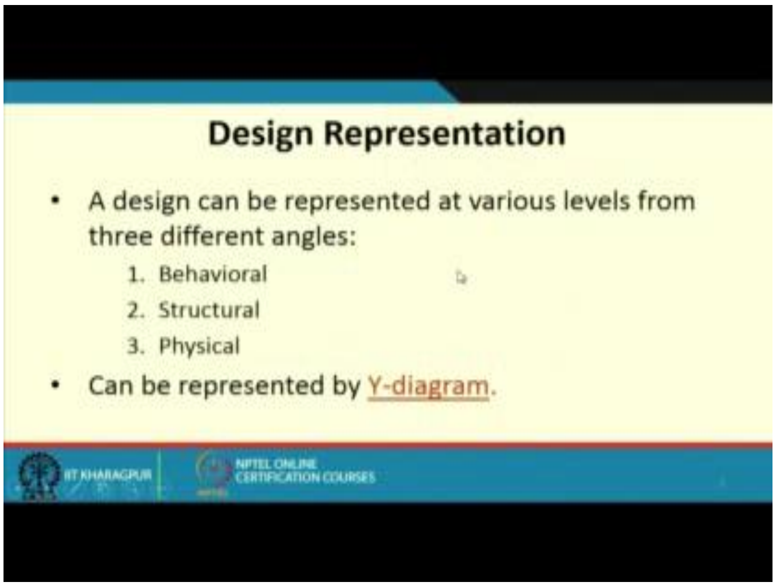
VLSI Physical Design

Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 02
Design Representation

So, let us continue with a discussion. So, in this second lecture the module we shall be talking about various design representations. So, the topic of this lecture is design representation.

(Refer Slide Time: 00:44)



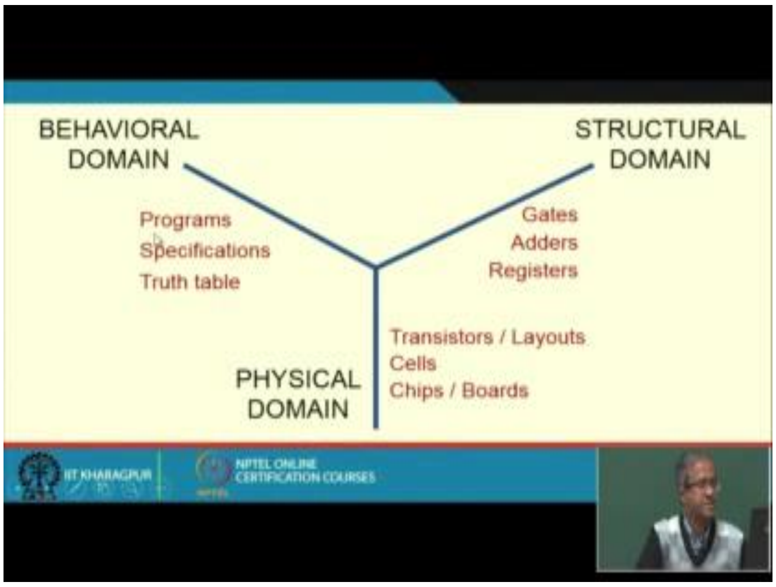
Design Representation

- A design can be represented at various levels from three different angles:
  1. Behavioral
  2. Structural
  3. Physical
- Can be represented by Y-diagram.

So, actually what we are talking about here is that design we have a design, this design can be at any level it can be at a very high level design, it can be an intermediate level design, it can be a very low level detail design, but what we want to says that design at any level can be visualized or viewed from 3 possible angles.

So, we say that the 3 angles are behavioral, structural and physical; behavioral means what the design is suppose to do; structural is how it is the implemented in terms of the circuitry or the net list; and physical is the actual implementation whether is the implemented as a chip, or as a module, or as a cell and so on and because of the 3 angles it was proposed that it can be represented by a so called Y-diagram. So, let us see I am a how this Y-diagram looks like, and how it can represent the various aspects of the design from this 3 different angles.

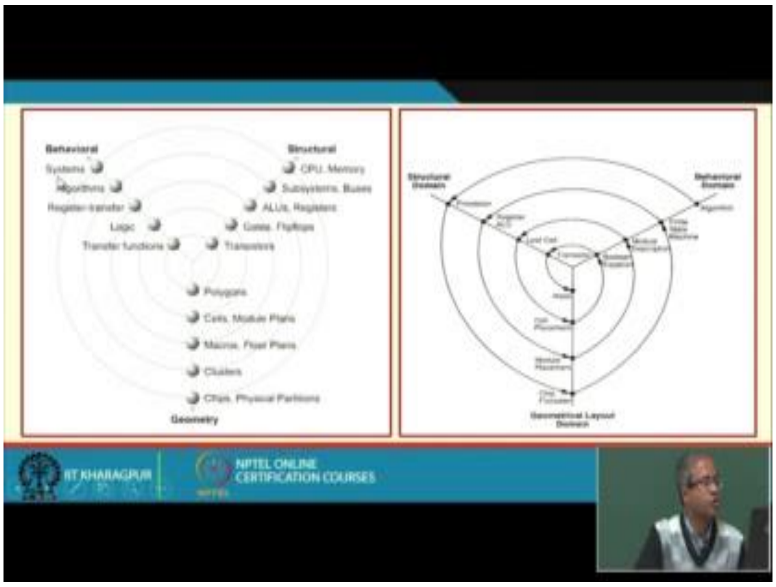
(Refer Slide Time: 02:02)



This is how a Y-diagram looks like where is you can see the 3 arms of the Y corresponds to the 3 angles of visualization behavioral, structural, physical.

