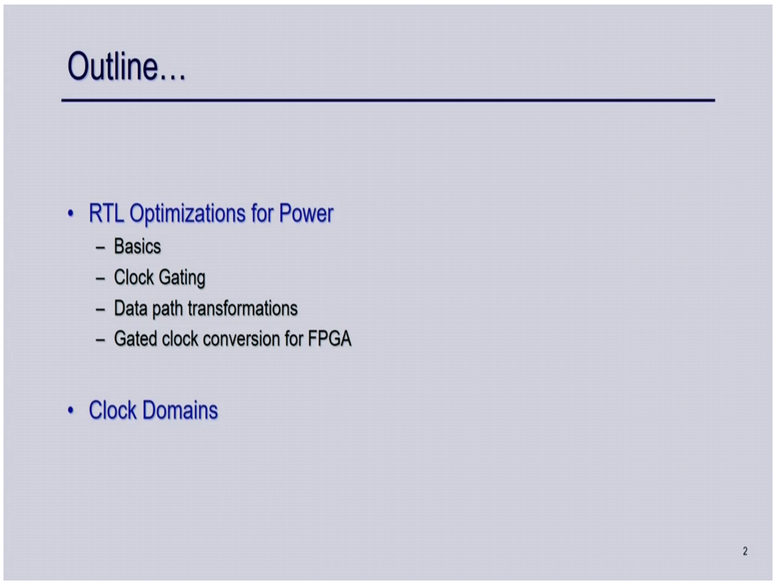
Optimization Techniques for Digital VLSI Design

Dr. Chandan Karfa
Dr. Santosh Biswas
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture – 09
RTL Optimizations for Power

Welcoming everyone; so, today we are going to discuss on RTL optimizing for power ok.

(Refer Slide Time: 00:36)



Outline…

- RTL Optimizations for Power
  - Basics
  - Clock Gating
  - Data path transformations
  - Gated clock conversion for FPGA
- Clock Domains

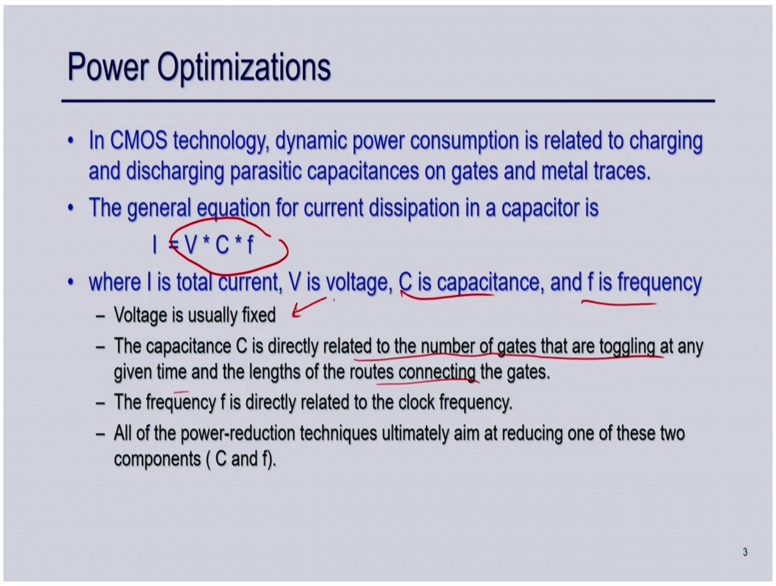
2

So, in today's discussion we primarily talk about what is the basics of power in a digital design and what are the kinds of techniques that we should apply just to improve the power efficiency of the design right or you or you can run the design in low power.

Also one more important aspect of a digital design is clocks and we most of the time we use multiple clock in our design, in our design needs multiple clock and that creates certain kind of problems right. So, we will also discuss about the clock domains in detail what are the issues with clock domains and whenever some signal crosses from one do domain to another domain what are the issues may arise and how to tackle those issues.

So, these are the two things primarily I am going to cover in this model.

(Refer Slide Time: 01:22)



Power Optimizations

- In CMOS technology, dynamic power consumption is related to charging and discharging parasitic capacitances on gates and metal traces.
- The general equation for current dissipation in a capacitor is

  $I = V * C * f$
- where I is total current, V is voltage, C is capacitance, and f is frequency
  - Voltage is usually fixed
  - The capacitance C is directly related to the number of gates that are toggling at any given time and the lengths of the routes connecting the gates.
  - The frequency f is directly related to the clock frequency.
  - All of the power-reduction techniques ultimately aim at reducing one of these two components (C and f).

3

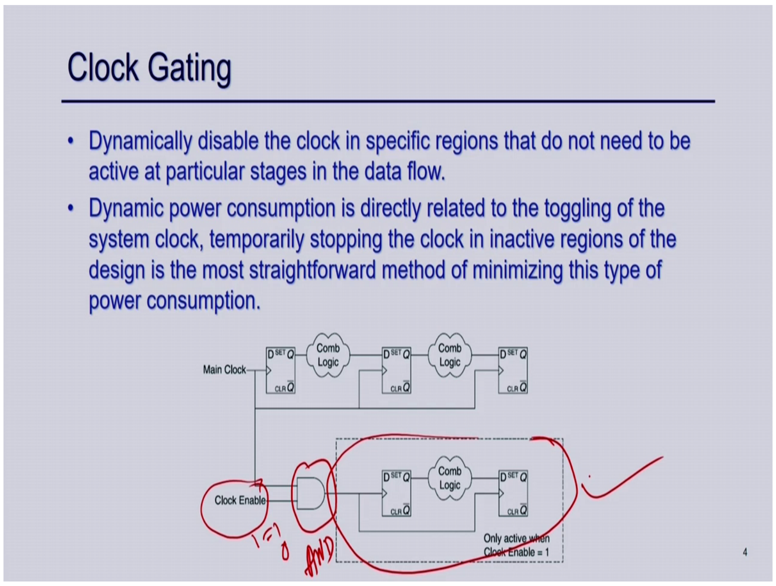
So, if you just look in to the power consumption of a circuit. So, in most of the digital design those are done using CMOS technology, the dynamic power consumption is related to the charging and discharging parasitic capacitances of gates and metal traces right. And if you just look very high level, that current dissipations which directed into the power is given by this formula is V into I; the I equal to V into C into f right where I is the total current, V is the voltage the voltage on which is the design is running, C is the capacitance and f is the frequency clock frequency. So, if you just look into an circuit usually voltages is fixed right.

So, we do not change much on voltage, but we will talk about that as well and the capacitance is related to the number of gates that are toggling at the give time and the length of the routing interconnections ok. So, in the very basic level or very high level then capacitance is depends on the number of toggling right. Toggling means when your data state change from 0 to 1 or 1 2 0. So, those are toggling upload digits circuits right so.

So, whenever there whenever you have lot of toggling your capacitance will be increased and your total power consumption will be will increase simultaneously right in proportionally. So, and the third parameter is the frequency what is frequency? Frequency is the clock frequency and it is also directly proportional the power consumption directly to related to the frequency of the clock.

So, primarily whatever the technique that we are going to talk about in related to the power optimizations or making the design run on low power is primarily targeting either reducing the capacitance or the frequency of those designs ok. So, capacitance means number of toggling in your design ok. So, these are the two key parameters and most of the optimism techniques that are applied in digital designs, they play on these two parameters only ok.

(Refer Slide Time: 03:45)



Clock Gating

- Dynamically disable the clock in specific regions that do not need to be active at particular stages in the data flow.
- Dynamic power consumption is directly related to the toggling of the system clock, temporarily stopping the clock in inactive regions of the design is the most straightforward method of minimizing this type of power consumption.

So, let us move on. So, some of the well known techniques that are applied in digital techniques and most well known is clock gating. So, what is clock gating? Clock gating is something in your design you try to switch off some portion of the circuit which is not active in a current moment of time right. So, it is very obvious that if you have big design not everything is running in doing the every part of the circuit is doing actual computer sense in all the clocks right in some clock some portion is active in some clocks some other portion is active and so on.

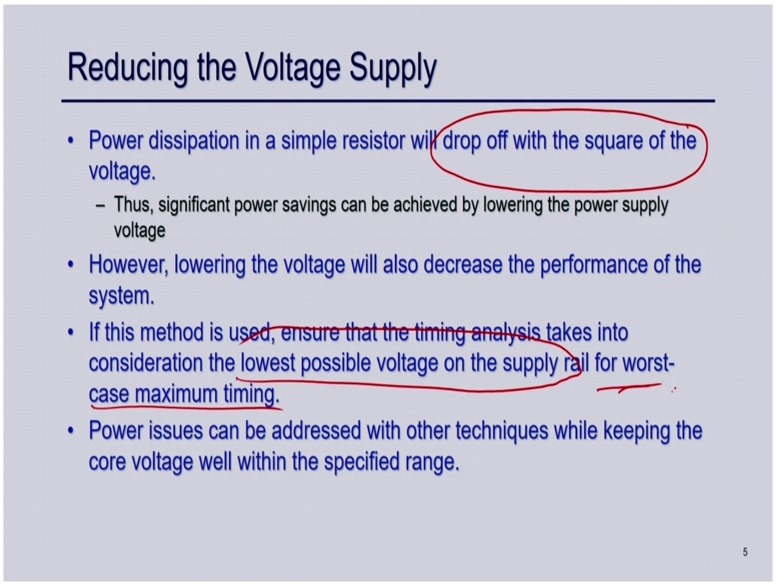
So, the idea is that if you can switch off the part which is inactive or which computer data is kind of unused for a particular clock then the number of toggling will not happen in that particular circuit that portion of the circuit and as a result your power dynamic power consumption of your circuit will reduce right. So, which is given by this example suppose I have this is one circuit and this is the part of the circuit which is not active in all the clocks right. So, you can actually apply a clock enable, which is when is one when this is one then this part is active whenever this is 0 this part is not active.

So, then I can actually get it this called clock gating, where I am just the system clock or the main clock is gated with this enable signal and gate and gate gated with that enable signal and is fare as the clock of this design right. So, whenever this enable is 0 the clock is something 0 so; that means, nothing no change in the data state of this register will happen, as a result your this circuit will be inactive and there will be no toggling of data as a result your power I mean power consumption in this of this part of the circuit will be less right.

So, this is what is called clock gating and this is very well known technique and is used widely in the specifically in the ASIC targets not for FPGA that will discuss in detail. So, for FPGA this is not a acceptable or that is not a preferable power optimization technique we should avoid clock gating; if your really design something for FPGA. So, this is one technique that effectively try to reduce the toggling.

So, effectively reducing the c playing with the value c right which is capacitance.

(Refer Slide Time: 06:10)



Reducing the Voltage Supply

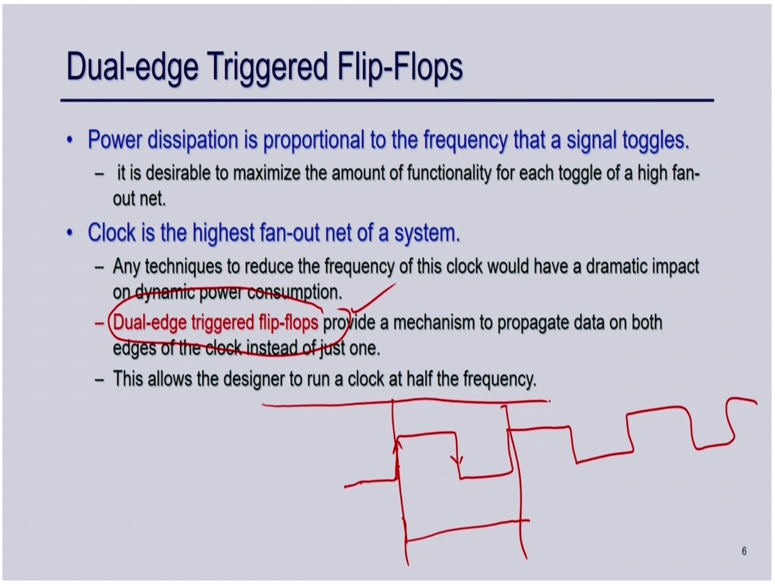
- Power dissipation in a simple resistor will drop off with the square of the voltage.
  - Thus, significant power savings can be achieved by lowering the power supply voltage
- However, lowering the voltage will also decrease the performance of the system.
- If this method is used, ensure that the timing analysis takes into consideration the lowest possible voltage on the supply rail for worst-case maximum timing.
- Power issues can be addressed with other techniques while keeping the core voltage well within the specified range.

5

Let us move on. So, next strategy is something reducing the voltage supply, as I mentioned earlier also that changing the voltage supply is not kind of advisable it is not a common practice, but what we can do, but this is very key factor because what we can see that this actually the total power actually depends on the square of the voltage. So, if your voltage reduces by 2, your power consumption will improve by 50 percent right. So, that is something by 2 it is 400 percent right so, or multiple of 4 times right; 4 times. So, that is something going to happen. But this is something is not usually happen. So, we have to what we have to make sure that, whatever the design you have done this is actually running by the lowest possible voltage supply right.

So, just to meet the worst case timing; So, you should not run your design in very high voltage even if your meeting time quite compatible, you should run your design as low voltage as possible just to meet the worst case timing. So, this is something also an voyage has to be reduce your power of your design right. So, reducing the voltage supply, but again this is not something is very commonly used ok.

(Refer Slide Time: 07:28)



Dual-edge Triggered Flip-Flops

- Power dissipation is proportional to the frequency that a signal toggles.
  - it is desirable to maximize the amount of functionality for each toggle of a high fan-out net.
- Clock is the highest fan-out net of a system.
  - Any techniques to reduce the frequency of this clock would have a dramatic impact on dynamic power consumption.
  - Dual-edge triggered flip-flops provide a mechanism to propagate data on both edges of the clock instead of just one.
  - This allows the designer to run a clock at half the frequency.

6

The third technique we are going to talk about is dual edge triggered flip flops what is that? So, what you have seen that the number of the power consumption directly related to the number of toggling of your design right. So, if you if you find some high toggling signals right.

