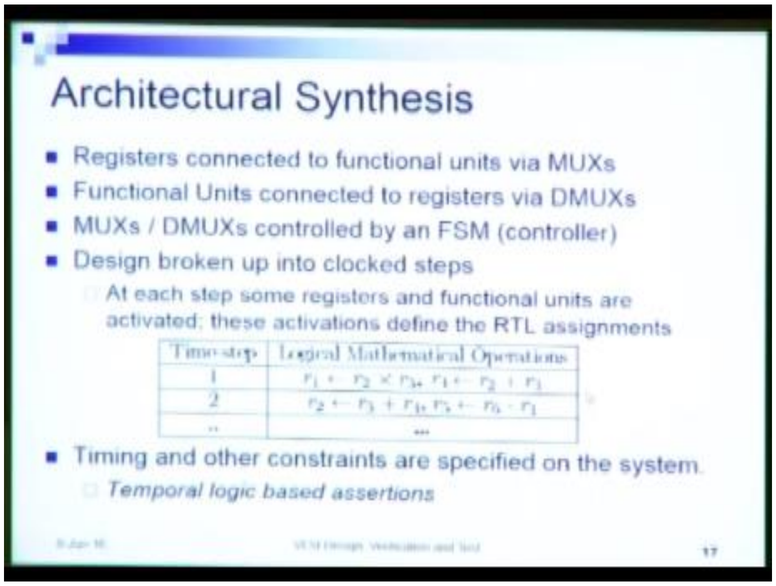
- Timing and other constraints are specified on the system.
  - *Temporal logic based assertions*

So the registers how we said there are registers and how will they to control how will they be controlled how will the this controller unit control as to what data should pass from the register when to which functional unit how will this be condoled it will be the registers that connected to the functional unit by a bugles the bugs is an any input one output device so therefore what will you do the registers float there outputs on the inputs of these markers and the output of the makes goes to the functional unit.

So therefore depending on which select lines are activated what times there the set of register outputs right so what are these register outputs will be one of this register outputs will be taken from the input to the output of the mass and therefore which load to the functional units right similarly the functional units are connected to the registers through d markers so my d marks is again one input n output device so the output of the functional unit now should go to a particular register it cannot go to all registers.

To a particular register where the output of this operation should be should legitimately reside so therefore the controller will appropriately select the particular output of the d marks so that the output flows from the so that this output from the function unit flows through the demarks to the appropriate register.

(Refer Slide Time: 13:31)



Architectural Synthesis

- Registers connected to functional units via MUXs
- Functional Units connected to registers via DMUXs
- MUXs / DMUXs controlled by an FSM (controller)
- Design broken up into clocked steps
  - At each step some registers and functional units are activated; these activations define the RTL assignments

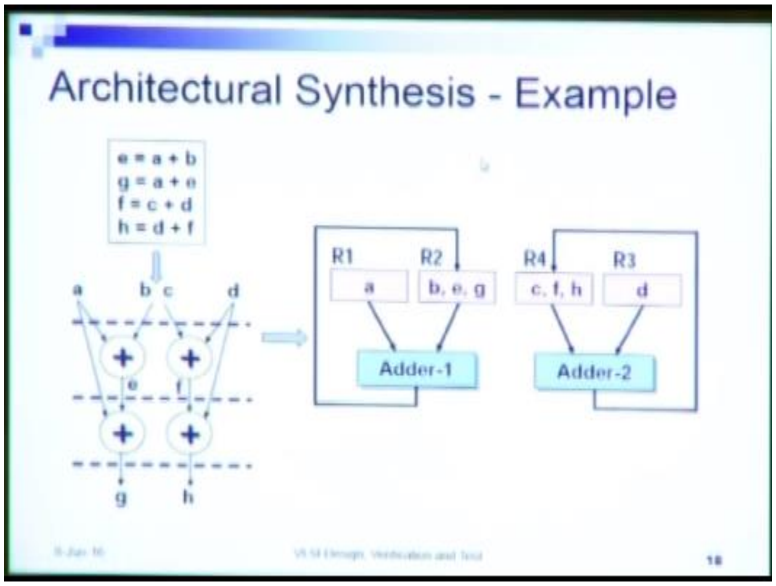
Time-step	Logical Mathematical Operations
1	$r_1 \leftarrow r_2 \times r_3, r_4 \leftarrow r_2 + r_3$
2	$r_2 \leftarrow r_3 + r_4, r_5 \leftarrow r_6 - r_1$
..	...