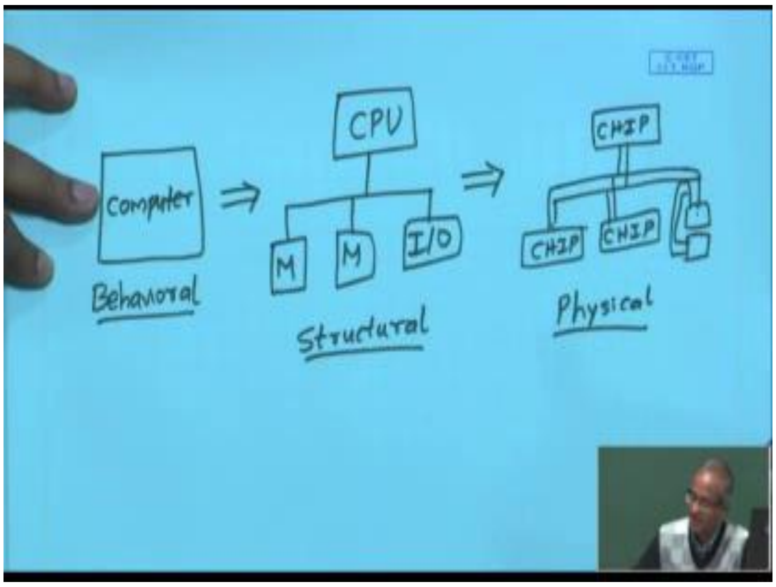
So, when you talk about behavioral, we basically talking about how circuit is suppose to behave. We are not going into detail as to how you are implementing it or how you are designing it. So, some typical ways to represent behavior are programs well as in high level description languages like very lower PHDL. Specifications in various form for example, Boolean equations can be specifications, truth tables, finite state machines and etcetera these are ways to specify behavioral. Structural domain says some kind of net list inter connections of registers, adders, gates and so on maybe higher level modules processor memory; and in the physical domain we are talking about chips or boards the basic cells transistors and so on.

(Refer Slide Time: 03:30)



So, any design can be viewed from any one of the 3 angles. Let us look at these 2 diagrams because these 2 diagrams actually show how we can interpret the Y-diagram. Just you look at this first diagram on the left, here as you can see we have drawn this imaginary circles these are concentric circuits, which are touching one of the points along each of the 3 arms of the Y. For example, you can see in the behavioral level we have a note called systems, the equivalent node in this structural level says CPU memory and equivalent one at the geometrical level says chips or physical partitions. So, what does this represent, I am just trying to show you.

(Refer Slide Time: 04:45)



Say I have computer that I want to design, this computer can represent my behavioral few. So, I specify how the computer should behave. Now the from another angle this computer can be visualized as comprising of a central processing unit CPU, it can consist of one or more memory modules, and may be an I O module and I O processor. So, from another angle the same computer can be viewed as an inter connection of very high level functional blocks like CPU, memory and I O this can be your structural view, but this same design when it comes down to physical view, it can be one chip representing the processor, there can be another chip representing the first memory, chip representing the second memory and may be 2 chips representing the I O, so you will be having some kind of inter connections between them something like this. So, this will be your physical.

So, what I mean to say is that the design at this same level of abstraction can be viewed from 3 different angles; One as the behavior, other as a net list, and the other as some components which are actually manufactured and the system is implemented by inter connecting them like chips in this case right. Similarly you can have some other examples or other level for example, if I give you some Boolean equations say f equal to a into b plus c, this represents of behavior, now I can represent this in terms of some gates that would be my structural behavior and this gates can be finally, implemented using either NAND or NOR or some kind of CMOS cells, that will be the physical representation, so we can have this view from any level of abstraction.

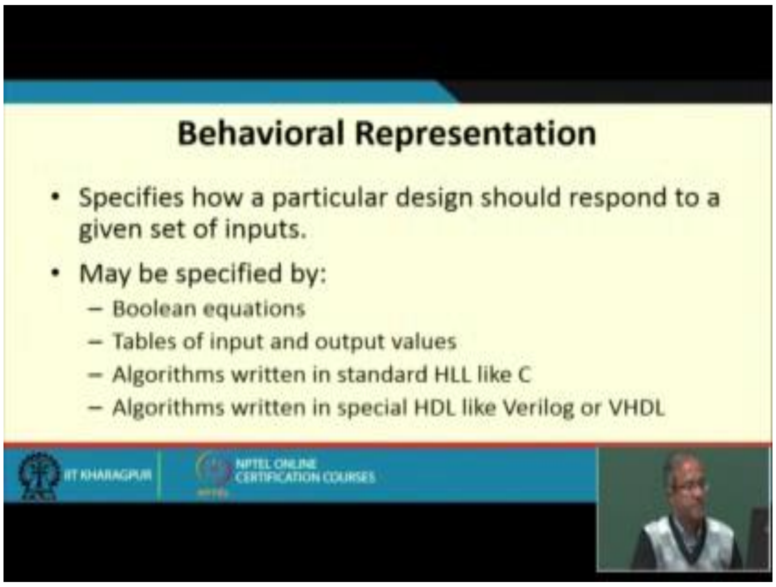
So, in this diagram if you come back to it; so as you move towards this center you are going into more and more detail for example, here you talk about logic here and in structurally you talk about a loose and registers, and here you talk about macros and floor plans. If you go down you talk about transfer functions, gates and flip flops, cells and module plans. So, as you move down towards the center, you are refining or design and making it more and more detailed; this is one way of interpreting the Y-diagram. Now this Y-diagram can also be expressed or interpreted in a slightly different way as the diagram on the right shows.

See here the arm here shows behavior, this is structural and this is physical or geometric domain. Now you see instead of circle we have drawn some kind of a helix which starts form here and it moves down towards the center. This actually represents top down design, how? It says that is start with a behavior lets and algorithm, this algorithm specification you first synthesis or decompose into structural design which consist some inter connection of as I said some processor, CPUs, memories, I O etcetera. Now this net list would translate in to physical domain where you do some kind of a chip level floor plant, like you decide that on your chip you can place your processor here, first memory here, second memory here the I O chips here. So, you do a tentative floor plan on the silicon, how you will be placing the different very high level modules here.

