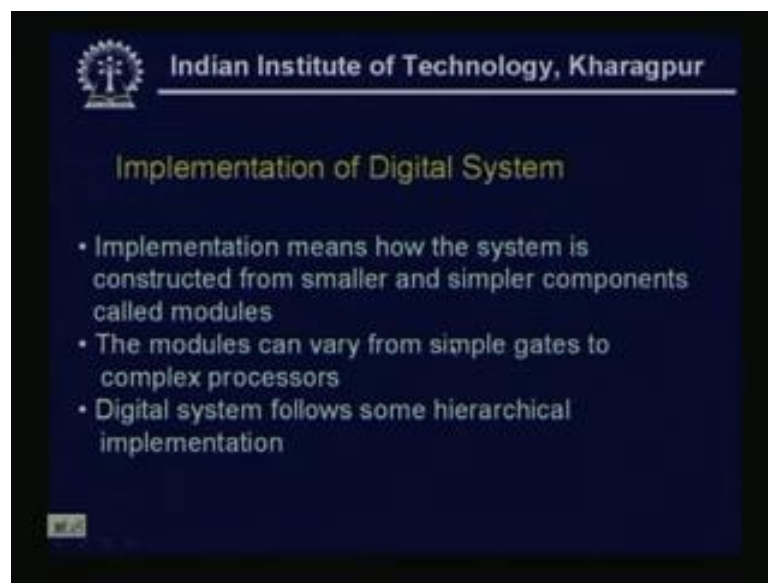


**Digital System Design**  
**Prof. D. Roychoudhury**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture - 02**  
**Introduction (Contd.)**

In the last class, the digital system design was introduced, we were discussing about the specification and the implementation of digital systems. Today, we will see what do we mean by implementation of digital system.

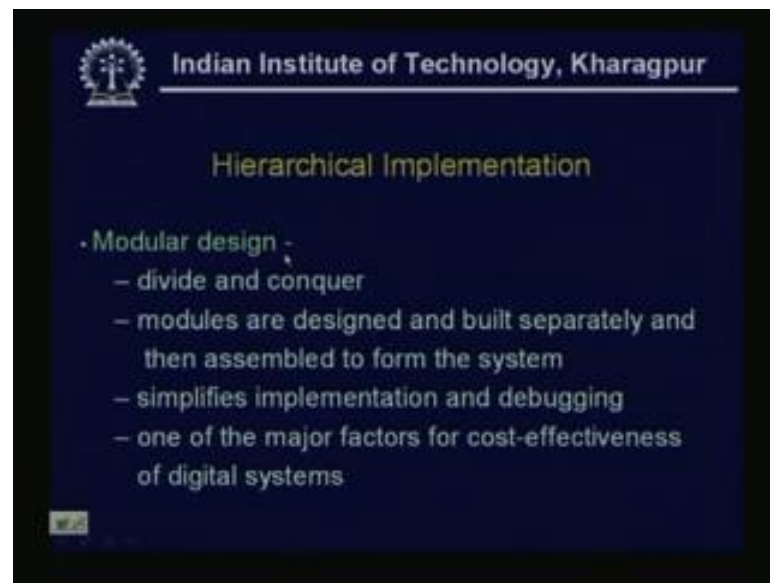
(Refer Slide Time: 01:08)



Now, implementation means, how the system is constructed from smaller and simpler components called modules, the modules can vary from simple gates to complex processes. That means, that one system has a related components, we have defined digital system like that. Now, these components again a set of components or more than one component; that is clustered together and we are calling that is as a module.

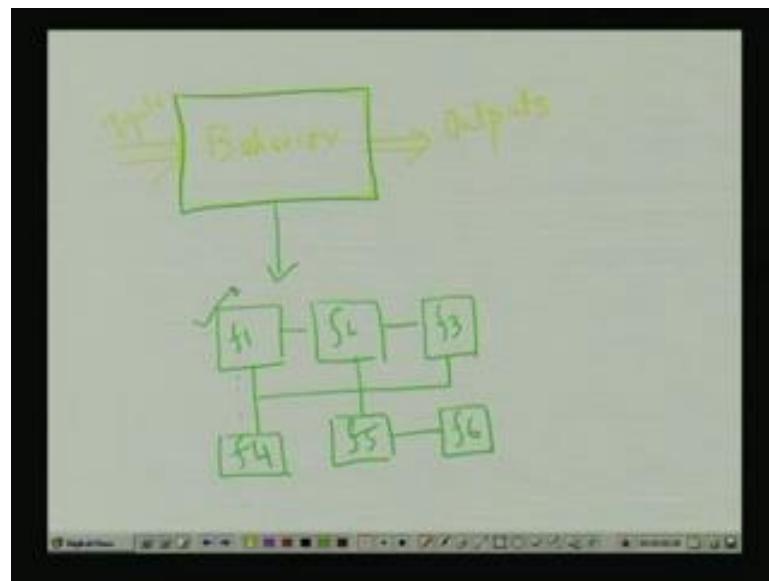
And, this module can be a set of gates a combination of simple gates or it can be a complex processor, now, digital system follow some hierarchical implementation. So, today we see that, what do was mean this hierarchy and how that one last digital system can be designed.

(Refer Slide Time: 02:17)



First, we see that, what do we mean by modular design.

(Refer Slide Time: 02:38)



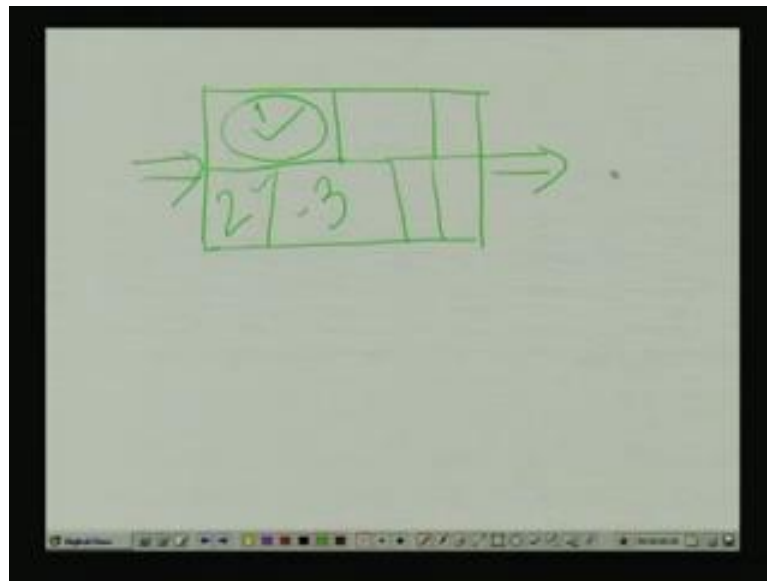
If we see, first we take one page and see that like the way we have discussed, that as if this some function or behavior of the system and some inputs are fed and outputs are taken out. Now, if this a large system, what we can do, we can break this thing as a some smaller blocks, say this behavior as if this is some smaller blocks, again it has it is each of the blocks has it is own function.

So, this is f 1, f 2 functional behavior means, it is normally, it is used and the same definition say f 5, f 6, say these are the some blocks. That means, these as if this last

system is broken into some smaller blocks, there must be some interconnection between these blocks.

So, this each smaller blocks, we are calling this is a module, so mainly this is divide and conquer means, the last system is divided and then the smaller module is designed first. Modules are designed and built separately and then assembled to form the sub system, why we are doing thing, because it simplifies the implementation and debugging.