- Timing and other constraints are specified on the system.
  - *Temporal logic based assertions*

Right so therefore the third point here is that the marks and the demarks are controlled by the FSM now the design is therefore broken down into a set of clock steps each again steps some of the register and functional units are and these activations defined the RTL assignments so therefore we have here three two times that shown at ti9me step one we have R1 = R2 X R3 and R4 = R2 + R3 and in the second times that we do R2 = R3 + R4 and R5 = R6 – R1 so therefore del there are a set of parallel register transverse at each time step, so this is the mathematical logical output of the architectural synthesis base and the structural output being the realization interms of registers, buses, memory function and units etc.

Now at this step because we have the banner register transverse and the time also specify for resetting the timing and the other constraints are can also be specified by in the system through a set of temporal logic assertions and hence we can verify that this whether this design is correct with respect to the timing properties that means a suppose I have an operation that should be done by the end of the 5th time step am I actually doing it. The time the temporal logic base assertions with verify that particular feature aspect okay.

(Refer Slide Time: 15:30)



Architectural Synthesis - Example

```
e = a + b
g = a + e
f = c + d
h = d + f
```

So now we take a small example to illustrate what we have been discussing over the last few slides, let us say that we have this small toy we have Ural description or an application at the input of the architectural synthesis technique, so these are four operations E = a + b, G = a + c, F = c + d and H = d + f so we see that the input E is dependent on this output of the previous step here this input F is dependent on the output here, so therefore we can first we said what do we do, we identify the hardware components that we resources that we require and we understand here that we require adders and registers to implement this particular behavioral description, now this behavioral description then will be scheduled.

So this process is scheduling this shows two time steps so what does it do, so in time step one if there is A and B as input and produces E as output and in time step 2 what happens this E is taken as input A is the other input and G is produced as output we understand the there is a dependents between this thing these two operations these two addition operation because this E cannot this E cannot be used until this output is produced at a previous step.

So we have a dependents here similarly we have a dependents at this step what do we do, so we have these two independent input C and D and in the first time step we do the addition operation output of the addition operation is this F and this F is taken as input in the next time step at this addition the next addition operation and the next time step D is the other input and H is finally produces output at the output of the second time step.

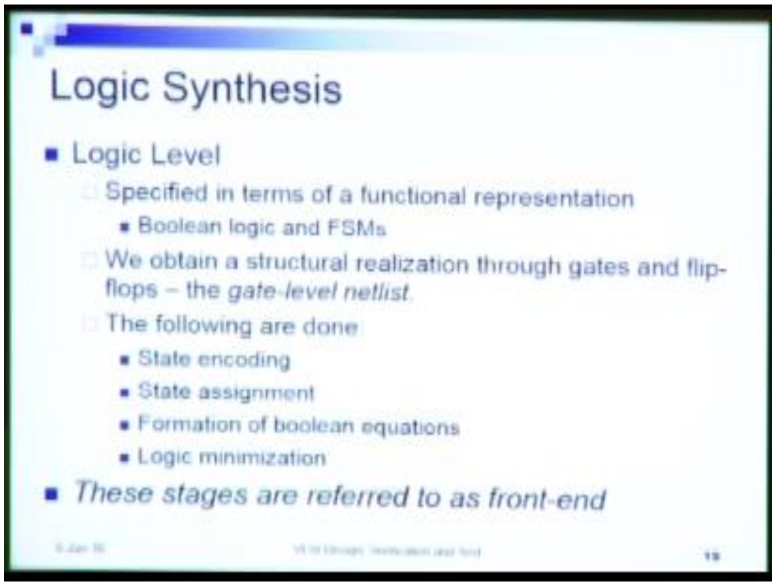
So we understand that this whole design can be completed in two time step however because there are two parallel additions in a particular time step we can be need atleast two adder units, two adder resources, now a particular do you realize mean so a sample realization of this schedule step is obtained here so this process is called allocation and binding, so we allocate different input data to different registers and different operations to different adjacent bind them, so for example we had used register 1 to store A.

And register 2 to store B, E, G so what we have let us see how we can realize this at time step one a time step 1 we have B and E as input to this adder and so I will have one marks so I will have a marks that takes the input to this adder and another one that takes the input to this one, okay. So then the output of this adder will then go through at D marks to register 2 so the demarks will tell will enable us to prevent the output from going to R1 and to go to the register R2 as desired.