Now from here again you go follow this part, this says finite state machines not only finite machines, it can be some kind of logic specifications also. So, each of this blocks you again go to behavioral domain and you try to specify what this blocks actually try to perform; what is the functionality of the CPU, what is the functionality of the memory and so on and each of these again will be traversing this part this synthesis, you will be translating them into processors, ALU registers etcetera register ALU multiplexor from here again you will go to placement of the modules refining the placement; again you go to the description of the modules; again you go to description like gates or some cells as will talk about later.

Then again how to place this cells, then again low level finally, transistor up to the layout mask. So, you see if you follow this kind of helix approach from the outer more cycle to the inner most one, you are actually following the at exact process which typically we follow during the design when we go through so called top down design approach, from the high level thing we slowly refine our design and we go into more and more detailed design fine.

(Refer Slide Time: 11:03)



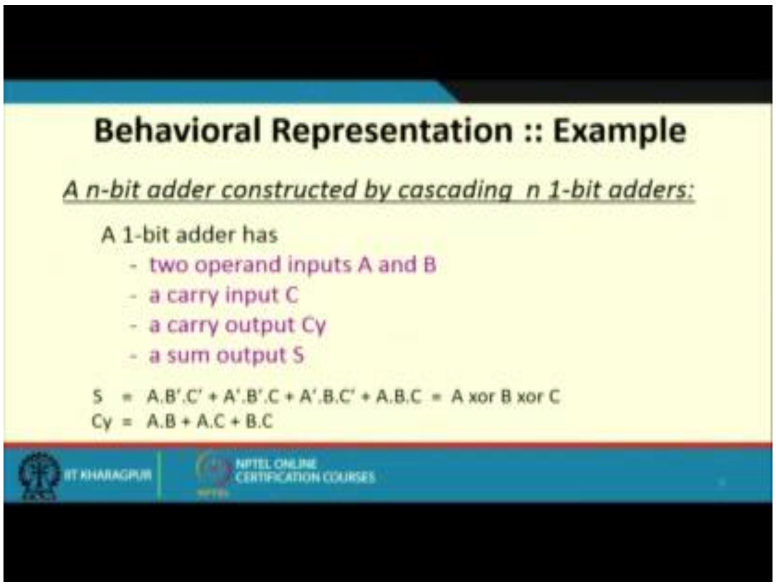
Behavioral Representation

- Specifies how a particular design should respond to a given set of inputs.
- May be specified by:
  - Boolean equations
  - Tables of input and output values
  - Algorithms written in standard HLL like C
  - Algorithms written in special HDL like Verilog or VHDL

So, let us look at the different levels of abstraction and see how in what this represents. Behavioral representation as I said this specifies how design should behave, how it should respond to a given set of inputs without specifying or telling how they are implemented; to just tell that I want this if I apply a an input of this, I should get an output of this is my behavior.

Now, this behavior depending on which level your specifying, can be specified in various ways some of this are mentioned here these are of course, not complete Boolean equations, you can have some kind of truth table, you can also have a finite state machine for sequential circuits, sequential functions; you can have some algorithm with description written in a standard high level language like C, or java, or C++, or you can also express this algorithms in the hardware description languages like Verilog or VHDL. So, behavior can be specified in any on one of this, but the point you notice that in behavioral specification you are not specifying exactly how to do you are just telling what to do, what I want, but exactly how to realize it we are not telling or is closing that.

(Refer Slide Time: 12:46)



Behavioral Representation :: Example

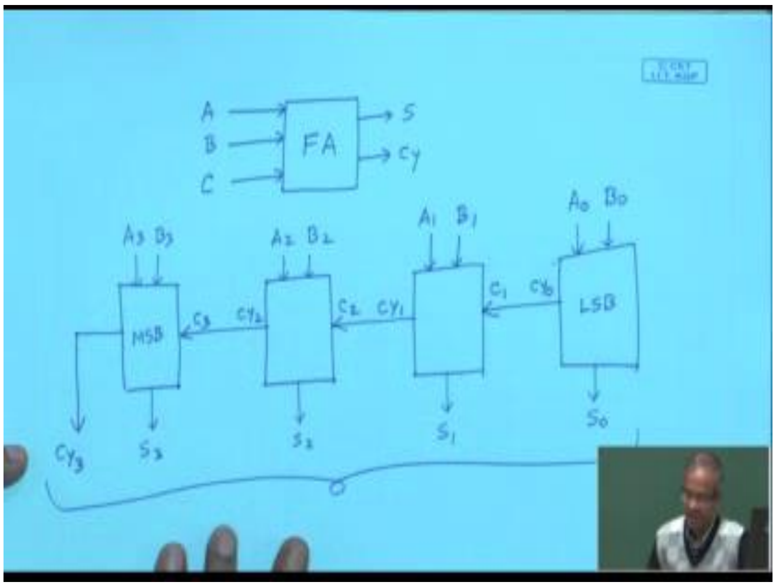
A n-bit adder constructed by cascading n 1-bit adders:

A 1-bit adder has
- two operand inputs A and B
- a carry input C
- a carry output Cy
- a sum output S

S = A.B'.C' + A'.B'.C + A'.B.C' + A.B.C = A xor B xor C
Cy = A.B + A.C + B.C

Now, let us takes some examples, let say we have an n-bit adder, which where constructing by cascading n 1-bit adders. So, how you do this? Let us first see.

