

Digital Systems Design with PLDSs and FPGAs
Kuruvilla Varghese
Department of Electronic Systems Engineering
Indian Institute of Science – Bangalore

Lecture-39
Xilinx Virtex Clock Tree



Welcome to this lecture on field Programmable Gate array is in the course digital system design with PLDSs and FPGAs, in the last lecture we have looked at the career logic in the Virtex, then we have looked at basically the routing resources and how the combination cute and sequences circuit or the data path maps into this FPGA resources we have looked at the dual port RAM.

And a few fitting examples in a we have looked at given some state machine and given some kind of circuit, how much resources it uses with an FPGA and also we have looked at a VHDL code and given the code what is a circuit is synthesize and how is really the signal path trace within the logic block or the slice of Virtex FPGA. We will have a quick run through those lecture slides before we get into today's portion.

(Refer Slide Time: 01:36)

Carry Chain52

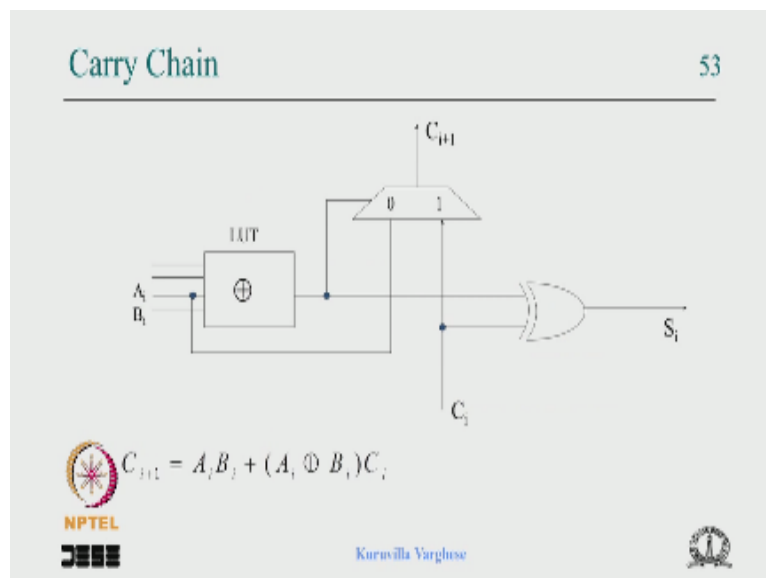
- Adder
$$S_i = A_i \oplus B_i \oplus C_i$$
$$C_{i+1} = A_i B_i + (A_i \oplus B_i) C_i$$
- Requires two lookup tables (S_i and C_{i+1}) at each stage.
- This along with routing makes adder big and slow
- Hence dedicated carry chain to make adder faster, implementing C_{i+1} .

Kuruvilla Varghese

So let us move on to those slides, so what we have told about the carry logic is at when you the basic operation in arithmetic addition which is used for subtraction, multiplication, division everything, so to make the arithmetic faster the addition has to be faster, but if you do implement the and you know that for each stay you need 2 outputs 1 is some output and another is a carry output.

And if you implement this in 2 logic look up tables then you need to interconnect them with the wires and all that. That will make all the kind of when you cascade the full adders, the total delay will be very high, so the advantages if this part is kind of hardwired or put it in to dedicate logic and that is what is carry chain in all about.

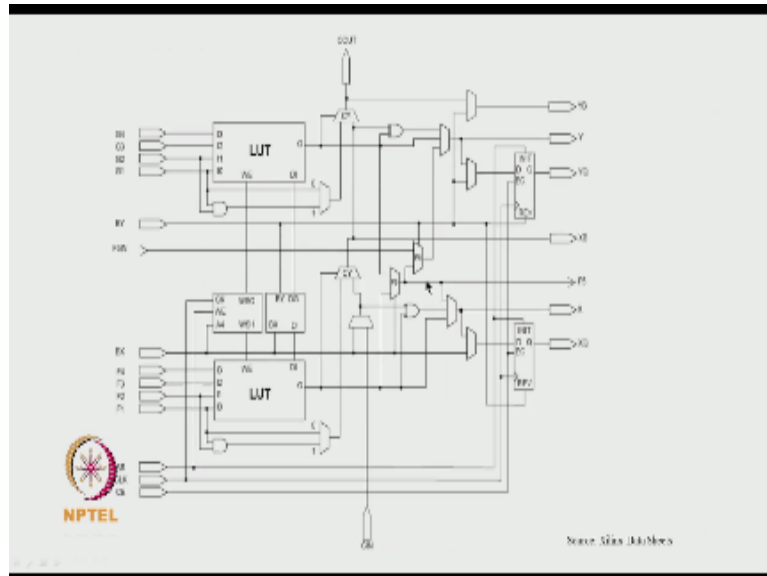
(Refer Slide Time: 02:42)



And we have seen that essentially I like for a particular stage I the input $A_i B_i$ is given to look up table and an xor is performed and that xor select Mux, so which one it is one the carry output is a carry input or if both are one to carry generated if that is not the keys okay. So essentially that implements this equation $A_i \text{ exclusive O } B_i$ is this path and $A_i B_i$ this path and $A_i B_i$ is this path.

And to generate the sum we use this particular A exclusive or B_i and external xor gate you know xor with the carry input to generate the sum. So you can imagine this goes to the next stage where $A_{i+1} B_{i+1}$ is input to the lookup table and that combine and generate the S_{i+1} and this logic there generate C_{i+2} and so on ok so that is how it goes and this is dedicated, this dedicated built in.

(Refer Slide Time: 03:46)



And we have seen that in the real the logic diagram of the slice where you can see all these you know this part and this part which form carry logic and this is exclusive OR gate external to it and that is a 1 that can be you can take it out or you can register through this particular flip flop okay.

(Refer Slide Time: 04:09)

Carry Chain

55

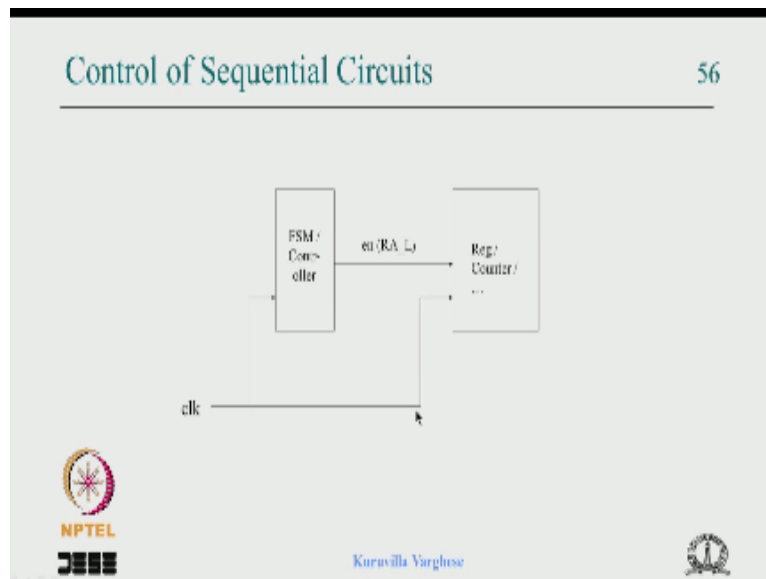
- For adders use the operator '+' to be able to use carry chains.
- For higher level functions like counters etc; synthesis tool infer and use carry chains.
- The AND gate combining A_i and B_i shown in Slice diagram is for partial product generation in multipliers
- In some FPGAs, carry chain has features to cascade (AND/OR) the LUT outputs.

Kurusilla Varghese

And the advice is that user +operator the tool will pick up this carry chain and use it you do not have to worry you do not have to write the kind of equation for the ripple adder or anything like that in fact if you do that it will not map to get into the carry chain we will use you know look up table, so whenever you want an adder using +operator and was some FPGAs this carry chain can be used as cascade chain.

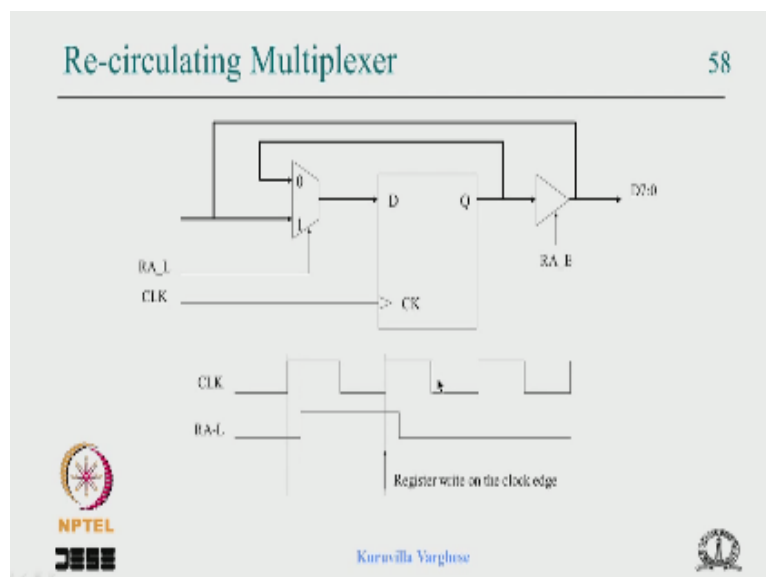
That is this particular output can be kind of and or with this particular output and sp on or do it really depends you know whether it is kind of positive logic or negative logic.

