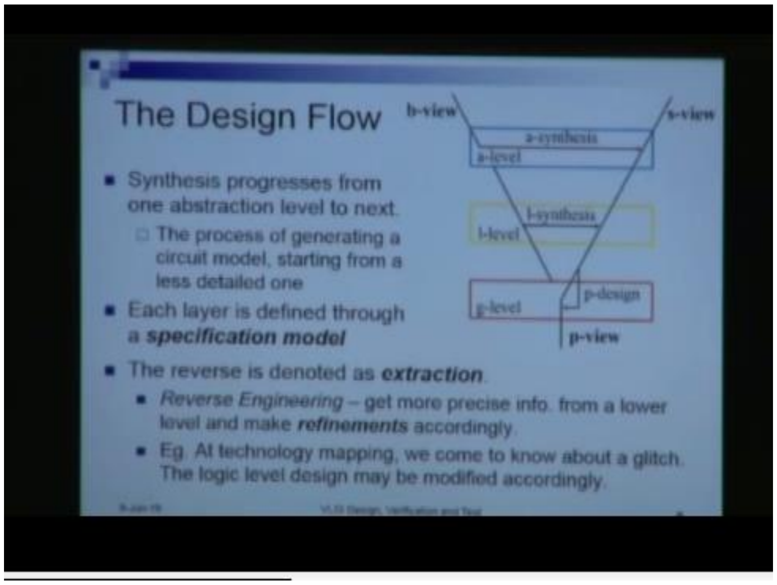
Hello, welcome to module 1 of lecture 2 of the course VLSI design verification and test. In lecture 1 we went through words IU of the full design VLSI design process, we saw that there are three important levels of abstraction at through which the design goes through the architectural level, the logic level and the geometric level.

And we saw different views through which the design goes through it is well. So we saw the behavioral view, the structural view and the physical view.

(Refer Slide Time: 00:39)

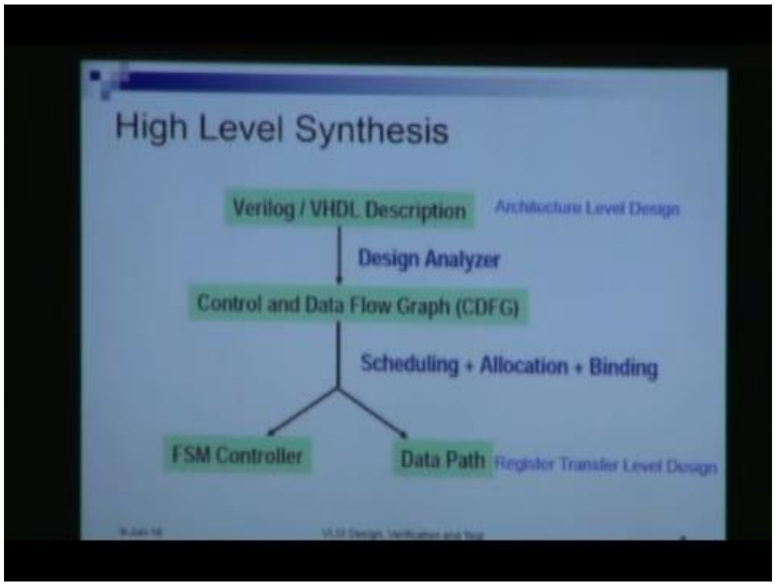

b-view
s-view
a-synthesis
a-level
l-synthesis
l-level
p-design
g-level
p-view

The Design Flow

- Synthesis progresses from one abstraction level to next.
  - The process of generating a circuit model, starting from a less detailed one
- Each layer is defined through a ***specification model***
- The reverse is denoted as ***extraction***.
  - *Reverse Engineering* – get more precise info. from a lower level and make ***refinements*** accordingly.
  - Eg. At technology mapping, we come to know about a glitch. The logic level design may be modified accordingly.

And we saw that the design goes through steps which are known as synthesis. We saw a few important synthesis steps of architectural synthesis, logic synthesis and geometrical synthesis or physical design.