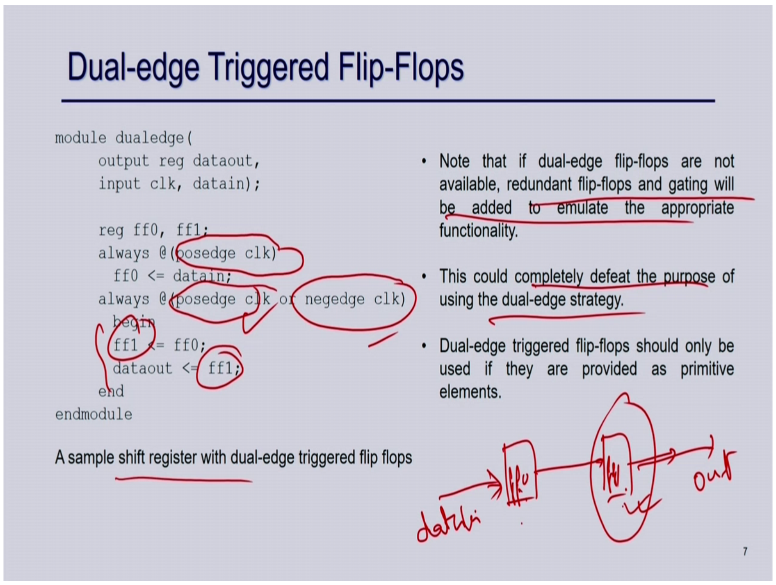
So, we actually toggles in every clock for example, clock right. So, clock actually toggles in every clock, I mean every cycle right. So, this is always toggle this is your clock and it always toggle. So, if you can reduce your frequency, the number of toggle of your clock will be half right. So, reduce will be also reduce. So, if you if your current design is running in say 100 megahertz, and if you want to make it half; that means, now you circuit will run in 50 megahertz.

So, number of toggle of clock will be exactly half right because now the clock is double of the previous one. So, your number of toggling of your clock will be half. So, this is something really important, but on the other hand if you if your circuit actually meeting timing requirement in 100 megahertz, and you make it 50 megahertz your design will not work right because your and if you are running for say same number of I mean same time period, that you will not finish your job right because earlier if you need say 10 cycle now it will only have 5 cycle. So, in 5 cycle you cannot execute all 10 operations in 10 cycles right.

So, then what you have to do you have to use the both rising edge of the clock right. So, this is your positive edge and this is your negative edge and you try to use. So, this is 1 clock period right. So, this is 1 clock period and there are 2 edges right usually we all the flip flops are active at the rising edge of the clock. So, whenever they are rising edge this data state of flip flop get changed.

So, if you can use dual edge triggered flip flop, which is what which actually update on both the clock, this is positive edge this is negative edge in both the edges if it is toggles right data state get changes, then we can actual so; that means, what happens the two operations can be executing within 1 clock period right so; that means, even if you make it your clock half, can effectively execute all of your operations in the prescribed time period. This is what is the use of dual edge triggered flip flop; So, using that we can actually run the whole design using the half of the frequency right.

(Refer Slide Time: 10:01)



Dual-edge Triggered Flip-Flops

```
module dualedge(
      output reg dataout,
      input clk, datain);

      reg ff0, ff1;
      always @(posedge clk)
        ff0 <= datain;
      always @(posedge clk or negedge clk)
        begin
        ff1 <= ff0;
        dataout <= ff1;
      end
endmodule
```

A sample shift register with dual-edge triggered flip flops

- Note that if dual-edge flip-flops are not available, redundant flip-flops and gating will be added to emulate the appropriate functionality.
- This could completely defeat the purpose of using the dual-edge strategy.
- Dual-edge triggered flip-flops should only be used if they are provided as primitive elements.

7

So, if there is an example of the shift register shift register, which is actually using dual edge triggered flip flop. So, what is happening here? So, this 2 bit shift register right. So, there are 2 register this is data in this is f f 1, f f 0, this is f f 1 and this is just connecting. So, basically whatever data is coming it is shifting every clock right that is what shift register is, but what is happening? This is actually updating in every posedge clock posedge. But this f f 1 is updating either in positive clock as well as the negative clock so; that means, in both the edge of the clock this data will be updated right.

So, every two cycles what is happening here this f 0 f 1 updated by f 0, and what about the content of f f 1 that is going to the output right this is out. So, this is data in. So, this is a shift register where this flip flop is dual edge triggered and this is just an example. So, dual edge triggered flip flop is something very useful, because it actually make your circuit run half of your frequency and which is improve your total power consumption of your circuit right.

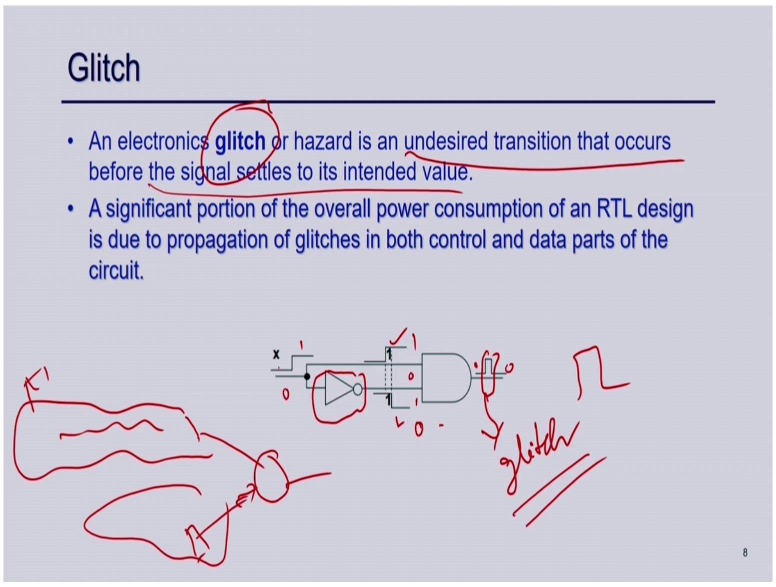
But one thing is to be noted here that not many of the technology support this dual edge triggered flip flop. So, if it is not supported you should not use it right. So, if say if your targeting some specific architecture of FPGA (Refer Time: 11:32) 2 or (Refer Time: 11:33) 6 or say Altera statis 5 something that may not have this particular dual edge triggered flip flop.

If it is not there then it has to be implemented using some redundant logic right. So, some redundant flip flop and gated logic just to eliminate these things because this is not exactly there and that actually defeats your purpose. So, you your purpose will not be served unless until part that particular device has dual edge triggered flip flop.

So, if it is have, we can actually utilize it and we can reduce our power. So, again this is basically technologic specific or target architecture specific, but if it is here have you can actually make your circuits like this, I mean you can update your registers in both positive edge negative edge clock. So, that this particular flip flop will may have to this dual edge triggered flip flop and effectively your power consumption will be reduced ok.

So, this is another tech technique. So, we have learned clock gating, we have learned I mean reducing voltage supply, then we have learned dual edge triggered flip flop concepts ok.

(Refer Slide Time: 12:37)



Glitch

- An electronics **glitch** or hazard is an undesired transition that occurs before the signal settles to its intended value.
- A significant portion of the overall power consumption of an RTL design is due to propagation of glitches in both control and data parts of the circuit.

So, and so, now we will move on to another concept called glitch. So, that also consumes lot of power right. So, what is glitch is basically hazard or undesired transition of the circuit right. So, before the signal settles to a actually intended value what is what is that? So, he here is an example. So, suppose you have a and gate where you are actually putting x and x bar. So, the output should be always 0 right. So, this should be always in zero state because x and x bar if you just do a and here it is always 0.

But what is happening here suppose initially my x is 0. So, I have 0 here I have 0 at this input and here is 1 because this is negation of this right. So, I have 1 here. So, because of that output was also 0 right and now I just move it make it 1.

So, I have now one. So, immediately this 1 is available here. So, 1 is available here, but since there are some combinational circuits here, it may not come exactly at the same moment and exactly at the same moment here right so; that means, earlier it was 1 right. So, because it was 0 it was 1 here. So, for a moment this is also 1, this is 1 and the output of this and gate will be 1 right. So, this is what is called glitch.

But this may be stand for a few second or something, but just after the moment this will become 0 whenever this will become 0 and this is 1 again this output will become 0 right. So, this is what is called glitch.

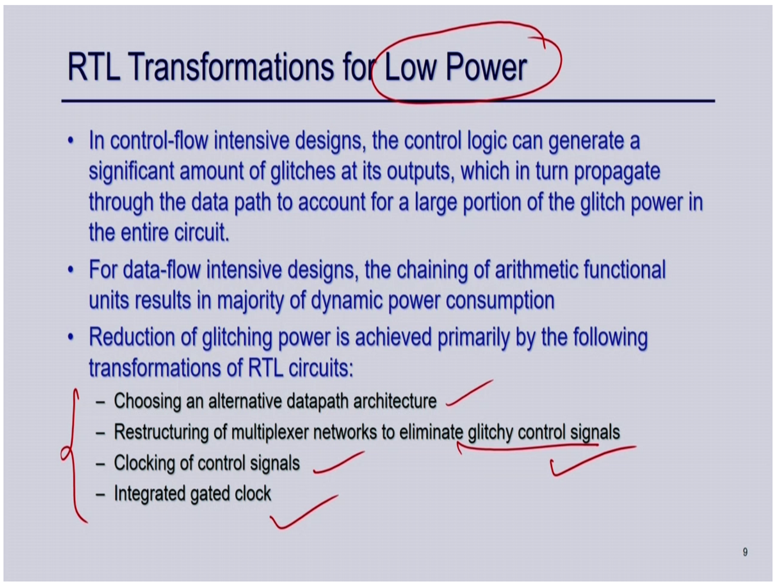
So, this is just a very simple example just to illustrate what is glitch, but what why it is happened. So, if you just think about the circuit, it has multiple input and this may be have a very long combinational delay right. So, this is coming from a register which is coming from a very very long combinational delay and this is very short right. So, then this is not expected that this both the signal will come at the simultaneously to this input of the registers.

So, there will be some delay ma mismatching delay combinational delay of the inputs, because of that this kind of spurious transaction will happen at the output and that is what is called glitch right and as I mentioned earlier also that power consumption it depends on this kind of this toggling. So, that is just going to be temporarily or momentarily toggles and this glitch actually consume some power right.

So, and so, basic idea is that we try to apply some kind of optimization or transformation of your circuit, which try to reduce the glitch of your design right. So, basically if you if your both the input or kind of balanced, then the chances of glitch is very less, but in reality that may not be always happen and because of some glitch might happen.

So, we try to update or modify my data path or the data path architecture little bit. So, that I can reduce this glitch, and which is effectively reduce the power consumption right.

(Refer Slide Time: 15:42)



RTL Transformations for Low Power

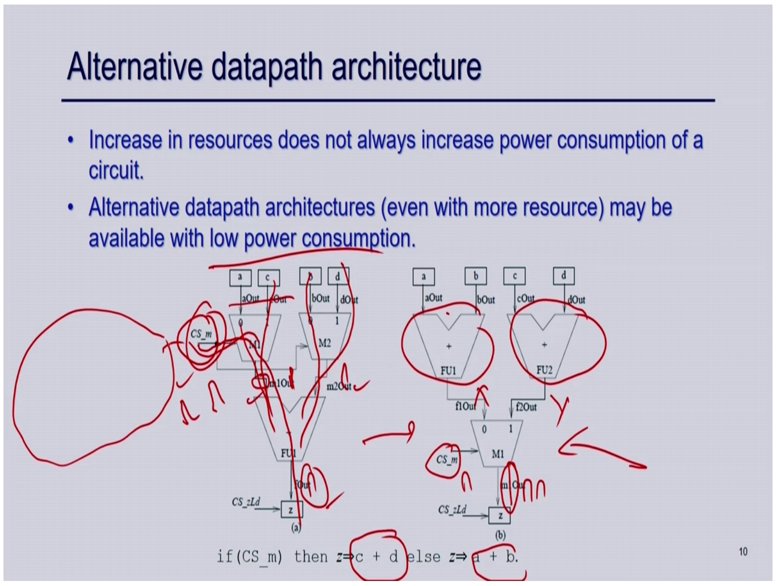
- In control-flow intensive designs, the control logic can generate a significant amount of glitches at its outputs, which in turn propagate through the data path to account for a large portion of the glitch power in the entire circuit.
- For data-flow intensive designs, the chaining of arithmetic functional units results in majority of dynamic power consumption
- Reduction of glitching power is achieved primarily by the following transformations of RTL circuits:
  - Choosing an alternative datapath architecture
  - Restructuring of multiplexer networks to eliminate glitchy control signals
  - Clocking of control signals
  - Integrated gated clock

9

So, based on that there are certain kinds of transformation can be applied on RTL circuit, just to improve the low power right. So, for example: choosing alternative data path architecture, restructuring the multiplexer networks to eliminate glitchy control signals, then clocking control signals and integrated gated clock.

So, this four techniques we are going to understand, which effectively try to reduce the glitch of your design right.

(Refer Slide Time: 16:09)



Alternative datapath architecture

- Increase in resources does not always increase power consumption of a circuit.
- Alternative datapath architectures (even with more resource) may be available with low power consumption.

if(CS_m) then z⇒c + d else z⇒a + b.

10

So, here is the first example. So, we have an perception that if your number of resource is high, your power consumption will be also high, but that may not be always true right. So, if you just think about an example here, I have 2 multiplexer here and adder here and in another architecture I have two adder one multiplexer.

But we assume that multi adder is more complex circuit then multiplexer. So, we can say that this particular circuit has more resources compared to this resource circuit right. Now, if you just think about what is happening here it is this control signal choosing between a and c and b and d here right. So, when this is 0, I am going to compute a plus b it will a will come here b will come here. So, it will a plus b output z equal to a plus b, when this is 1 this c will come here and d will come here. So, it is c plus d.

