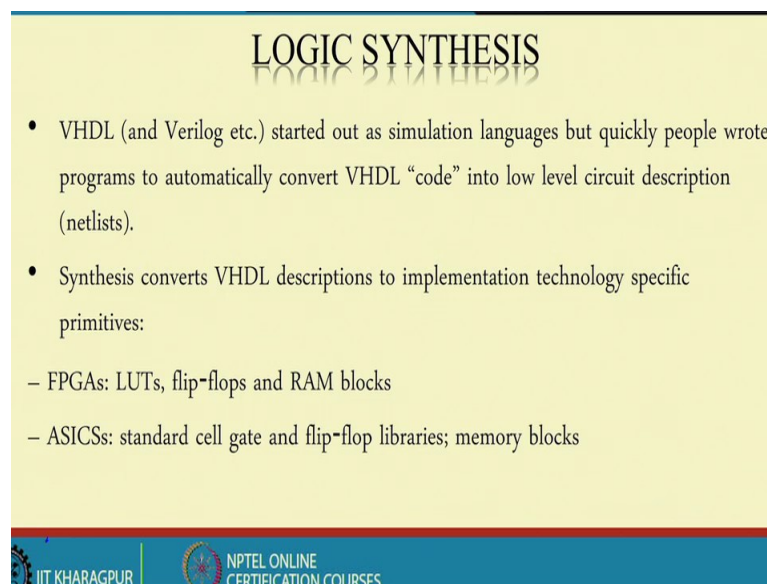


Architectural Design of Digital Integrated Circuits.
Prof. Indranil Hatai
School of VLSI Technology Indian Institute of Engineering Science and
Technology, Shibpur, Howrah

Lecture - 03
Introduction (Contd.)

Welcome back to the course on Architectural Design of ICs ok; so, this is the third lecture on introduction.

(Refer Slide Time: 00:25)



LOGIC SYNTHESIS

- VHDL (and Verilog etc.) started out as simulation languages but quickly people wrote programs to automatically convert VHDL “code” into low level circuit description (netlists).
- Synthesis converts VHDL descriptions to implementation technology specific primitives:
 - FPGAs: LUTs, flip-flops and RAM blocks
 - ASICs: standard cell gate and flip-flop libraries; memory blocks

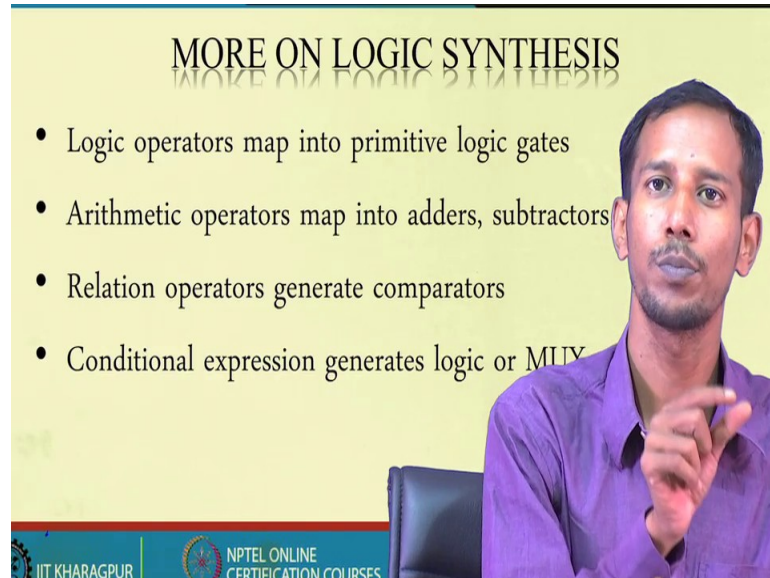
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, yesterday we have seen this in the; that means, FPGA design style it has to start with specification, then you have to; that means, the (Refer Time: 00:37) RTL specification or this a detailed design then there are function simulation and then synthesis, then timing simulation and then you can go for the device programming part.

So, and what I said that this logic synthesis, logic synthesis is the part for converting from the logical expression to the gate level net list ok. So, just more on to this what is mean, what does it means this logic synthesis just more add to on that particular perspective. It has to start with this VHDL or Verilog, then this synthesis procedure that for this process that basically converse this VHDL description to implementation technology specific primitives like in for FPGA it has to be implemented using LUTs, flip flops or RAM blocks in ASICs it has to be; that means, converted into the

corresponding standards cell gates or flip flop libraries or memory blocks if it is available ok.

(Refer Slide Time: 01:43)



The slide features a light green background with the title "MORE ON LOGIC SYNTHESIS" at the top. Below the title is a bulleted list of four items. On the right side of the slide, there is a photograph of a man in a purple shirt, who appears to be the speaker, gesturing with his hand. At the bottom of the slide, there is a blue banner with the logos of IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES.