(Refer Slide Time: 01:01)



High Level Synthesis

$$\begin{tikzcd}
\boxed{\text{Verilog / VHDL Description}} \arrow[d, "\text{Design Analyzer}"] & \text{Architecture Level Design} \\
\boxed{\text{Control and Data Flow Graph (CDFG)}} \arrow[d, "\text{Scheduling + Allocation + Binding}"] & \\
\boxed{\text{FSM Controller}} \quad \boxed{\text{Data Path}} & \text{Register Transfer Level Design}
\end{tikzcd}$$

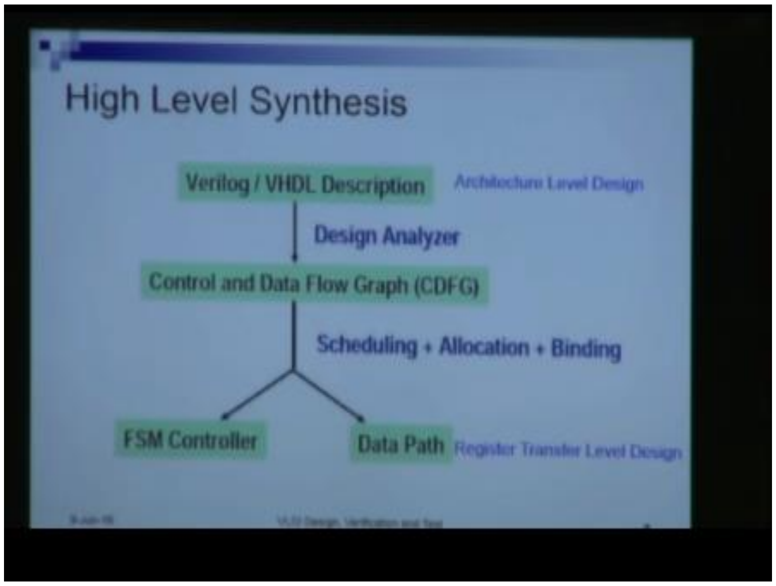
And now today what we will do is that we will take a more detailed look into the first important design step that is architectural synthesis or high level synthesis. So before going into this step, let us say that the initial design starts even earlier than this. The initial design which we may say as application level will also contain a specification of the entire application. Now this application will also be designed in terms of a set of processors which we will be communicating.

However, for all of these if we look from the perspective of an entire embedded system all these processors we may not need to implement them with hardware. Some of them, it will be sufficient for us to implement in software, which means that for those processors, for those blocks of code we would not, we will only use a general purpose processor and execute a high level program like the C language program for the process and run it and that will be sufficient.

So this step is called hardware, software partition. Now the future sense starting from the architectural synthesis which we are concerned with here will be concerned with only the hardware synthesis part. So the software is taken care by a different process, different set of different flow we will not take here.

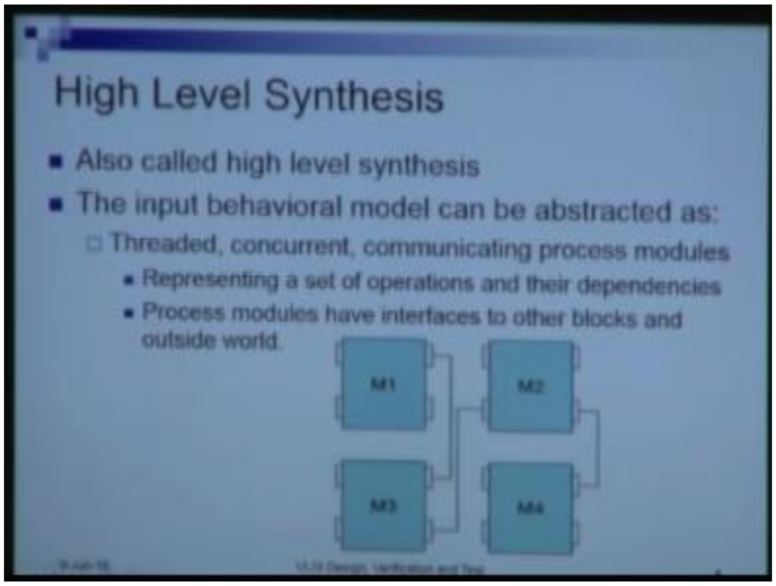
So the hardware also, at the beginning of the architectural synthesis step we will have a set of concurrent communicating processes each such process will be described formally using a programming language or high level hardware description language program in Verilog, BHD and etc, and the high level synthesis set we will take it from the architectural level design and will end up into a register transfer level design.