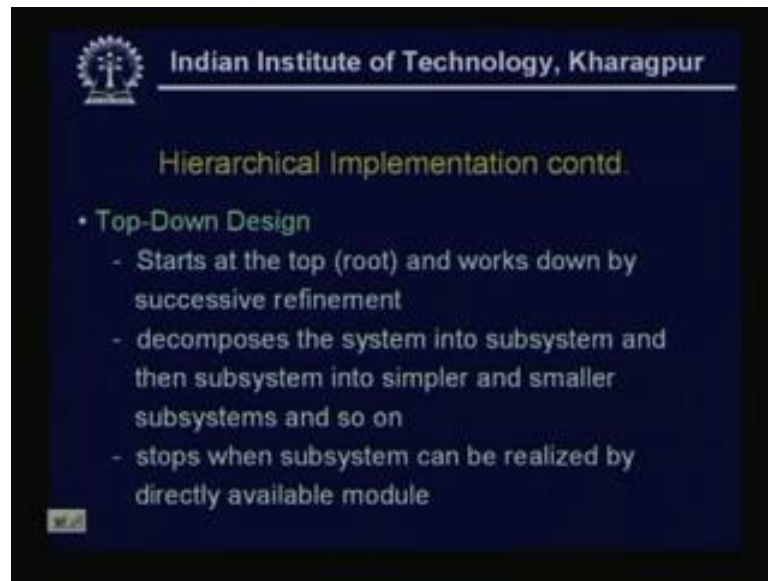
(Refer Slide Time: 05:43)



Because, always it is true that instead of designing a large system, if I partition the thing into smaller blocks, say I am partitioning in this way and then first to design this one. This module say, this is module 1 or design this module, say module 2, first it will be easier, another thing is, if there is some problem; that means, just to debug, if the full system is a is not working, if it does not work properly.

Then, first we will identify that whether that each module the smaller modules are working or not. Then it must be instead of testing the whole system or instead of debugging the whole system, the debugging of smaller modules are always much better, it is advantageous. Now, one of the major factors for the cost effectiveness of digital system that comes from this modular design.

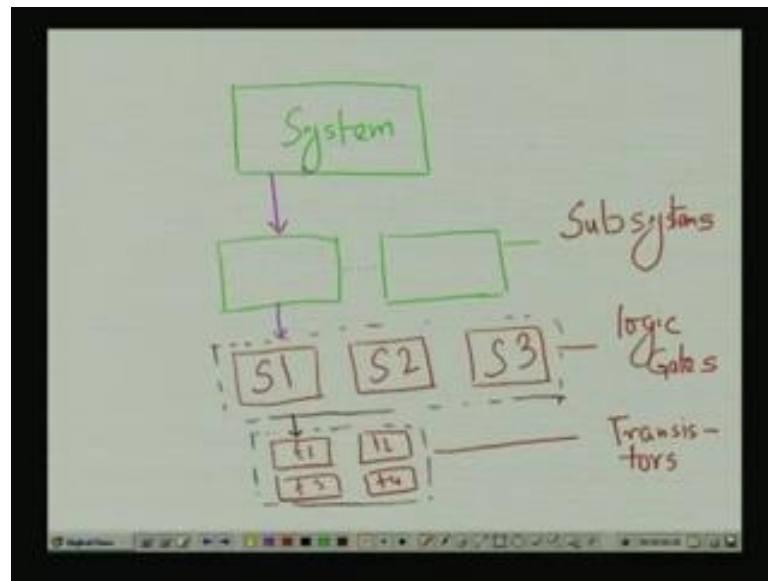
(Refer Slide Time: 07:08)



Now, we see, what do we mean by top down design approach, so it starts at the top root and works down by successive refinements and decomposes the system into subsystem and then subsystem into simpler and smaller subsystems and so on. When, it will stop because, what we are doing, one system, first we are breaking or partitioning into smaller subsystems. Then, one subsystem, again we are breaking that smaller subsystems, so when we will stop.

So, we stop when subsystem can be realized by directly available module, so these modules, what just now we have discussed that module, that as if this modules are designed and built separately and these are available to us. So, when get the subsystem as the modules, which are available then we will stop that thing, so this is top down design approach, now we see pictorially.

(Refer Slide Time: 08:37)

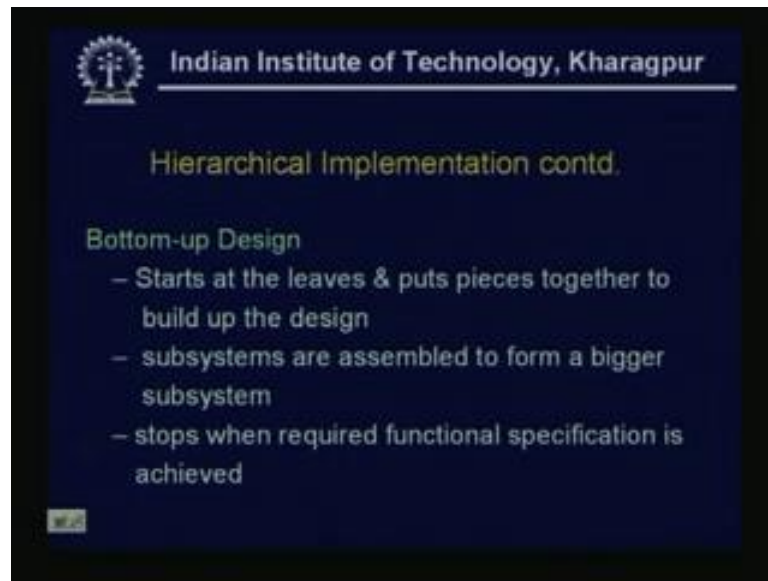


Then, say I have first 1, say some I am taking that this is the system that I want to design. Now, top down approach means, as if these are the in the second level, say as if these are some blocks, say some smaller subsystems, these are smaller subsystems. Now, say each block, so this is number of subsystems are there, now as if these modules, again partition into some smaller modules, which are actually realized by some gates logic gates, these have a logic gates.

Then, again this is a set of gates, say this is S 1, S 2, S 3 then, this S 1 is again partition into say some transistors. So, these are a combination set of say a set of transistors say t 1, t 2, t 3, t 4, so why it is called top down as if the for this system, this is first we have, it is coming from this direction, it is partitioned into subsystems. Then, say this subsystem only this one is broken as a group of smaller subsystem, which consists of a number of gates.

Now, again say it is coming from that, here we are getting that n of number of as if smaller, it is again partitioned into smaller sub blocks and these are nothing but a set of transistors set of transistors. So, we are coming from the top, the system is given and then, how we can realize the system, how we can implement that thing, so this is called the top down approach.

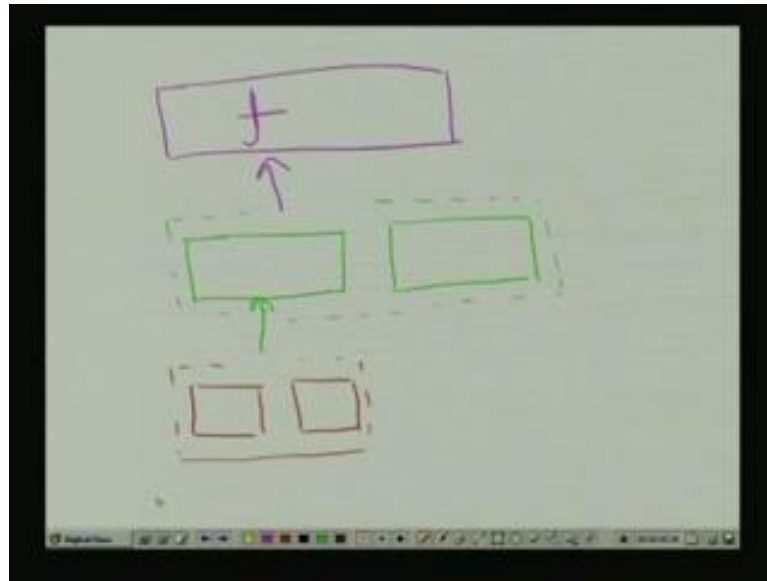
(Refer Slide Time: 12:11)



Now, another is the bottom up design, now in the top down approach, all ready we have seen that, we have started from the top, that as if that is the root, which is given that, which system I have to design. And then, I am partitioning and partitioning smaller and smaller subsystems, until I get a realizable module. Now, this is totally a reverse approach that, it starts at the leaves and puts pieces together to build up the design.