So, either it will computing c plus d or a plus b and based on this control signal will decide that output will be either c plus d or a plus b. So, in the alternative architecture what I have done, I just do operations plainly right a plus b here c plus d here and based on this control signal I decided to select whether a plus b or c plus d. So, effectively these 2 circuits are same they are executing the same operations, but their number of resource is different right.

But now assume this particular this control signal is glitchy right. So, control signal is glitchy means what. So, this is coming from some comb long combinational input and that that inputs are may be unbalanced delay, because of there may be lot of glitch is happening here right. So, this might happen to this particular control signal.

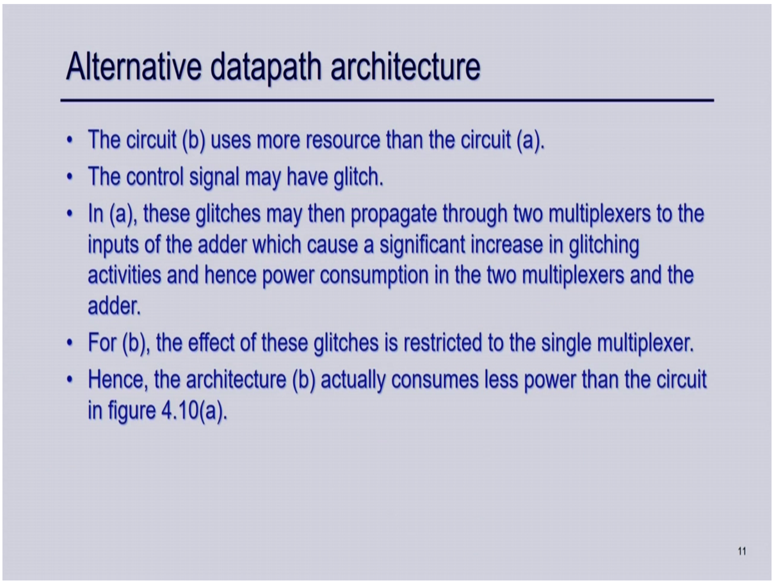
So, now this control signal is from the control path is moved to the data path. This is your data path and this is moving through this data path. So, since there are some glitch here though even though this input are stable because of there is glitch here the output of the this multiplexer have some glitch what does it mean? It is suppose to select a, but for a movement it will selects c, similarly it supposed to select b it momentarily it will selects d and vice versa. So, there will be glitch at this point at this point similarly at this point as well.

Because for a momentary the output will be different, but finally, it will be settle down. So, this is what is so; that means, this control glitch and the control signals move through this propagated to the output through a long chain, and it will create lot of glitch here, glitch here, glitch here, glitch here, glitch here right on the other hand here this particular glitchy signal is still there, I and it only effect the data flow between this I have only this glitch here, I do not have any glitch here I do not have any glitch here. So, that means, I try to redefine my architecture I try to restructure my data path.

So, that I can move my control signal as close to the output possible so that I the effect of the glitch in the control signal does not go into more into the data path, and as a result the total number of glitch in the resultant side will be less and hence the total power consumption will also have a is less right.

This is just an example based on your based on your design, you might have a different scenario and you have to consider a different scenario all together, but this is just to illustrate a fact that some time even with the mode resource you can have achieved less power right. So, this is one alternative transition, we talked about which also achieve less power ok.

(Refer Slide Time: 19:36)



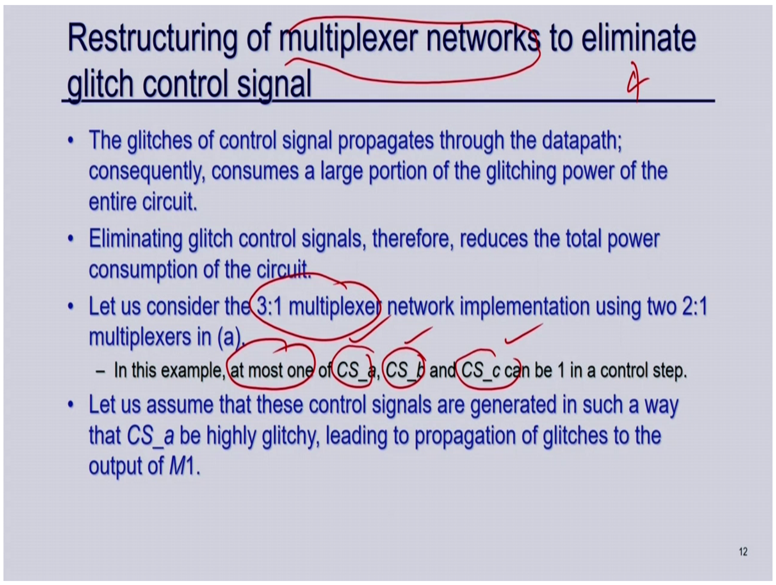
Alternative datapath architecture

- The circuit (b) uses more resource than the circuit (a).
- The control signal may have glitch.
- In (a), these glitches may then propagate through two multiplexers to the inputs of the adder which cause a significant increase in glitching activities and hence power consumption in the two multiplexers and the adder.
- For (b), the effect of these glitches is restricted to the single multiplexer.
- Hence, the architecture (b) actually consumes less power than the circuit in figure 4.10(a).

11

So, we will move to the next technique which is called restructuring the multiplexer network just to eliminate glitchy signal right.

(Refer Slide Time: 19:38)



Restructuring of multiplexer networks to eliminate glitch control signal

- The glitches of control signal propagates through the datapath; consequently, consumes a large portion of the glitching power of the entire circuit.
- Eliminating glitch control signals, therefore, reduces the total power consumption of the circuit.
- Let us consider the 3:1 multiplexer network implementation using two 2:1 multiplexers in (a).
  - In this example, at most one of *CS_a*, *CS_b* and *CS_c* can be 1 in a control step.
- Let us assume that these control signals are generated in such a way that *CS_a* be highly glitchy, leading to propagation of glitches to the output of *M*1.

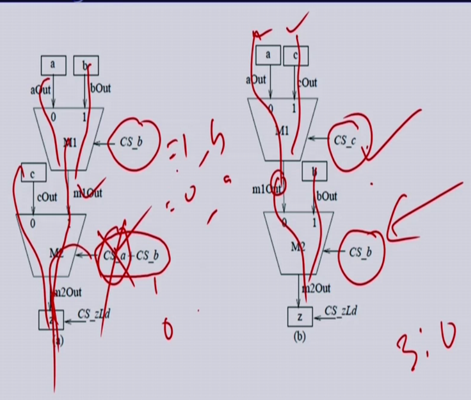
12

So, as I mentioned in the our previous example also this control signal is glitch, the CS m is glitchy because of that it is propagated to data path and we have lot of glitch in the data path as well. So, what we can do actually if there are such kind of control signals are there, it might be a situation where you actually possible to remove that particular control signal all together and instead of that use some other control signal which is not glitchy at all right.

So, here is an example suppose I want to implement a 3 is to 1 multiplexer right and I have three control signals CS a b and c. So, this is selecting between a b c.

(Refer Slide Time: 20:25)

Restructuring of multiplexer networks to eliminate glitch control signal



13

I input a b c I am going to select one of them. And I have 3 control signals CS a CS b and CS c which at a time at most one of them will 1 because at I want to select one of them. So, if CS a is 1, I am going to select a, if CS b is 1 I am going to select b, if CS c is 1 then I am going to select c. So, at most one of them will be one; that means, I am going to select one of them. So, this is what is multiplex in behavior right.

So, now I want to implement this particular circuit using 2 is to 1 multiply multiplexer right. So, this is 1 I have doing this. So, suppose I am using CS b to select between a and b. So, if it is 1, b will come here if it is 0, then a will come here right. So, it will select between a and b, now I can use either c here right. So, if I use c I can use a or b right CS a or b, if one of them is true then this data will come here.

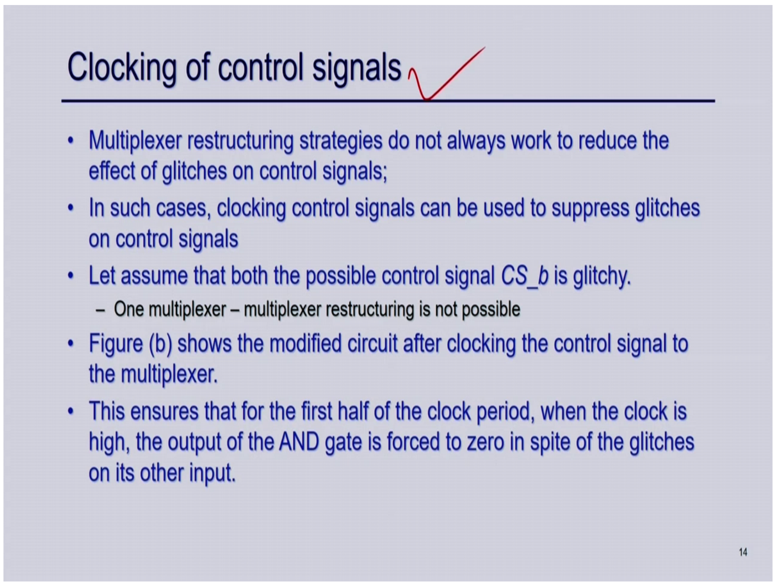
So; that means, a or b will come here while it is already decided a or b here at this point or if it is 0 then c will come here right. So, this what it is the purpose of the circuit right this is what the implementation is. On the other hand let us assume that this CS b CS a is a glitchy control signal right. So, CS a is a glitchy control signal so; that means, it is now propagated through the data path right, but you can have it another possible because this I know that CS b and c and a are only at most one of them is 1 at the point. So, I can actually remove this I can use it c s underscore c to use instead of a right.

So, what I have done here I used c s underscore c just to select between a and c, if it is 1 then c will come if it is 0 then a will come and now I can use CS b, if it is 1 then b will come and if it is 0 then either a or c wherever selected here based on this volume will come here.

So, still this will do the 3 is to 1 multiplexer behavior both of them are doing the same thing, the advantage here is there is no CS a which is a glitchy signal. So, again I will remove glitchy signal. So, that the total toggling of the circuit will be less and hence I can achieve less power.

Again just this is an example very simple example just to illustrate the fact, but in real circuit this can be a challenge to identify this glitchy signal and restructuring the multiplexer right, but this is always possible this is just a give a pointer that you can actually think about this kind of optimization just to remove your glitchy control signal from your design I move on.

(Refer Slide Time: 22:56)



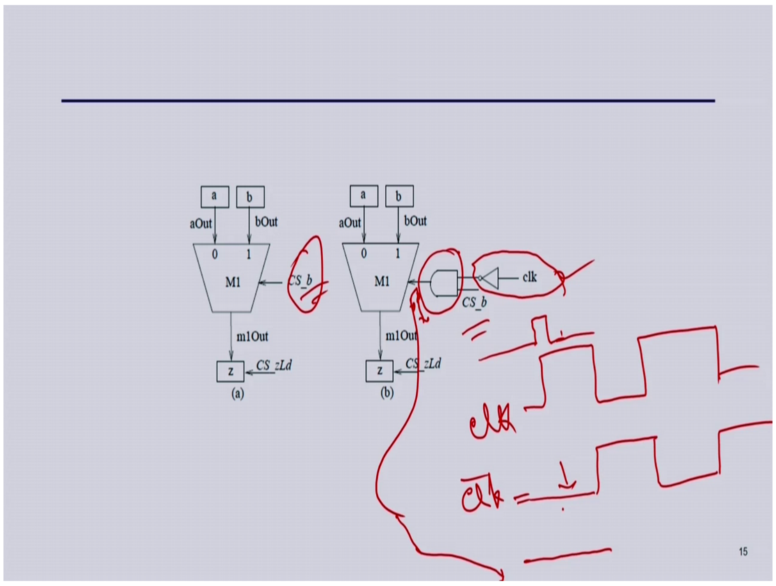
Clocking of control signals

- Multiplexer restructuring strategies do not always work to reduce the effect of glitches on control signals;
- In such cases, clocking control signals can be used to suppress glitches on control signals
- Let assume that both the possible control signal *CS_b* is glitchy.
  - One multiplexer – multiplexer restructuring is not possible
- Figure (b) shows the modified circuit after clocking the control signal to the multiplexer.
- This ensures that for the first half of the clock period, when the clock is high, the output of the AND gate is forced to zero in spite of the glitches on its other input.

14

So the next one I will talk about the clocking of control signal right.

(Refer Slide Time: 23:01)



So, far what we have done here I tried to remove this CS a because that is a glitchy signal, but there may be scenario where cannot remove right for example, here this is a very single marks and this is glitchy right we cannot remove it, because we need a marks. So, what we have to do here. So, what we can do here I can actually negate the clock and I can make and gate and I just and this CS b, the same control signal with the negate clock negation of the clock and that will actually very helpful just to remove glitches.

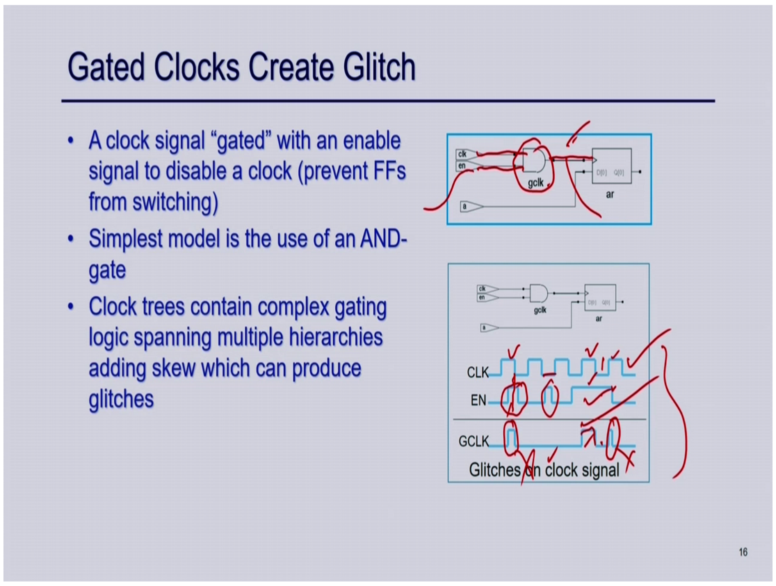
Because what will happen here. So, you try to understand the glitch happened at start of the clock right, because whenever the data changes may be some data is arrived some is not arrived. So, there will be some moments that is glitch and then it will be settle down. So, whenever you have a use negative edge of the clock so; that means, whenever this clock is 1 right this clock is 1.