(Refer Slide Time: 13:09)



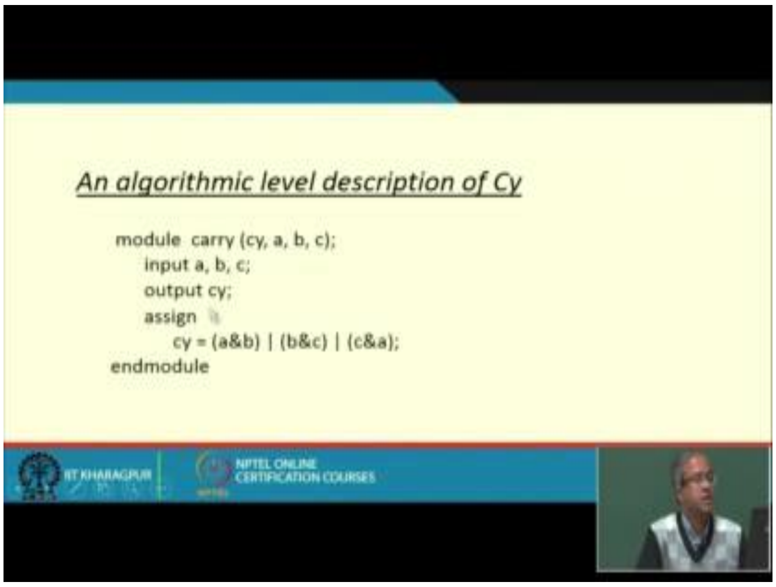
Let say we have a full adder; so how does a full adder work? Full adder has 3 inputs; the 2 inputs A and B 2 bits and may be a carrion C, and the outputs are some and carry. Suppose if I want to design a 4 bit adder the so called ripple carry adder what I do? I use 4 such full adders, and I cascade them in a chain like suppose this is my list significant bit, this is my most significant bit and these are the 4 full adders. So, what

I do? I connect my least significant bits A 0 and B 0 here, A 1 and B 1 here, A 2 and B 2 here, A 3 and B 3 here, and you get as the output this some from this side less significant some then S 1, S 2, then S 3.

Now, the carry outs, the carry out from this C Y 0 goes to the carry in of the next stage. Similarly C Y 1 goes to C 2, C Y 2 goes to C 3 and finally, C Y 3 that comes out that will be your carry out of the final addition. So, this is how 4 bit adder works, this is cordial ripple carry adder because the carry might ripple through the various stages before it comes out right. So, you can see this was might behavior, and this is structural level description where I used full adders at the basic building block and I have inter connected them right; which has example as shall show this on this slide or. So, here the 2 description that I have shown here, these are the functional descriptions of a full adder.

So, whenever full adder, the sum can be defined as this Boolean equation or A XOR B XOR C and the carry is defined as A B or A C or B C; these are the expressions for sum and carry in a behavioral fashion.

(Refer Slide Time: 16:11)



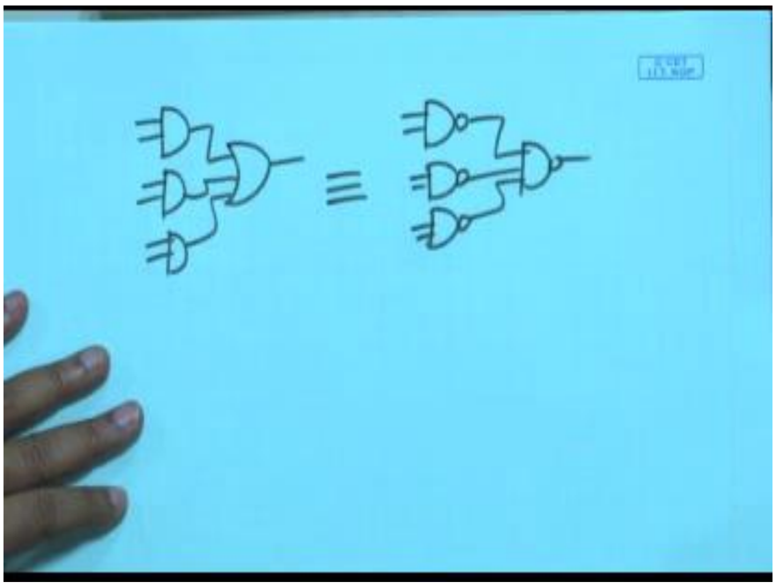
An algorithmic level description of Cy

```
module  carry (cy, a, b, c);
    input a, b, c;
    output cy;
    assign
        cy = (a&b) | (b&c) | (c&a);
endmodule
```

Now, you see this is the first look of a specification of a circuit module using a hardware description language; this is very long, this is how verylog program looks like. So, I said the carry function is a b or a c or b c. So, you see how I have written a module for the carry, you see this syntax is very similar to c like a function I define I start with a key word module, then the name of the module, then the parameters. The full adder will have 2 outputs carry out this can these not a full adder only the carry part, the output will be carry and the inputs will be a b and c.

So, I declare the input variables, I also declare the output variables then I assign this is AND and this is OR, a AND b OR b AND c OR c AND d; now you must say that I have mentioned AND and OR. So, this is as good as structural I have mentioned the gates, why I call it behavioral description?