So, it starts given a subsystem given a system that I want to design that as if, we are starting from the leaf and this subsystems are assembled to form a bigger subsystems and when it stops, when required functional specification is achieved. So, here what we are doing that, totally reverse concept.

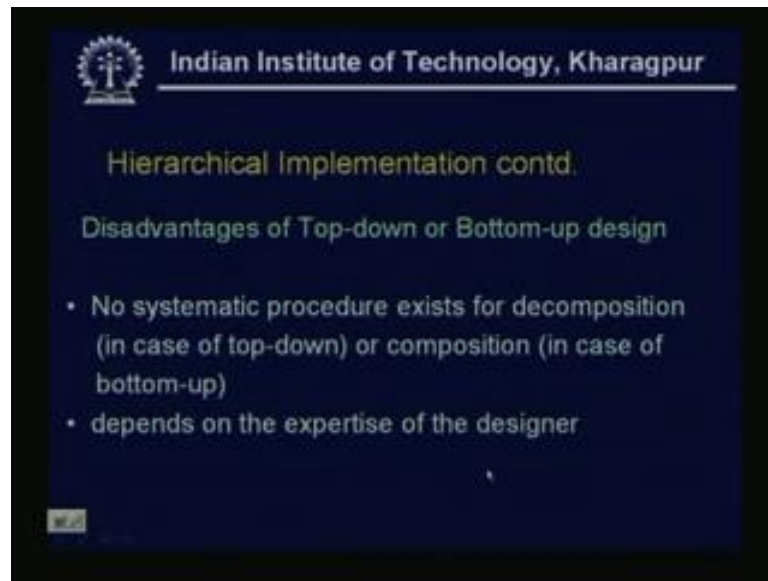
(Refer Slide Time: 13:15)



Say as if, what is available realizable modules, say that small modules, say this is some transistors or it can logic gates also, say I am taking this one, this is a set, first these are this two are assembled. Then, say we are these are assembled, I am getting a bigger subsystem and then again say from another, I have got another bigger subsystem. So, these are these are some bigger subsystem.

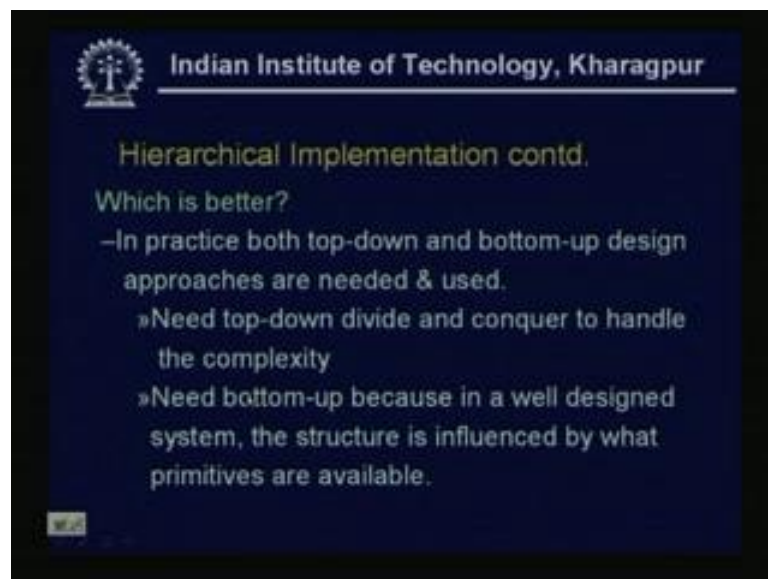
Now, again I am going up, say that I got the specification, that I want to realize, means the behavior; that I want to implement, that my function  $f$ , so I stop here. So, actually here, we are approaching from the bottom and stops, when the behavior of the function is given.

(Refer Slide Time: 14:46)



Now, what is the disadvantages of top down or bottom up design, so there is no systematic procedure exist for decomposition, see in case of top down approach, actually we are decomposing. And in case of bottom up, we are assembling the smaller modules, so when we will stop, actually there is no systematic procedure for this, it totally depends on the expertise of the designer. So, this is a very big disadvantage of top down or bottom up design.

(Refer Slide Time: 15:27)



Now, which is better, because all are modular design, top down or bottom up, one we are approaching from top, one we are approaching from the bottom. So, in practice, that

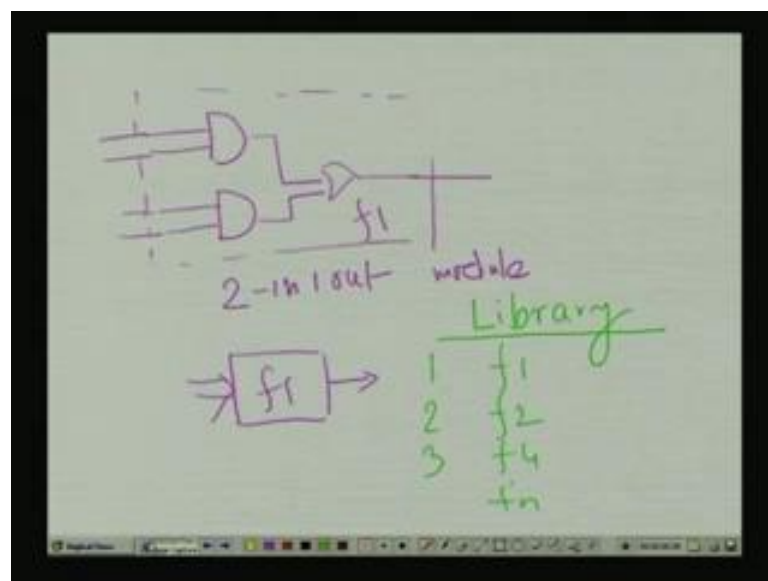


actually both top down and the bottom up design are used they are needed, so top down is normally used to divide and conquer to handle the complexity. Say, if it is a bigger system, all ready I mentioned that to debug, debugging is a big problem.

That for a large system, if it does not work correctly, then where is the fault, just to detect that thing, to identify the fault is a big problem. Rather, if we partition the full system and we take separately the each smaller module and we check separately, then it will be easier to identify, that where the fault is. So, normally for and another thing is that, that to design the smaller modules, will be much more efficient and obviously, complexity will be much lesser, if we handle only one smaller modules.

And the bottom up is needed, because in a well designed system, the structure is influenced by what primitives are available. This is very important, because now a day, what we are calling that available realizable module, so now a days this is called the library, one available library is there. What do we mean by library? So, library means that all ready some smaller modules, say that are actually designed and realized.