Then what happens now we have at the end of first time step we have this E in register are to NA as we just to we have not registered R1 then at the second time step A and E are given to adder 1 and then what happens, the output of adder one is now G this G then goes back to the register R2 at the end of time step 2, the same thing happens for the other adder for the other adder first C and D is the input at time step 1 C and D are driven to adder 2.

The result of addition is F and is driven to a register R4 and not to R3 by using a de-multiplexer at the output of adder 2 and then what happens is that F and D are driven at time step 2 to adder 2 again and output H is produced which goes back to R4 so at the end of time step 2 we have G and H in register are 2 and R4 add that correctly implements the behavior that we want to have.

(Refer Slide Time: 21:03)



Logic Synthesis

- Logic Level
  - Specified in terms of a functional representation
    - Boolean logic and FSMs
  - We obtain a structural realization through gates and flip-flops – the *gate-level netlist*
  - The following are done
    - State encoding
    - State assignment
    - Formation of boolean equations
    - Logic minimization
- *These stages are referred to as front-end*

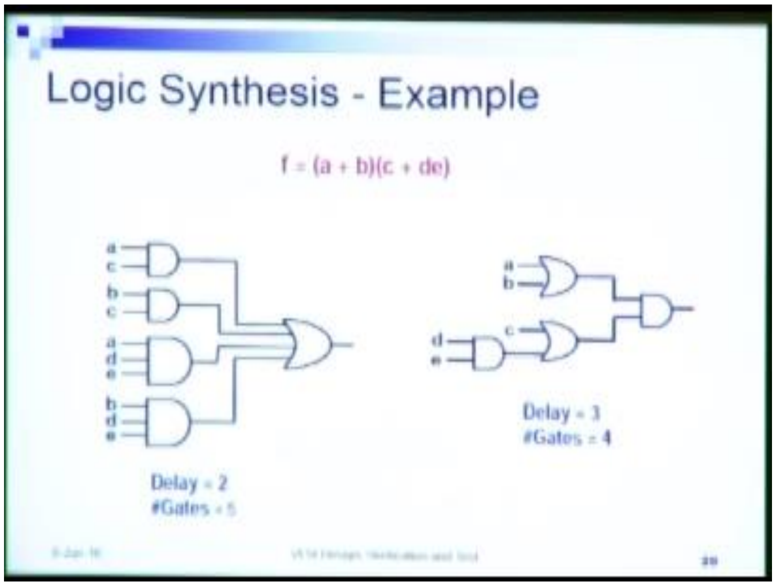
So after the architectural synthesis step after a look at the architectural synthesis step we will go in deep early to the architecture synthesis step in the next lecture. Now we take a look at what logic synthesis does, so at the logic level what do we have, we have the register transfer level design, the register transfers the parallel register transfers at the output of sorry, at the output of the architectural synthesis step we also have the controller so we have the FSM of the controller so we have the FSM of the controller, so this RTL level transfers.

The RTL description as well as this FSM has then to be encoded in terms of a Boolean expression, in terms of Boolean equations and also this FSM has to be we have to do a state encoding assignment and realized in terms of a digital circuits so the FSM and RTL description have to be taken and corresponding digital circuit has to be made out of that. So at the logic level the input is in terms of a functional representation, the functional representation is Boolean logic and FSM and what is the output, the output is a structural realization through gates and flip flops.

So what we have is called a gate level net list at the output or the logic synthesis step. So what do we do here, we do the following we do state encoding state assignment formation of Boolean equations and logic minimization so at the end of the logic synthesis step the front end of the logic synthesis step we have a minimized digital circuit corresponding to the functionality that we have at the beginning of architectural synthesis step.

So that architectural synthesis step at the beginning of the logic synthesis step was transformed into a set of Boolean equations and those Boolean equations where then realized in terms of gates and the output of the synthesis step and the front end of the logic synthesis step.