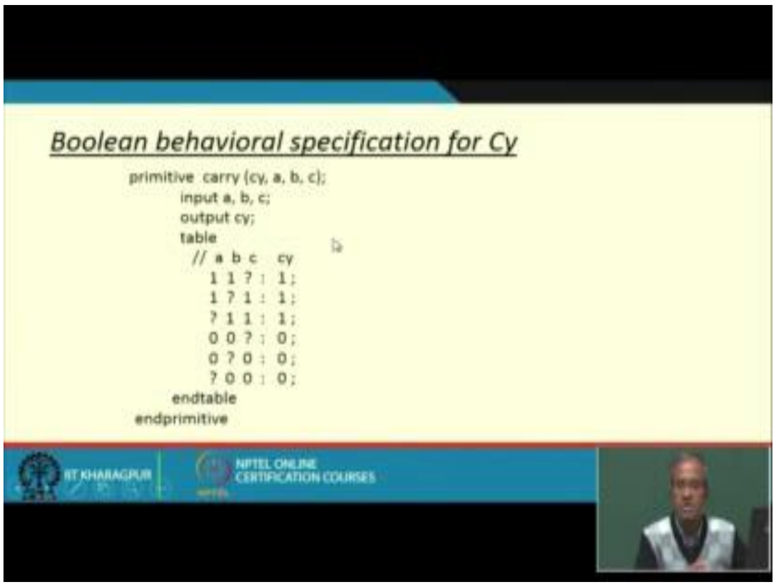
(Refer Slide Time: 17:41)



The reason is if I specify it like this, so I really do not know how this will be finally implemented. It can either be implemented as 3 and gets, implementing the 3 product terms followed by an OR or it can be implemented using NAND gates alone; both are equivalent, both these designs are equivalent. So, I really do not know when we are finally, synthesis in the design into the gates, whether you arrive at this or you arrive at this it depends on your cad tool, how it will actually translate your specification into the final gate level net list.

So, that is why I say that this is behavioral level description, not a structure level because I have not mentioned the exact gate types, I have mentioned the Boolean equation a b or b c or c a.

(Refer Slide Time: 18:43)



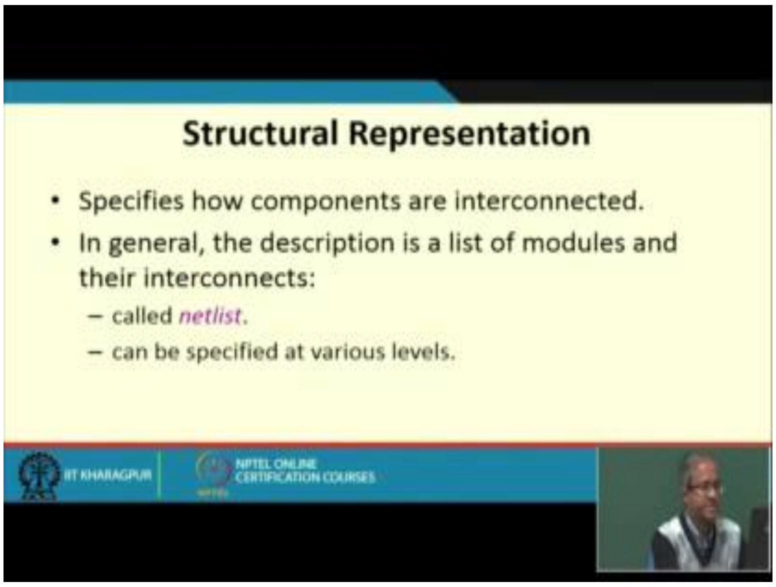
Boolean behavioral specification for Cy

```
primitive  carry (cy, a, b, c);
        input a, b, c;
        output cy;
        table
          // a b c   cy
             1 1 ? : 1;
             1 ? 1 : 1;
             ? 1 1 : 1;
             0 0 ? : 0;
             0 ? 0 : 0;
             ? 0 0 : 0;
        endtable
  endprimitive
```

Similarly, this is another way of specifying the behavior; here I can specify the truth table. So, here also I specifying the same specification for the caddy input and output and the convention is the inputs will come first and then the output. So, there is the table and n table construct, I give the inputs, then colon expected output. If sum of the inputs are do not care I can put question mark; like if a and b are 1 and 1, then irrespective of c the carry will be 1.

Similarly, if b and c as 0 and 0 then irrespective of a, the carry will be 0. So, here I can make my truth table short in this way right. So, this is also way to represent the carry function in a behavioral fashion fine.

(Refer Slide Time: 19:40)



Structural Representation

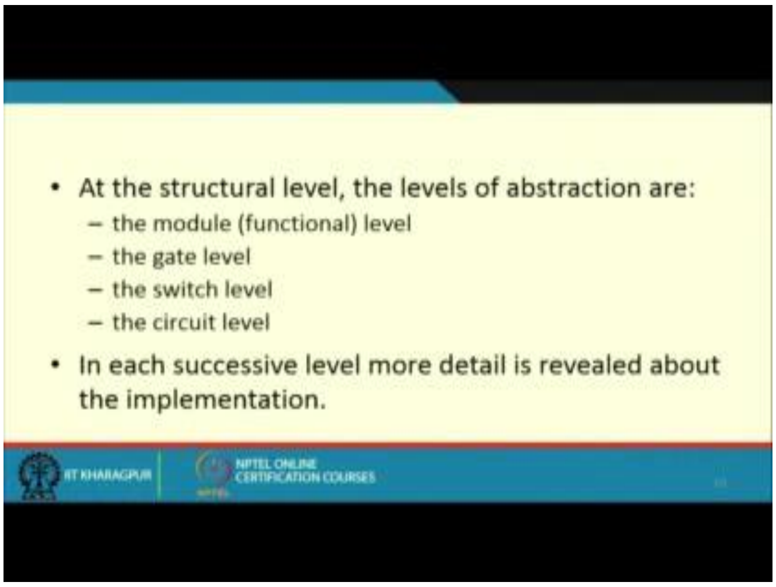
- Specifies how components are interconnected.
- In general, the description is a list of modules and their interconnects:
  - called *netlist*.
  - can be specified at various levels.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, let us move to the structural representation. So, structural representation essentially tells you how certain modules are components are interconnected, you specify some interconnection like these 2 circuits I have shown, these are examples of structural descriptions, here I have shown there are 4 3 NAND gets and 1 OR gate they are interconnected together, here I have shown 4 NAND gates interconnected together. These are examples of structural descriptions.