So, at this moment this will be 0 right. So, this clock bar will be this is clock this is clock bar and this will be 0 and then this is 1 right this is just opposite of this. So, at this moment since this is 0, even if there are some glitch here in the CS b that will not propagate to this signal right. So, if you just draw it I just draw it clearly you can understand that.

So, suppose this is my signal there is some glitch here. So, this is my CS b right this is actually 0, but there is a glitch here. So, what will happen? Since this is 0 the output will be here. So, it will effectively no glitch at this point if this is a this is that signal right. So, even if there is a glitch that will not propagated to this data path.

So, I can actually eliminate this kind of glitch using this just clocking the control the clock, the negation of the clock and ending with that clock b and give as feeder of the select line of the mark. So, that this glitch will be eliminated by the negation of the clock right so, this is another technique just to remove glitches, again and hence it will also reduce the power of your power consumption of your design right.

(Refer Slide Time: 25:16)



Gated Clocks Create Glitch

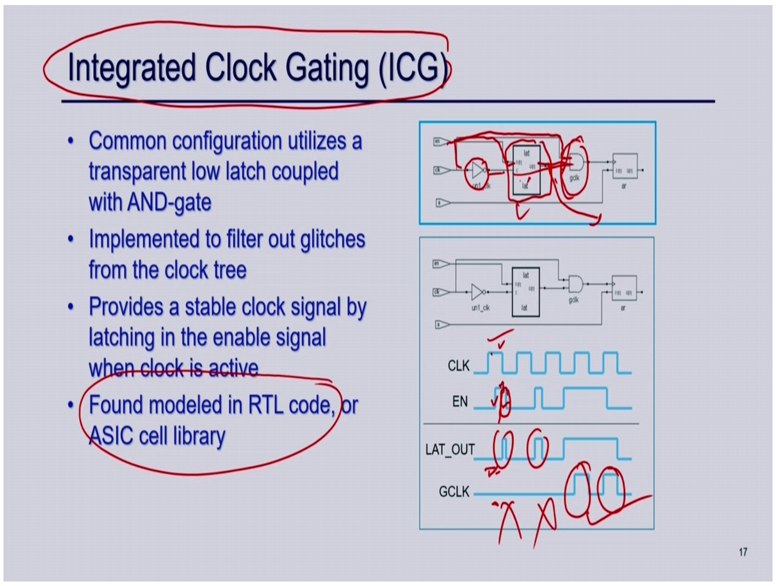
- A clock signal "gated" with an enable signal to disable a clock (prevent FFs from switching)
- Simplest model is the use of an AND-gate
- Clock trees contain complex gating logic spanning multiple hierarchies adding skew which can produce glitches

clk
en
gclk
ar
CLK
EN
GCLK
Glitches on clock signal

16

So, this is what is another technique. So I will move on and I will talk about the another technique which is called integrated clock gating.

(Refer Slide Time: 25:26)



Integrated Clock Gating (ICG)

- Common configuration utilizes a transparent low latch coupled with AND-gate
- Implemented to filter out glitches from the clock tree
- Provides a stable clock signal by latching in the enable signal when clock is active
- Found modeled in RTL code, or ASIC cell library

lat
gclk
ar
CLK
EN
LAT_OUT
GCLK

17

So, before that let us introduce the fact. So, what is happening here? So, even gated clock create glitch in clocks clock signal right. So, why? Because earlier clock was directly connected to this is. So, this is an example of clock gating right you can see here this my clock and this is ended with the enables clock.

So, this is the gated clock and then this is feeded to the clock of the design right. So, this is a clock gating. So, now, may be this enable is may be glitch, because I do not know this is enable may be glitchy and that glitch will propagated to this output of the clock and hence the registers right. So, how we can prevent that? So, you can the example is written here. So, this is my clock and this is my enable you can see here this is something the glitches right and this is the actual data one.

So, what is happening here? Since this is also the clock is 1 here and this part of the clock this part of the clock is one. So, glitch will create here. So, for example, here so, this is this is something this is 0 and there is glitch. So, it will not be propagated because this is ended here, but here what is happening here is this is. So, this is sig signal because this is 1 this is 1 the output will be 1 this is 0 this is 1, but the output will be 0. So, this is the output signal this is the clock.

But here this is 1 and this is also 1. So, there is this momentary glitch. So, which are the redundant things? So, I the wanted things only this is redundant this is redundant. So, this glitch will be created in the in the clock path right. So, gated clock will create that. So, how to avoid that right? So, that is avoided by integrated clock gating; what is integrated clock gating? So, let us look into this diagram it will be clear. So, earlier I used and gate just to put and I put clock and then I right. So, that is what I have done here right and gate clock enable.

So, what I am doing here I just put a latch here I just put a latch here and what I am doing here is enable is the input to this data and this is the enable output is going to this clock and the clock is directly connected to this gated clock right and this negation of the clock is going as a clock to this latch right and this is really helpful just to remove this glitches. Because what is happening here since this is a latch whenever this is 1 the clock is 1 whatever the data that will updated when it is 0 it will not change right.

So, this is how it will work. So, suppose this your clock and this is those glitches are there and what will happen here. So, since this is clock 1 and this is negation of clock. So, nothing so, this there is a glitch here right you can see here. So, for this part so, this is clock 1; that means, clock bar is 0 so that will not be propagated in the latch out, but this part will be propagated right.

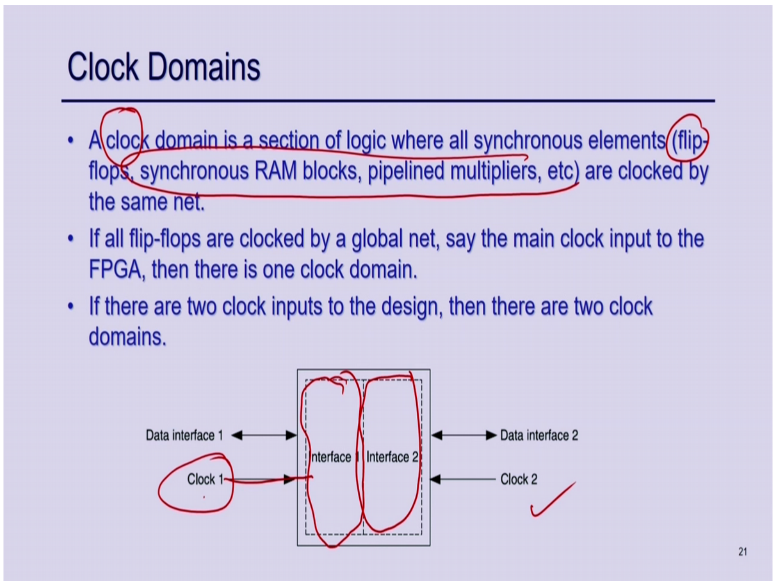
So, this is the latch part right. So, this will come; at the latch out at this point, but at this point clock is 0 right. So, this glitch will not affect eliminate out at this point. So, what is happening here; so, whatever the glitch is in the negative part of the clock that will be eliminated by the latch and whatever the positive part of the clock sorry.

Negative part of the clock will eliminate in the and gate itself and whatever in the positive part of the clock that will be eliminated at the negative in the in the lat latch right because the negation of the clock is using here I mean I am just illustrating it again you can understand. So, there is a glitch here which is in spanning over some part of the positive clock and the negative clock right.

So, since this is clock one. So, and this is negation of the clock. So, your clock bar. So, this part of the glitch will not be propagated to the latch out right because this will be eliminated by the latch, but this part will not be eliminated by the latch because at this point latch is 0. So, this will be propagated, but in this that will be eliminated by the and gate because at this point clock is low. So, this will not propagated to the actual gated right.

So, effectively I am going to get the correct percent all this spurious things will be eliminate right. So, this is what is called integrated gating clock gating, and what is happening here I just use a latch at the in the clock, and I just use a negation of the clock at the clock of this latch, and I put the enable through the latch to the and gate right. So, through this I can actually eliminate those glitch in the control signal as well right and this is also very well used technique for ASIC library just to element bleach in the control signal this is also one of the technique right.

(Refer Slide Time: 30:11)



Clock Domains

- A clock domain is a section of logic where all synchronous elements (flip-flops, synchronous RAM blocks, pipelined multipliers, etc) are clocked by the same net.
- If all flip-flops are clocked by a global net, say the main clock input to the FPGA, then there is one clock domain.
- If there are two clock inputs to the design, then there are two clock domains.

Data interface 1
Clock 1
Interface 1
Interface 2
Data interface 2
Clock 2

21

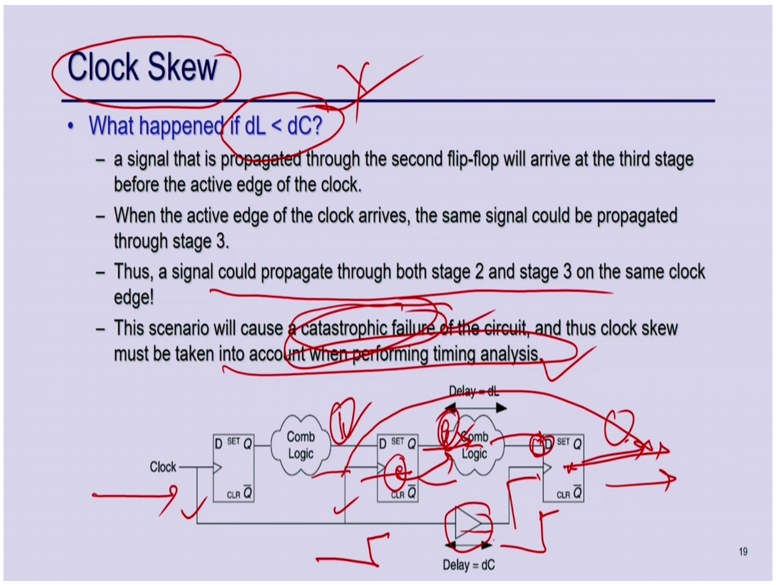
So, far we have discuss various technique right. So, what we have discussed? We have discussed about this clock gating, we have discuss about this lowering the voltage dwell estimated flip flops, then I have talked about different kind of optimization technique like RTL optimization technique restructure in multiple multiplexer network or you choose a different architecture or you clock the control signal or you do the integrated clock gating.

So, these are the all techniques that are discussed and those are actually used for use to reduce the power consumption of your circuit right. So, I will now discuss about this clock domains and then again I will come back to this clock gating and specifically for FPGA architecture, what is the problem of this clock gating in FPGA architecture and how it is avoided by FPGA synthesis tools that at the end of this lecture, but for the time being we are going to discuss about the clock, and different clock domains.

When you have multiple clocks what is the clock domains, what are the issue we face because of this multiple clocks and approach design. And remember clock is very important part of digital circuit and whenever you have multiple clocks you are using you have to be very very sincere, very very careful designing your design circuits.

Otherwise what will happen your circuit will become metastable, and you have a inconsistency (Refer Time: 31:35) of your circuit right. So, we will talk about this clocks and clock domains now.

(Refer Slide Time: 31:38)



Clock Skew

- What happened if dL < dC?
  - a signal that is propagated through the second flip-flop will arrive at the third stage before the active edge of the clock.
  - When the active edge of the clock arrives, the same signal could be propagated through stage 3.
  - Thus, a signal could propagate through both stage 2 and stage 3 on the same clock edge!
  - This scenario will cause a catastrophic failure of the circuit, and thus clock skew must be taken into account when performing timing analysis.

So, move on. So, when you are talking about clock; that means, clock is something your driving your digital circuit right. So, it actually driving all the registers and most of the register or flip flop is estigate whenever there is a positive ways, your circuit will struggle right your circuit state will change right. Now, we will talk about what is clocks cube clocks cube right what are the issues.

Because primarily we are going to talk about the, what are the issues with clock you should be careful, but first thing we are going to talk about the clocks cube right what is clock cube? Is something the same clock is feeding to all the register of your circuit right. Suppose your circuit has only 1 clock not multiple clocks, but our expectation of asynchronous circuit is that that particular clock will reach all the register at the same time right.

If it is happen there is no problem, but in reality that may not happen right because of the some delay as we discussed about this clock gating or and even the clock path has to be reroute through some through some reroute right so; that means, the it may not happen that that all clock actually reach through all the register at the same point of time, there might be some delay. And let us assume a scenario here and that is what is called clocks cube right.

So, whenever it is the difference between that register I mean clock that is achieving I mean arriving to particular flip flop, the difference in time between them right. So, that is what is called clocks cube and what have what might happen practically, suppose there are 2 register 3 registers in my design and this 2 clocks are actually coming simultaneously there is no problem right. So, there is no problem in here, because I am they are coming in the same time and.

But suppose there is a scenario there are some delay here and; obviously, there are some combinational circuit between 2 registers and this delay of this combinational circuit is d L and delay of this clock is d C and d L is less than d C right; that means, the combinational circuit delay of this combinational circuit is less than of the dealing the clock path; that means, what this data arrive early, then this clock arrives right.

So, that is something is the problem and that is very serious issues because what will happen because of that that the same propagated through the stage 2 and stage 3 at the same age of the clock which is very very serious issue and have a catastrophic failure of the circuit right.