(Refer Slide Time: 23:36)



Logic Synthesis - Example

$f = (a + b)(c + de)$

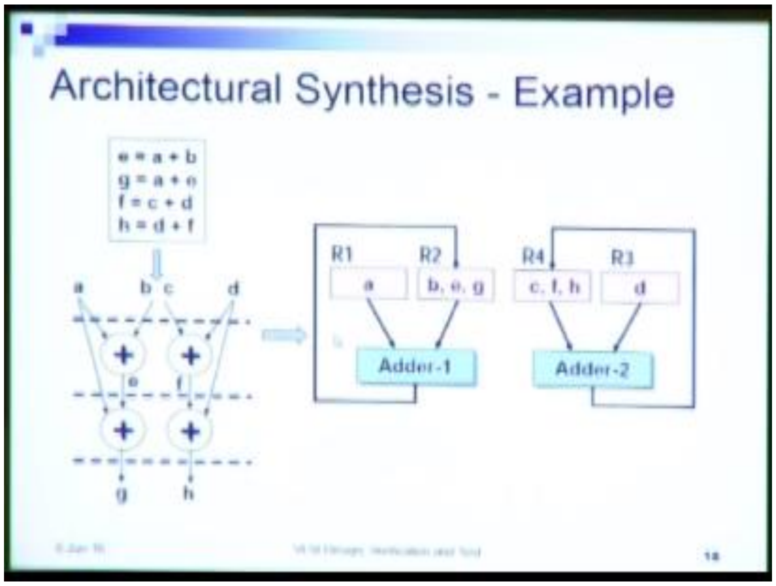
So here is an example, let us say we want to implement the function this following Boolean equation (a+b) (c+de) so one way is to implement here so we have two different implementations of the same Boolean equation that we have as input. So if we implement it in the form that we have here what we have is the direct transformation of this so a and b we have an OR gate and this one we have de an AND gate the output then passes through c and the output of de we have this OR gate and finally we have the output at the output of this AND gate.

We see that the delay this is the three stages there are three stages in this net in this logic circuit and the delay of this circuit is 3 and the number of gates required are 4. Now that same function can be implemented as follows, so we have ac if we just break this up we will gate this and with the same function can be implemented as follows and we see that there are two stages in this circuit and so the delay of this circuit is 2, so what we have is the lower delay is a lower delay realization of this Boolean equation.

So this both this realizations have the same output and implements the same functionality however this implementation has higher delay but lower number of gates and this one has a lower delay but higher number of gates we have 5 gates required here, so therefore the area of this circuit will be higher than the area of this circuit however the delay of this circuit will be lower than the delay of this circuit.

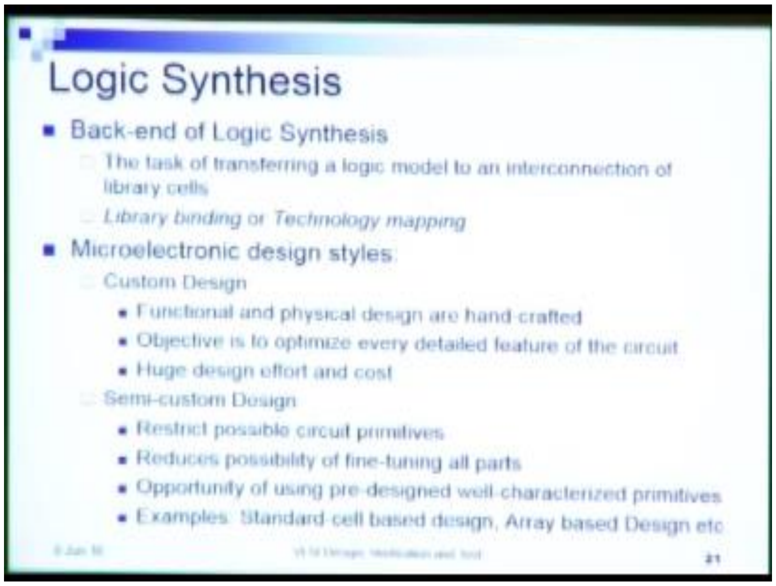
Hence, we can say that this will be lower in performance but also lower in the area consume this will be higher in performance consuming lower delay but also higher in terms of area consume, right this is called the micro level optimization of the circuit. We can also do such optimizations at the architectural level and such optimizations will therefore be called the macro level optimizations of the circuit.

(Refer Slide Time: 26:18)



For example, if we take this example we said that this one for this realization to be required two adders we could easily have done a serialization and done this whole thing using one adder in four types steps, so if we had just one adder we could do the same thing in four type steps so then obviously it will require lower area just requiring one adder hardware unit however it will require four time steps to complete the whole operation and hence it will be lower in performance. So here we also have an area verses delay trade of.