Now such a structural description is as I said nothing, but a list of modules or components, and how they are interconnected. This is sometimes called a net list, now a net list can be specified at various levels like gate level, transistor level, higher level and so on.

(Refer Slide Time: 20:55)



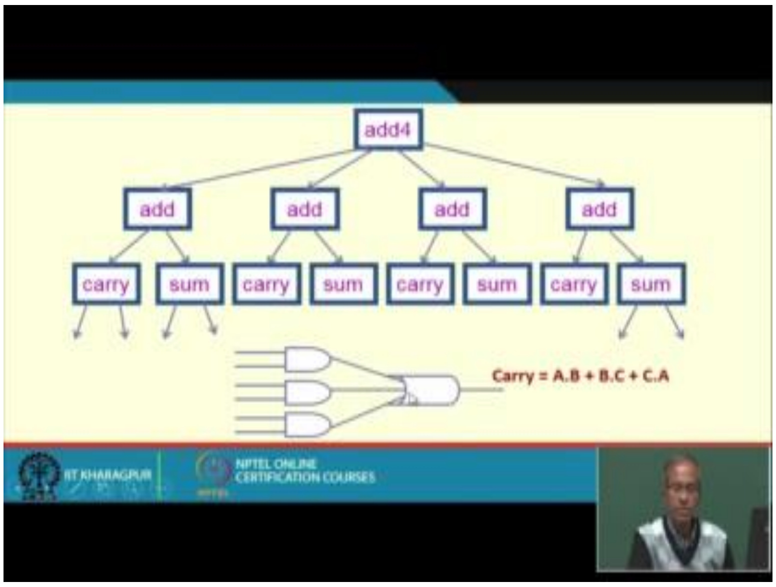
- At the structural level, the levels of abstraction are:
  - the module (functional) level
  - the gate level
  - the switch level
  - the circuit level
- In each successive level more detail is revealed about the implementation.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, we shall take an example. So, at this structural level as I said there can be various levels of abstraction like you can have a very high level module, like you if you just come back to this example I showed earlier. Here I had 4 full adders, and they where interconnected. So, this is a net list of full adders, this is at the functional or module level where my full adder is the module then I can have the gate level.

So, at the gate level I have a set of gates and the interconnection as I have shown in the diagrams here gate level. Similarly I can have switch level; switch level means the transistor level. So, each of the gate can be converted in to a transistor and circuit level means it is a further level of refinement, where you have not only the transistors, but also some you can say parasitically means like registers capacitors and so on. So, you have a circuit comprising of transistors, registers, capacitors typically that is called the circuit level view of the design. Now as you can see as you move towards the module level down to the circuit level, more on more details get revealed as you move down we get more and more detailed functionality, and clearly more accurate representation of the design or the behavior.

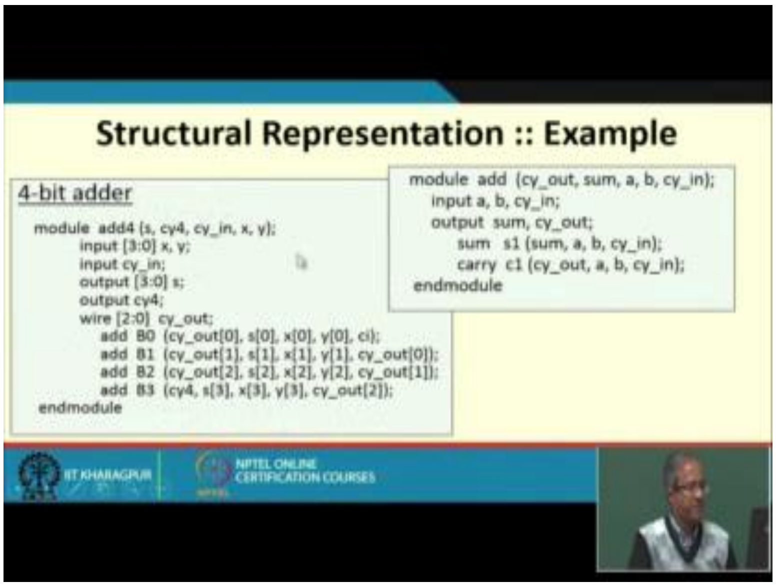
(Refer Slide Time: 22:40)



So, let us come back to the 4 bit adder once more. So, this is the hierarchical description of how we have created this structural description. The 4 bit adder let us call it add 4, the 4 bit adder we had created by using 4 full adders, these are 4 full adders and each of the full adder can be implemented by a carry module and a sum module. The second full order can also be implemented by a carry module and a sum module and so on. So, as you can see I have seen earlier that the carry module can be implemented like this 3 AND gates and an OR gate with the function like this, similarly this sum can be implemented as A XOR B XOR C.

So, now shall see how we can create structural description of this 4 bit adder using sum description language like verylog.

(Refer Slide Time: 23:47)



Structural Representation :: Example

4-bit adder

```
module  add4 (s, cy4, cy_in, x, y);
        input [3:0] x, y;
        input cy_in;
        output [3:0] s;
        output cy4;
        wire [2:0]  cy_out;
           add  B0  (cy_out[0], s[0], x[0], y[0], ci);
           add  B1  (cy_out[1], s[1], x[1], y[1], cy_out[0]);
           add  B2  (cy_out[2], s[2], x[2], y[2], cy_out[1]);
           add  B3  (cy4, s[3], x[3], y[3], cy_out[2]);
 endmodule
```