MORE ON LOGIC SYNTHESIS

- Logic operators map into primitive logic gates
- Arithmetic operators map into adders, subtractors
- Relation operators generate comparators
- Conditional expression generates logic or MUX

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So,; that means, you have this particular information in your library. So, those particular, whatever logical expression you are describing using the HDL language during the synthesis that will those gates or those operators or those memory, memories will be called to implement your particular system or the particular expression which you have mentioned in the (Refer Time: 02:17) EDA tool ok. So, that is the process for logic synthesis ok.

(Refer Slide Time: 02:25)

The slide is titled "WHY SYNTHESIS?" in a large, bold, serif font. Below the title, there is a list of bullet points. The first bullet point is "Automatically manages many details of the design process:", followed by two sub-bullets: "– Fewer bugs" and "– Improved productivity". The second main bullet point is "Abstracts design data (HDL description) from any particular implementation technology". The third main bullet point is "In many cases, leads to a more optimal design than could be achieved by manual means." Below these, there is a green bullet point that says "• BE CAREFUL WITH SYNTHESIS:", followed by a green sub-bullet that says "– SYNTHESIZED HARDWARE DOES NOT MEET DESIGN SPECIFICATION!". At the bottom of the slide, there is a blue footer bar with the IIT Kharagpur logo on the left and the text "NPTEL ONLINE CERTIFICATION COURSES" on the right.

- Automatically manages many details of the design process:
 - Fewer bugs
 - Improved productivity
- Abstracts design data (HDL description) from any particular implementation technology
- In many cases, leads to a more optimal design than could be achieved by manual means.
- BE CAREFUL WITH SYNTHESIS:
 - SYNTHESIZED HARDWARE DOES NOT MEET DESIGN SPECIFICATION!

So then, why do I need to do this synthesis? So, if this synthesis process is basically automatically manages many details of the design process, it reduces the number of bugs. Bugs means that; that means, the fault you have in your system and this also improve the productivity, productivity means.

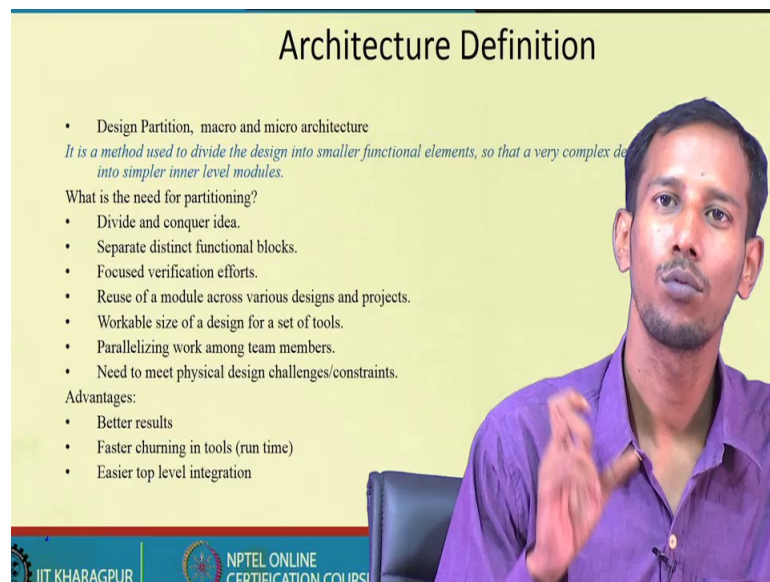
So, synthesis once you done if you; that means, the performance wise it will be much more on the higher side, then this abstract design data from any particular implementation technology in many cases leads to a more optimal design than could be a, than could be achieved by manual means. That means, this see this is all about this automatic or this tool based synthesis process and apart from this; that means, synthesis process means it has to pass through this optimization phase as you are putting the consent to it to the tool.

But manually optimization sometimes you can do the mistake ok. So, that is why automated tool wise synthesis optimization is sometimes it is better, but whenever you are doing this automated synthesis at that time, very much you have to be very much careful because this synthesized hardware does not meet the specification. Whenever this particular circumstance happens at the time you have to be more careful, how you have to be more careful? You have to be more, you will be knowing in that it is not meeting the specification after; that means, doing the simulation or sometimes what happens

during the optimization, tool based optimization sometimes this logic representation that changes ok.

So, to check all these things you have to be more cautious about doing this automated synthesis though will learn more on this manual optimization, tool also do the optimization so both the things we have to ok. So, whenever you are following or we are using this tool based optimization. So, at the time on the background some of these are already running ok. So, as a designer or as a researcher I have to know both the things that is the manual optimization, if I know the manual optimization on the background what is happening how it is basically the tool is doing the optimization that also will be very much known to me.