(Refer Slide Time: 05:02)



And we also have seen in a sequence circuit or FSM control the register or counter and in our sequence example we have seen that can kind of enable the data path to re-circulate the Mux, so when the enable signal from FSM is 1 something input goes to the flip flop, otherwise it is re-circulated.

(Refer Slide Time: 05:24)



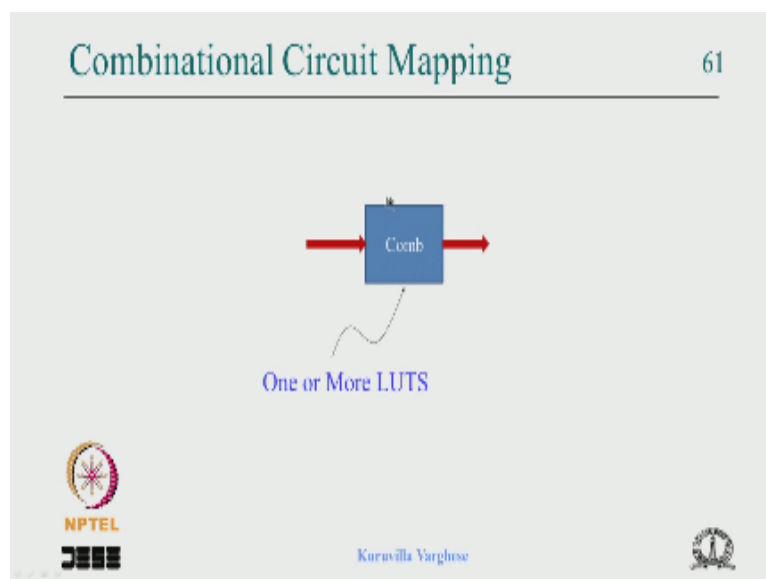
Now in FPGA this re-circulating Mux is built into the flip flop and the select line of the re-circulating Mux is available as clock enable or something when does call EC enable clock ok. This are this all CE we know some call CE and some call EC. So if you write VHDL code like this up on the clock is some kind of signal is 1 q gets d, then automatically that control

signal is connected to clock enable because one level of re-circulating Mux is all already available.

But definitely if you have and you know you have else control signal then another Mux come outside which will be implemented in the lookup table than in the flip flop. So this normally may one average you will have mostly 1 control and that is built into the FPGA quite good because otherwise for a 2 to 1 mux unnecessarily the lookup table will get used and by building that into the flip flop.

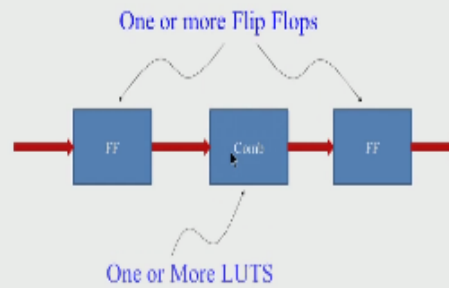
The lookup table is same, otherwise a 4 input look up table will be kind of reset for in a 3 input logic to those 2 to 1 mux will have 2 input and 1 select line input, so it is a three input logic but 4 input look up tables used to implement that ok.

(Refer Slide Time: 06:58)



And when you have some combination circuits in a FPGA will be map 2, 1 or more look up tables you know maybe 2 look up tables will be cascaded or both will be combined with our way of all of them is combine using a F5 a fifth Mux which is cascade with something else and so on ok, that you can kind of work it out.

(Refer Slide Time: 07:22)



Kurusilla Varghese

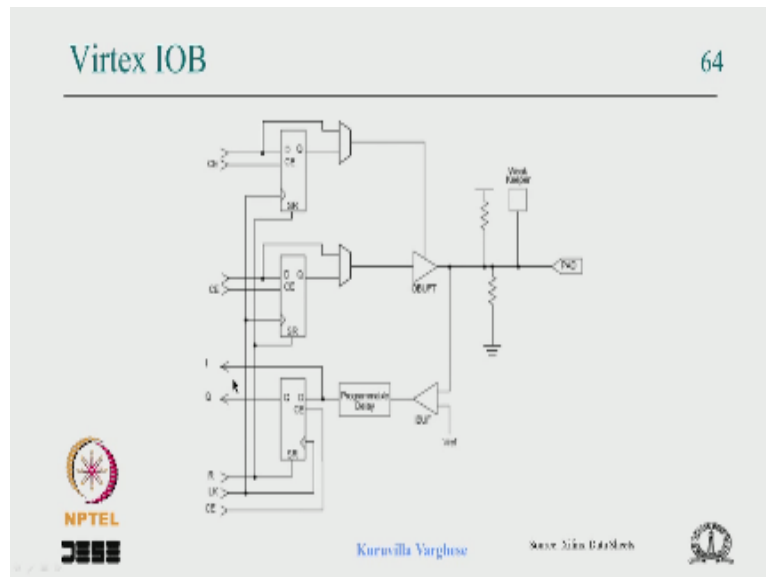


And when you have a sequential circuit mapping you will have some flip flop combination circuit flip flop, be with the sequential circuits or datapath see like in the case of sequential circuit the flip flops are kind of you know the source and destination can be in one place and in the case of data path it is some thoughts register and separate destination register but path itself is different in sequential circuit.

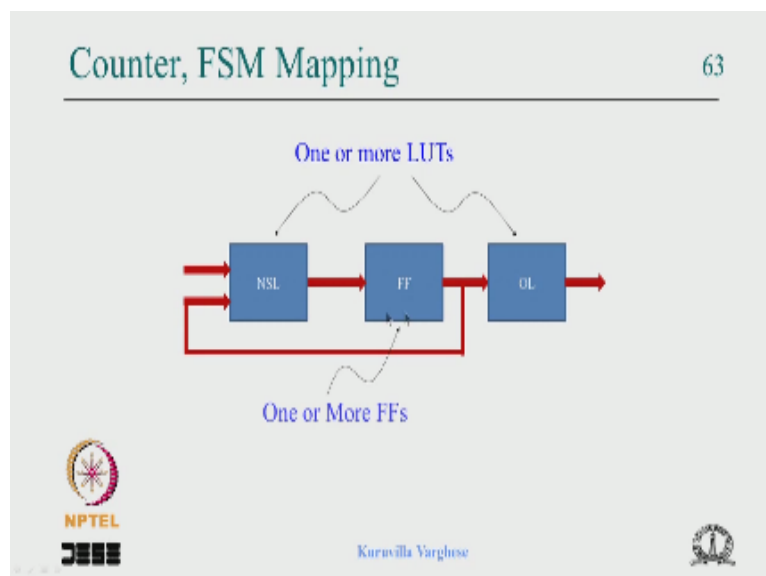
So the flip flops get implemented in 1 or more flip flops, combination circuit will get implemented in the look up tables and again the flip flop will go to the flip flops of the slice ok and it will be good the bus we can hope is that this combination circuit cute follow with the flip flop can get implemented in you know the same slice ok depending on the number. You know depending on the number of flip flops it might occupy multiple slices.

And this would require some extra flip flops from a slice you know it all depends on the complexity unless sequential circuit how it gets with same thing with the sequential circuit or finite state machine you have flip flops and the next state logic, so this you can imagine that coming in a kind of you know if you go back to this diagram here you have the lookup table follow with the flip flop ok follow up with flip flop.

(Refer Slide Time: 09:01)



So definitely so you have the combinational circuit, the look up table followed with flip flop.
(Refer Slide Time: 09:06)



So that can get implemented and that output can go to the look up next slice and so on. So that is how the real sequential circuit map in FPGA, this we have seen I/O block essentially it is a tri state buffer which allows in an output path and input path and a tri state path ok and this can be combinational so as I said that there are FPGA wires here, so the output can come directly or through a flip flop.

So you can get synchronize and go out that a good idea because otherwise already it is coming through some combination circuit there will be lot of delay from the clock edge and you know it is going out of the chip it can suffer for further delays, so will be useful to

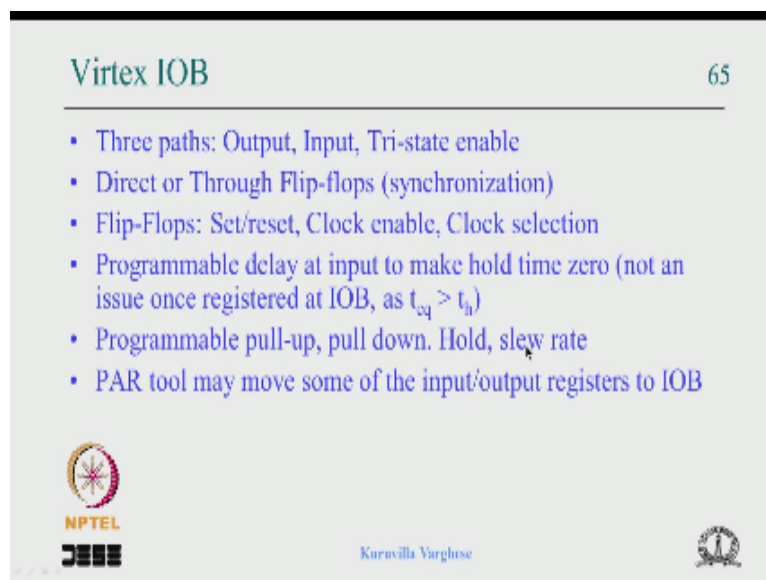
synchronize with clock because it appears with no delay after the clock, then maybe it can be reason the receiving end or go to some combination to get free synchronise.