Where I will explain that what is happening here suppose since. So, suppose whatever the clock is coming here. So, suppose I am talking about the clock 2 right. So, clock two. So, in usually what will happen in clock 2 whatever the data here that will come here and whatever the data of this register that will go out right in clock 2. In clock 3 whatever the data of this will come out here, and whatever the data that of this clock 2 that is feed here and that that will come out of this register now right.

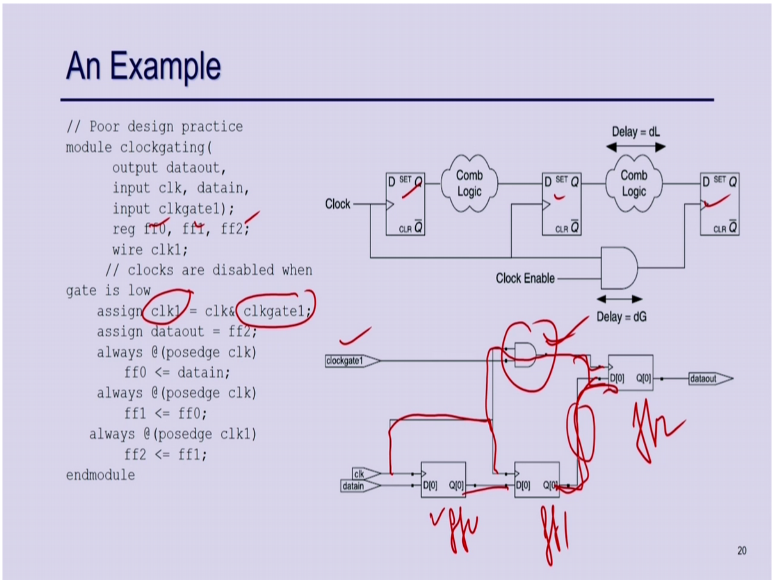
So; that means, 1 the whatever the data is coming at input in after 1 latency 2 latency 3 and go to the output right so; that means, the same data is not. So, if you have 1 data here 1 that will not be propagated here as well as the here right because that is something is not allowed. So, whatever the data here it will come in 1 in next clock here then it will come here right, but because of this problem this d L less than d C that may not happen right what will happen here? Suppose I am talking about say clock k or clock i.

So, the data whatever data here it will be available here and since that data the clock is not still reach here that clock is coming little bit late. So, this data is already processed and ready here. So, whenever this clock has come here the same clock whatever the clock here because of that data comes here so that particular. So, as I discussing that because of this delay in the clock whatever the data is process here that is that will arrive at this point and by the time this clock receive here reach here that is ready at the input of this clock and that will fade out of this register right.

So, this is something; that means, the same data whatever is here is ready here which is something it is not expected for digital circuit and that is something called a serious problem to the circuit right. So, this is something has to be handled account by the timing analysis right. So, when you are doing the timing analysis that has to be taken care if you have a clock really have a clocks cube that has to be taken care otherwise this kind of problem will happen right.

So this is the problem with clocks cube and that has to be taken must be taken care during the timing analysis of the design ok.

(Refer Slide Time: 36:21)



An Example

```
// Poor design practice
module clockgating(
      output dataout,
      input clk, datain,
      input clkgate1);
      reg ff0, ff1, ff2;
      wire clk1;
     // clocks are disabled when
gate is low
    assign clk1 = clk& clkgate1;
    assign dataout = ff2;
    always @(posedge clk)
       ff0 <= datain;
    always @(posedge clk)
       ff1 <= ff0;
   always @(posedge clk1)
       ff2 <= ff1;
endmodule
```

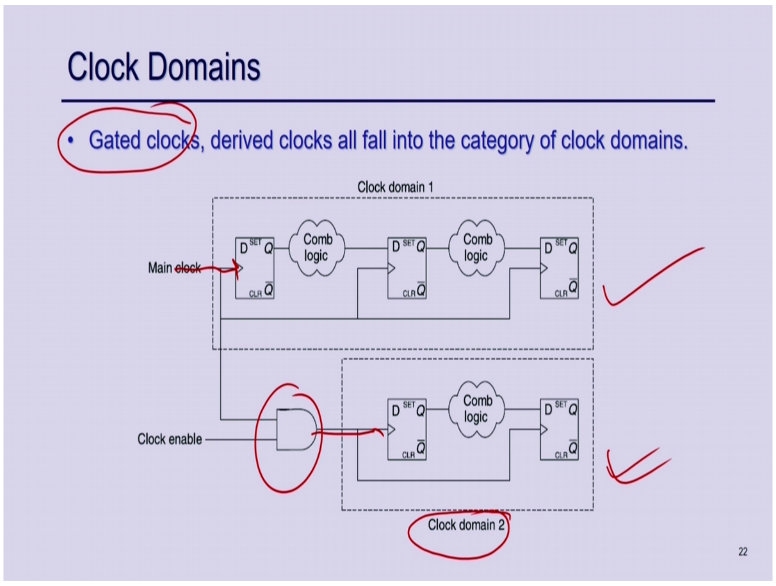
So, it is an example that why this is happening here. So, I have 3 set of register for 0 f 1 and f 2 this is f 0 0, this is f 1 and f 2 and you can see here this clock 1 is ended with a clock gate 1 right. So, this is clock gate 1 is a signal which is ended here. So, this is f f 0, this is your f f 0, this f f 1 this is your f f 2 and you can see there is no delay here there is no delay here, but there is a delay here in the clock path. So, this is your clock it is coming here and it will go through this and gate and this will register, but this data will come immediately here right.

So; that means, there is a delay in the clock path and there is no delay in the combinational path. So, there is; that means, is actually depilating this scenario in this scenario you might reach to this problem right. So, this is one issue we will talk about how to avoid this kind of thing. So, basically just to avoid this week have to the timing analysis we have to add some kind of constant. So, that this clock pure is such that this may not happen right. So, that you can actually avoid this scenario right and now we will talk about the clock domains.

So, as I mentioned you might have to use multiple clocks in your design right. So, clock domain is what? The set of registers flip flops and this pipeline multipliers those are actually driven by a clock right. So, all the registers of a particular portion of the design driven by a clock so, that is a domain of this clock right.

So, in this particular scenario I have 2 clocks right. So, this domain is driven by this clock 1 this, but is driven by the clock 2 right.

(Refer Slide Time: 38:06)

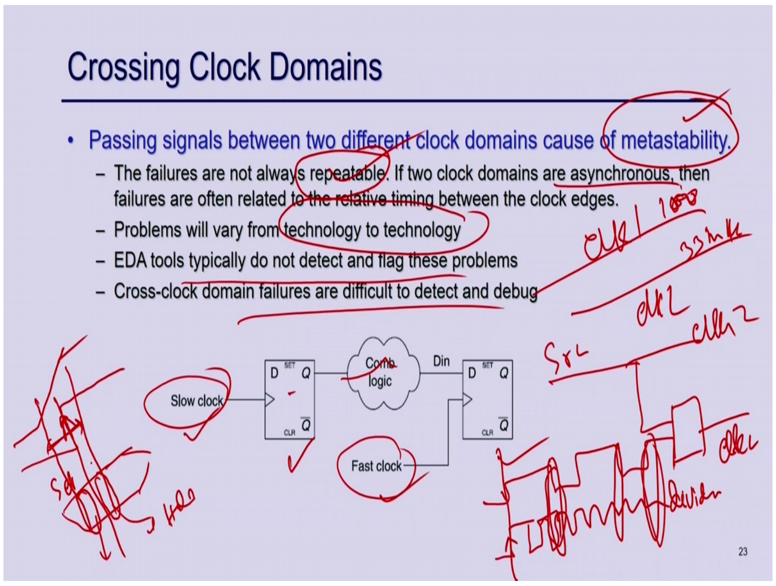


Clock Domains

- Gated clocks, derived clocks all fall into the category of clock domains.

And so, if a multiple clocks is fine but the problem is that whenever you are.

(Refer Slide Time: 38:09)



Crossing Clock Domains

- Passing signals between two different clock domains cause of metastability.
  - The failures are not always repeatable. If two clock domains are asynchronous, then failures are often related to the relative timing between the clock edges.
  - Problems will vary from technology to technology
  - EDA tools typically do not detect and flag these problems
  - Cross-clock domain failures are difficult to detect and debug

When you are actually passing a data from 1 domain to another domain, that will create a problem. So, just to specifically say that even the gated clock actually create a different clock domain right. So, since this is the main clock it is coming to this particular clock and this is gated here and go as a clock to this, that synthesis tool as in this is a different clock domain right. So, this is clock domain 1 this is clock domain 2.

So, even this gated clock or derived clock will actually create different clock domains right. So, the problem will happen here in the clock domain is something, whenever you are passing a data from 1 clock domain to another clock domain. So, when you have multiple clock domains there must be some communication between the clock domain otherwise we do not need that right.

So, if there two things are totally independent there is no problem, but most likely whatever the data is computing here that is going to be used in the next clock domain right. So, that is what is happening here. So, this is running in slower clock, this is running in faster clock and you are doing some operations and updating them and now what will happen here right. And this is this unless you have taken some results, that will actually create a metastability in your design and which is serious issue.

And the problem with this metastability of this kind of circuit is this kind of failure is not repeatable right it may be that if then it is not repeatable in the sense that it may be that if your target technology is and because this it may varies technology to technology right because it may be happen that in some particular FPGA architecture it is not repeatable, it is not happening because of the delay of that particular clock or the devices and the some parameters, but in a very faster architecture this might arise right and also even when this particular this 2 clock are really asynchronous what does it mean by asynchronous? They are actually have a different sources right.

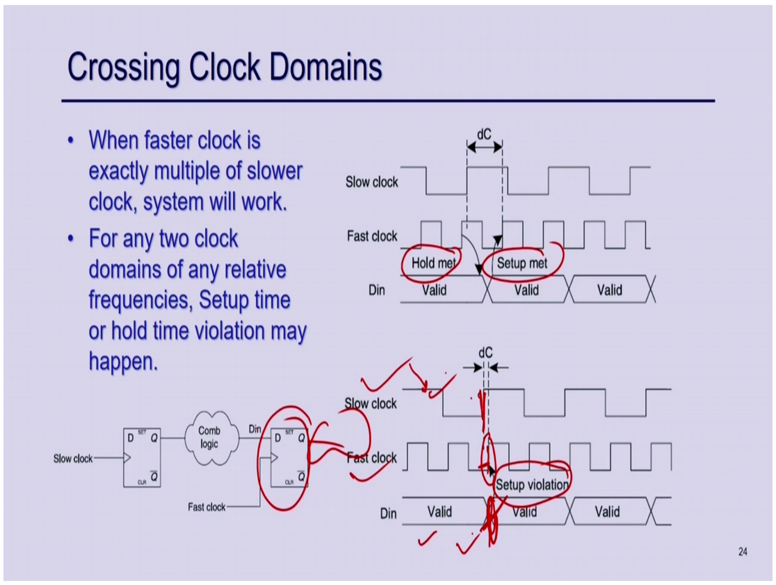
They are not actually related, this clock 2 are not at all related. So, what is synchronous may be they are 2 clocks they may be actually generating from the same clock source and there may be some clock divider or something and then this is your clock 1 say this is a divider circuit and this is your clock 2 right.

So, that is a source is 1 and you have 2 clocks. So, that that is a kind of synchronous clock though although I mean they have 2 different sources, but even specifically if this particular and there are 2 different clock right they are 2 asynchronous this is your clock 1, and this is your clock 2 and this may be say 100 megahertz and this is say 100 megahertz and this is say 33 megahertz they are not related at all right. So, they are not their face is also not aliened also right. So, this kind of in this scenario, you never know how exactly their there this rising is really aliened.

So, if they are aliened there is no problem right. So, this is a faster clock this may be like this and may be some scenario this may a line in some scenario they may not a line some scenario, they may not a line and something so; that means, are not repeatable in some scenario it may occur some scenario it may not occur right and the much problem is that this EDA tool does not typically do not detect this kind of problem right. And effective they are actually very difficult to detect and debug this kind of clock domain processing problem and because of this metastability happening due to clock domain processing usually do not flagged by the synthesis tool and even I mean even the simulation tool does not give you any hint.

So, and they are not repeatable they may not happen sometime, and even for technology to technology they might may happen or may not happen. So, this is very serious issues and this has to be handled with at most care when you are designing your circuit right. So, we will talk about that.

(Refer Slide Time: 41:59)



Crossing Clock Domains

- When faster clock is exactly multiple of slower clock, system will work.
- For any two clock domains of any relative frequencies, Setup time or hold time violation may happen.

So, whenever. So, there is a very specific example here. So, what is happening here why this metastability happen right. So, that is that is a question I am going to ask you right the problem is that whenever we know the concept of hold time and setup time. So, whenever the basic concept is whenever the clock arrive right though data must be ready at least before sometime right. So, it must be ready.

So, this is your at least sometimes before that clock comes it should be ready. So, this is setup time and similarly before after the clock at least sometime it that it should hold that. So, which is called hold time so that you have a chance to that register content get updated right. So, this is called setup time and this is called hold time right this is hold and this is say time, and in if the data toggles in this pure right. So, this is at this is the boundary at this boundary even the clock comes at this point. So, this is the point your data state does not change right, but if your data state changes here then what will happen?