(Refer Slide Time: 05:26)



The slide is titled "Architecture Definition" and features a speaker in a purple shirt on the right side. The text on the slide is as follows:

- Design Partition, macro and micro architecture

It is a method used to divide the design into smaller functional elements, so that a very complex design is split into simpler inner level modules.

What is the need for partitioning?

- Divide and conquer idea.
- Separate distinct functional blocks.
- Focused verification efforts.
- Reuse of a module across various designs and projects.
- Workable size of a design for a set of tools.
- Parallelizing work among team members.
- Need to meet physical design challenges/constraints.

Advantages:

- Better results
- Faster churning in tools (run time)
- Easier top level integration

At the bottom of the slide, there are logos for "IIT KHARAGPUR" and "NPTEL ONLINE CERTIFICATION COURSE".

So, this architecture definition so we are this course is all about this architectural designs of ICs first the thing is that architecture definition. So, what does it means ok. So, this you have to build one system so at that time the design will be for a larger design it has to be partition into two, into two architecture you can just partition it, one macro architecture and another is the micro architecture.

So, what is that? It is a method; this design partitioning is a method which will be used to divide the design into smaller functional element. So, that a very complex design is split into simpler and inner level modules; that means, it is; that means, if I if I; that means,

just work on a be a very big complex circuit. So, it will take me to verify or to describe that particular system, it will take too much time.

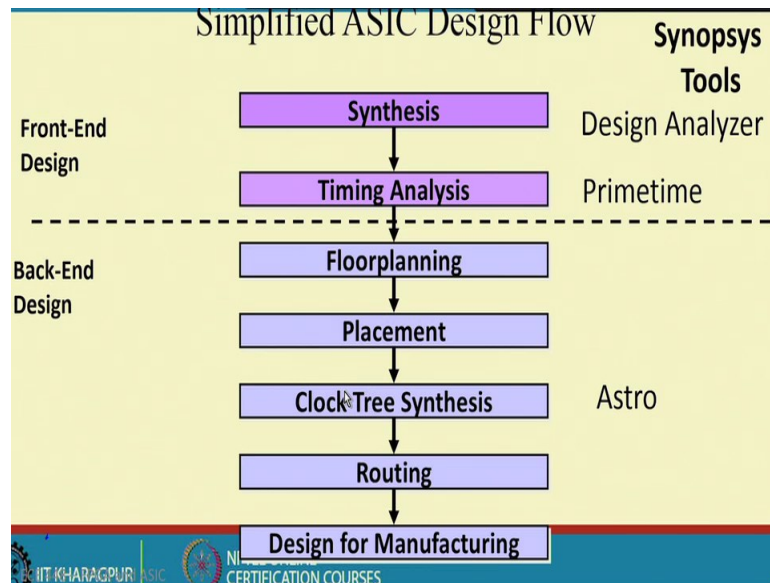
So, instead of that if I split that particular blocks into several chunks and each of this chunks will be monitored or as a designer you can monitor each of this things very easily and once each of this chunks or each of this blocks or this sub blocks, under the main blocks, each of this sub blocks are verified correctly at that time you put, you just integrate them all in all and you build the particular main module or the main block ok.

So, then what is the need for partitioning? So, here the idea is that, divide and conquer rule, ok. So, instead of doing the whole thing at a time simultaneously you just divide the things in to several blocks and do it parallely ok. So, that means, the time also for building the particular blocks, the time also I can enhance by simultaneously doing the sub blocks and separate distinct function, functional blocks focused verification efforts. Reuse of a module across various designs and projects, workable size of a design for a set of tools, parallelizing work among team members, need to meet physical design challenges and constraints.

So, you see there is several means, several advantages of doing the partitioning. So, the verification that becomes easier and suppose one block is used for different other application to or for. So, if I block wise if I just divide it and that particular block that will be used many of the system if it is required ok. So, once you develop, you use it for different application where it is required ok.

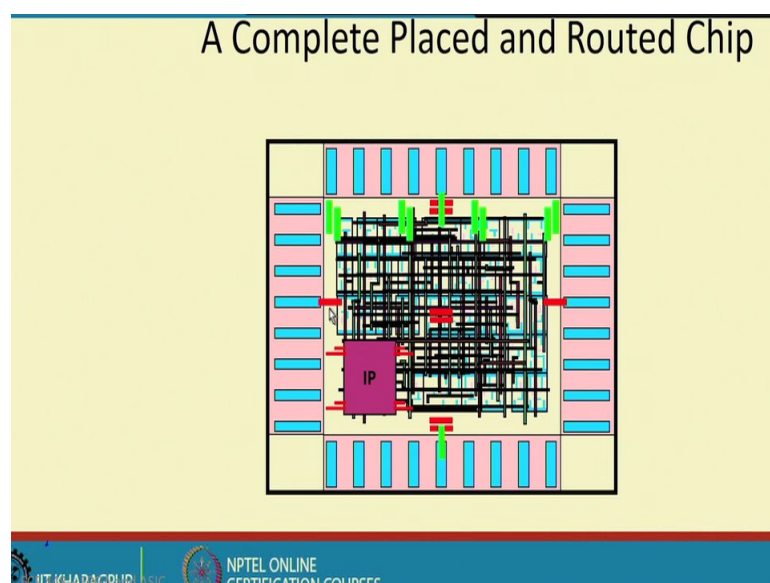
So that means, as I said that a partitioning basically you it you it helps to parallelize your work so; obviously, it reduce the design time.