Similarly input path can be directly taken or can be synchronised and the all these flip flops clock enable and the clocks and clock can be chosen from multiple clocks, they have reset and set and this programmable delay can be added to make the whole time 0 because some time this signal is coming from external source it could be single by signal but then trial the whole time can create problem.

Because making the whole time is stuff, so we can add some delete make 1 such synchronize the whole time will not be much of a worry ok and similarly the tri state path you have combinational path or the register path and as we said that I know it can be pulled up or pull down because when you try sitting a signal it is floating then the noise can you get picked up all the input can switch.

So that can be avoided through the large pull up and not pull down or program a hold circuit which weekly holds on to the last value and so to avoid the noise pickup.

(Refer Slide Time: 11:23)



The slide is titled "Virtex IOB" in green text at the top left, with the number "65" in the top right corner. It contains a bulleted list of features in blue text. At the bottom left are the NPTEL and JEE logos. At the bottom center is the name "Kurusilla Varghuse". At the bottom right is a small circular logo.

Virtex IOB 65

- Three paths: Output, Input, Tri-state enable
- Direct or Through Flip-flops (synchronization)
- Flip-Flops: Set/reset, Clock enable, Clock selection
- Programmable delay at input to make hold time zero (not an issue once registered at IOB, as $t_{cq} > t_h$)
- Programmable pull-up, pull down. Hold, slew rate
- PAR tool may move some of the input/output registers to IOB

NPTEL JEE Kurusilla Varghuse

And it support various I/O standards.

(Refer Slide Time: 11:27)

- Various IO standards
 - LVTTTL
 - LVC MOS33, LVC MOS25
 - LVC MOS18, LVCOMS15, LVC MOS12 ...
 - PCI33, PCI66
 - ...
- Some IO standards require a Reference voltage for Inputs
- Banks of I/O pins support some of the IO standards

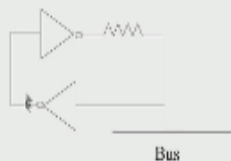


And this is a hold circuits flip flop with latch the input path.

(Refer Slide Time: 11:31)

Week keeper (Hold)

67

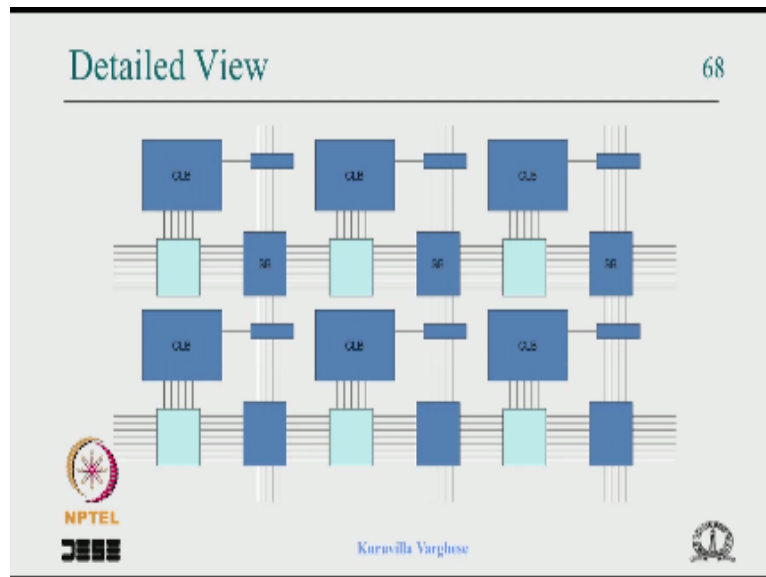


- Hold circuit hold the previous state of the bus, but provides a weak drive so that it could be driven to '0' or '1'.
- This avoids unnecessary switching of inputs by noise, if the bus would have been left in high impedance.



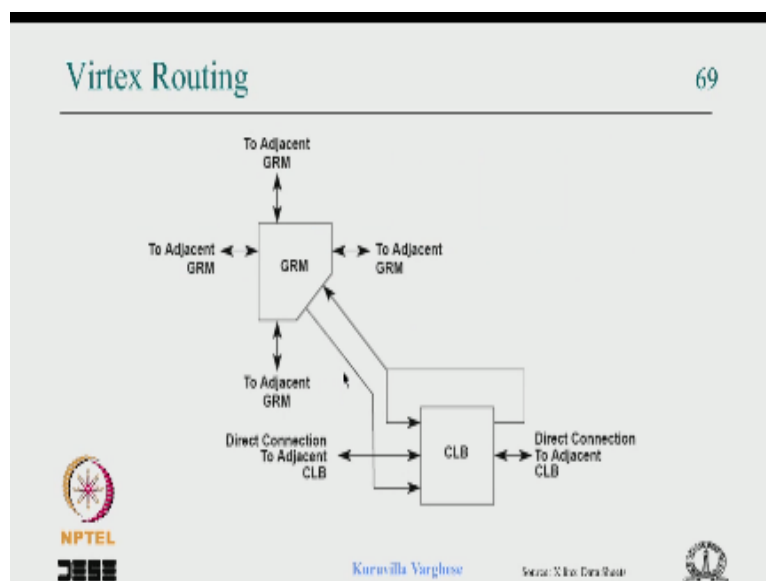
You can see the output is given as well as rest and so really does not matter even though it is holding driven by a flip flop, that is you can kind of pull it down or pull it up with low distance there is no kind of VCC ground short circuit because of this resistance, but it avoids the switching you know used the device.

(Refer Slide Time: 11:56)



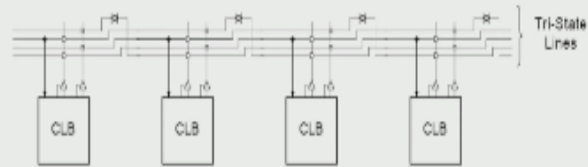
And this is the main you know the wiring diagram you have main switches and this is the logic block and this input to the logic block output from the logic block.

(Refer Slide Time: 02:09)



As a picture shows that are directors running from this switch to the CLB and the adjacent CLBs are connected with dedicated wires that means this CLB output can go to the input of the CLB and this CLB will be received input from the CLB output without going through the wire. So that can be very fast.

(Refer Slide Time: 12:30)



- For Busing and Multiplexing it is better to use tri-state gates than multiplexers



And there are some statistics of the number of wires which is available then we have seen the bus lines to tri state gate plus CLB to form the bus which is not available in the kind of current FPGAs, so you want to do muxing have to do the AND/OR muxing than the tri state gate musing. Then we have looked at an FSM example with two input, 3 state, and 2 Mealy output.

(Refer Slide Time: 12:55)

Fitting Example: FSM

- FSM, with 2 inputs, 3 states, and 2 Mealy outputs. How many CLBs to fit in?

- State Variables: 2 flip-flops (3 states)
- NSL: 2 state variables + 2 inputs = 4 inputs
- OL: 2 Inputs + 2 state variables = 4 inputs
- 2 LUTs for NSL
- 2 FFs for state variables.
- 2 LUTs for OL

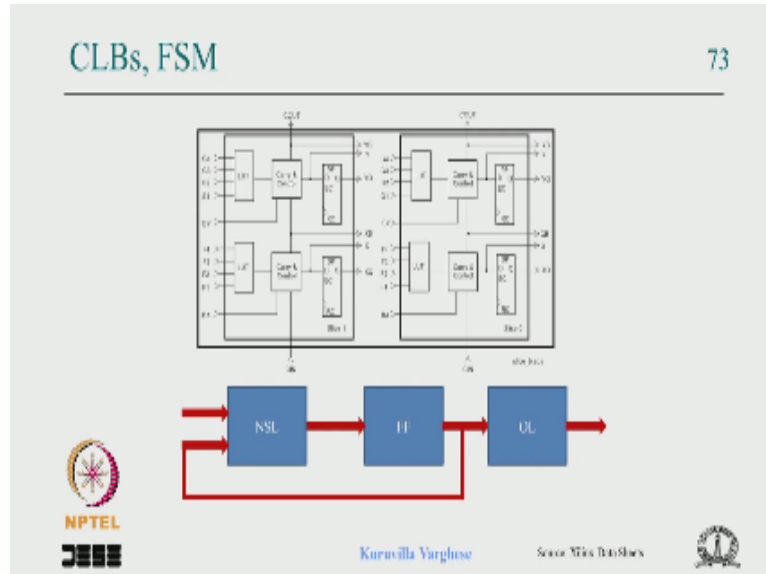


This requires 1 CLB minus two FFs In fact if output is registered still it can be accommodated in one CLB



We work out how much CLB it requires to occupy to implement this and we have come out with basically 1 CLB minus 2 flip flop you can kind of refer to the last lectures discussion to get in the detail.

(Refer Slide Time: 13:19)



And we have looked at so that is a kind of it implements.