(Refer Slide Time: 27:01)



Logic Synthesis

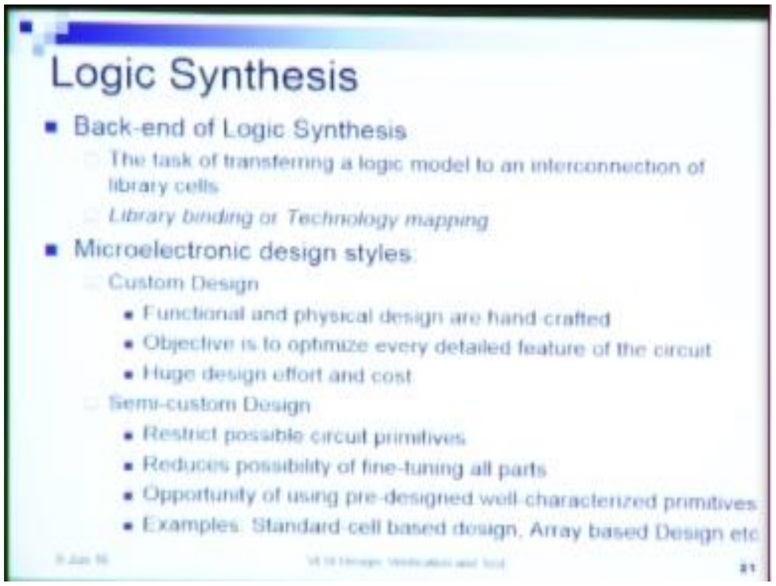
- Back-end of Logic Synthesis
  - The task of transferring a logic model to an interconnection of library cells
  - *Library binding* or *Technology mapping*
- Microelectronic design styles
  - Custom Design
    - Functional and physical design are hand crafted
    - Objective is to optimize every detailed feature of the circuit
    - Huge design effort and cost
  - Semi-custom Design
    - Restrict possible circuit primitives
    - Reduces possibility of fine-tuning all parts
    - Opportunity of using pre-designed well-characterized primitives
    - Examples: Standard-cell based design, Array based Design etc

[illegible] VLSI Design: Verification and Test 21

Now we come to the back end of the logic synthesis system, the back end of the logic synthesis step does this it does the task of transforming the logic model to my interconnection of library cells, this is popularly known as library binding or technology mapping. So it will take you from the gate level design to the transistor level design. Obviously, you have many different design in terms of the semi conductor technology here TTL, CMOS, Bi-CMOS extra also they can be different design styles here you can have custom design and semi custom design for example.

In a custom design the functional and physical design are all handcrafted so everything is done by hand and hence the objective is to optimize infelt detail future of the circuit.

(Refer Slide Time: 28:05)



Logic Synthesis

- Back-end of Logic Synthesis
  - The task of transferring a logic model to an interconnection of library cells
  - *Library binding* or *Technology mapping*
- Microelectronic design styles
  - Custom Design
    - Functional and physical design are hand crafted
    - Objective is to optimize every detailed feature of the circuit
    - Huge design effort and cost
  - Semi-custom Design
    - Restrict possible circuit primitives
    - Reduces possibility of fine-tuning all parts
    - Opportunity of using pre-designed well-characterized primitives
    - Examples: Standard-cell based design, Array based Design etc

8 Jun 16 VLSI Design, Verification and Test 21

Obviously the circuit is very complex it is huge if it is huge then such a design is impractical to be done it is impractical because it will incurred huge design effort and cost with respect to this a we have semi custom design in a semi custom design the objective is to restrict the possible circuits primitive choices that we have, now we said that if the custom design we can optimize up to the last transistor right but in this you cannot do so you have set of puddle computer and design unit cells of implementation.

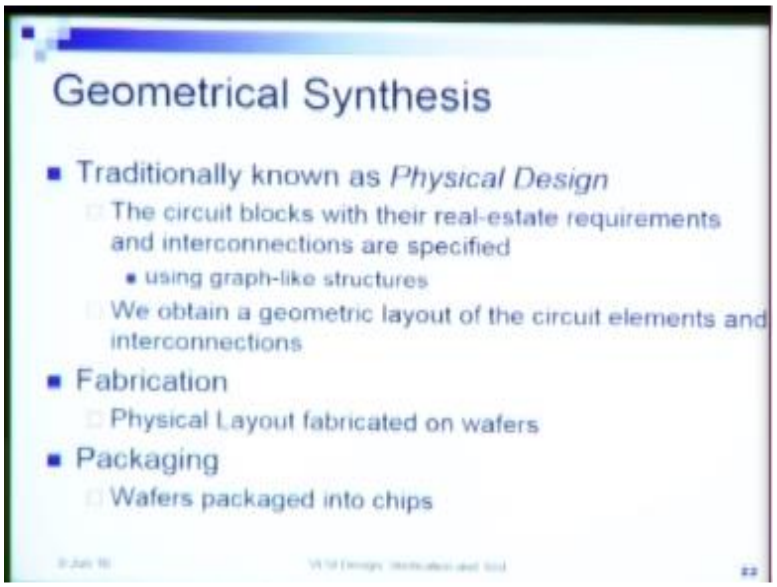
And that cannot be handed that cannot be twit with by hand and this restricts the circuit this restrict that one of choices that you have but that also preserve the opportunity of handling very large scale designs and also it also gives you the opportunity of using puddle design and well characterize primitives because these cells are primitives that we have her can be optimize with respect to design because that will be used in all designs.