(Refer Slide Time: 09:16)



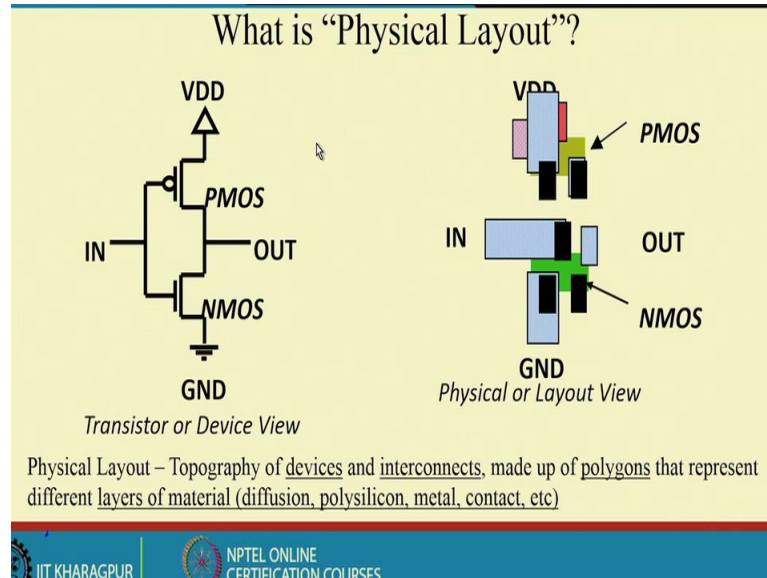
So, what I said that that this is this ASIC design flow. So, in ASIC design flow this floor planning, placement then this clock tree synthesis, routing and then for this send it to the fabrication lab. So, this backend part, this part is basically changes from the FPGA design flow the this front end part or this specification, then this architectural definition, then this synthesis, then simulation, then timing analysis all these part is basically common in both of the case both of the design style. But these portion, this device programming or in ASIC design these has to be do it manually, where you are getting that thing in readymade there in FPGA design style.

(Refer Slide Time: 10:14)



So, this is just one example of one complete placed and routed chip under the; that means, if I follow this ASIC designs style.

(Refer Slide Time: 10:25)



So, this is the; that means, I am talking about this layout, layout. So, what is this layout is nothing, but this physical; that means, view of the corresponding circuit. So, suppose this is one, this transistor level; that means, schematic of one inverter and this is the physical description of that particular inverter ok. So, here this particular things is basically related to PMOS and this particular things is basically related to NMOS and then there is the connection as there is a connection common between these gates and then this is, this for the out there is a common source and from there is a out.

So, this information this for PMOS and for NMOS this information is very much common or this information will be provided by the technology library which you are using. So, this will be different for different vendors. So, you use those; that means, information for particular vendors and you have to after this; that means, the whole suppose this inverter circuit I have built.

So, this particular geometrical description file that will be sent to the corresponding vendors only, it is not that the vendors like tsmcs are there or umc are there some of the; that means, companies are already there. So, if I use tsmc one particular; that means, vendors this information I cannot send it to the other vendors for the fabrication ok.

So, otherwise it will be they it will be; that means, not recognizable by their manufacturing process.

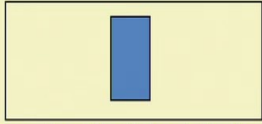
(Refer Slide Time: 12:26)

Process of Device Fabrication

Devices are fabricated vertically on a silicon substrate wafer by layering different materials in specific locations and shapes on top of each other

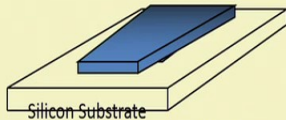
Each of many process masks defines the shapes and locations of a specific layer of material (diffusion, polysilicon, metal, contact, etc)

Mask shapes, derived from the layout view, are transformed to silicon via photolithographic and chemical processes



Layout or Mask (aerial) view

→



Wafer (cross-sectional) view

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So; that means, this is the process of device fabrication I will not go details about this; that means, one thing this geometrical; that means, files you has to be sent to the fabrication. So, then this fabricating lab or this fabrication lab they have to follows through the device fabrication procedure and they after that, they build the corresponding system for me.