(Refer Slide Time: 13:25)

Fitting Example: Counter

- 8 bit up counter with parallel load feature
 - State Variables: 8 Flip-flops
 - Incrementer uses carry chain
 - NSL: 1 state variables + load + 1 din = 3 inputs per state variable
 - NSL requires 8 LUTs
 - This requires 2 CLBs (4 Slices)

And we have looked at the 8 bit counter with parallel load features, so this is the count of the 8 flip flops with the +1 implemented and we said that this look up table with the carry chain can implement +1 and flip flop. So the 8 flip flops and 8 bit lookup table with a carry chain that requires. So it is basically it requires 2 CLBs.

(Refer Slide Time: 13:48)

Signal Paths in CLB

76

```
library ieee;  
use ieee.std_logic_1164.all;  
  
entity test is  
  port (a, b, c, d, e, f, g, h: in std_logic; z: out std_logic);  
end entity test;  
  
architecture arch_test of test is
```

```
begin
```

NPTEL
JEE

Kuruvilla Varghese



And we also have looked at a code with kind of a to h input.

(Refer Slide Time: 13:53)

Signal Paths in CLB

77

```
process (a, b)  
begin  
  if (a = '1') then z <= '0';  
  elsif (b'event and b = '1') then  
    if (c = '1') then  
      z <= (d and e and f and g) xor h;  
    end if;  
  end if;  
end process;  
end arch_test;
```

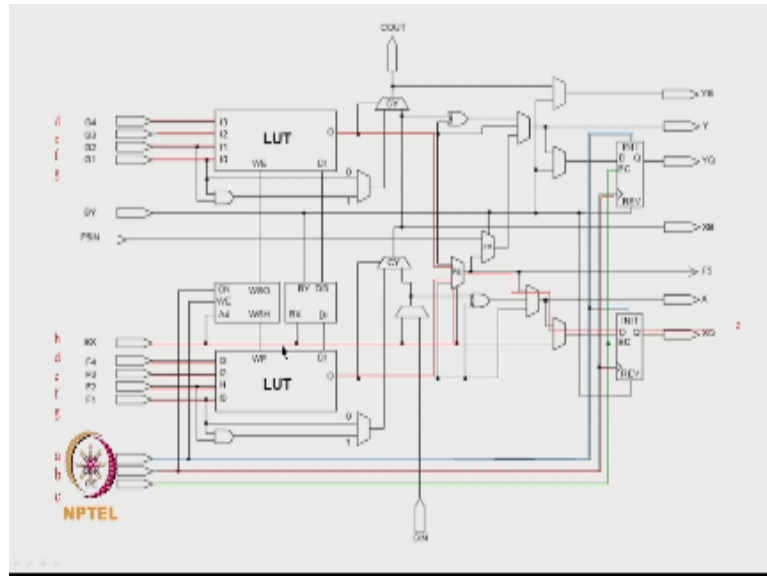
NPTEL
JEE

Kuruvilla Varghese



And we said it is a process a is reset, b is the clock, c is a control signal and z is d and e and f and g and xor h. We have discussed this will be a 2, 4 input lookup table cascade, c go to the enable clock of the flip flop, d is a path a reset ok.

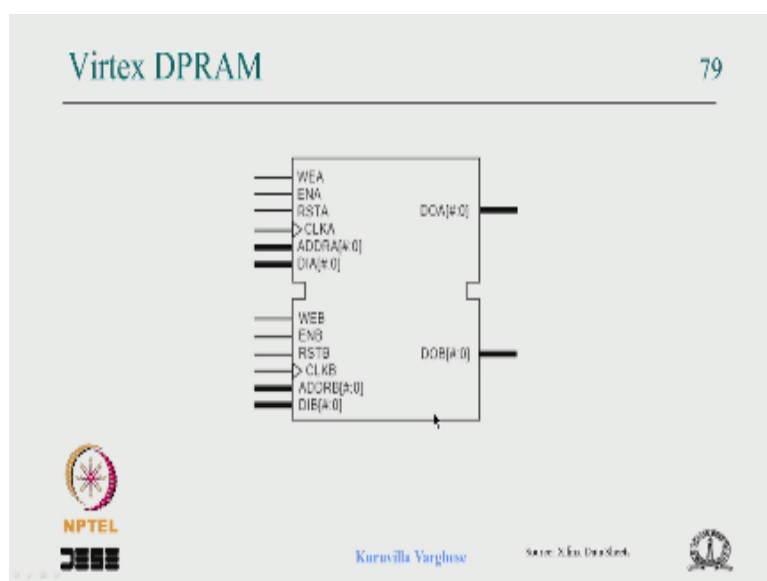
(Refer Slide Time: 14:15)



So we have shown that you know if taken this flip flop and we need 5 input look up tables, d, f, g, goes to same to look up table h is a fifth variable that selected at 5 Mux, so that is combined and that output is router to the flip flop and flip flop gets you can see the clock which is b reset which is a reset is a this blue line and clock enable which is e, c which is c, so that is how it get map.

As I said you can write this code, select a Virtex FPGA, Spartan 3 FPGA and synthesizer implemented go to the floor planner open up then you can see assuming the CLB which is implement then you can see this wiring connection inside there, so you can verify that is working even if the time permits I will show that this tool this exercise I will show in the tool.

(Refer Slide Time: 15:18)



And we have looked at the dual port RAM and 2 dual port RAM are available because I am blocked RAM with two ports with input and output separated. So you could write through the 1 port and read from another port or we can write using both port, does not matter but only thing is that there should not be conflict, in a both not try to access the same location. So if somebody is writing and your reading through another port ok.

This sport then if it is a same location then the read value may be wrong and if you try to write to the same location we are not sure which one will succeed, so there is a conflict. So that should be avoided but otherwise it is a 2 dual port and we have discussed little bit about multiport memory either not very difficult to implement that but these are kind of hard code and this can be combined.

You know this basic blocks are available this can be combined in various ways to implement say different with different like I know this will be proposing to 8 then you can combine to the two of them in parallel to implement into 16, both of them can be combined to implement double the size which will involve some muxing as I seem in the lookup table. That is what part of the block rap.

But you should know that there is a basic granularity which it is the size of the width and depth. So if it to some arbitrary with an arbitrary size some block and get wasted. So you would see that though the you might look into FPGA data sheet and find what is the total size of the memory available, but if you choose arbitrary depth and width you will see that some memory is wasted ok.

So that you need to take care you should find what is a basic granularity basic size in terms of the depth and width. So if you choose an integer multiple of the depth and width then everything will be no wasted, but if it is kind of fractional on that then some gets wasted there is because these are kind of cascading or gang to the block level not at a very small or small size levels of the basic clocks are combine together to build that.

(Refer Slide Time: 17:54)

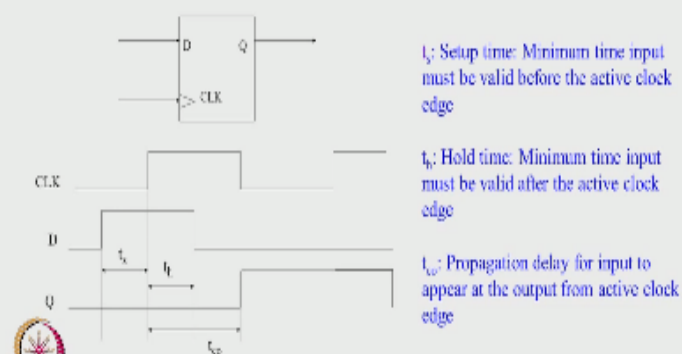
- True Dual port Memory
- Each port can be read/write, read or write
- Synchronous reads and writes
- Can be combined for larger widths and depths
- Instantiated through Core Generator Tool
- Conflict on simultaneous read/write to a location, read data could be wrong
- Can be initialized in VHDL code



So that should be kept in mind, so this is 2 dual port memory is what can be read or write or either you know 1, 1 is read and 1 is write, it is all synchronous read and synchronous write everything gets written on the clock edge and we combine for large width and depth and it can be instantiated with tool, so when you want use that core generator tool that will give the instantiated template which can be cut and paste it in your code.

And this is the conflict and basically this memory when you used and you initialise it with some data and that initialize can be specified in the VHDL code ok.

(Refer Slide Time: 18:43)



So that is what we have seen in the last class may be I have gone little fast. So I repeated the part you please look at the last lecture and this lecture. Now let us come to another part we have discussed this topic metastability like for a flip flop to LAX or to register the input data

to the output. The input has to specify some timing with respect to the clock head. That means when the clock edge comes the data has to be set up sometime before it.

And after the clock edge should be held on for some more time. So that the value at this point gets enough it fully copied or transferred to the output ok. So there is a window around the clock edge where in the data has to be stable okay. The time before the clock edge is call setup time, time after the clock call this whole time. Then if that is met with the propagation delay t_{co} the output appears after the clock edge.

And if it does you know if you do not need that then there is a chance that this could be wrong this would not be a full copy of the input in the worst case it can get stuck in between and we have discussed this phenomena and we have seen how to avoid that by synchronisation we have looked at the single seat synchronisation double speed multi change and lot of techniques.

And we have also looked at a little bit you know how that probability, so that a essence of the timing of the input of the flip flop.