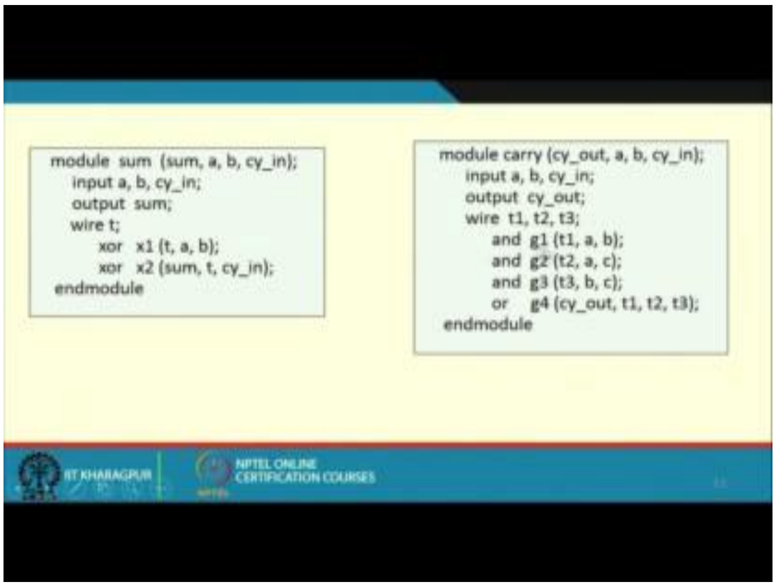
```
module  add  (cy_out, sum, a, b, cy_in);
    input a, b, cy_in;
    output  sum, cy_out;
       sum   s1 (sum, a, b, cy_in);
       carry  c1 (cy_out, a, b, cy_in);
 endmodule
```

Let see this. This is the top level description of the 4 bit adder. So, you see. So, at the miss at the highest level, what does your adder represent? You are adder will have sum carry out, carry in and 2 inputs, X and Y; your sum inputs X and Y or all 4 bit numbers that you represent here in terms of the array notation 3 colon 0 means X and Y or 2 arrays, whose index values can go from 0 to 3. Similarly s is also and an array, but the input carry seen and the output C Y 4 they are single bit, and intermediate connections like if you look at this full adder again there were 3 intermediate connections between the full adders.

So, this full adders intermediate connections we can defines some wires, and we instantiate 4 copies of full adders add represents of full adder whose description is here as you can see, this is the full adder what you have carry out and sum as the outputs a b and cy in as the inputs, you have a sum module you have a carry module. In this sum module we have sum as the output these are the inputs, carry module cy out is the output a b cy in the inputs. So, you instantiate or we include 4 such add modules and you see just by giving the variable names appropriately you provide with the interconnections right, your output of this B 3 cy out 2 will be the input of this c out 2 a B 2.

Similarly, this will be connected here, this will be connected here. So, the connections you can specify by giving the names appropriately. So, as you can see here and this diagram once more, the output of one stage is coming as the input of the other; output of one stage is coming at the input of the other. So, in this same way here you can specify the variable names appropriately, so that this interconnection can be specified. Now you will see this is still not a complete description, because we have not defined sum and carry; we have defined add we have use add 4 times to create a 4 bit adder, but now these are sum and carry.

(Refer Slide Time: 26:31)



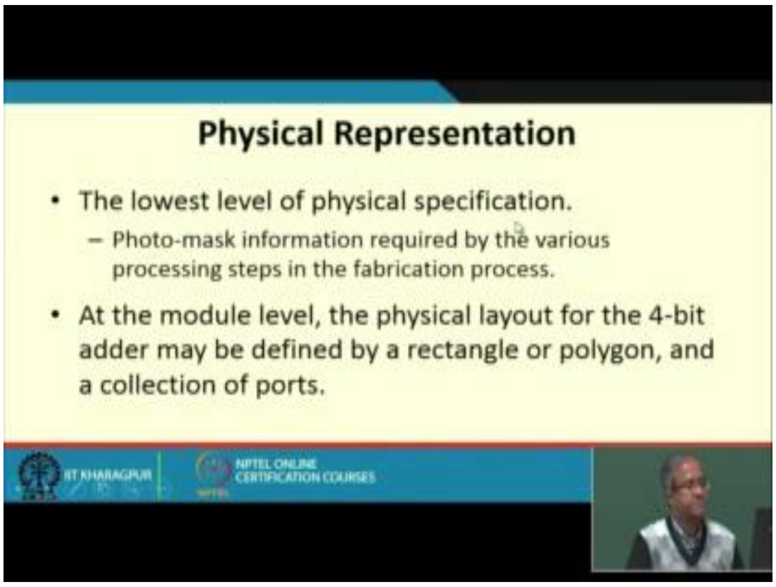
```
module  sum  (sum, a, b, cy_in);
   input a, b, cy_in;
   output  sum;
   wire t;
       xor   x1 (t, a, b);
       xor   x2 (sum, t, cy_in);
 endmodule
```

```
module carry (cy_out, a, b, cy_in);
    input a, b, cy_in;
    output  cy_out;
    wire  t1, t2, t3;
        and  g1 (t1, a, b);
        and  g2 (t2, a, c);
        and  g3 (t3, b, c);
        or     g4 (cy_out, t1, t2, t3);
 endmodule
```

You see here we have defined carry in terms of sum gates, we have not used the logic expression any more a b or b c or c a.

We have said that there are 3 AND gates and an OR gates, the convention is and t 1 a b means a b or the inputs, t 1 is the output. Second one a c or the inputs t 2 is the output here b c input t 3 output, and the final or gate t 1, t 2, t 3 are the inputs and cy out is the output. So, this is the pure structural description, because we have specifying the exact gates to be used and how they are interconnected. Similarly this sum where gives 2 exclusive or gates first one generates or output t, second one takes XOR of t and c generate sum. So, you can see these examples actually shows you that how we can create a structural description of a full adder down to the gate level, from the behavior 4 bit add a description down to the gate level, you can have a pure gate levels structural description.