(Refer Slide Time: 12:53)

Logic Synthesis

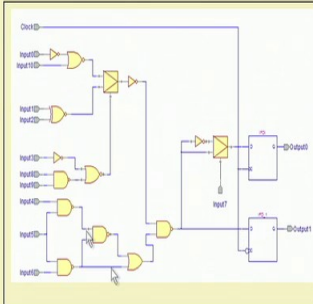
VHDL description

```
architecture MLU_DATAFLOW of MLU is
    and A1 STD_LOGIC;
    and B1 STD_LOGIC;
    and Y1 STD_LOGIC;
    and MUX_0, MUX_1, MUX_2, MUX_3 STD_LOGIC;
begin
    A1 <= A when (NEG_A = '0') else
        not A;
    B1 <= B when (NEG_B = '0') else
        not B;
    Y <= Y1 when (NEG_Y = '0') else
        not Y1;

    MUX_0 <= A1 and B1;
    MUX_1 <= A1 or B1;
    MUX_2 <= A1 xor B1;
    MUX_3 <= A1 xnor B1;

    with (L1 & L0) select
        Y1 <= MUX_0 when "00",
            MUX_1 when "01",
            MUX_2 when "10",
            MUX_3 when others;
end MLU_DATAFLOW;
```

Circuit netlist



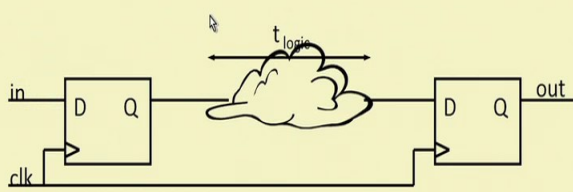
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, this logic synthesis is that here is this VHDL code which I have; that means, this is the logical expression which I have written using VHDL language and this is the corresponding gate level net list or the circuit level implementation of the corresponding circuit, what I have described using VHDL.

(Refer Slide Time: 13:16)

Critical Path (1)

- Critical Path – The Longest Path From Outputs of Registers to Inputs of Registers


$$t_{\text{Critical}} = t_{\text{FF-P}} + t_{\text{logic}} + t_{\text{FF-setup}}$$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, after that before to start with these things, we are basically very much cautious consider about this in any VLSI design style there are three things you must have to be considered that is speed, power and area. So, this three are the major concern in any system design or VLSI system design.


So, when ever we are talking about speed at that time this critical path comes. So, these are the basic things which you have to know from the beginning, because we will use this things in the later on. So, what is this critical path, the definition of the critical path is that this is the longest path from the outputs of the register to the input of the registers. Means what? Suppose this is one registers, this is one registers and inside of this register there are some, this combinational circuit or it has to pass some of the logic gates. So, this is; that means, the critical path is that the input of the register, sorry this input of the register to the output of the register. So, what does it follows this t_{critical} is then follows $t_{\text{flip flop}}$ plus this is the t_{logic} or the delay for this logic gates and then this is the $t_{\text{flip flop setup}}$ time.

So, what, why I need to consider this the delay about this t flip flop setup and then this t flip flop of this particular circuit.

(Refer Slide Time: 15:15)

Critical Path (2)

- Min. Clock Period = Length of The Critical Path
- Max. Clock Frequency = $1 / \text{Min. Clock Period}$


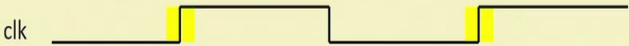


We will come to the later slide. So, once you got the critical path, so the maximum minimum clock period is the length of the critical path and then the maximum achievable clock frequency equals to 1 by this minimum clock period. So, this is the definition of maximum clock frequency and this is the definition of the critical path.

(Refer Slide Time: 15:46)

Clock Jitter

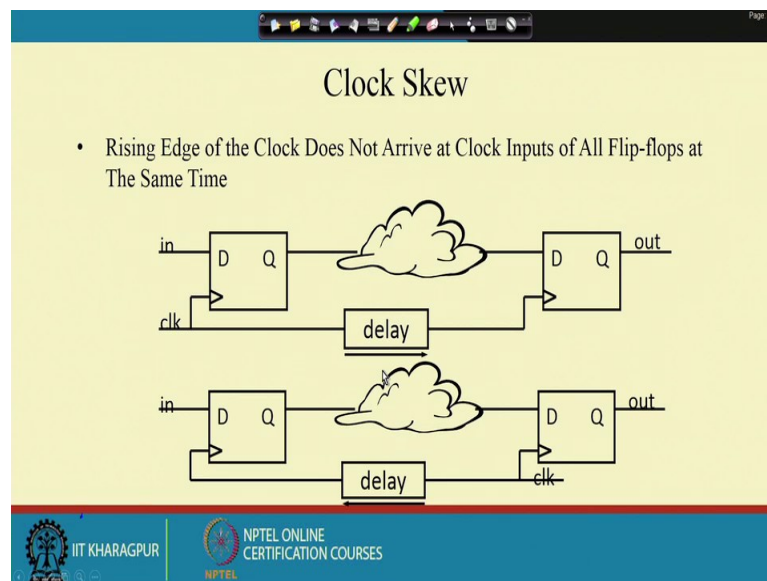
- Rising Edge of The Clock Does Not Occur **Precisely** Periodically
 - May cause faults in the circuit



So, then another information; that means, another important part of this is the clock jitter. So, it happens that we consider or we for a synchronous design we consider that each of the system will be work on a particular age of the clock. So, it is not on not that only rising edge or the falling edge also, but it will it has to occur sorry it has to work on the edge of the clock, ok.