(Refer Slide Time: 17:26)



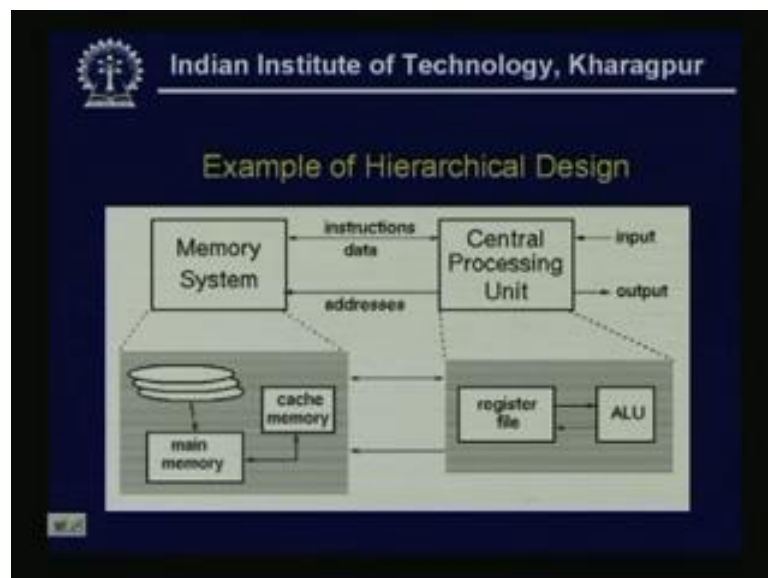
Say, if we take an example, say I have, I am taking two, two input and get, simple example, I am thinking, two, two input and get and one, two input or get. Now, what we see that, we can tell that, this is one module, normally often used, it is very much used in every design. So, this is my 4 inputs of the module, one smaller modules and one output, so two input, two in and one out module, this is one module.

Now, instead of designing every time the smaller modules, what we can do, we can design and test and we make, this is as a one module which has two input and one input. And this is the function, say  $f_1$ ; this I am telling this is one subsystem one module, now once, it is designed, we are telling that, this  $f_1$  is a realizable, available module. Now, say this, I have some library is there, now library means, all ready pre-designed, pre-tested modules are there.

Say 1 is that,  $f_1$  there and its specification, all ready we have defined the specification of one system or subsystem, say some  $f_2$  is available, similarly some  $f_4$  and up to  $f_n$ , same number of. Now, what we will be doing that, if now I want to design a system, then these are all my library is with me; that means, all this  $f_1$  to  $f_n$ , this  $n$  number of modules are available, these are pre-designed and pre-tested modules.

So, what I will be doing, that instead of designing again, designing from the scratch. I will just take from this thing from my library and I will put that thing as a block, so that is why it is very much helpful. So, that is why bottom up means, that as if we are taking, what is available, that library is available, what modules are all ready at there, we are just selecting some modules and then we are assembly together to get a larger modules. So, that is why that, both top down and bottom up designs are important.

(Refer Slide Time: 20:32)

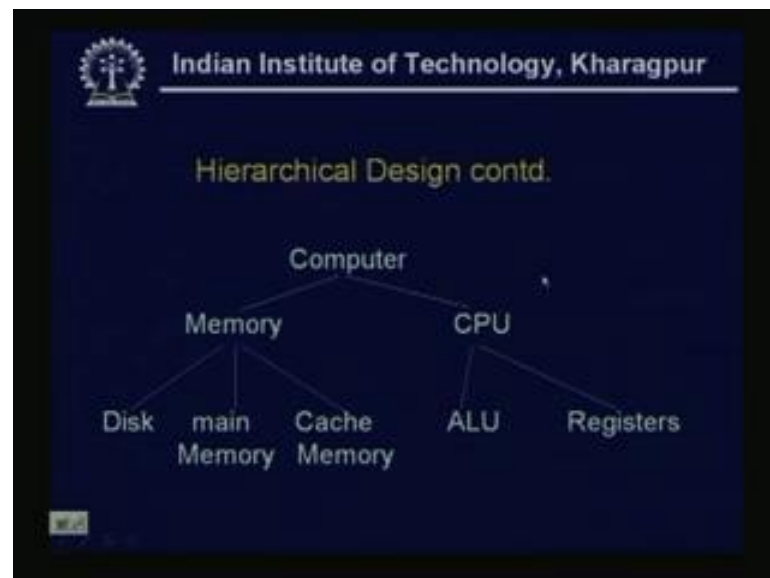


Now, we take a example of hierarchical design, this is a very common example of our computer system, digital computers. So, it has all ready we have seen, the last the same picture, it is our memory and this CPU and that instructions and data, that are passing

between memory and CPU addresses are address lines are there. And see this memory system, actually these we can take that, as if this is a larger subsystem, because this full thing is a system, my digital computer is a system.

So, it has now I am considering as if two subsystems memory systems and CPU system, now again this memory system, this is a subsystem, this is partition, this can be viewed as if this is a combination of two other smaller subsystem, that is called the main memory and the cache memory. Disks are can be a third party, we will see later, similarly the CPU this is a larger subsystem and again, if we partition this thing, that some smaller subsystems. These are register file and ALU, the arithmetic logic unit, so this is a very good example of the hierarchical design.

(Refer Slide Time: 22:13)



So, now we see, if was think that, this is as a tree and computer is the root computer is a root means that, I want to design a digital computer. So, this is my system, so first it is partitioned, it is partitioned into two, this is my memory and this is my CPU. Now, this memory is a larger subsystem, so it is partitioned into smaller subsystems, this is disk main memory, cache memory. Similarly, the CPU I can think is a combination of two smaller subsystems, the arithmetic logic unit and the resistors.

(Refer Slide Time: 23:06)



So, we have some idea, so in this class, what we will see that or what we will read, that how we can design this smaller subsystem. And then, we will assemble or the reversing also thing also top down approach given a larger the system, how we can partition into smaller one, how it can be designed, so we will learn in this class. Now, another thing that is very important nowadays is the digital design process and what is the current trend.

So, all of we know that now the dimension, mainly the size of the circuits are changed enormously and due to the advent of VLSI, that very large scale integration circuits. Because; that means, that within one smaller area, even the millions of logic gates or you can think that millions of circuitry, I can put. So, one thing is very clear that, it is almost impossible to design it manually and that is why, now the automation or the computer automation is needed.

So, for that purposes that this some computer aided design tools are introduced in the process due to its increased size and complexity. So, instead of doing manually if we use this CAD tools and for the different levels, because if it is a that lower level, just now what I mentioned that, it is a say transistor level or in the upper level, it is a gate level or even in the upper level, it is a block level. Means, some of the gates are clustered to form a module.

Then, in every level, so what will be the flow well, if it is a top down or if it is a bottom up, what will be the flow that, I want to discuss. So, designers want to standardize the

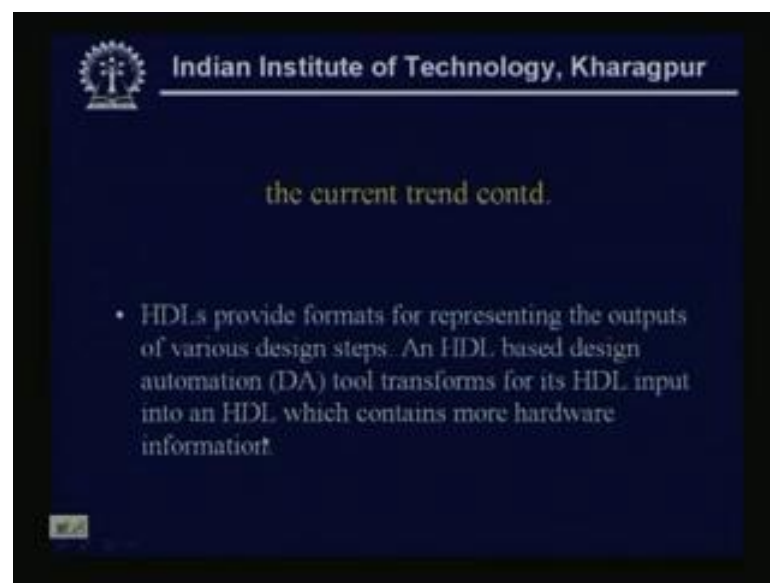
design procedure, starting from the design idea, to get the design implemented thus so called design flow. Now, new CAD tools are based on hardware description language to improve the process.