So you cannot puddle design you can free design this units optimize it to the best extend possible and then use it. So the trade of is that you cannot optimize with respect to the last transistor possible however you can handle much bigger this designs and you can optimize across this limits. So there are many different examples of semi custom design for examples standard cell based design handed based design etc.

In standard cell is design you have standard cell you have a library of standard cells and you use this cells to design your circuit so you design your circuit and then fabricate hand based design on the other hand comes pre fabricated which pre diffusion pre wire so it comes as a at array of gates for example fell programmable gad anis so it comes as an array pre fabricated pre diffuse pre wired gads and you have to program this online to achieve the logic that you want to implement.

So how do they do that they use something call anti fuses to short circuit and open circuit and appropriate connections between this gad arrange to realize appropriate digital or to prepare functions. After the logic synthesis step become to geometrical synthesis.

(Refer Slide Time: 31:11)



Geometrical Synthesis

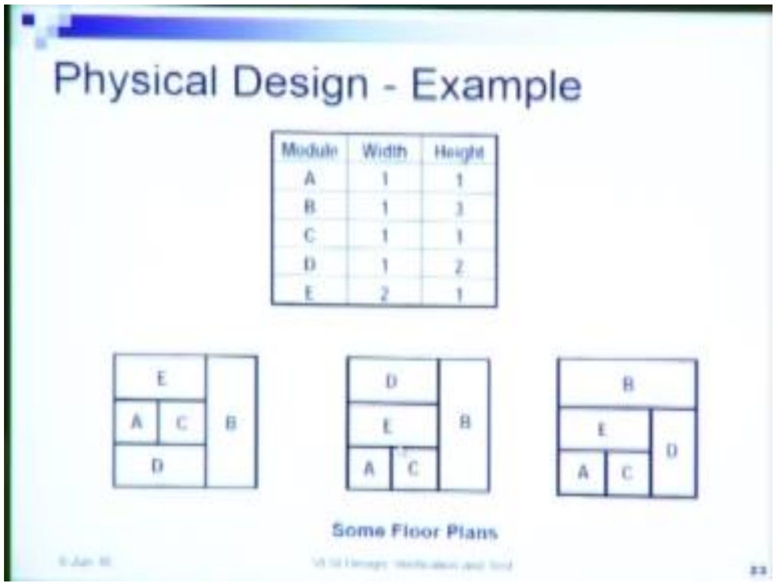
- Traditionally known as *Physical Design*
  - The circuit blocks with their real-estate requirements and interconnections are specified
    - using graph-like structures
  - We obtain a geometric layout of the circuit elements and interconnections
- Fabrication
  - Physical Layout fabricated on wafers
- Packaging
  - Wafers packaged into chips

This step is popular to known as physical design and this step the circuit blocks or circuit locks with then real estate requirements now what we have is that for each component we have designed and we know what is the expected area on the subtract that I will require to implement this law, the circuit blocks with the real estate requirements and then inter connections has specified using graph live structure.

So this net list as I said before are graph like structure well each note is a component and the edges define connection between components. It just had hypocrites define connection between components, so we obtain the geometrical layout of the circuit element so this is the behavioral this is the input to the physical designing step is this structure this graph like structure and the output is what like the geometric layout of this circuit elements along with the inter connection as that as will be obtain on the subtract.

So after you have obtain the layout you actually fabricate this layout on the way for and obtain the physical chip which is the in the chip is that package and then you it is ready to wanted the is in ready to wanted.