(Refer Slide Time: 20:32)

Minimum Clock period 82

Data path

$$t_{clk} > t_{co} + t_{comb} + t_{setup}$$
$$t_{co(min)} + t_{comb(min)} > t_{h(max)}$$

Here we are considering the data path from first flip-flop to the next. We are estimating the minimum clock period for proper latching of data on to second flip-flop

NPTEL
JEE

Kunavilla Varughese

Now when you take a data path and this time in place to find out what is the maximum clock and basically to avoid whole time and all that take a data path were there is a source register the output of which was going through combination circuits and reach the destination 3 and we know that when a clock comes here, at the same time the destination also get a clock. So but the data like whatever maybe the data that there is captured.

But the input would like because of with respect to this clock the output changes and he propagate through combination cute and reaches here by the time the same clock at this gone already and now we know that the next clock edge come and we have to have some setup time before it for to be captured. So we always in this case analyse the timing with respect to 1 for coming here and data get transferred here go through the combination circuit.

And it is set up before the next clock edge, so a clock edge to clock timing is analysed to find the clock period okay the minimum clock period which is to be greater than $t_{co} + t_{com} + t_{setup}$, we also know that when clock edge comes the input asked to remain there after the clock ok. So we should make sure that the input when a cock comes here both at the same time the input changes after $t_{co} + t_{com}$ that be greater than the whole time ok.

Because that is with respect to same edge is coming there and there is a whole time after the clock edge, so but the data is changing here going to change here after t_{co} and t_{com} , so that $t_{comin} + t_{com} \min$ should be greater than the whole time otherwise the whole time is violated with respect to the same clock. So the basically from the setup and hold time we are you know kind of coming out with this the setup time decide this cock period.

(Refer Slide Time: 22:55)

83

Minimum Clock period

- Sequential Circuit / FSM

$$t_{clk} > t_{co} + t_{comb} + t_{setup}$$

$$t_{co(min)} + t_{comb(min)} > t_{h(max)}$$

Kuruvilla Varghese

The whole time decide this hold time violation condition ok in the data path, same thing for a sequential circuits for some where the critical path is from some salts register through a combination circuit which is next to the destination register ok. So the situation is saying ok

when you look at the individual flip flop, the situation is same from a source register through a combination circuit to the destination register the clock period should be $t_{co} + t_{com} + t_{setup}$.

Similarly the whole time variation will be $t_{co} + t_{com}$ should be greater than the whole time and when we mind you should remember that in all these analysis the clock period analyse from one clock edge to the next clock edge and the whole time is analyse for the same clock edge okay like when the clock comes the data changes with t_{co} t_{com} here and the same clock there is a whole time in do.

So this thing should happen after the clock edge, that is basic what is basically express this in this inequality. Now one thing you should remember that we are assuming 1 we are making a very kind of some basic assumption in all these analysis in the minimum period and hold time violation you know you can think what is this assumption we are making. Assumption is not very realistic which we are making is at the clock reaches at the same time to the source and destination flip flop ok.

In a picture may not be very obvious because you know you are writing a block diagram and it looks me but you should remember that in a chip in which area maybe the source registers in one corner and the destination register is other corner and the clock has to travel to various path to reach there and that can show for delay all along. So the clock that could be skew between the clock ok.

The skew is the relative delay between the clock arrival, that means the clock arrives here at 100 nanosecond and clock arrives is here at 1 or 2 nanosecond due to the wire delay, then there is a skew between them to nanoseconds skew between them ok that can affect because, so we have to bring in that skew into analysis of the clock period as well as a whole time violation ok. That is important for realistic analysis.

(Refer Slide Time: 25:41)

- Previous analysis assumes that the clock reaches at flip flops at the same time, it is not practically true, as the wire delay and buffer delay gets added.
- This creates relative delays between pair of flip flops or registers
- For analysis it is important to consider the clock skew between flip-flops/registers where there is a data path between them.
- Clock Skew:
 - Difference in arrival time of the clock at the flip flops

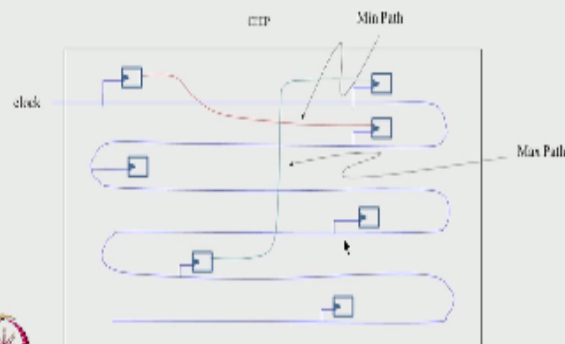


Kunavilla Varghese



So that is what is stated here, so in all analysis we have initially assume that the clock reaches the flip flops at the same time, but we have to consider the relative delays due to the wiring and that is cause skew which is the difference arrival time of the clock at the flip flops and we will see what happens when there is a skew between the flip flop.

(Refer Slide Time: 26:07)



Kunavilla Varghese

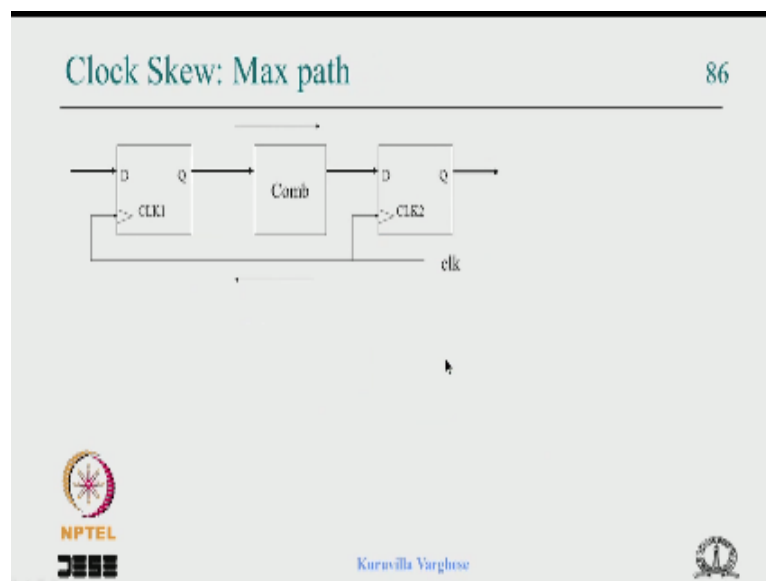


And take assume that there is a chip and there are flip flops kind of scattered around and assume which is not very realistic that the clock is coming from pin which is going to the flip flop okay like that you know like chain ok, which is not a very good started, started to highlight the problems and just putting some kind of fictitious case and you see the that there are two scenarios say take this a flip flop source and destination where the clock is going from the source to the destination like that.

And the data is also going from the source to destination and this problem is called min path problem which we are going to analyse the timing and there is another situation where you see this is a source register where the clock get for it when it reaches the destination register but the data is going back from the destination to the source ok. So all say this is the author of the data is concerned this is a source register and the destination register.

So here the scenario is that data and the clock are in the opposite direction which is called Mux path ok this is called Max path problem and here this case the data and the clock **is** kind of travelling in the same direction so that is called a min path problem.

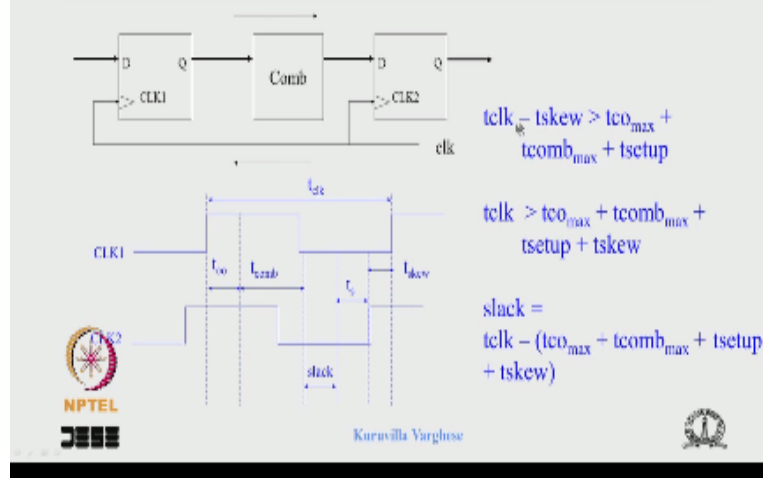
(Refer Slide Time: 27:46)



So we will analyse both of these and see what happens when was first used the max path problem where the data goes from the source register to the destination register, but the destination that comes earlier today ok. So this is a little better way opposite case the resource registered drive the destination register with the data but as far as the clock is concerned the destination they just get the clock fast and the source a get the data and in the clock later.

So let us put that picture for analysis of this is a case we have a source register as far as data is concerned output of which is going through a combination circuit and reaches the destination itself and the clock of the source of clock 1, source the destination clock 2 and you see that the data is moving this way and the clock is moving in the opposite direction. Now the other fallout is that the clock 1 is the delayed version of the clock ok. So let us put that picture.