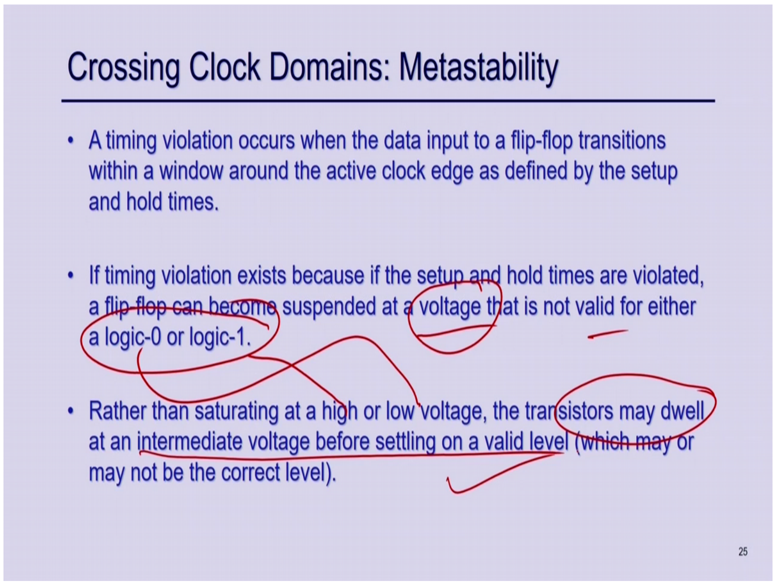
It changes from 0 to 1 or 1 to 0 and that is called the metastability and that is happened when you have some data is going from a slower clock to faster clock. Because since there between this clock their edges are not align their edges are not aligned. So, whatever the data coming here is may not sync with this particular clock right because of that this kind of problem will happen and that is actually the problem will happen here is something right.

For example, here this is your slow clock this is your fast clock you can see this is there is a change. This is the d C delay in that particular clock, this is what I talked about earlier also and here this is the faster clock has come and this data is output coming from this.

So, this is related to this clock right. So, whenever this is come this data get updated. So, here from here, but it is changing here right. So, at this point whenever the data is changing the clock is also changing and that is called setup violation right. Because at this point data should not change because this data is coming from the clock 1 domain, which is slower clock and this data is dependent on this clock and this might change based on this of this positive way of this clock and at that moment may be the scenario that this clock is also changing.

So, that will actually create a hold time violation or setup violation right and that is a serious issue and because of that what will happen?

(Refer Slide Time: 44:34)



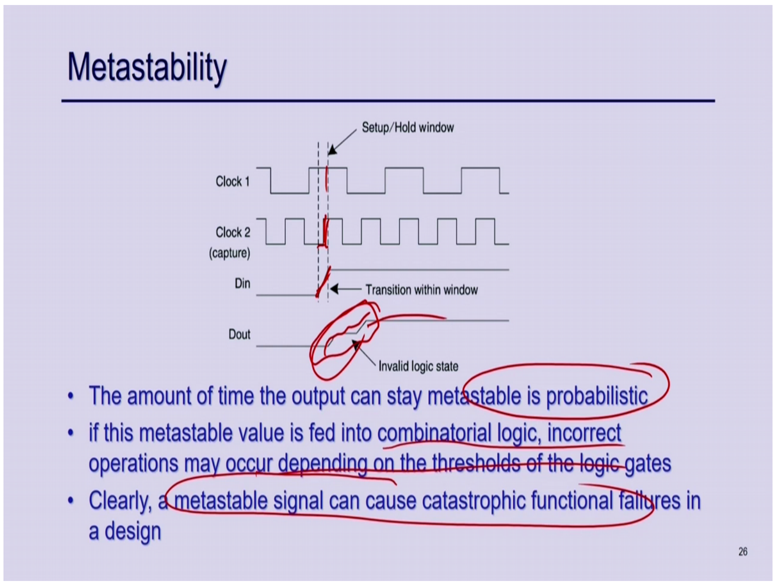
Crossing Clock Domains: Metastability

- A timing violation occurs when the data input to a flip-flop transitions within a window around the active clock edge as defined by the setup and hold times.
- If timing violation exists because if the setup and hold times are violated, a flip-flop can become suspended at a voltage that is not valid for either a logic-0 or logic-1.
- Rather than saturating at a high or low voltage, the transistors may dwell at an intermediate voltage before settling on a valid level (which may or may not be the correct level).

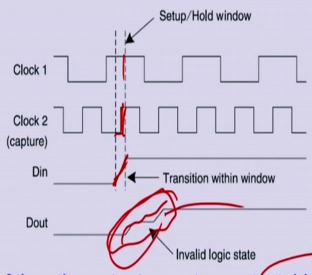
25

And because of that your data will become metastable right.

(Refer Slide Time: 44:35)



Metastability



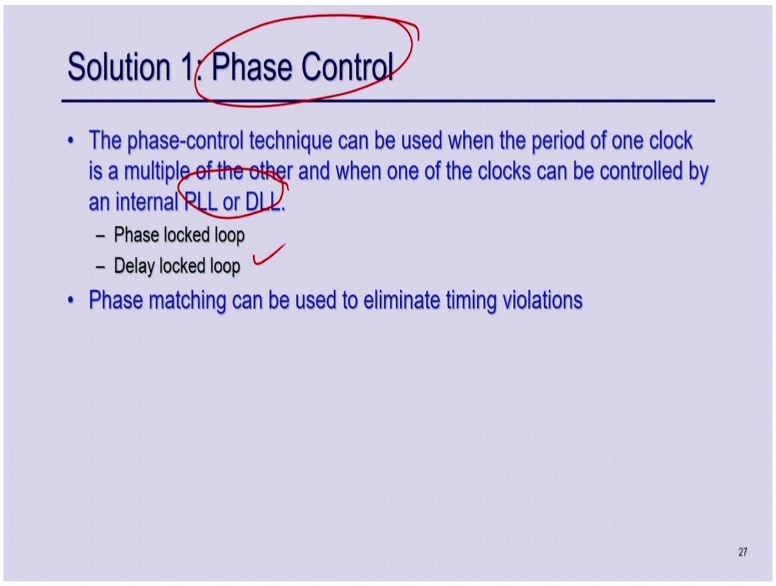
- The amount of time the output can stay metastable is probabilistic
- if this metastable value is fed into combinatorial logic, incorrect operations may occur depending on the thresholds of the logic gates
- Clearly, a metastable signal can cause catastrophic functional failures in a design

26

So, which is called unstable; this is what same example. So, at this point the clock has come and data is changing and because of that you never know what is that? So, in this state I your data neither in 0 state or logic state in digital I always assume either it is 0 or 1 right either it is 0 or it is not in between of that and this is actually 1 means high voltage high and I mean I mean by the set up suspended at a voltage which is not neither 0 or 1 right.

So, that is that is not a expected state and that is called metastablity and that actually this particular metastability will through go through this and this will go through this circuit following this right. So, it will actually go through the combinational circuit following this following these gates also right.

(Refer Slide Time: 45:22)



Solution 1: Phase Control

- The phase-control technique can be used when the period of one clock is a multiple of the other and when one of the clocks can be controlled by an internal PLL or DLL.
  - Phase locked loop
  - Delay locked loop
- Phase matching can be used to eliminate timing violations

27

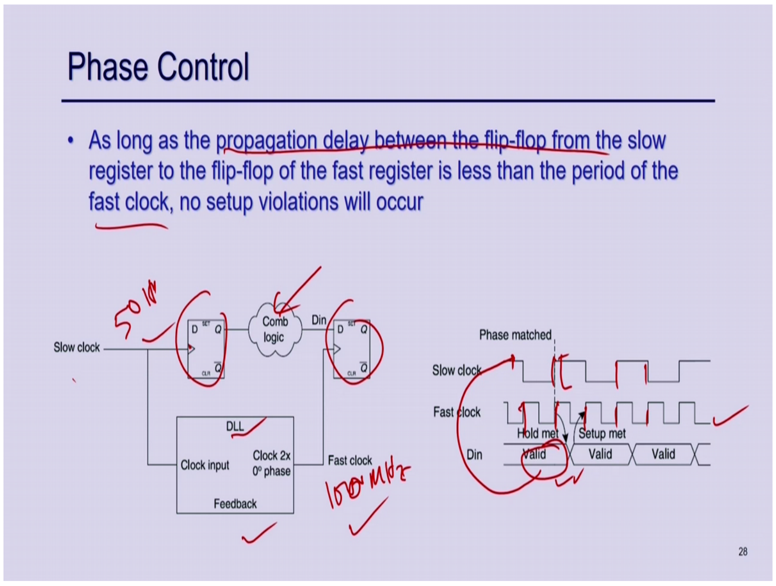
So, this metastability will propagate and may propagate to through the output right. So, because of that this is something. So, high voltage means 1 and low voltage means 0 right. So, it might be some dwelling between the intermediate voltage and this may reach to some value I do not know whatever the value will be coming here, some state after sometime right.

When the data stable it will go stable and that is what is called metastability, and this may carry to the output also and this is may be probabilistic as I mention this whenever, I do not know because this 12 clock is asynchronous, when this will come and this will come there may not after this they may sync or may not sync or the data may become a metastable or may not metastable as I mention earlier also. So, that is something probabilistic and this will be fed into logic and it will be generate a output incorrect output.

So, this is something metastable signal can be catastrophic functional failure of a design. So, this is something has to be avoided right and how we can avoid that? So, there are 3 primary techniques that has to be applied just to avoid this metastability be between when your some signal is going from 1 clock domain to another clock domain, which have a different clock cube right. So, the first technique is phase control. So, that this is applicable when this one particular clock other particular clock is multiple of one another right.

So, one is 100 megahertz and another is 50 megahertz then what we can do? We can use the PLL or DLL that means, phase locked loop or delay locked loop of the specifically for FPGA just to generate the other clock right.

(Refer Slide Time: 47:12)



Phase Control

- As long as the propagation delay between the flip-flop from the slow register to the flip-flop of the fast register is less than the period of the fast clock, no setup violations will occur

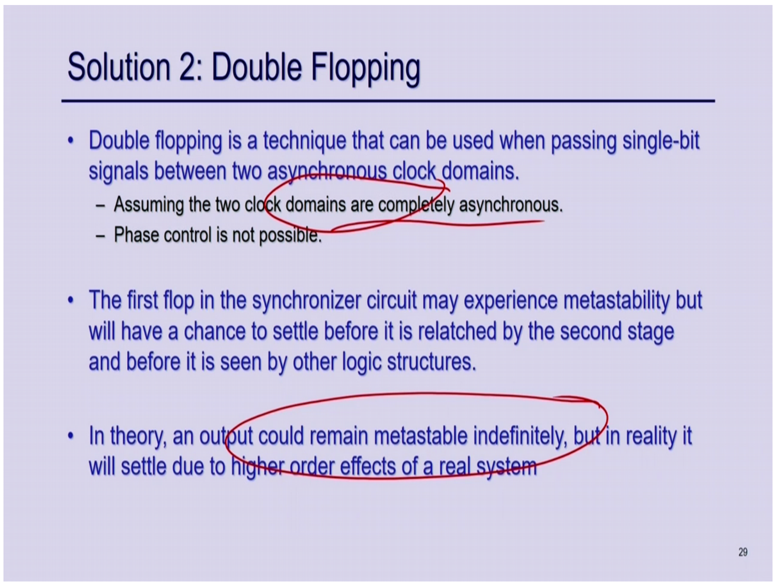
So, here is the slower clock. So, this is say 50 megahertz and what I am doing here I just use a DLL to generate the 100 megahertz clock right.

So, this is 100 megahertz clock and the advantage of having that. So, this actually aligned; because this is just twice of this clock is just twice of this right. So, ideally all this edges should match right and this DLL ensure that that it make this clock aligned. So, all this positive edge should be aligned right that is what is happening.

So, as long as the propagation delay between that flip flop this slow flip flop to the this propagation delay between this slow flip flop or the fast flip, flop is less than the clock period of the fast clock right of the 100 megahertz of this clock, you should not have any problem right; because this are always aligned.

So, the data whatever is generating will be this is this data is dependent on this particular clock and since they are always aligned. So, whatever the data change will happen some point from here and that will not violate set up time or hold time. So, this is what is applicable this phase control is using the phase PLL or DLL when the clocks are multiple of each other.

(Refer Slide Time: 48:24)



Solution 2: Double Flopping

- Double flopping is a technique that can be used when passing single-bit signals between two asynchronous clock domains.
  - Assuming the two clock domains are completely asynchronous.
  - Phase control is not possible.
- The first flop in the synchronizer circuit may experience metastability but will have a chance to settle before it is relatched by the second stage and before it is seen by other logic structures.
- In theory, an output could remain metastable indefinitely, but in reality it will settle due to higher order effects of a real system

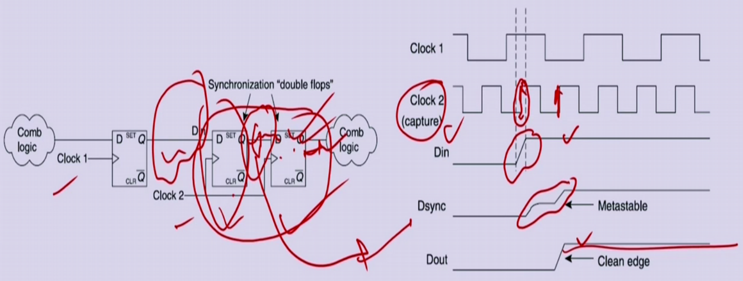
29

But if your clocks are not multiple of each other and they are completely asynchronous right.

So, they are they do not have any relation right one is 33 megahertz, one is 47 megahertz, one is 11 and one is 13 there is no relation. So, in this scenario we cannot apply the this phase control technique what we can do? I can use a double flopping right.

(Refer Slide Time: 48:45)

Solution 2: Double Flopping



When using the double-flopping method, it is important to specify timing constraints such that the signal path between the first and second clock domain is ignored during timing analysis.

30

So, the technique is called double flopping. So, again this is that clock 1, this is clock 2 and what I am doing here in the clock 2 I am actually use instead of 1 clock I mean 1 flop what is happening other scenario right. So, this your normal scenarios 1 clock.

I am now I am going to use 2 flops if the destination clock domain right. So, it may be faster or slower. So, the advantage here is that whatever the metastability happen here that will happen here, and that will not propagated at the output of this register right. So, what is explained in this register?

So, this is your capture clock; that means, clock 2 and there may be some at this whenever this positive triggers are there may be changes in data right that and that might get some metastability at this point which is this d C, but that will not be propagate it to this register because this register, where this data will come to this register in the next because this is a faster clock in the next clock right.