(Refer Slide Time: 32:43)



Physical Design - Example

Module	Width	Height
A	1	1
B	1	3
C	1	1
D	1	2
E	2	1

Some Floor Plans

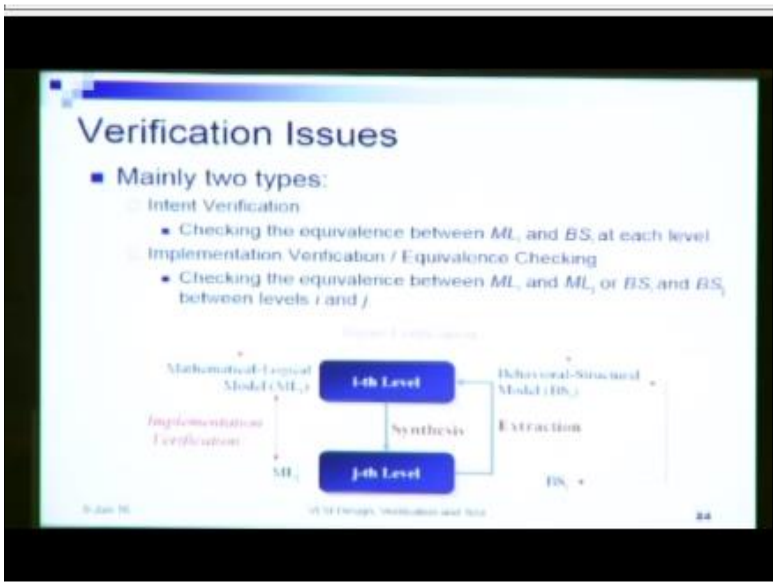
As that example of the physical designs so let us suppose the assume that we have four module five modules so the first module has width one and height none so here we have the modules we have lots specify the inter connections in this very simple example but we assume that any real example will have inter connection requirement between them so here we have five modules, module a as a width of one and a height of one.

Module b has width of one and a height of three module c has a width of one and height of one module d has a width of one height of two module e has an width of two and a width of one and w have three defer at layout plans for the same set of modules, here see this one the first one it has width of 2 and a height of 1 a has a width of 1 height of one c has a width of one and height of one d has a height of 2 and we have wrote it this at obtain like this.

D has a this is the height here basically height of 2 and the width of one and b has a height of 3 and the width of one okay so what we see is that because we have really been able to obtain three different floor plans primarily because that inter connections are not so stringent here obviously when we have actual connection restrictions inter connection restrictions all these different layouts may not be feasible all these differently layouts may not confirm to the requirements that I have let us say the critical delay that I require between any two connections the net length.

The length of between two components may or may require to have bounded by a couple bound it cannot go beyond a certain limit because timing restrictions will then be violated so the signal must pass from say module A to module B within three time units but the length of the wire connecting module A and module B have let us say cannot provide timing bound it requires at least four units so then I have to place a such in such a way that the distance between module A and module B will be such that the violate can be reduced. So that the timing specification can be better.

(Refer Slide Time: 35:27)



Verification Issues

- Mainly two types:
  - Intent Verification
    - Checking the equivalence between ML_i and BS_i at each level
  - Implementation Verification / Equivalence Checking
    - Checking the equivalence between ML_i and ML_j or BS_i and BS_j between levels i and j

Mathematical-Logical Model (ML_i)
I-th Level
Behavioral-Structural Model (BS_i)
Implementation Verification
Synthesis
Extraction
ML_j
J-th Level
BS_j

At each step of the design we also said this was the design flow in an actual at each step of the design right from the initial specification to the implementation pre silicon we have verification to be done so we need to verify that the design is correct at each stage of the design like so when we are doing a synthesis from less detail one to more detail one errors can come up and we need to verify that there are no errors respect to the specification that I have.

So there are two main types of verifications that we can have one is the intent verification that is checking the equivalent between the MLI and the BLI at each level so this is with respect to the figure that we have here so the figure says that the Ith level and then from the synthesis process I will come to the Jth level so this is more structural in respect to the Ith level so this is less detail this is more detail this is more we have this is more structural because this is the synthesis process in this direction.