(Refer Slide Time: 28:56)



So you have a clock 2 which is reaching here and clock 1 is a delayed version of the clock 2 and that delay is called skew. For whatever reason there could be wiring delay, buffer delay, and all that in the clock. So there is a skew between the clock 1 and clock 2 and mind you the clock 2, clock 1 is a delayed version of the clock 2, the clock showing complete up. Now you assume that there is a clock coming to the clock 1 and clock 2.

So by the time the same clock we need not worry because $t_{skew} + t_{com}$ is greater than the same clock. So by the time this first clock comes at both places. There is no time to latch whatever you get it go on and the data comes later. So the analysis from this clock to the next clock, when the $t_{skew} + t_{com} + t_{setup}$ should be greater than this the clock period is ok. But you see here the clock 1 is here and the clock 2 is here.

But to come early by an amount skew and the skew is the delay in this wire is ok. So the clock to come early to the clock 1 so basically the timing resources from this clock at 2 this particular clock at here ok now that clock and so the analysis that the time for time from this clock at 2 this clock at this clock period minus t_{skew} now that has to accommodate this $t_{skew} + t_{com} + t_{setup}$.

So you get an expression $t_{clock} - t_{skew}$ should be greater than $t_{com_max} + t_{comb_max} + t_{setup}$ and so the t_{clock} is greater than $t_{com_max} + t_{com} + t_{setup} + t_{skew}$, so what happens is that the clock period is increasing because of skew and the frequency of operation goes down if there was no skew we set the clock is greater than $t_{co} + t_{skew} + t_{setup}$ when there is a skew that gets added up, so that the clock has to accommodate that skew.



And the fallout is that the clock period goes down the frequency operation goes low than without skew, so that is one disadvantage of the skew in the max path because of the skew the clock period goes down or clock period goes up and the frequency of operation comes down because of this skew, the frequency of operation of the data path comes down which is not a good idea. So if there is a because of this skew you are forced to choose the clock period which is larger which reduces the frequency of operations.

So one need to analyse all the path and what is the maximum skew in the in max path situation and choose add that to the clock period so that you get the corrected clock period and we know that the whole time also will kind of change. Because it is easier plus 3 come now I should have the disk you should bring into the come into the whole time violation, so that can that be analysed.

(Refer Slide Time: 32:33)

Clock Skew: Max path
87

- Analysis for data path from first flip-flop to next
- We assume $t_{co} + t_{comb}$ is greater than the hold time of flip-flop
- Hence, when a clock edge comes to both the flip-flops, new data from first flip-flop arrives at the second flip-flop after the clock edge, even after the hold time and won't get latched in second flip-flop
- But, we estimate the clock period such that when the next clock edge comes to second flip-flop, data from the first flip-flop due to current clock edge get latched in the second flip-flop


Kuruvilla Varghese


So let us choose the other problem where the data path and the clock is going in the same direction ok. So you are you see the clock 2 is a delayed version clock 1, so you have a clock 1 here clock 2 with skew ok, so the clock 2 come later. Now we have in a better position because if you analyse from 1 clock at to the next clock edge you have $t_{clock} - t_{skew}$ because you are going from this edge to this edge.

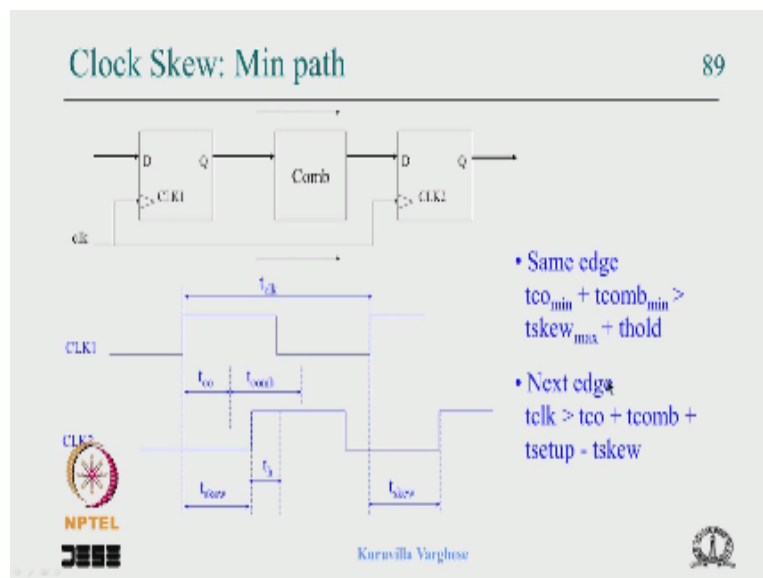
$t_{clock} - t_{skew}$ should be greater than $t_{co} + t_{com} + t_{setup}$ ok, so which is good because now the t_{clock} and t_{skew} goes to the other anticlock will be $t_{co} + t_{com} + t_{setup} - t_{skew}$, so the clock period comes down the frequency of operation goes up ok, but this is not a great concern

because it could be Max path which reduces the frequencies one need to choose the particular condition it is a the the max path not Min path of the clock frequency is concerned.

But it is greater danger when you look at the diagram is not with respect to you know one clock edge to the next clock edge say because of this clock use getting delayed the normally we assume that it is coming at the same time and by the time because of the charges clock at the data will change $t_{skew} + t_{com}$ and we are assuming it is greater than the whole time. So by the time the whole time window is gone nothing happens to this.

But you asked you see here the situation is that the clock to is getting delayed so maybe the $t_{skew} + t_{com}$ can you get no catch up with the whole time window of this clock 2 ok. So that is what is shown here maybe because of this delay the whole time window comes here if this is minimal $t_{co} + t_{com}$ is minimal may be it can violate the whole time ok. So the whole time violation condition is that $t_{co} + t_{comin} + t_{com}$ should be great $t_{skew, max} + t_{whoel max}$ ok.

(Refer Slide Time: 35:07)



So that is the issue with the clock skew you can create the whole time violation and if it happens you see there is no, there is no way in changing the clock period and solving the problem because if you increase a clock period is not going to solve the problem because the problem is with respect to the same clock edge, ok so it does not like when you go from one side to other edge the clock period come into picture.

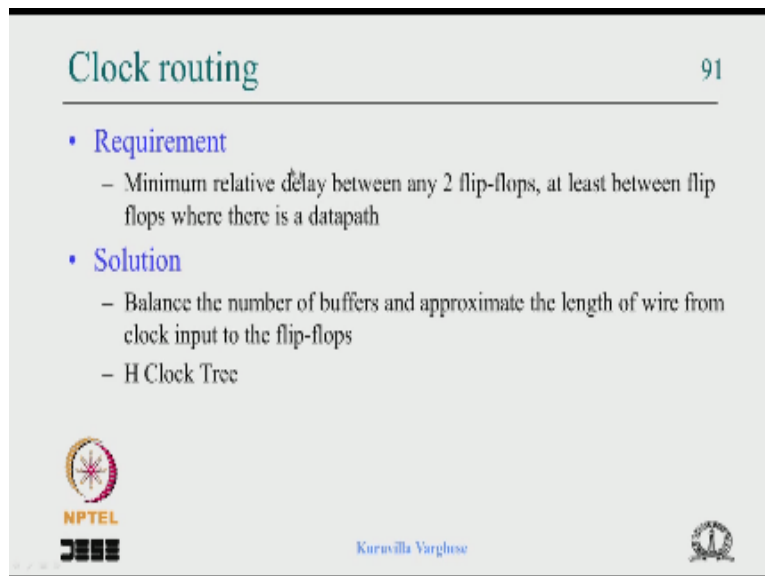
But here we are talking about the CMAT are reaching you know the one of the destination rejection later and which can cause a whole time violation, so here the $t_{co} + t_{com}$ should be

kind of great than $t_{skew} + t_{hold}$ there is a violation happens and because this t_{com} gets into the whole time window change in the clock period want help it, but what is it that you can add additional delay.

So that the data at this point comes after the whole time. So this is the issue with the clock skew it create Max path and min path and Max path you can bring down the clock vacancy min path can have the whole time violation. So essentially when you route the clock in a VLSI chip one has to make sure that the clock reaches every flip flop without Mux relative delay between them ok.

There is relative delay between them then all these issues crop up, so it is advantages to route the clock which is the clock reaches all the flip flops which is all spread within a chip do the routing such that the relative delays at the flip flop clock is minimal ok, that is basic requirement of clock 3 rerouting.