So, it is whenever the clock has been; that means, generated. So, at that time it is not that all the edges of the clock occurs or at the same time; that means, these particular edge is not at all precise for all the clock edge occurs for a; that means, ongoing clock ok. So that why, what happens, it may cause faults in the circuit. So, that is why whenever we are using one clock source at that time we have to be more cautious about this jitter free clock, for the proper work of the system.

(Refer Slide Time: 17:08)



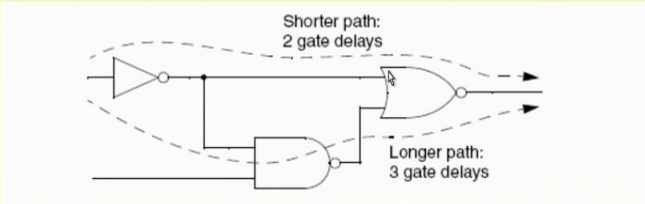
So, then another important parameter is that clocks skew. So, clocks skew is that the rising edge of the clock does not arrive at clock inputs of all flip flop at the same time, means what? That means, rising edge of the clock does not arrive at clock inputs of all flip flops at the same time when this type of; that means, scenario happens, here you see this clock is applied in this particular first flip flop then second flip flop with some delay. So, delay means if I use gated clock in any of the flip flop so at that time the clock which is basically applied directly and the clock which is applied after delaying or after using

some gated logic. So, there will be a difference of arrival of the clock edge to those particular flip flops.



So, at that time this clocks skew occurs ok. So, edge it, you can see that there are this two, basically this two flip flops are connected via some logic. So, the clocks are basically also differs the arrival of the clock is also differs. So, this may create some problem to the system ok.

(Refer Slide Time: 18:47)

Static Timing Analysis Review



- Tools will calculate all paths from sequential start point to sequential end point.
- The worst case path will be used for Setup analysis, and the best case path will be used for hold analysis.
- All paths are considered for design rule checking

 IIT KHARAGPUR  NPTEL ONLINE CERTIFICATION COURSES

So, then another thing is that this static time analysis. So, here you see this is one particular circuit ok, just we will consider that thing later.

(Refer Slide Time: 18:57)

False and Multicycle paths

- False path
 - Very slow signals like reset, test mode enable, that are not used under normal conditions are classified as false paths
- Multicycle path
 - Paths that take more than one clock cycle are known as multicycle paths.
 - Have to take define the multicylce paths in the analyzer and it takes those constraints into account when synthesizing

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, then in another two things are there which is that false path and multicycle path, false path is that very slow signal like reset, then test module enable that are not used under normal condition are classified as false path. And what are these; what is this multicycle path? Paths that make more than, that take more than one clock cycle are known as multicycle path and here has to, have to take define the multicycle path in the analyzer and it takes those constraint into account when synthesizing.

(Refer Slide Time: 19:40)

Multicycle path - Example

Multiplier takes 5 clock cycles to produce a result.