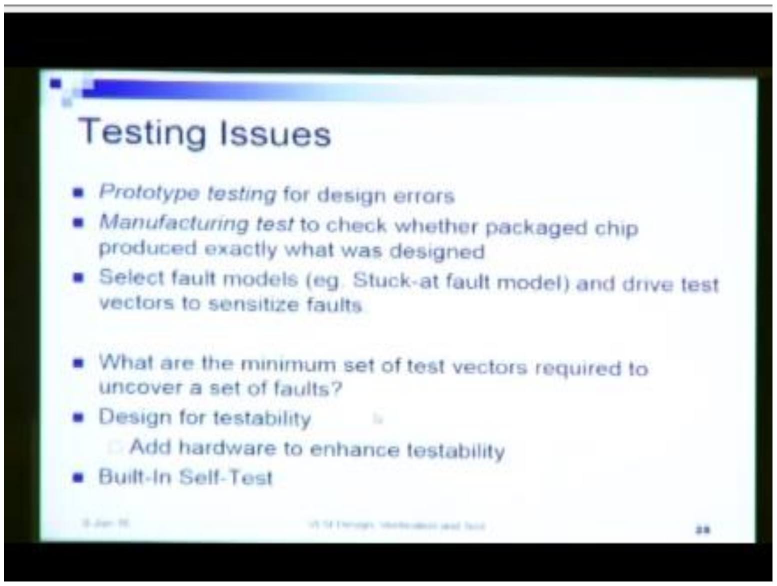
And in each of these levels we can have a mathematical analogical model of the specification so at the Ith level I had ML I as the mathematical analog specification at level j I have this energy as the mathematical analogical specification correspondingly we have one BNL structural module this BSI represents this BNL structural model at the Ith level and the BSJ represents the BNL structural module at the Jth level okay.

And we say that what is inter verification, inter verification is to verify that whatever I have at the BNL structural model that the Ith level that is correctly we will cooperate the requirements of the specification I have so I will mathematically analogical model the specification which will incorporate with all the properties that the design needs to satisfy and then we will verify that all the properties all the tiny properties say we have utility properties functional properties all these properties are the correct with respect to this mathematical analogical specification.

So this is verified whether whatever I get intending to do is correct so whether the indent is correct or not is the indent verification as against indent verification or equivalent study what is this tell me this verification does what does this do it is checking the equivalent with MLI and MLJ or BSI or BSJ at two distinct level so is the implementation correct with respect with two mathematical specification at the distinct level is the synthesis with regarding respect to the structural specification at two distinct level.

This is verified by checking the equivalent between these two levels that is called equivalent checking.

(Refer Slide Time: 38:54)



Testing Issues

- *Prototype testing* for design errors
- *Manufacturing test* to check whether packaged chip produced exactly what was designed
- Select fault models (eg. Stuck-at fault model) and drive test vectors to sensitize faults
- What are the minimum set of test vectors required to uncover a set of faults?
- Design for testability
  - Add hardware to enhance testability
- Built-In Self-Test

And we also have testing issues involved at the prototyping level as well as the manufacturing level so we have prototype testing and manufacturing testing so prototype testing for design errors so if I have design errors the last place to check with respect to driving a set of test vectors and seeing whether the outputs are correct always under all conditions is by actually putting the driving test vectors through the prototype and finding that over all test vectors overall test inputs to the circuit the output of the circuit matches with respect to the expected output.

So this is called product testing and manufacturing test is to check whether the package is produce exactly what was design so after the fabirification application we have we might incorporate with errors in the design like the number the correct chips that we design with the percentage of correct chips that we design with respect to the total number chips manufactured is referred to the EN so manufacturing effects will basically determined what is the means how many are correct chips after actually fabrification process and packaging.

So manufacturing test checks whether the package chip produced what was design now how will you test you test with respect with certain fault models for example you can stuck at fault models check whether certain inputs are always stuck at one on one short circuit always stuck at one are always stuck at zero right so how you test that so these faults discovered by certain test vectors sensitized these faults so if I assume that a certain gate input always at one I have to appropriate test vectors to find out whether appropriate test vectors to find out whether the output is wrong for example if it is a orchid and one of the input is always stuck at one and then if we drive a zero, zero.

To this test vectors to this orchid output is still one we can understand that one of these inputs must be stuck at one so one of the very important issues in this thing is to find out one the minimum test vectors we required to uncover a set of faults we understand that any circuit any reasonably sized circuits will have an infinite numbers of inputs possible right so we cannot test a circuit foe all possible inputs that are possible so we have to find out the research here is to find out the find out the minimum test vectors that are essential that are required that can sensitize a given number of faults a given set of false we have.

 And we also have another aspect with the research is also done with respect to the design testability so in the hardware itself we have certain extra hard ware we higher hardware so that the design becomes more easily testable and we also have built in self test that means that the chip will automatically test whether the certain parts is correct is correctly functioning and that is called built in self test with this we come to the module 03 the last module of lecture 01.