(Refer Slide Time: 03:12)


High Level Synthesis
Verilog / VHDL Description
Architecture Level Design
Design Analyzer
Control and Data Flow Graph (CDFG)
Scheduling + Allocation + Binding
FSM Controller
Data Path
Register Transfer Level Design

So what we will start with first the fail of VHDL discretions will be there then the design analyzer which is again program will transform this program into it is control and data program given this transfer given this control and data flow graph we will perform scheduling allocation and binding on them and ultimately obtain a controller and the data path.

(Refer Slide Time: 03:38)

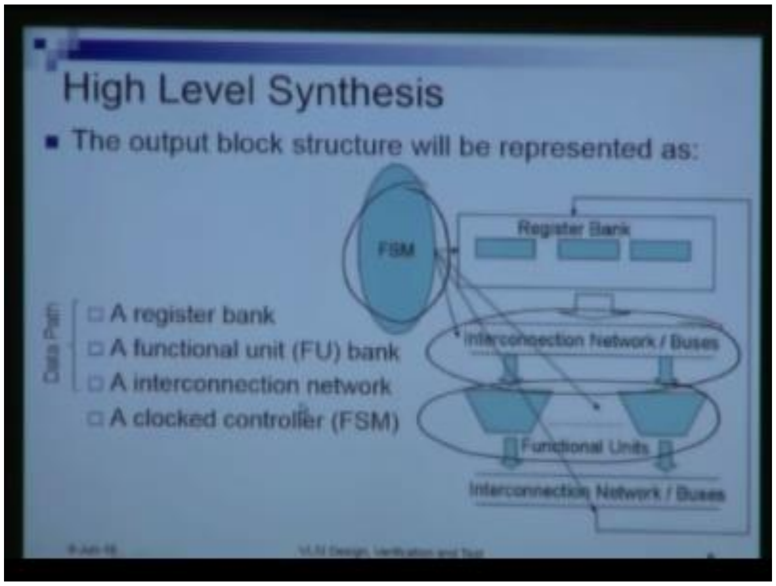


High Level Synthesis

- 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.

So architectural synthesis is also called high level synthesis the input behavioral model is abstracted as straighter conference communicating process so these process represent the operations and the data how the data is transformed through this operations and process modules as interfaces to other block and the outside worlds as we said yesterday.

(Refer Slide Time: 04:03)



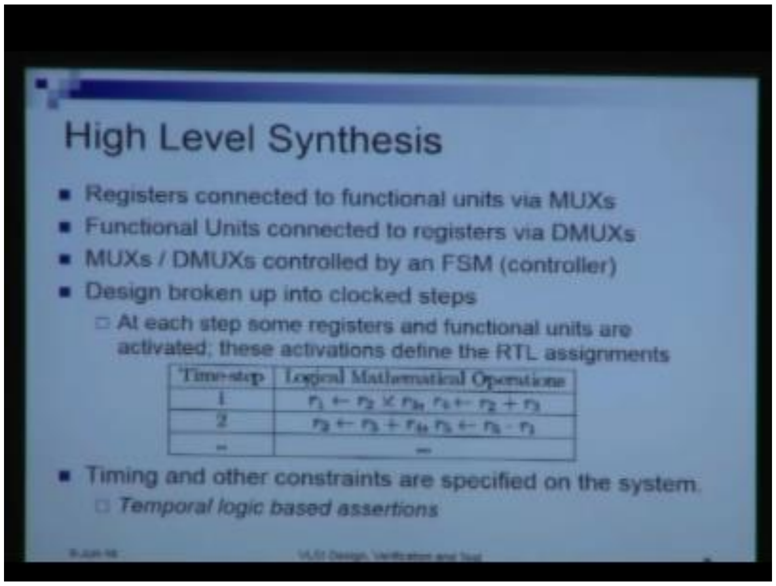
High Level Synthesis

- 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
VLSI Design, Verification and Test

And the output of the high level syntheses step is a data path and control path the data path is composed of a set register set of registers are n register bank a functional unit bank and interconnection network so this is the register bank this is the functional unit bank this is the interconnection network okay and along with that we will this controller FSM which will control operations the flow of data in this data path.