(Refer Slide Time: 27:51)



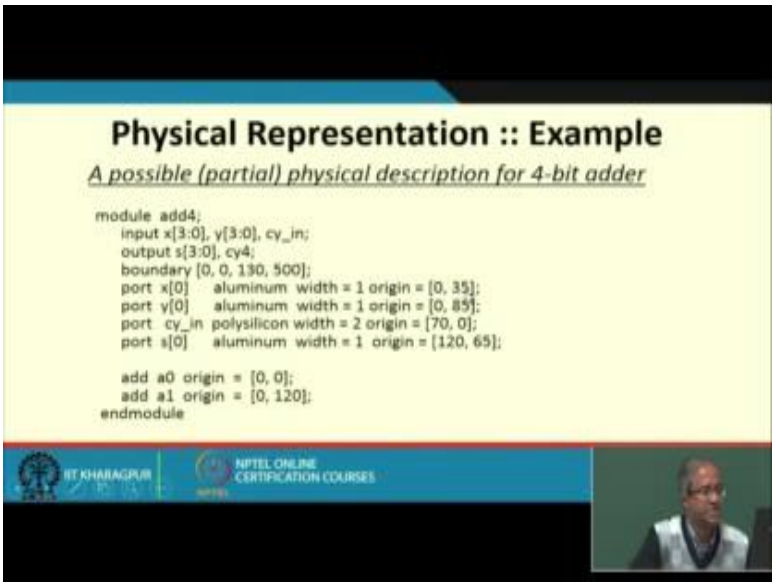
Physical Representation

- The lowest level of physical specification.
  - Photo-mask information required by the various processing steps in the fabrication process.
- At the module level, the physical layout for the 4-bit adder may be defined by a rectangle or polygon, and a collection of ports.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Now continuing at the lowest level, you can have the physical representation. So, here at the physical specification as I said that at the lowest level to specify the layout as the collection of some rectangles, as I had shown in that example earlier the rectangles can be in various layers; it can be in a metal layer, aluminium, it can be diffusion layer, poly silicon layer, various different fabrication layers are there. So, you have to specify the exact geometries of this layer and in which layer it belongs to, that will comprise of your physical description from which you can create the fabrication masks and you can go to the fab to fabricative chip using those masks. So, as I said the physical layout is nothing.

(Refer Slide Time: 28:55)



Physical Representation :: Example

A possible (partial) physical description for 4-bit adder

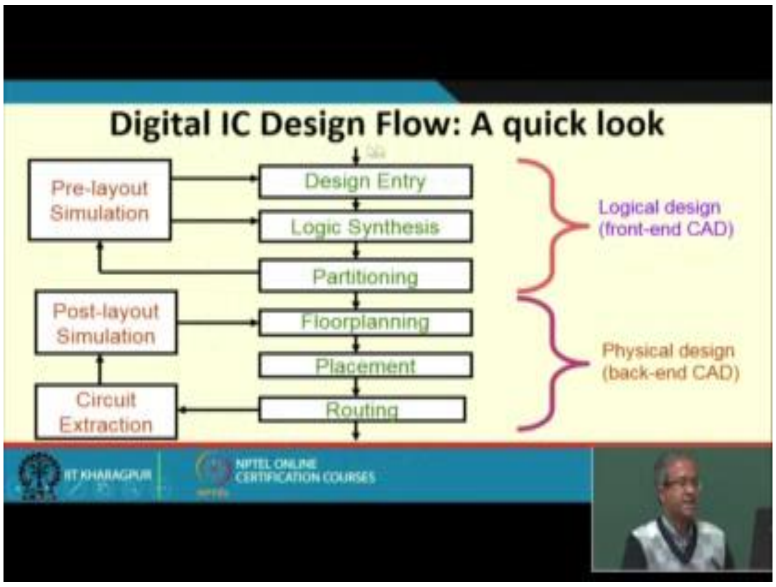
```
module add4;
    input x[3:0], y[3:0], cy_in;
    output s[3:0], cy4;
    boundary [0, 0, 130, 500];
    port x[0]    aluminum  width = 1 origin = [0, 35];
    port y[0]    aluminum  width = 1 origin = [0, 85];
    port  cy_in  polysilicon width = 2 origin = [70, 0];
    port s[0]    aluminum  width = 1  origin = [120, 65];

    add a0 origin = [0, 0];
    add a1 origin = [0, 120];
endmodule
```

But a collection of rectangles or polygons, where this is not a complete description this is the representation, this is only a partial description I have shown, just to tell you how the very log description for the physical representation can look like.

You see here you can specify some line called port, which is carrying a signal x 0 which is on the aluminum layer, whose width is 1, whose origin is at this coordinate. So, in this way you have some primitives for the ports, the boxes, lines everything where you can exactly specify which layer it is running on what is it starting coordinate and what is it size. So, in this way you can generate the final layout as well, this will be final used for fabricating the chip.

(Refer Slide Time: 29:49)



So, I think we have got a fairly good idea. Now in this I means final diagram design digital I C design flow let us have a quick look again, there are various steps of the processing design entry, logic synthesis, partitioning, floor planning, placement routing these we shall be discussing in the next lectures.

You see this whole design process can be divided broadly into 2 parts. The first few steps are called front end cad or logical design, the next few steps are called physical design or back end cad. Now as part of this course, we are more I means concerned about the physical design aspects, so we shall be assuming that we already have a circuit net list available to us, and from there we shall be carrying on this steps of floor planning, placement, routing and the other analysis steps and of course, that every step you need to carry out simulation, this things also we shall be discussing later. So, we can carry out simulation before your layout is created or after your layout is done then also you can extract this circuit and you can do something called post layout simulation which will of course, be much more accurate. So, roughly this is how things work.

So, I think with this way we come to the end of this second lecture. So, we shall be you continuing our discussion in the next few lectures as well.

Thank you.