(Refer Slide Time: 37:12)



Clock routing 91

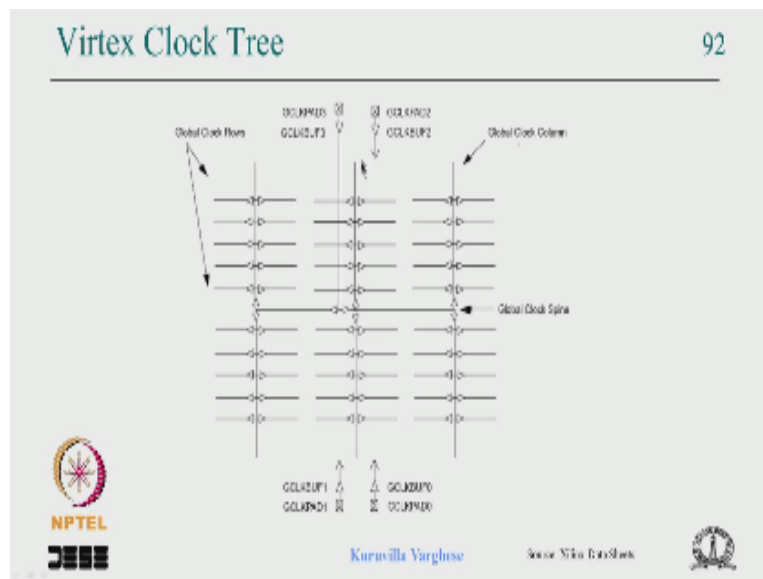
- **Requirement**
 - Minimum relative delay between any 2 flip-flops, at least between flip flops where there is a datapath
- **Solution**
 - Balance the number of buffers and approximate the length of wire from clock input to the flip-flops
 - H Clock Tree

NPTEL JEE Kuvaila Varghese

So the requirement is at minimum relative delay between any two flip flops at least between flip flop and were there is a delay data path ok. So we have 2 flip flop we should make sure that there is minimum delay between the arrival time of the clock at least between the registers where there is a data path, if there is no data that no cause of worry. So that that is a requirement, so that trick is to balance number of buffers was that there is an input where the clock get buffered and reaches the flip flop.

So we have to assume that the number of buffers in reaching from the input pin at the flip flop should not because of the flop and the number of wires you know the number of wire set number of buffers are identical any second and one solution will not be the solution there is one solution is called H clock tree where the clock is routed like an H ok. So in a VLSI chip when you do the clock router routing have to make sure that the clock tree is like a H 3 clock 3, but in an FPGA is built into the FPGA fabric, so that is what is the clock tree and we can see that.

(Refer Slide Time: 38:34)



Suppose there is a clock pin which is buffer at the beginning and it comes vertically down then it goes horizontally like H and then you can branch you put the buffers then you go up in the horizontal line you go vertically ok and whenever you brand you put buffers and now the water from the vertical line to each flip flop him branch out of with buffer okay. Now the advantages that suppose there is a flip flop here and a flip flop here.

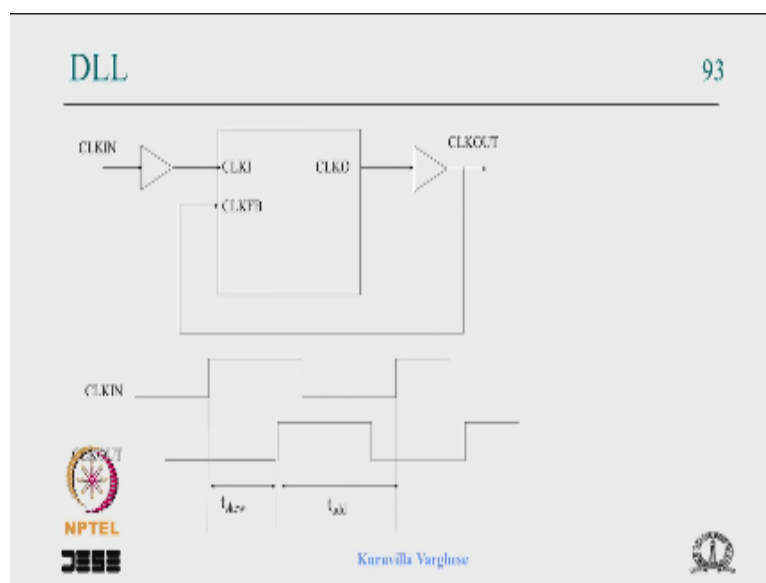
So you from the input when you look at the number of buffers say 1,2, 3 and 4 in reaching hear you say 1, 2, 3 and 4, so the flip flop here and flip flop here as the same number of buffers from input pin and similarly the you see the wire segment length 1 then 2 and 3 but here it is 1, 2 and 3. So the wire segment also somewhat identical, so essentially we can assume that the skew between the nth point are less than that is what is require.

That could be a delay between the input pin and the n, which we are not worry it will not be able it is a relative delay between the flip flop that should be minimal. So that is why the property is used in a FPGA to built in and their dedicated pin which is driving this clock trees

so you should connect the clock oscillator or the clock input to those particular pins and these are this is about the vertex clock 3 but you know in the recent period I could be proper is which is kind of FPAG.

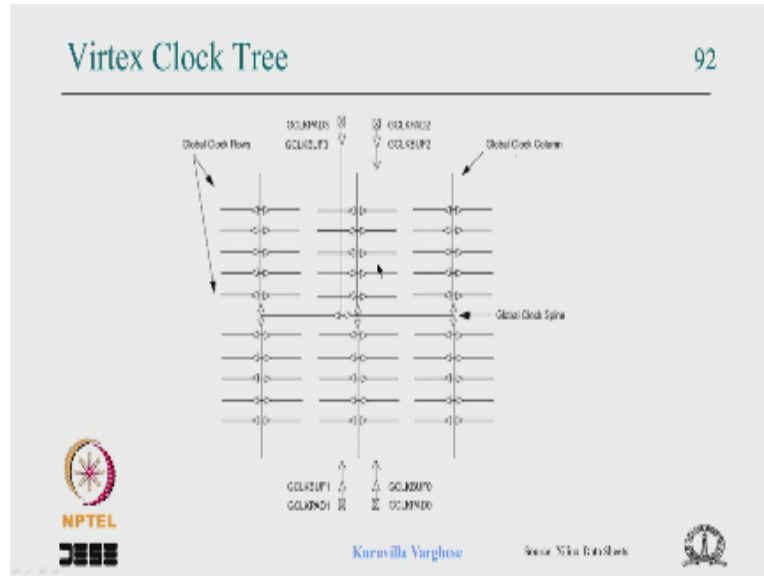
A fabric is quite big may not be able to clock by a single clock or in such complex opposite to be multiple clock domain so sometime you need to clock by the user clock and or other is a clock which is which is limited to an area of the slice of the CLB of the flip flop I mean of the FPGA, so that should be kept in mind. So whenever you want use a dedicated clock pin use a dedicated clock tree and so on.

(Refer Slide Time: 40:56)



And are we should look at the really locked loop which is available in FPGA which is used to remove the skew between the input clock and output clock and assume that there is an input pin like that and there is a clock tree which we are using the kind of clock lot of flip flops. Now because of this the buffer delay and all there could be skew between like you put a because of loading in a there a lot of flip flops and from the input to the output there could be lot of delay due to loading.

(Refer Slide Time: 41:33)



And that can be avoided by the delay locked loop where the input is connected to the input of the delay locked loop output is going to the clock tree which is feedback to a feedback. So what does the delay DLL do is that you have a clock input and assume that the clock output is skew by some amount, so what it does that it cannot be anywhere cannot bring anything back, but what I can do is that suppose this is a clock period then there is a skew. What does is that the input is for the delayed by the clock tree minus q .

So this edge you know get in a synchronised to this end, so that that skew let the skew is removed so the DLL is used for these skewing an input clock output clock. So you can remove the skew between them using the DLL and so that is the idea of the DLL which kind of add delay to remove the skew. You do not need a PLL because DLL is much simpler to implement and PLL but in PLL you know that the block diagram of the PLL you must learn.

(Refer Slide Time: 42:52)

- In a DLL, input clock is delayed for de-skew
 - In a PLL, a VCO synthesizes a clock synchronous to the input clock
 - DLL adjusts the phase of the input clock.
 - PLL synthesizes the clock of same phase and frequency as that of the input clock.
 - PLL has the problem of working with a limited range of frequencies, but in FPGAs clock frequency may not change in most cases.
 - PLL also cleans up the input jitters.
- Xilinx Virtex 5 has PLL blocks in addition to DLL in DCM.



Kurusilla Varghese



And basically in a DLL input clock is delayed, but in PLL always you find the phase difference between the input and output and that is filter and that is given to a voltage control oscillator to synthesise a clock with synchronize to the input, so in a PLL the VCO voltage control oscillator synthesise a new clock okay, of the same phase and frequency as that of the input clock the only problem with the PLL is that it will lock on to only a range of frequency at the input depending on the field using inside.

And also take some time you know get the locking or get output in phase takes a while but DLL is much more quick in doing the d skewing but one advantage of the PLL kind of newly generating the output. So even if the input or some get that would not be available and output, so for various frequency synthesis we should conserve PLL than DLL and like in the current GPGAs you have PLL blocks in addition to DLL in the digital clock manager.

(Refer Slide Time: 44:15)

- PLL
- Digital Clock Manager (DCM)
 - DLL for de-skewing
 - Phase shifter
 - Frequency multiplication / division
- Clock Buffers, Muxes (Glitchless)



All these can be connected in clock path
Clock pins, Clock tree



Kuruvilla Varghese



So like current FPGA and PLL and digital clock manager and digital clock manager contains DLL for de-skewing it has free so the input clock can be shifted by 90 degree, 180 degree and so on. It has frequency multiplication and division which is available within the clock manager and there are clock buffer Muxes, with glitches that means there are 2-1 Mux where we can switch between the clock.

And that should create a not in glitch because depending on when you are switching there could be additional adjust the output. So that glitters switching which essentially means that those clock input should be kind of other switched output should price to the corresponding clock, so there will be 2 flip flop is synchronisation crisscrossing and all that we can imagine the circuit.