(Refer Slide Time: 04:49)



High Level 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

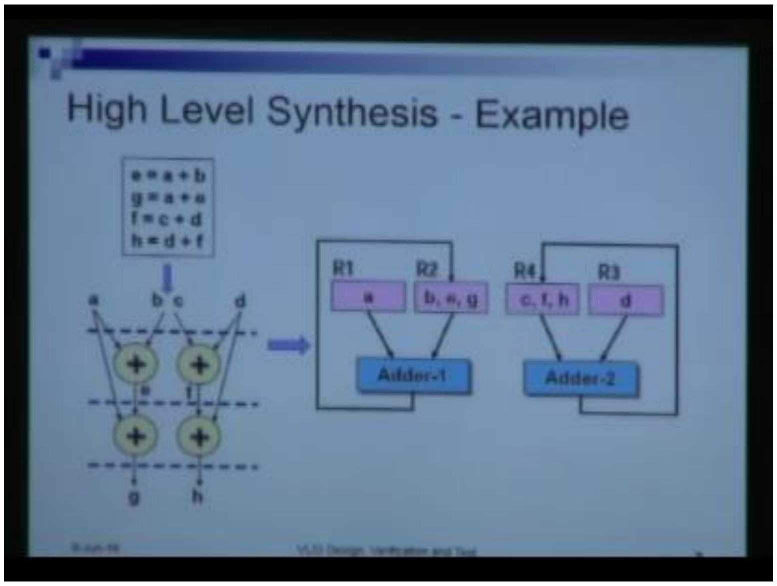
Time-step	Logical Mathematical Operations
1	$r_1 \leftarrow r_2 \times r_3$, $r_4 \leftarrow r_2 + r_3$
2	$r_2 \leftarrow r_3 + r_4$, $r_5 \leftarrow r_6 - r_1$
--	--

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

VLSI Design, Verification and Test

So yesterday we said that registers are connected to functional units by bugs the functional units again are connected to the registers demarks marks and demarks are controlled by the HSM controller and after this design process after the high level syntheses process the design gets broken down into a set of steps into a set of clock steps and then each set some registers and functional units are activated and these activations defined RTL assailments so we also sad yesterday that we will have a set of parallel register transfers r1 = r2 X r3 and r4 = r2 + r3 times take one every times step 2 we have r2 = r3 + r4 and r5 = r6 – r1 so although I set this yesterday but I did not possibly take a concrete example showing the use of this bugs demarks etc.

(Refer Slide Time: 05:53)


High Level Synthesis - Example

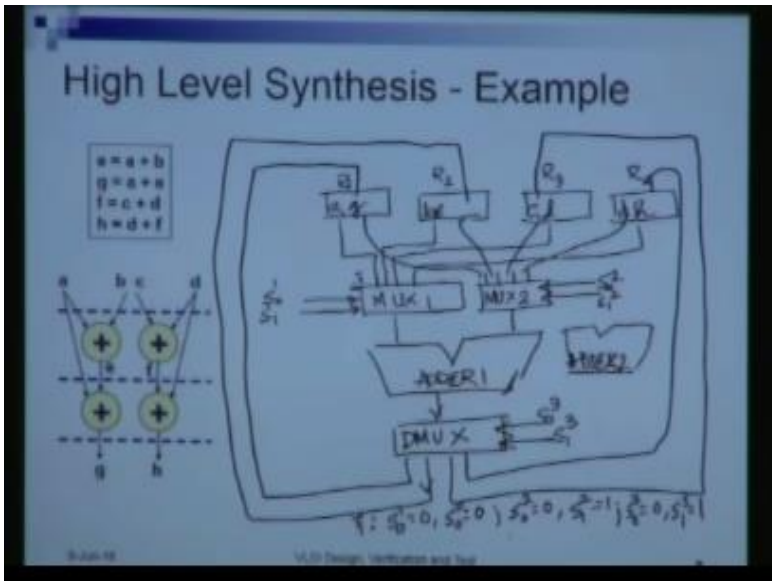
```
e = a + b
g = a + e
f = c + d
h = d + f
```

So I just go through yesterdays example again once more and show you where the marks and demarks are before proceeding further so I said yesterday that I have a basic block this is a basic block and corresponding to this basic block let us say we had arrived a schedule and in this schedule we said that the first statement A in the first statement we said E = A+B so it happens in time step 1 these are two time steps these are this has two time step.

In time step 1 what happens A and B are added and E is produced and in time step 2 this A and E are added the same step at occurs A and E are added and G is produced, if this cannot be done in time step 1 because E is dependent because the E is on the right hand side in statement 2 and is only defined that is on the LHS of this synthesis of statement 1, similarly on the other part and we said that to implement the schedule.