Now, what do we mean by this hardware description language, because so far we know that programming language, that we call that software languages, say our C, Pascal, C plus plus, Java. Now, see here I want to design a circuit a digital computer, say my components are wire that transistors or if it is a some of the transistors, I assemble it will be a gate 1 and gate 1 and gate.

So, how we can specify, that as it is a tool, that computer tool means, say it is software, so I have to tell that tool or the software, that this is the specification of the circuit or this if we consider the system. Then, you can tell that, this is a specification of the system, means this behavior, I want to realize, so for that purposes, that some hardware description language; that means, that as, it is name implies, this is the language.

That will tell us, this is a block and get this represents the flip flop, this is a wire, like that, this is clock. So, this is the same language HDL languages are proposed, normally VHDL, verilog, these are the different type of a languages we use.

(Refer Slide Time: 27:46)



Now, this HDL's provide formats for representing the outputs of various design steps, An HDL base design automata tool transforms for it is HDL input into an HDL output, which contains more hardware information. So, just now what we told that, there are

some a design flow, we mark this line, this very important that, what will be the design flow.

(Refer Slide Time: 28:43)



Now, see if we start, that from the design idea, means the problem that I want to solve, so these I am telling, this is my design, idea and then this is called some behavioral design. Behavioral design means, so first thing what problem, I want to solve or what system, I want to design. So, the behavior of the design that should be clearly told in this step, the first step should be that thing.

That, we are defining as a behavioral design, what will be the output, then the outputs are flow graph or pseudo code like that, then, this is a form that behavioral stage, then some smaller blocks, some blocks we are partitioning, this is called the data part design. And what is output, these are some bus and register structure, later I am giving example of this thing and then it is a logic design. So, this is again modular, that again I am partitioning into smaller blocks.

And these are these blocks are now synthesized or realized by logic gates AND, OR NOT, etcetera. So, now puts are gate wire list or normally this is called the net list, this is called the net list and now these gates, I want that actually, this is a again, it is partition because one gate, I can always realize by some transistors. So, this is we called the physical design, so the output is transistor list; that means, a set of transistors assembling together form a gate.

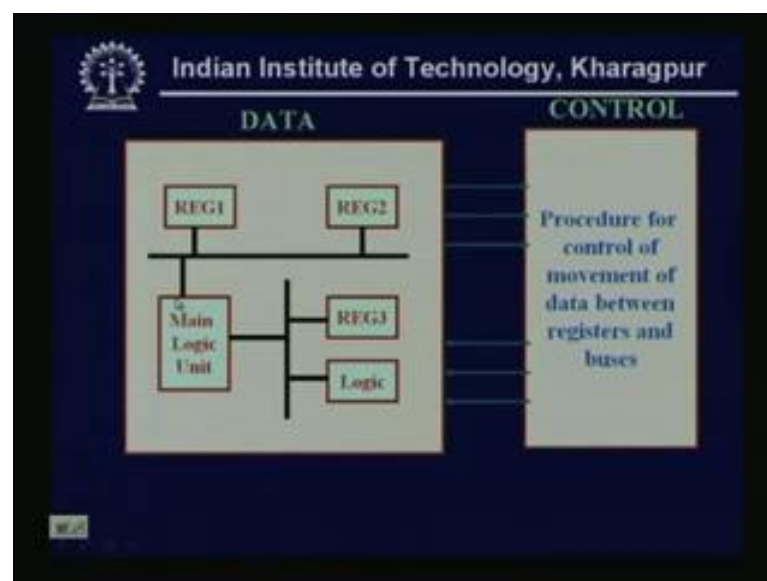
And this, we call that layout means, how they are placed, the set of transistors, how they are placed, that within the specific area, say that my chip, what normally we have seen that IC chip. So, that this is it is called that layout how it is placed, how it is partitioned, what will be the size, this is called the layout. Now, it is it can be manufactured, so this is some manufacturing stage and ultimately, we get our IC chip or board.

So, these we can tell, that this is our digital design flow, that means if we summarize, that starting from the design idea, design idea means that, what system I want to design, say our digital computer or a digital calculator, that can be your design. So, first we will get the behavioral design, what will the function, then data for design, that it will be some block diagram design, specifies each block specifies some bus resistors structures.

Then, it is logic design means that, the same thing is realized by a number of gates and wires. Then, again it is simplified or it is realized by transistors, because ultimately the transistors, I had to on silicon, that IC chip that is nothing but see piece of silicon. So, they are actually the transistors will be fabricated, so then, it will be manufactured and we get the digital IC, so this a digital design flow.

Now, just now, that in the design flow, say that data path design, that once the behavioral design is unambiguously written or defined, then the using some bus and register structures we can take the data path design.

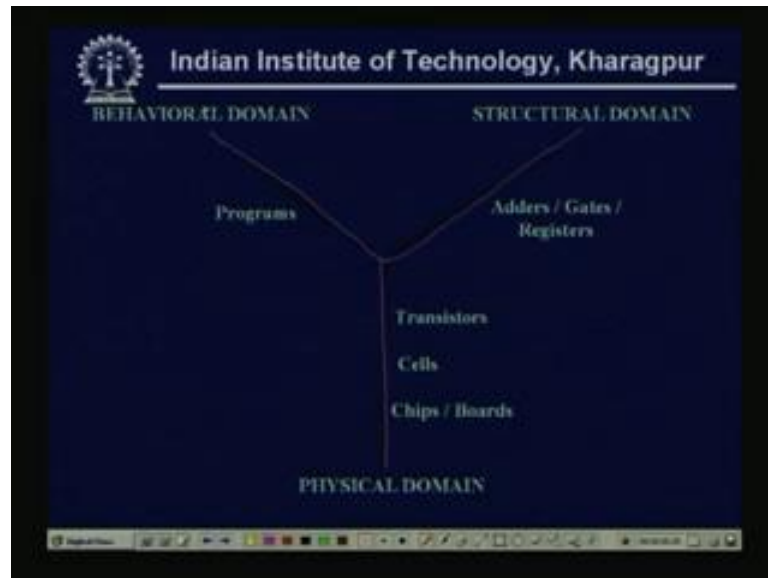
(Refer Slide Time: 33:20)



So, see this is my data and these are nothing but register; now just we can think as if these are some storage. So, there are several storages are there and there are some

interconnections, this is my main logic unit and these are connected with the procedure for control of movement of data between registers and buses means. These are my control unit, so this is the data path of design or what we call that that bus and register structure.