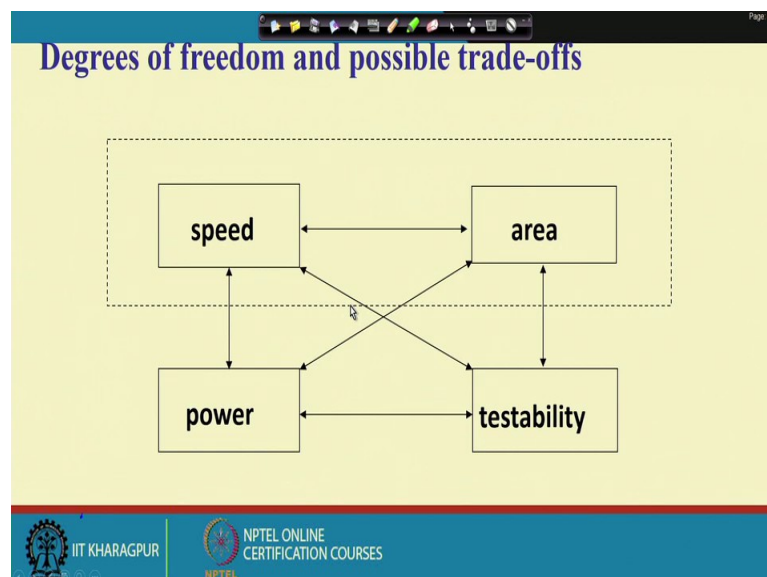
Figure 13 - Multi-Cycle Example

```
pt_shell> set_multicycle_path -setup 5 \  
-to [get_pins Capture_Reg/D]
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, what is the meaning of this? Multicycle path means suppose here I am having these particular this flip flop, this two flip flop that are basically connected in between of that they there is one multiplier, this multiplier takes 5 clock cycle ok. So, that is why the; that means, the data transfer from these to that, I have to wait for high clock cycle here to; that means to properly arrive the data here ok. So, the that is why this particular, this particular path is known as multicycle path, on whenever I am doing the synthesis at that time I have to mention that this multicycle path is from this Q 1 output to the, this D 1 sorry, D 2 input with multicycle of 5 ok. So, that we have to mention and this is in this if you use synopsis tool set this is a common for to do that ok.

(Refer Slide Time: 20:54)



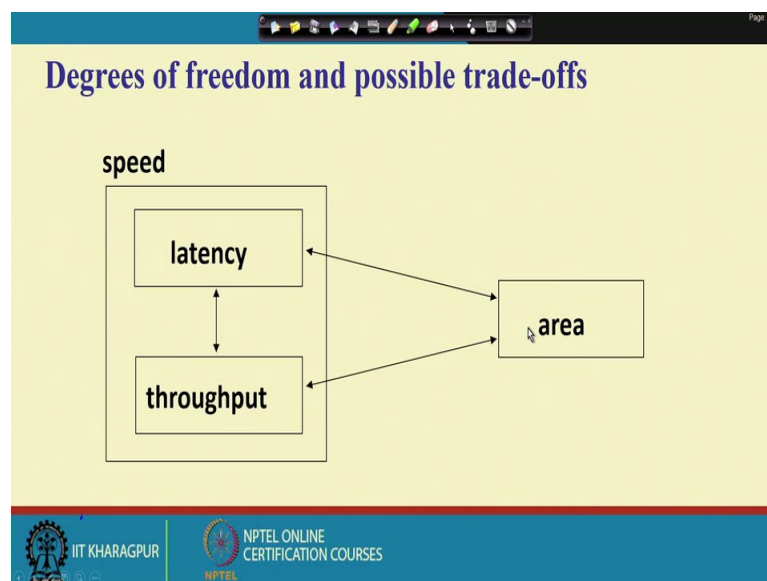
So, as I say that there is a; that means, very much; that means, relation between this speed power and area this three are the major concern of the main constant in your VLSI system design. So, another thing is that that is the testability, so each of this; that means, each of this four area, speed, power and testability each of these are basically interconnected to each other and they have trade off among them. So, why they have trade off among them; that means, it is not that at I can achieve the speed and area both at a time ok, for on the; that means, in the linearly like an increase speed and area both because they have the trade off, either you can go for the speed if you increase the speed.

So, at that time you have to be loose the area or if you just reduce the area at that time you have to lose the speed ok, but in area and power if you reduce the area. so at that

time; obviously, you can shape the power and speed wise also if you increase the speed; obviously, you will; that means, require more of the power. And for testability options so for it is not that, that it is not that what I said is time to pressure, the time to market pressure that are that becomes very much important nowadays. So, that initially whenever you have designed the system at that time the test option you have to, from the beginning you have to put that test option to the circuit itself ok.

So, the after fabrication the testing of the particular chip that becomes very much easier ok. So, whenever you are putting the additional test logic to your system, for building or to make one fault free or bug free chip. So, at that time; obviously, you are putting or you are; that means, compromising with speed power and area. So, that means, their testability is also basically, there is a tradeoff between these speed power and area ok. So, that is why there is all these fours are connected to each other.

(Refer Slide Time: 23:37)



So, as I said that whenever there is; that means, if you try to increase the; that means, if you try to decrease the area. So, at that time you have to lose in terms of speed. So, speed can be in terms of latency or in terms of throughput and if you just increase the speed at that time you have to put the parallelism here, ok. So, at that time parallelism means more of the; that means, you can make multiple copies of that each of the processing element and you can process it simultaneously. So that means, each of the processing element you are copying means, more the number of processing elements you are

copying or you are putting to do the same work to finish your job early so that means, you are increasing the area, ok.

(Refer Slide Time: 24:28)

Recommended rules for Synthesis

- When implementing combinational paths do not have h
- Register all outputs
- Do not implement glue logic between blocks, partition
- Separate designs on functional boundary
- Keep block sizes to a reasonable size

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSE

So, then this is, not at all.

(Refer Slide Time: 24:33)

Avoid hierarchical combinational blocks

Block A: reg1 → Combinatorial Logic1 → Block B: Combinatorial Logic2 → Block C: Combinatorial Logic3 → reg2

Not recommended Design Practice

The path between reg1 and reg2 is divided between three different block

Due to hierarchical boundaries, optimization of the combinational logic cannot be achieved

Synthesis tools (Synopsys) maintain the integrity of the I/O ports, combinational optimization cannot be achieved between blocks (unless “grouping” is used).