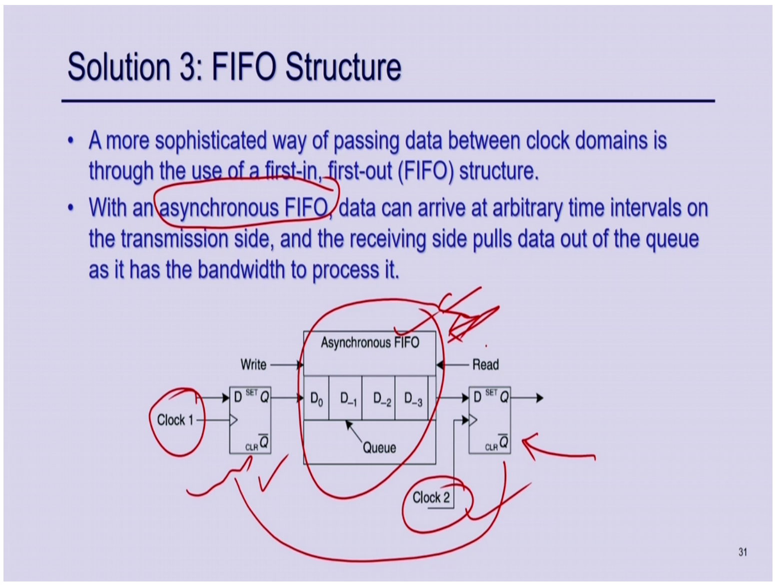
So, and until that point this data gets stable and the output will get stable right. So, at the output this metastability will not propagate it right. So, this is something simple or the most convenient or the most easy way to avoid this metastability between multiple clock domain right. So, in theory even you can prove that this metastability may remain independently or it may occur in this particular even in this scenario, because you might assume that their clock is very fast.

So, that I mean whenever this next clock come this metastability will come here and that might come here also right, but in practice that may not happen right. That may be the scenario in theory, but in practice this is very useful and when you can use this technique just to avoid metastability at the output because of clock domain processing right.

One point to be noted here whenever you have this double flopping method, you have to use the timing engine should not to do any optimization in this training because this is a redundant part and that has to be mentioned that between the first clock and the second clock domain even this timing synthesis right. So, sorry this is first clock. So, this point right. So, this point the first clock to second clock domain is ignored during timing synthesis because that is not required because that will be taken care by this technique right.

So, this is that solution 2.

(Refer Slide Time: 51:15)



Solution 3: FIFO Structure

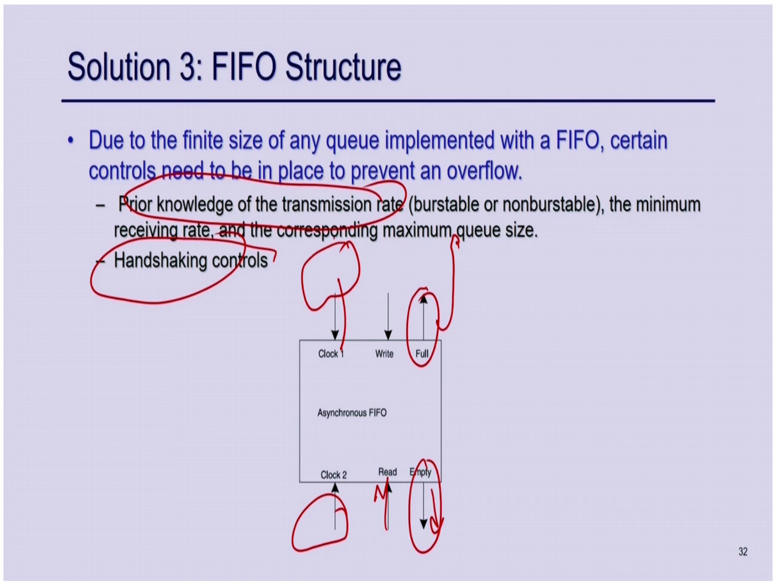
- A more sophisticated way of passing data between clock domains is through the use of a first-in, first-out (FIFO) structure.
- With an asynchronous FIFO, data can arrive at arbitrary time intervals on the transmission side, and the receiving side pulls data out of the queue as it has the bandwidth to process it.

And the third solution is the FIFO structure asynchronous FIFO which is the conventional first in first out technique right it is basically appears everywhere right. So, whenever you just think about a shopping mall, where we I mean that billing point right that is we have a queue right when; that means, the speed of the person who do the billing when his speed may be different from the frequency which the customer comes right. So, if the customer frequency is high there will be long queue if the frequency of the biller is fast there will be no queue right.

So, that is something so; that means, the frequency of the biller verses the customer coming customer is the two different domain and their frequency may be different and we have a queue first in first out and that is what is this asynchronous FIFO; that means, if you just replace the same scenario here, I have this is the customer side. So, their writing to this data and this is the biller side who actually access this data process the data one by one.

So, the way it is happening is that I mean whatever the data is coming here they will keep writing in this particular asynchronous FIFO in clock 1, and this will be reading this data in the speed of clock 2 right and there is no problem because this two are totally and there is no direct communication between this two and this two right and what is happening here? But one point has to be taken care is that what is this FIFO size, because you know that if the speed is very high and reading speed is very low this will become very long right and in hardware you have to fix the size and it may be that in the scenario that (Refer Time: 52:49) case it will cross the size of the FIFO. So, it will over flow right so; that means, it will create a problem.

(Refer Slide Time: 52:59)



Solution 3: FIFO Structure

- Due to the finite size of any queue implemented with a FIFO, certain controls need to be in place to prevent an overflow.
  - Prior knowledge of the transmission rate (burstable or nonburstable), the minimum receiving rate, and the corresponding maximum queue size.
  - Handshaking controls

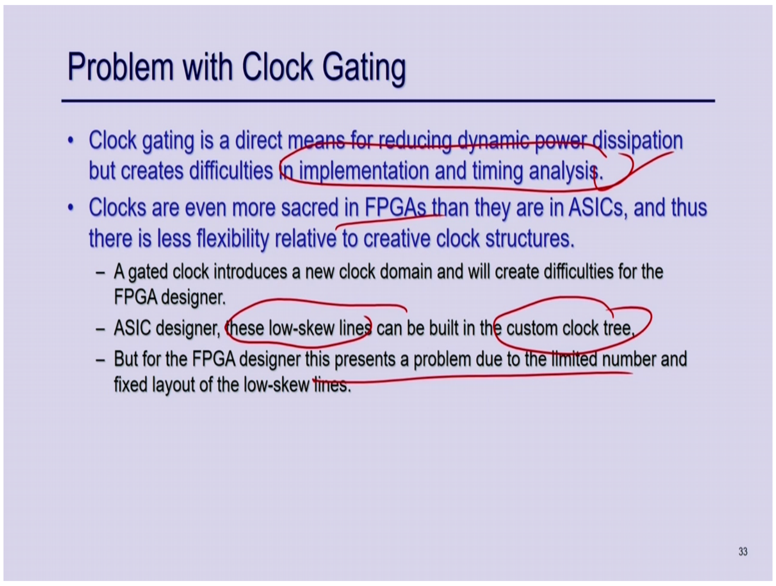
Clock 1
Write
Full
Asynchronous FIFO
Clock 2
Read
Empty
32

So, there are two techniques can be done one is something that if you know the prior knowledge of the transmission rate I know that whatever the scenario the FIFO size should not go beyond 100 or 200, then I can use those that size FIFO just to avoid this over flow right.

Similarly, the other convenient technique is the hand shaking. So, that when this is full the this process will not write anything and whenever this is empty this process will not read anything right. So, there will be some signal called full and empty. So, when this full signal coming this domain from clock, wall will not write wait to write unless this there some vacant person available.

And similarly, this empty is one; that means, it is not going to read from this clock 2 domain will not read anything from this because there is no data here. So, this is kind of handshaking through which also I mean this over flow problem can be avoided. So, this is the technique 3.

(Refer Slide Time: 55:51)



Problem with Clock Gating

- Clock gating is a direct means for reducing dynamic power dissipation but creates difficulties in implementation and timing analysis.
- Clocks are even more sacred in FPGAs than they are in ASICs, and thus there is less flexibility relative to creative clock structures.
  - A gated clock introduces a new clock domain and will create difficulties for the FPGA designer.
  - ASIC designer, these low-skew lines can be built in the custom clock tree.
  - But for the FPGA designer this presents a problem due to the limited number and fixed layout of the low-skew lines.

33

So, in a as a summary so, this clock domain crossing has a serious issue and create a metastability in your design and whenever you have a clock domain crossing; I you have to use one of this 3 technique first technique was that phase control if the clock is multiple of each other you should use PLL or DLL of your FPGA.

Just to avoid that or in general you should use double flopping for that destination clock. So, that you can avoid the metastability propagating through the output and the third technique is more generic or more worst way is this FIFO you use a FIFO between 2 different clock domain, and you do this writing completely independent of each other right. So, this is something the 3 technique and this are the issues that will faced by the clock multiple clock domain ok.

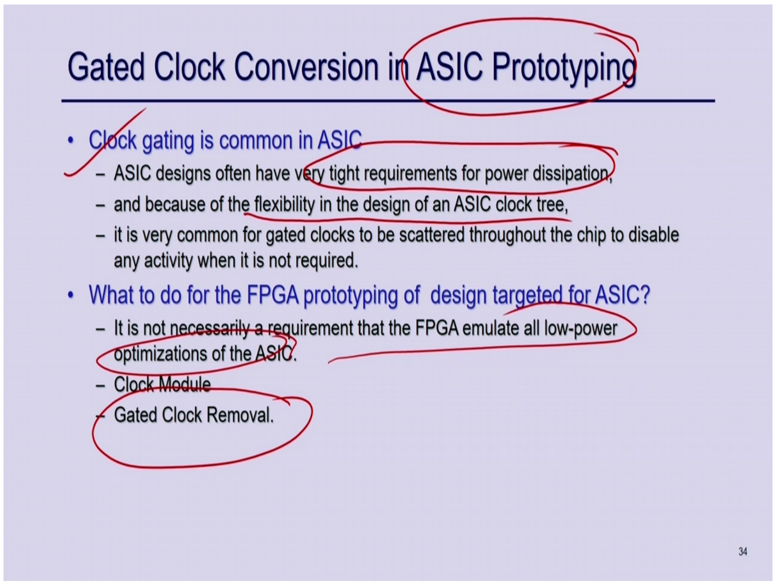
Now, I will come back again to this clock gating. So, we have talked about that clock gating is a very useful approach to reduce the power of your design and now, but the problem of clock gating is that it create a different clock domain right as I discussed earlier also and that is actually problem for the timing analysis tool and implementation specifically for FPGA in FPGA there we have only one master clock line. So, if you have multiple clock, they will be routed through the normal (Refer Time: 55:08) structure and that will create lot of delay and lot of problem.

So, multiple clock domain is not a practice for a FPGA target specifically this gated clock. So, that may not be feasible in a FPGA target right. So, on the other hand in ASIC, because they have some flexibility multiple clock tree circuits are there and we can actually apply clock gating, but it will come with some problem that we have already discuss early that that has to be taken care the timing analysis tool has to be taken care of those problem that will come due to clock gating that has to be there.

But for FPGA this is a big problem that we cannot use this clock gating I mean at all almost right. So, that and you also remember that FPGA is a very power hungry module which where we this power optimization does not do work because it is a redundant technology right.

So, that re-routable interconnection, re-programmable logic so, all are kind of redundant circuits are there. So, this power this FPGA engine is very power hungry. So, low power technique for FPGA, this kind of thing does not work. So, people who are actually working on FPGA, they do not bother much on this power issues for FPGA target.

(Refer Slide Time: 56:32)



Gated Clock Conversion in ASIC Prototyping

- Clock gating is common in ASIC
  - ASIC designs often have very tight requirements for power dissipation,
  - and because of the flexibility in the design of an ASIC clock tree,
  - it is very common for gated clocks to be scattered throughout the chip to disable any activity when it is not required.
- What to do for the FPGA prototyping of design targeted for ASIC?
  - It is not necessarily a requirement that the FPGA emulate all low-power optimizations of the ASIC.
  - Clock Module
  - Gated Clock Removal.

34

But on the other hand in FPGA this ASIC there may be very tight recommend of power dissipation right then they have to use some kind of clock gating technique just to reduce your power and also they have some flexibility of clock tree ASIC clock tree using that you can actually meet your timing or you can remove the problem that is because of the coming because of the clock gating.

So, for ASIC this clock gating is very common and this is not very common for FPGA prototyping sorry for FPGA design. So, whenever you are designing something for ASIC you might think of having a gated clock in your circuit, but when you are designing something for FPGA, you should avoid use clock gating in your design right.

But now the question is when you are actually doing the prototyping ASIC prototyping what is ASIC prototyping? So, you are have a circuit which is actually your design for FPGA and it is a very big circuit you just think about the processor which is a very big circuit before fabrication, you try to simulate your actual circuit and try to see whether this is working or not right.

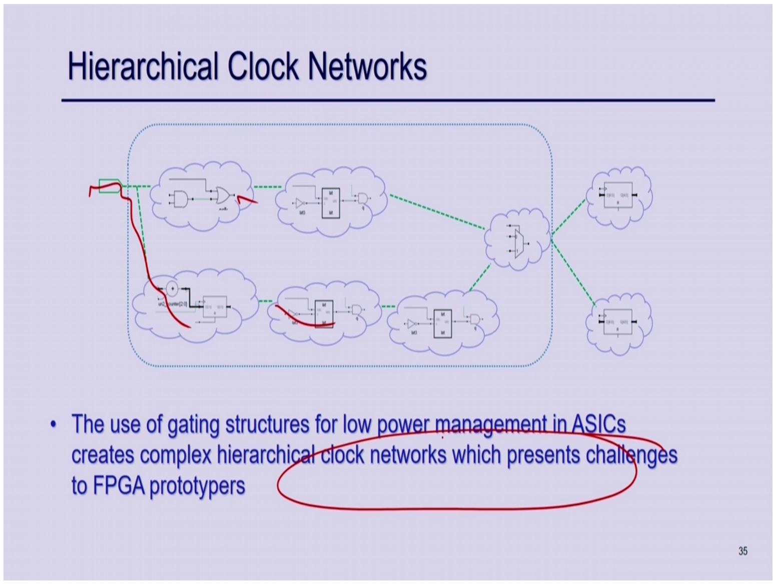
So, when that is very common practice is done by any chip provider like Intel or say other companies which actually run the whole processor in a FPGA board right. So, there where the multiple FPGA stack and stack together may be 10, 15, 16, 80 FPGA board are stack together and the whole processor there may be million line of (Refer Time: 57:59) code that the whole processor circuit is synthesis into a FPGA board right and then that is become a process.