(Refer Slide Time: 34:33)

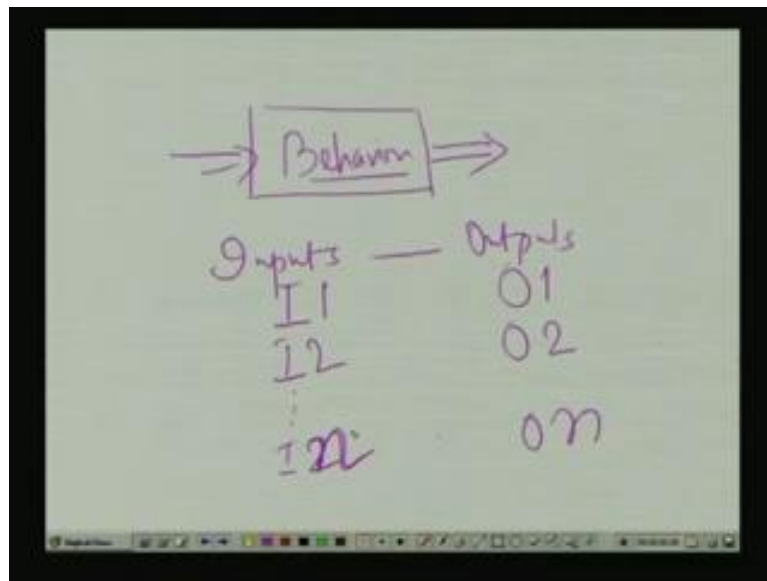


Now, another very important thing, that we can think, that is the Y diagram, see these we are that, as if that, all ready we have a discussed that, the modular design approach, main the top down or the bottom up approach. Now and parallely, when I introduced the design flow, I told that the different type of definition, that behavioral or whether it is structural, whether it is in the physical means that, when it is realized by gate or transistors.

So, that different approaches and to get or to achieve the goal, to get the design to be implemented that one Y diagram is normally defined; say this is my Y diagram. So, here this is a behavioral domain; that means, the how many ways, I can solve a problem.



(Refer Slide Time: 36:11)



So, behavioral that that one says behavioral domain means, all ready the way, the system I have defined this is behavior, that behavior means, that if these are my inputs, then this is the output. That means, if  $I_1$  my input,  $O_1$  is my output, if  $I_2$  is the input, then  $O_1$ ,  $O_2$  is the output. So, in this way that for a specific inputs that, what should be the, this should be  $I_n, O_n$ .

So, what should be input output specification that clearly defines the behavioral domain, if it is given then, what we can do, that I can write that programs to get this behavior, so I can get these things, again that Y diagram, I am telling. So, now once it is given; that means, that the program, then it comes to our that physical domain, it comes to this side, if these arm, there are three arms and this arm is that, that it is realized by transistors.

Then, cells and ultimately it comes to that digital chip or the board this we are calling the physical domain, another is a structural domain, means the arm; that we are representing. We call that given a system to be implemented, always that it is a similarly of the top down approach. That, I can partition the this thing into some smaller blocks or smaller modules and this can be, that some say adders, gates, registers, so these arm, this is a structure.

So, if the structural domain, then from the adders, gates, we come to this point and obvious, we know that adder, whether is adder, gate, register, decoder, any other functions, always that we can implement by a or we can realize by a set of transistors.

So, this is a very well known wide, we call that Y picture, or Y structure, how we can use or how we can implement the digital design, how we can approach.

(Refer Slide Time: 39:24)



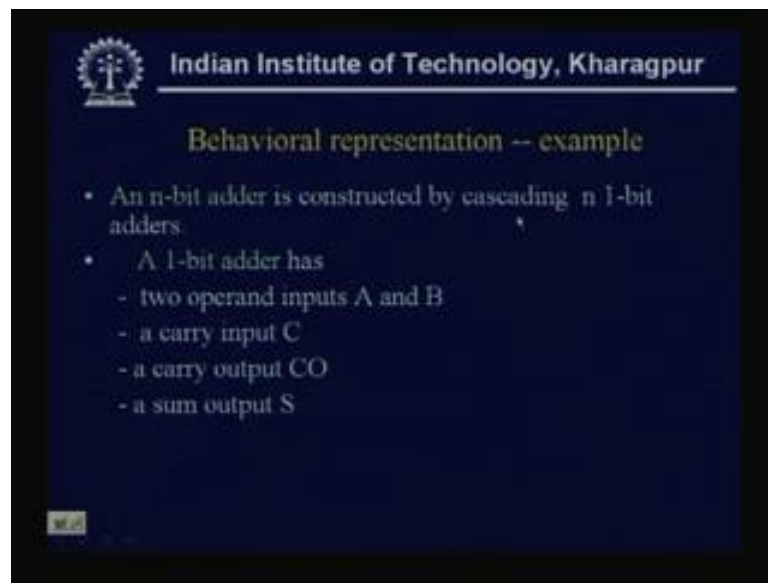
So, behavioral representation, now how behavioral is representation describes, how a particular design should respond to a given set of inputs. So, mainly we are discussing about the CAD tools, because nowadays that just for it is size, it is a huge number of circuits, so how we can represent this behavior of a systems. So, behavior maybe specified by Boolean equations, we will see later, we will read these Boolean equations.

Then, tables of inputs and output values, just now I have shown, always the behavior can be represented that, if this is the input, this will be output. If I 1 is an input, O 1 is the output, I 2 is input, O 2 is the output. So, this set of input and outputs clearly defines the behavior.

Another way, I can tell that algorithms written in standard high level computer language or in special hardware description language or that one behavior easily I can describe by computer language, which is nothing but some English type language. We know that software language, the C language is English type see, using this computer language, always I can do that thing.

Now, here one thing is very special, because here it is not software, I have to specify my circuit, the transistor, the gate, the decoder, the adder and multiplier, so for that I need hardware description language or we call the HDL instead of the software language. So, in this way, that I can feed my behavioral representation to the CAD tool.

(Refer Slide Time: 41:21)



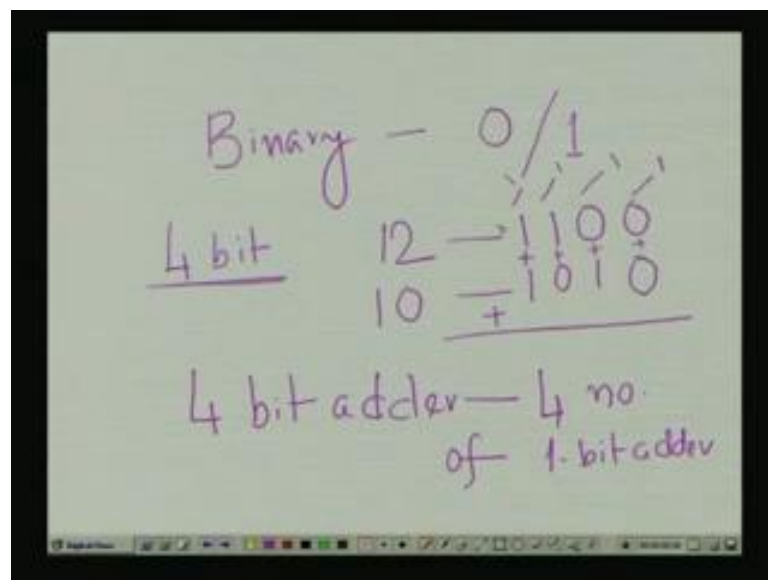
Indian Institute of Technology, Kharagpur

Behavioral representation -- example

- An n-bit adder is constructed by cascading n 1-bit adders.
- A 1-bit adder has
  - two operand inputs A and B
  - a carry input C
  - a carry output CO
  - a sum output S

Now, I take example, then it will be clear, what do we mean by behavioral representation, say an n bit adder is constructed by cascading n 1 bit adders, what is a adder.

(Refer Slide Time: 41:45)



Binary - 0/1

4 bit

$$\begin{array}{r} 12 \rightarrow 1100 \\ 10 \rightarrow 1010 \\ \hline \end{array}$$

4 bit adder - 4 no. of 1-bit adder

So, just I want to add nothing but say n bit means, yesterday we have seen that binary only two digit, we will use 0 or 1. Now, say 4 bit, some 4 bit numbers, I am taking, our decimal 12 means, it is actually 1 1 0 0 and say our decimal 10 is 1 0 1 0. So, this is the 2 4 bit numbers, I want to add, this is I am calling that 2, 4 bit adder. So, what I need, I need 4, 1 bit adder, this is 1, bit 1, this is 1 bit, this is 1 bit and this is 1 bit.