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, some of that means, whenever we are doing this particular synthesis procedures. So, at that times some of the things you have to be avoid, you have to be avoid these hierarchical combinational blocks, like here if you see I have this register 1 and register 2, among them I have three blocks, this is block 1, this is block 2, this is block 3. So, the

path between reg 1 and reg 2 is divided between three different blocks, due to hierarchical boundaries optimization of the combinational logic cannot be achieved. So, synthesis tool maintain the integrity of the I O ports and combinational optimization cannot be achieved between blocks. So, that means, instead of putting different sub blocks or this sub combinational block in between the register to register connection, you put it into, you club it into a that means, common combinational logic. Where a, the synthesis tool can auto easily apply the optimization procedure and it can gives you a better result.

So, it has to; that means, as a designer it you have to avoid this type of hierarchical combinational logic if you are following this particular, this type of particular system design and the how to handle the combinational path ok.

(Refer Slide Time: 26:04)



Recommend way to handle Combinational Paths

Recommended practice

All the combinational circuitry is grouped in the same block that has its output connected the destination flip flop

It allows the optimal minimization of the combinational logic during synthesis

Allows simplified description of the timing interface

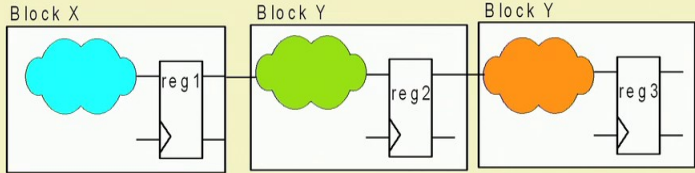



So, if you just see another logic; that means, the two things two registers it has. So, combine all the logic 1, logic 2, logic 3 level into one system and you this is the block A and this is the block B. So, here you see this is block A, this is block B and then block C, instead of that you just put it, this one as a block A and this one as block B or block C.

So, here you see all the combinational circuit is grouped in the same block that has its output connected the destination flip flops. It allows the optimization, optimal minimization of the combinational logic during synthesis and allows the designer to simply description of the timing interface ok.

(Refer Slide Time: 27:07)

Register all outputs



Register all outputs

Simplifies the synthesis design environment: Inputs to the individual block arrive within the same relative delay (caused by wire delays)

Don't really need to specify output requirements since paths starts at flip flop outputs.

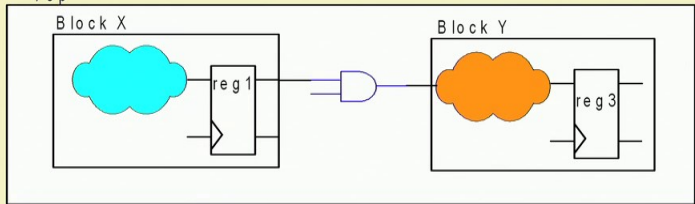
Take care of fanouts, rule of thumb, keep the fanout to 16 (dependent on technology and components that are being driven by the output)

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, or else what you can do; that means, if I am having this kind of; that means, combinational logic. So, then I can put the registers here, after each of the combinational logic I can put the registers and so; that means, what does it means or this will basically avoid the, the problems which occurs during the synthesis, automated synthesis ok.

(Refer Slide Time: 27:39)

NO GLUE LOGIC between blocks



No Glue Logic between Blocks, no matter what the temptation

Due to time pressures, and a bug found that can be simply be fixed by adding some simple glue logic. RESIST THE TEMPTATION!!!

At this level in the hierarchy, this implementation will not allow the glue logic to be absorbed within any lower level block.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, for today this is it ok.

(Refer Slide Time: 27:41)

Page 7/7

Separate design with different goals

Top

The diagram shows two registers, reg1 and reg3, each with a clock input and two data outputs. reg1 is connected to a blue cloud labeled 'Time critical path'. reg3 is connected to an orange cloud labeled 'Slow Logic'.

reg1 may be driven by time critical function, hence will have different optimization constraints

reg3 may be driven by slow logic, hence no need to constrain it for speed

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, next day in next lecture we will see some of the example ok. So, how we can, how we can apply our; that means, tweak or some of the; that means, tricks to get the, suppose one specification is I have to meet. So, and algorithm is fixed or the algorithm has been developed by the designer ok. So, as an architectural designer how I can use some of the tricks to make or to meet the specification in a easier way ok. So, that will see in the upcoming that means, lectures ok. So, for today this is it.

Thank you.