Now you install operating system say windows liners on this top of this and you execute you play you right program, you executive some you browse internet you do whatever the that your computer can does right. So, that is something called prototyping, that you before the actual fabrication of the big circuit you dump that the whole thing in FPGA and you run it for some time just to see it is creating something problem or not right.

So, that is just give you some confidence that the circuit that you are going to fabricate has no issues right that is called ASIC prototyping. And now you think about a circuit of that that scale and it has lot of gating or something, and you just try to do it in FPGA then this clock gating will create a serious problem right you cannot map all those clocks to that to the FPGA board. So, then this FPGA synthesis tool try to do what is the most thing it try to is that it try to do the gated clock removal, we try to remove this gated clock from your circuit so that it that particular whole design can be synthesis able for RTL.

Because you cannot at this point you cannot remove the gated from your actual circuit then the whole purpose is lost right. So, that is something does by this FPGA synthesis tool like simplify flow premier from synopsis sagilings or altera synthesis tool right and it is not necessary requirement for the FPGA module that all low power optimization of ASIC will be will be met by the FPGA this synthesis. Because the primary objective is to check the function it invert the, but the low power requirement things cannot be verified using FPGA right.

(Refer Slide Time: 59:48)



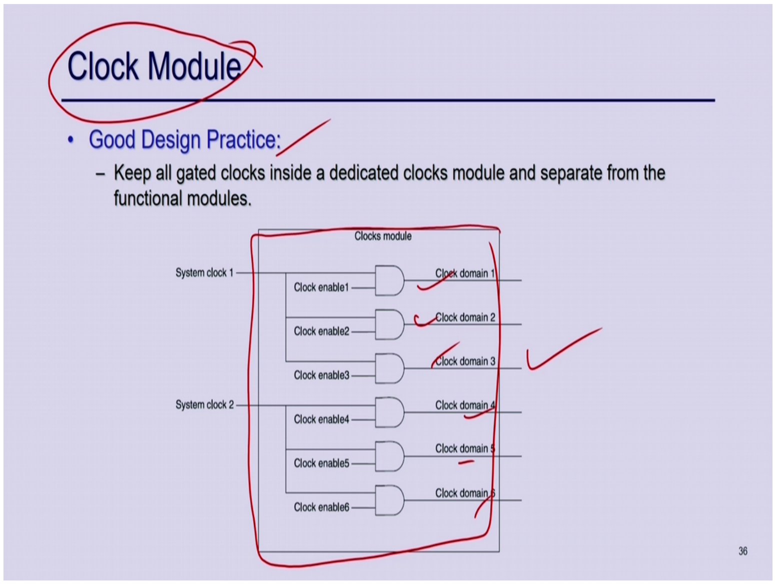
Hierarchical Clock Networks

- The use of gating structures for low power management in ASICs creates complex hierarchical clock networks which presents challenges to FPGA prototypers

35

So, those the kind of strategy you should take about that here is that problem that 1 clock may go to multiple hierarchies through multiple hierarchies multiple layer, and that will create a serious problem for the FPGA prototyping I just already explained that.

(Refer Slide Time: 59:59)



Clock Module

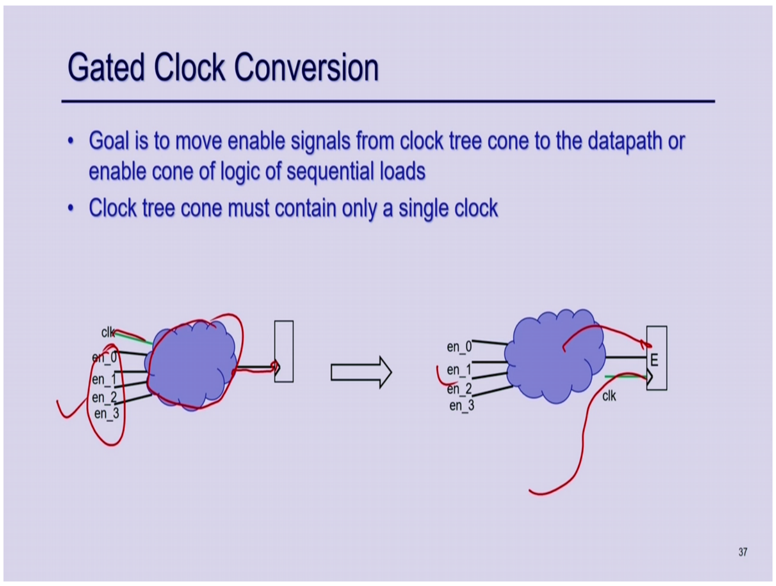
- Good Design Practice:
  - Keep all gated clocks inside a dedicated clocks module and separate from the functional modules.

36

So, the strategy you should about that when you are designing your f p this ASIC circuit, you should create a clock module. it is a good design practice that you keep your all clock in a single module whatever the. So, you create say 3 clock domains here 4 another 3 here. So, there are 6 clock domains, but you keep create all these module everything in a one module. So, that this is something whenever the FPGA synthesis is happening it is easy to detect all your clock domain easily and can be processed easily right.

So, this is kind of good design practice that you should follow when you are actually thinking about you are having a ASIC prototyping of your design FPGA prototyping of your designing right.

(Refer Slide Time: 60:38)



Gated Clock Conversion

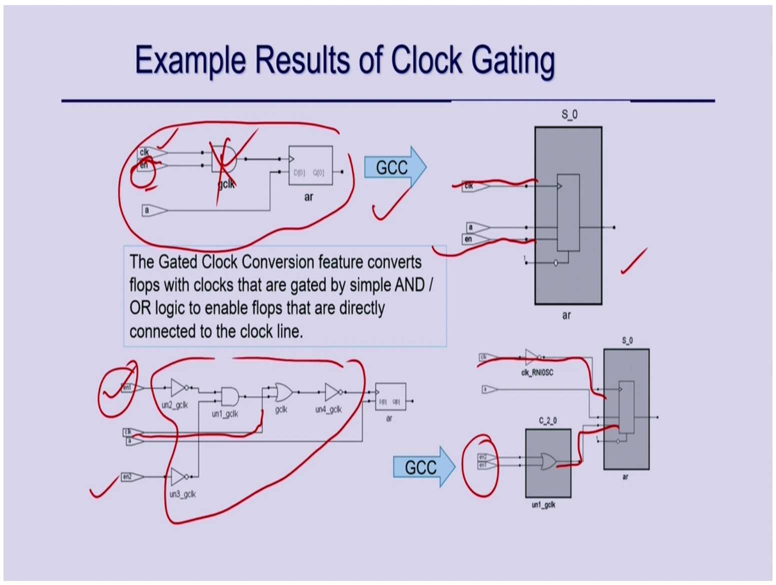
- Goal is to move enable signals from clock tree cone to the datapath or enable cone of logic of sequential loads
- Clock tree cone must contain only a single clock

clk
en_0
en_1
en_2
en_3
en_0
en_1
en_2
en_3
E
clk
37

The second thing that synthesis tool the FPGA synthesis tool that is a the gated clock conversion it is just the reverse of that; it does the reverse of gated clock. So, whenever there is a clock which is gated with several enable signal, it may be one enable or multiple enables and there is b combinational logic and that become your clock it just do the reverse it just remove out this all this enable and put them at the enable of the clock and clock is coming directly to this I mean this is just the reverse of clock gating right.

The, you just put enable to the enable signal, the all the enable logic to the enable signals right this is what is called gated clock conversion and that is related to the FPGA.

(Refer Slide Time: 61:16)



Example Results of Clock Gating

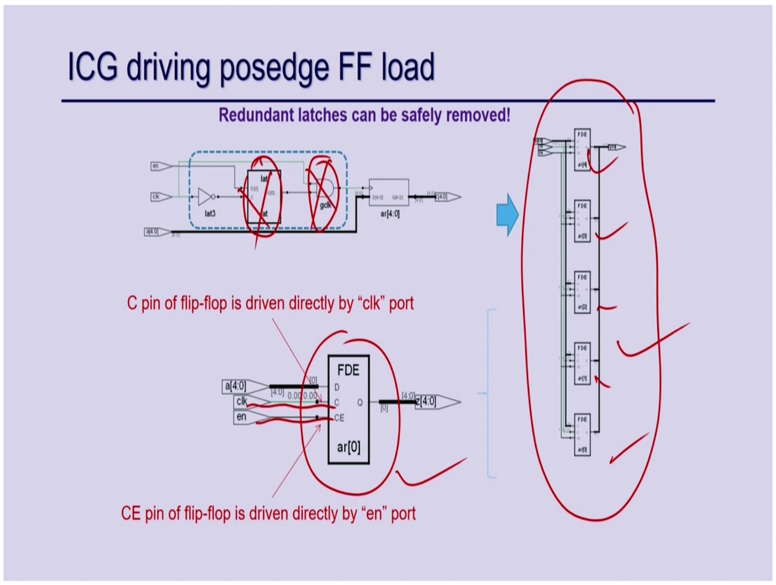
The Gated Clock Conversion feature converts flops with clocks that are gated by simple AND / OR logic to enable flops that are directly connected to the clock line.

So, here is an example suppose you have this circuit here where you have a gated clock and this is clock enable, if you just synthesis this with a simplify pro of synthesis of synopsis and if you enable this is gated clock confirmation then enable what will happen here it will actually move this remove this and gate it take this enable and it will connect directly this enable to the enable point, and this clock directly to the clock point of this particular flip flop right.

Similarly, if you have bigger logic, here you can see here there is enable 1 here and enable 2 and there is lots of logics here and this is basically gated clock right this is the gated clock, it is actually able to detect this circuit and remove everything optimize this and effectively put this enable 1 this circuit effectively means all of or one another to. So, you do the optimizations and put this as the enable and clock is directly feed into the this register right.

This is what this tool automatically does that this is called automated gated clock conversion for ASIC prototyping ok.

(Refer Slide Time: 62:20)



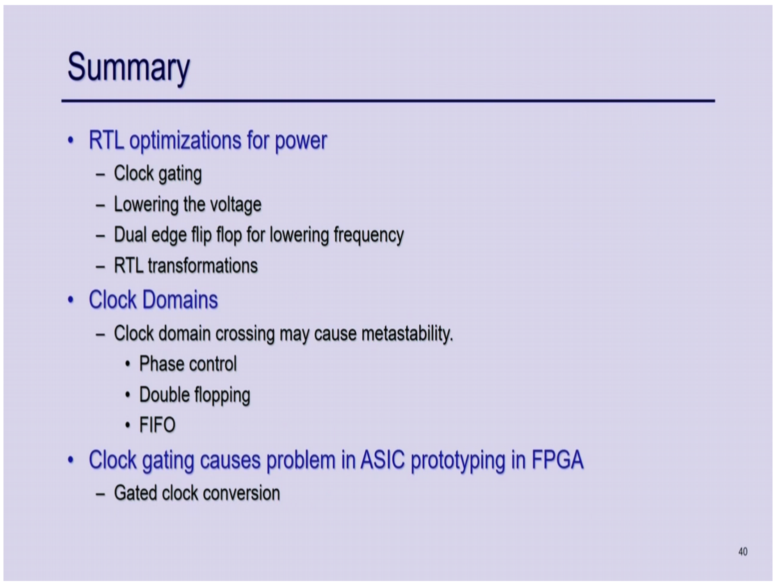
ICG driving posedge FF load

Redundant latches can be safely removed!
C pin of flip-flop is driven directly by "clk" port
CE pin of flip-flop is driven directly by "en" port
FDE
ar[0]
a[4:0]
clk
en
D
C
CE
Q

And I talked about this integrated gated clock also where I use a latch as well as the and gate and if you just through do through this synthesis tool, what will happen again you just remove this and gate and this latch and again this enable will directly connected to the enable point and this clock directed to clock point this is what is will happen in synthesis real synthesis right this is real synthesis whether I just give a give a capture one of this register.

So this is a five bit register this is five points right. So, there five bits and this is just a one sample of that right this what the synthesis tool does just to remove the gated clock things right.

(Refer Slide Time: 62:58)



Summary

- RTL optimizations for power
  - Clock gating
  - Lowering the voltage
  - Dual edge flip flop for lowering frequency
  - RTL transformations
- Clock Domains
  - Clock domain crossing may cause metastability.
    - Phase control
    - Double flopping
    - FIFO
- Clock gating causes problem in ASIC prototyping in FPGA
  - Gated clock conversion

40

So, in summary you today we have discuss about this several optimization technique or transformation technique that are related to power, we have discuss about clock gating we discuss about this lowering the voltage of your design, we discuss about this dual edge triggered flip flops for lowering frequencies, we discuss about different kind of RTL transformation just like this changing the multiplexer architecture or it this removing some glitchy control signals or connected clock to that glitchy controls and integrated connect clock techniques those are techniques that are improving the power consumption of your circuit.

We have also discuss the clock domains issues with the clock domain clocks cube problem and then we talked about this metastability because of that clock domain crossing and we talked about three specify technique like phase control double flopping control, and FIFO control issue I mean solution to avoid this metastability of this clock domain crossing problem right. And then we talked about this, what is the problem with clock gating when you are doing ASIC prototyping in FPGA and how it is actually avoided using gated clock conversion. With this I will complete this particular module.

Thank you.