So, I have add this 1 bit, I have to add this 2, 1 bit, I have to add this to 1 bit, I have to add this 2 1 bit, that means at 4 bit adder. Now, what I can tell, 1, 4 bit adder can be realized, because it is hierarchical, so 4 number of 1 bit adder, this is as a whole 2, 4 bit numbers. I want to add this 1, 4 bit number, this is 1, 4 bit number and for that what I want actually 1, 2, 3, 4 number of 1 bit adder.

So, an n bit adder is constructed, say if n equal to 4, I have given the example. So, for an n bit adder is constructed by cascading in cascading n 1 bit adders. Now, A 1 bit adder has two operand inputs A and B, a carry C, a carry output CO and a sum output S, what do we mean.

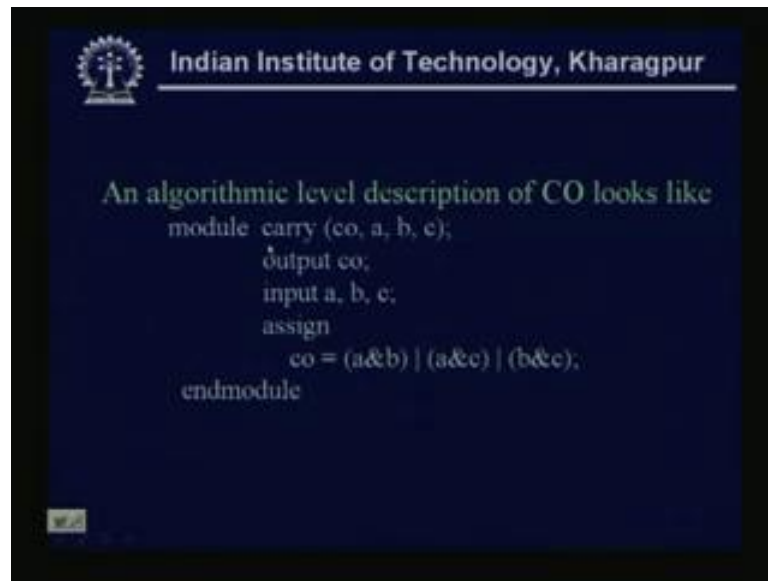
(Refer Slide Time: 44:35)

The diagram illustrates a 1-bit adder. On the left, it shows the addition of two 1-bit numbers: 0 and 1. The result of the addition is 1, which is labeled as the 'Sum'. Below the sum, it shows 0, which is labeled as the 'Carry Output'. On the right, it shows the addition of two 1-bit numbers: 1 and 1. The result of the addition is 1, which is labeled as the 'Sum'. Below the sum, it shows 1, which is labeled as the 'Carry Output'. The diagram also includes a 'Carry' input of 0 at the top right.

Because, see now 1 bit adder, I am considering 1 bit; that means, I want to add 0 and say 1 this is 1 bit 2, 1 bit. Now, what I want, so these are the two operands, now if we add this thing, say I am I am getting 0 plus 1, say my sum is 1, currently we assume and that say carry is 0. So, this is my sum, what is my carry, I am telling carry output is 0, why carry output is needed.

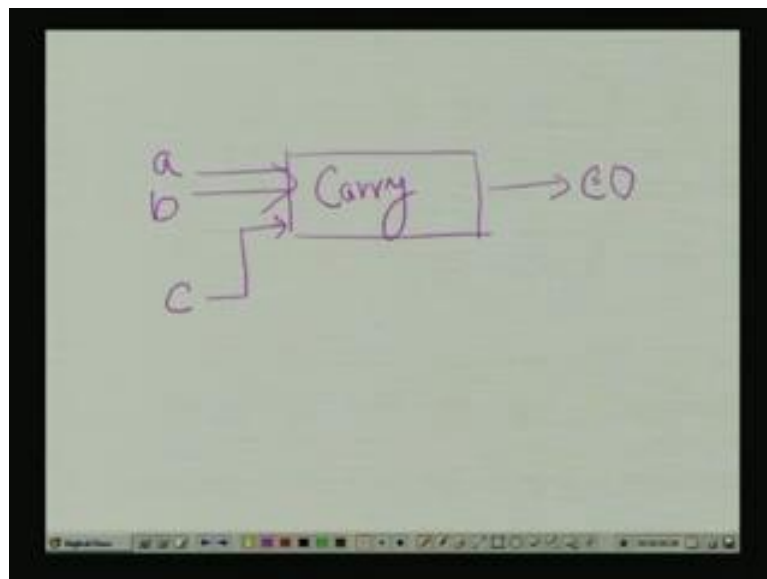
Because, if it is 1 bit, if it is for 2 bit, then I know that, I have to add this thing and the carry will be added with the next bit, so that carry output will be propagated, so I need carry output. Now, when we are considering the sum of these two bits, see here it is two operands, other operands are actually there and that is my carry input. So, this is illustrated in here, the two operand inputs A B a carry input C a carry output CO, a sum output S.

(Refer Slide Time: 46:07)



So, the same thing, that if algorithm level description say algorithmically, we are telling stepwise, actually I am defining a module, say a module and this a carry module. This is a carry output; that means, as if one subsystem I am defining and these are the three inputs, why three inputs, that input ABC, AB are the operands and C is the carry in, so these are the three inputs. And then, that this assigns some musings, this is my behavior; that function, how I can generate the carry output from here.

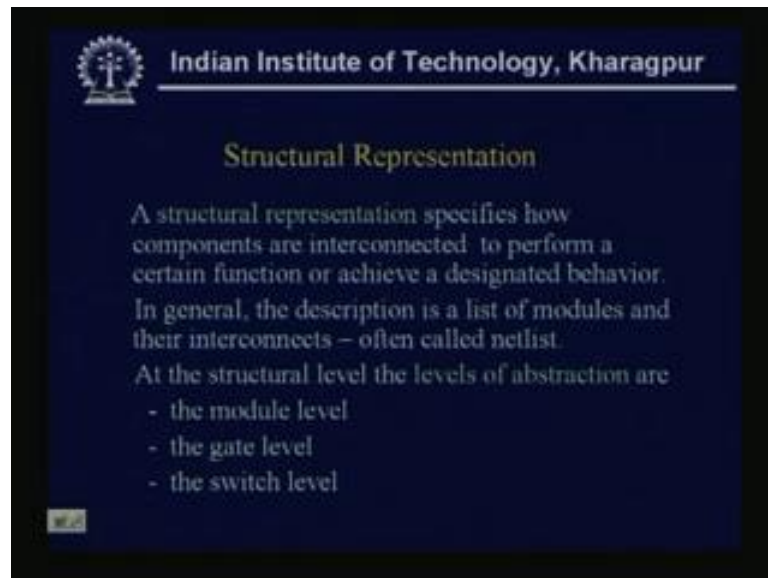
(Refer Slide Time: 46:59)



So, if I see only the carry output; that means, the way we have designed the digital system, that this is my carry module or you can tell that, this is my subsystem that carry

module, it has three inputs. This are my a and b, that I want to add that from the previous one, another c is coming, this is my carry in and only one input that carry out, so this is inputs, outputs behavior, so this is a system.