So the Mux are available and all these devices you know the PL, DLL, clock buffers, Muxes everything can be connected in the clock tree path like between the clock pin and the clock tree and depending on where you require that ok. So that is the basic resources available with respect to the clock management the PLL, the DCM digital clock manager which contains the DLL and which does phase shifting frequency synthesis this skew and all that.

It has clock buffer, Muxes and all that and this work with pop in and you can insert does between the clock pin and the clock tree.

(Refer Slide Time: 45:59)

- Resources

- Buffers
- DLL / PLL
- Block RAMs
- DSP Blocks

- Usage

- Vendor library components
- Inferred by synthesis tool, when possible
- VHDL attributes with code



NPTEL

JEE

Kurusilla Varghese



So this is a little bit of special resources, so you have buffers, DLL, PLL, DCM, block RAM, DSP blocks, all these can be you know instantiated with the when the recompenses are in the case of Xilinx know you have the code tool which can generate this the template code for instantiating it and sometime you write a code and synthesis tool info Xilinx that is a memory and will put it in the block memory or this computation can go into a DSP block.

All that is decided by the synthesis tool and sometime it can write is that instead of using the vendor library component you can write the VHDL code for a memory and it can attributes use attributes in VHDL code to say that use block RAM for that particular code of the memory write in subject code will be in a you will be kind of implementing quotable code, if you use vendor library component you translate this FPGA design to another FPGA from another vendor.

You have to change those components or if you are taking this FPGA design ASIC which is usually the case because the ASIC designers will do the prototyping in FPGA and move to the ASIC. So if used when the library component all that asked be changed in the ASIC designed that can be avoided by using VHDL and attribute.

(Refer Slide Time: 47:34)

- JTAG: Prototyping (PC : Board)
- Master Serial:
 - Configuring from a Serial PROM
 - Embedded boards
- Slave Serial
 - Works as a slave to master FPGA connected to a serial PROM
- SelectMAP



8 / 16 bit wide synchronous slave configuration of FPGA
Suitable for FPGA Interfaces to a CPU



Kurusilla Varghese



And so that is it and we are going to have a look at the configuration of the Virtex FPGA, so previously we had a look at essentially we looked at the metastability the clock period analysis and hold time violation analysis for a datapath and sequential circuits.

(Refer Slide Time: 48:01)

- Previous analysis assumes that the clock reaches at flip flops at the same time, it is not practically true, as the wire delay and buffer delay gets added.
- This creates relative delays between pair of flip flops or registers
- For analysis it is important to consider the clock skew between flip-flops/registers where there is a data path between them.
- Clock Skew:
 - Difference in arrival time of the clock at the flip flops

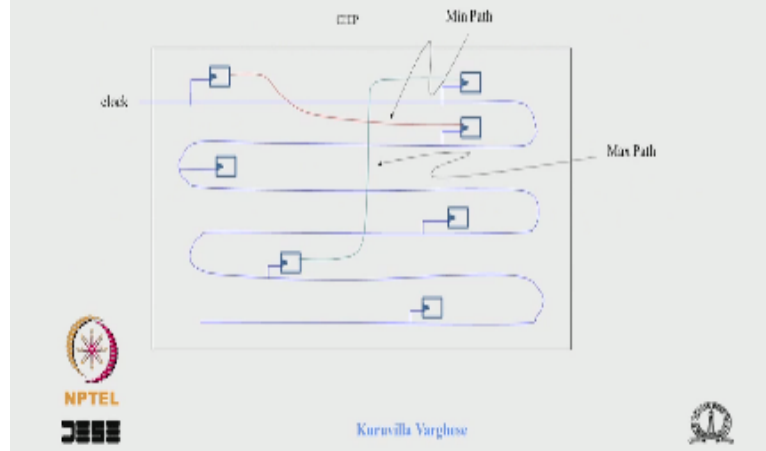


Kurusilla Varghese



And we said we analyse the creatures at the same time not the case.

(Refer Slide Time: 48:06)



In real life so we have put a real life case we have put Mux path and min path and we introduce skew between them and analyse and we have seen that it is troublesome in the max path if there is a problem then you have to reduce the clock frequency in mean path hold time violation is that we have to increase a combination delay and we have looked at the park with routing which gives the minimal kind of relative delay between the flip flop clocks can you look at the DLL which is used for this skewing.

And we have seen what is the difference between PLL and DLL and we also look at what current FPGAs has in terms of the flip flop and the source of flip flop and so let us look at and how these resources are instantiated with VHDL, that is what we have seen. So let us look at the configuration of FPGA briefly.

(Refer Slide Time: 49:06)

Virtex Configuration

97

- JTAG: Prototyping (PC ↔ Board)
- Master Serial:
 - Configuring from a Serial PROM
 - Embedded boards
- Slave Serial
 - Works as a slave to master FPGA connected to a serial PROM
- SelectMAP
 - 8 / 16 bit wide synchronous slave configuration of FPGA
 - Suitable for FPGA Interfaces to a CPU



Kurusilla Varghese



and

And implementing myself to the Virtex and briefly tell you about current FPGA more option than the Virtex of FPGA which is available, so to be in current I will talk little bit about the Spartan 6 with respect configuration. So basically when you are doing the prototyping you can use a clock call JTAG which is different name you say committees name which has come out with this particular port which is called joint as action group.

That is also called TAG which is text in a test access board that is also called boundary scan because that was developed that was developed for the continuity check on PCB with very complex packages like BGA are you know that in a in PCB when you mount chips earlier you had the packages of LTC package and you want to check the continuity you can prove the source pin and the destination pin.

When you put prob one of the source one of the destination now in a package like a BGA or chip the patch come under need the chip and because of the multilayer PCB this goes maybe want and output goes deep in within the layers and through the inside layers of the PCB goes to the destination and reaches IC and when you have a PCB there is no way to kind of prob it because mostly when you manufacture PCB after the PCB manufacturing the chip get mounded .

Then comes to the testing site you know it is not that other PCB is manufactured then it comes back for continuity check and goes again for another component mount because that that gets delayed because unnecessary you are not transport is involved, so PCB gets fabricated and chips get mount on it and so the boundary scan is a test port which is come for the continuity check through the boundary of the chip you know electrical not by probing.

So you can call a kind of improving inside the boundary of the chip through the text access port. So that is used for even FPGA programming, so basically this port is a serial port with minimum number of line not waste, number of pins, so it has a clock pin, data input, data output and you a have a mode select to control a small FSM inside and when you are prototyping you can program and it sound like FPGA from a PC to a small dongle.

Dongle is a small circuit like a kind of a USB pen drive or a very small circuit. So normally you have a USB dongle where output goes to board and you can program reprogram. So this is very useful at the time of prototyping because you are working now I know for the various

iteration. So you do not want to permanently program that into FPGA, so you just program at tested then we work on the design reprogram and all that that is what is a tag is used.

And it is possible to program multiple FPGA trujet 1 because at the multiple devices can be cascaded because there is a data input and data output, data output of the first FPGA can be connected to the data input of the second and so on. So data can you know program a simple device and multiple devices which cascaded, then there is mast serial mode which is like JTAG.

JTAG is a standard which has boundary scan but master serial is a kind of prototyping port from the Xilinx which has a clock very simple clock which a clock and data input ok that case of the master the clock is given by the FPGA to a serial form ok. So literally FPGA clocking the data out of the PROM is a serial PROM which gets in the FPGA program ok and this is very useful for deploying an FPGA based embedded board in a field.

Because in the field you cannot insert that program this FPGA to from my PC. So the configuration is stored in a PROM which is connected with a serial mode to the FPGAs and the power on FPGA gives the clock and program itself ok and slave serial to the master and slave depends on who gives a clock ok, if the FPGA is giving the clock from us but in a Slave serial what happen is that if we get the clock from outside ok.

That is it not that the prom gives a clock but you can think of a serial mode where the FPGAs are cascaded, so 1 FPGA access a master clock in the PROM, also clocking the slave FPGA, that is why this life serial is I know use and there is and these all are serial mode which flow, there is a parallel mode where may be a CPU, a microprocessor can program an FPGA to an 8 bit of 16 it wide path which is synchronous.

So this is a case when you have a processor board and you have an FPGAs, you do not need an additional PROM, the processor will have some flash memory to store its embedded software or the form, so along with that you know the same memory can store the configuration for FPAG and so an 8 bit 16 bit wide path you know you can program at this CPU can program this FPGA.

And in the Virtex FPAG there is only is a master mode kind of you know FPGA and the select this particular parallel path, but in the current FPGA as in slave mode. So I think that I will show one at least one or two example of these one select map and these two together I will show the producer and we also look at in the next lecture what are the current FPGAs supporting in a nutshell and with detail about the configuration.

So that is about the configuration 1 for prototyping 2, 4 type 2,3,4 kind of in the field how to deploy this in the field and the select map is in a parallel programming for parallel configuration which is faster much faster than the serial configuration ok. So I would like to wind it up and you coming to the kind of end of this lecture set on the FPGA lecture module on the FPGA what is left is maybe little bit about the configuration.

And bit about the debugging and a slight a brief look at others FPGAs then we will wind up the FPAG lecture. So please go through the lecture on the FPGA try to understand and grasp it, so that you can when we wind up when we put together a very clear what is happening when I show the tool and so on. So I wish you all the best and thank you.