So we will do an allocation and binding so first of all we have four registers R1, R2, R3 and R4 in which I have A, B, C and D and then what happens is that what we have decided is that after time step 1 and time step 1A And B these two should be activated and then the output will go to R2 and not to R1 forR3 and R4 similarly will happen so what we will just take an example now.

(Refer Slide Time: 07:44)



So what we said is that initially we have four registers R1, R2, R3, R4 okay and we have we said that a marks will control which of the register input should go to a function unit, okay and what we have in situation like this and initially I have this a,b,c and d in this four registers and this MUX have say so this MUX 1 has select inputs s0 s1 or say 1 s01, s11 and this one similarly has this four inputs from this four registers similarly okay, and will also be similarly controlled by two selective inputs say s02 and s12, okay and the output as I said will contain a DMUX which will decide the output of the adder should go to which particular okay, so this will also have two select inputs say s03, s13 right.

So now what should be the set of control signals that we should have, if you want let us say the e output to be here and the g output to be here and first see let us say we have c and d let us say we want this c h you c and d, c and d to be here and the final output h to be here, if this is so then what should be the register transfers that I should what should be the values of these select inputs so t1 say what should be the select inputs I want to select one for MUX1 for the adder1 what do I want.

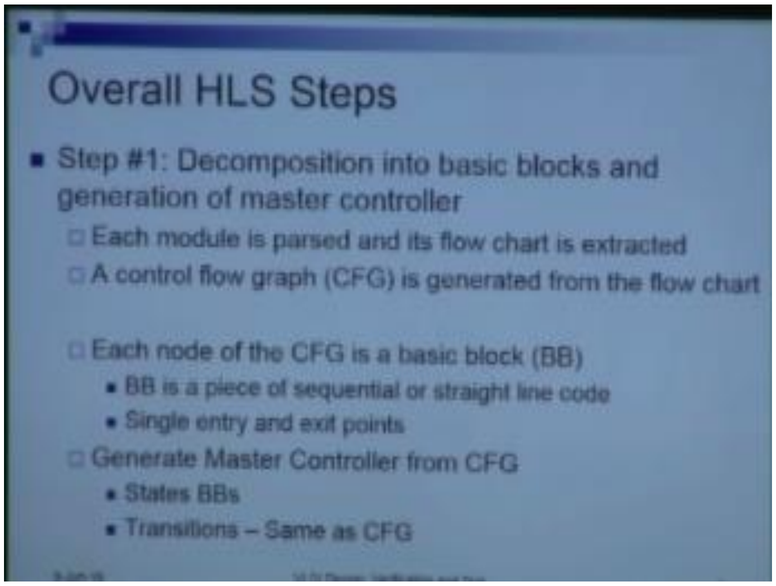
Similarly this will again I have not drawn the other part there will be an adder2 here which will again have this two inputs and two MUX and one DMUX for it that I have not shown but let us stay for adder one what will happen is 011 let us say this is i0 i1i2 i3 so r1 which we selected when marks one is for all when select inputs and 00, so r1 will be selected when selected it could a say 00.

Here I want to select for marks two I want to select r2 and that will be selected when it is 01 so s20 = 0 and s1 2 = 1 right so then what will happen in the first time step I will be able to select r1 and r2 for marks one if it is select r1 marks two will select r2 and that for times step and hence adder want if it is activated in time step one will get the correct inputs get that correct inputs a and b.

Then what will happen that time step one the output should go to r4 so therefore the deface should also select s03 = 0 and s1 3 = 1 so why because if it selects this select inputs the output of the adder will go so this particular line it goes to r2 will be selected and the output e will get store in to part 2 this has design. So similarly I will be able to and the therefore in times step one I will be able to this activate the register transfer r2 = r1 + r2 or using these control c+ right.

So similarly I will be able to get the set of the register transfers should proper control signals at by appropriately selecting the select inputs of this marker and demark I will be able to appropriately select control signal so that the appropriate register transfers start activated okay.

(Refer Slide Time: 14:48)



Overall HLS Steps

- Step #1: Decomposition into basic blocks and generation of master controller
  - Each module is parsed and its flow chart is extracted
  - A control flow graph (CFG) is generated from the flow chart
  - Each node of the CFG is a basic block (BB)
    - BB is a piece of sequential or straight line code
    - Single entry and exit points
  - Generate Master Controller from CFG
    - States BBs
    - Transitions – Same as CFG

So with this example now we go and look at the overall high level synthesis steps so with this we come to the end of module one of lecture 2.