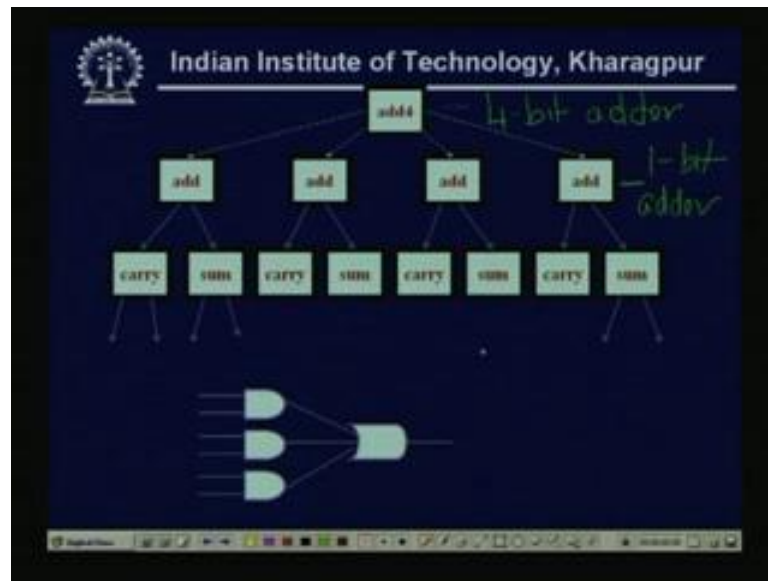
(Refer Slide Time: 47:46)



Now, that is a structural representation, so structural representation specifies, how components are interconnected to perform a certain function or achieve a designated behavior. In general, the description is a list of modules and their interconnects often called net list; that is if a number of modules and there are some interconnections between them specific interconnections that together form another larger module and that gives a structural representation of the system.

So, the structural level; the levels of obstructions are the module level, the gate level and he switch level. The module level, all ready we discussed; then it is module can be realized by realized by gates, gates can be realized by transistors, actually in between, I can write other transistors. Transistors can be mainly, nothing but actually switch and transistor are same the transistors are nothing but switch that on or off.

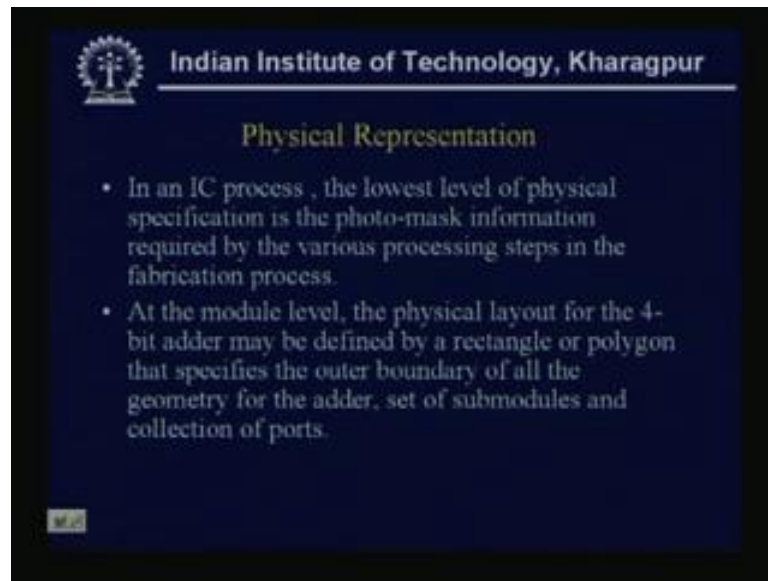
(Refer Slide Time: 48:53)



Now, pictorially if we see the same example, so this is my 4 bit adder, now it is partitioned into 4, 1 bit adder, so these are all this level, this is all 1 bit adders. Now, 1 bit adders, what will the components, so there will be some, I have to realize a carry, I have to realize a sum, so 1 carry output and 1 sum. So, for each one, it is same that carry and sum, so in this way, I can say a 4 bit adder, I can hierarchically, I can implement.

And another thing just notice, see here the all carry and sums that modules are same similar structure, so these what we mention that, this is the duplicate of modules and in digital design mainly, this is the advantage. That the same type of one large system can be realized by a similar modules and it is very important or it is very advantageous to fabricate, the similar modules or in a smaller area, that what we call that VLSI or simple IC's also.

(Refer Slide Time: 51:13)



Now, the last step that physical representation, means that in an IC process, the lowest level of physical specification is the photo-mask information, means that physical representation means, that the full system, it is partition into blocks, then it is realized by gate, then it is realized by transistors. And now that we are using the CAD tools, means all the things are automated, this is a digital automation.

So, just to a digital IC, that in the fablab, we have to send the photo-mask information, means that which transistors will be fabricated on the silicon and how this is nothing but a photo-mask information. So, this is required by the various processing steps in the fabrication process, now at the module level, the physical layout for the 4 bit adder. The example, I have given maybe defined by a rectangle or polygon; that specifies the outer boundary of all the geometry for the adder.

Because, actually if I want to design a 4 bit adder, that is nothing but a when you will see, this is a piece of silicon, say a rectangle silicon piece. Then, now 4 bit adder means that a set of transistors. So, in the 4 piece of silicon, that how the transistors are fabricated and where what will be placing, mainly that information is the photo-mask information and this is the set of sub modules and collection of ports, mainly that information we have to sent to the fab.

So, now the main thing, that if I want to summarize, that main thing what is a there, that given a digital system or if or my aim is to implement a digital design, that starting from





So, these we are calling our physical design, so this is my physical design up to this, this is physical design and this is my logical design. So, this is the overall introduction of our digital system design course, so next time, we will start the course.

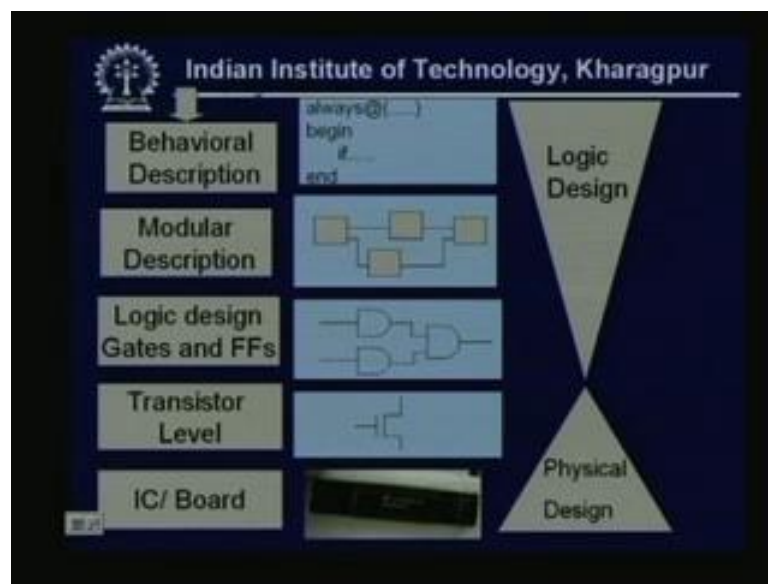
Thank you.

**Digital System Design**  
**Prof. D. Roychoudhury**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 03**  
**Digital Logic-1**

Today, we will start discussing on digital logic part one, but before that, I want to revisit a quick summary of the previous two lectures.

(Refer Slide Time: 58:32)



Mainly, the previous two lectures, I give a introduction of digital system design, what do we mean by the digital design. So, starting from the design idea that, if I want the digital IC's, so first it should be the behavioral description, means that given the problem, how I describe the behavior of my problem. Normally, nowadays the CAD tools are used for that purposes.

Thank you.