

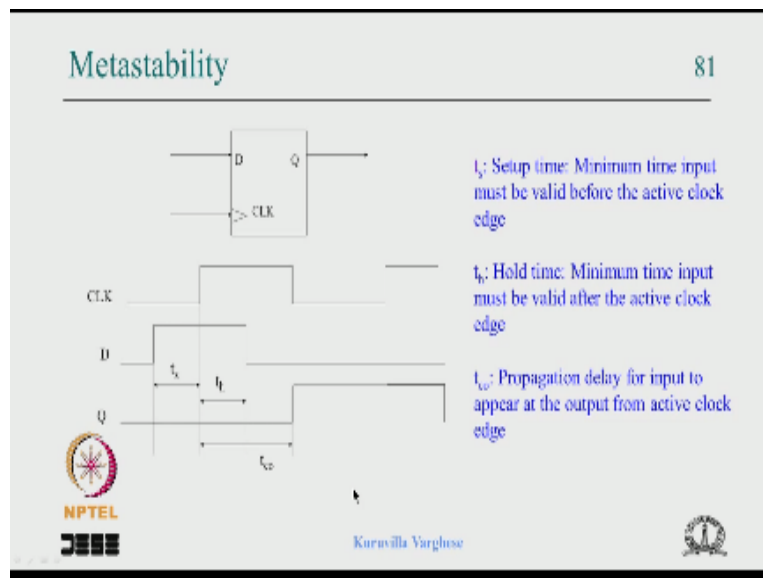
**Digital Systems Design with PLDs and FPGAs**  
**Kuruvilla Varghese**  
**Department of Electronic Systems Engineering**  
**Indian Institute of Science – Bangalore**

**Lecture-40**  
**FPGA Configuration**

Welcome to this lecture on field Programmable Logic array gate arrays in the course digital system design with PLDs and FPGAs, in the last lecture we have looked at the clock tree essentially some basic related to the clock tree why the clock trees are especially clock trees are required and we were analysed the basic timing of data path and sequential circuit in the presence of skew.

And then that is a kind of you know setting the background for the clock tree which gives the minimum skew between the end point and then we have looked at the special resources like DLL, PLL clock managers and all that and we have started the various method of configuring or programming the FPGA. So before continuing with the configuration of programming we will briefly look at the last lecture portion and then we will get on today's part.

**(Refer Slide Time: 01:38)**



On the last lecture we have looked at basically the issue of metastability for a flip flop not get in metastability the input has to meet the setup and hold time.

**(Refer Slide Time: 01:48)**

## 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 Kuruvilla Varghese

And for a datapath or a registered to register path from that require when we develop the inequality for the minimum clock period and we developed the condition for not violation of the whole time ok.

**(Refer Slide Time: 02:09)**

## Minimum Clock period 83

• Sequential Circuit / FSM

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

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

NPTEL JEE Kuruvilla Varghese

And same thing with the sequential circuit or FSM where the register to register path is between the state registers, expressions are saying because if you are kind of show it explicitly you know separate the source and destination it exactly look like this. Now what was missing in this analysis was that we assume in all these in arriving at this expressions.



We assume that the clock is reaching at the source and the destination at the same time which cannot be true in a chip not depending on which way the clock flows the destination could be

lagging or leading the source in terms of the clock arrival. So that should be taken into consideration when analysing this expression.


**(Refer Slide Time: 03:07)**

**Clock skew** 84

- 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





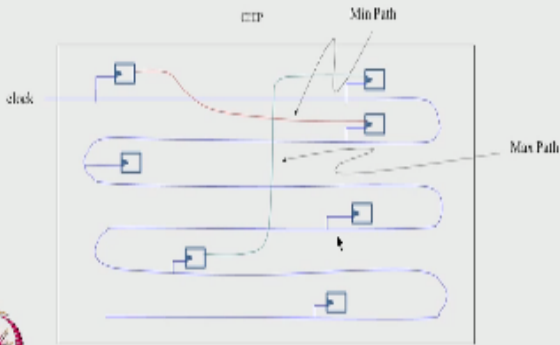
Kurusilla Varghese




And that is what we have done.

**(Refer Slide Time: 03:11)**

**Max Path and Min Path** 85

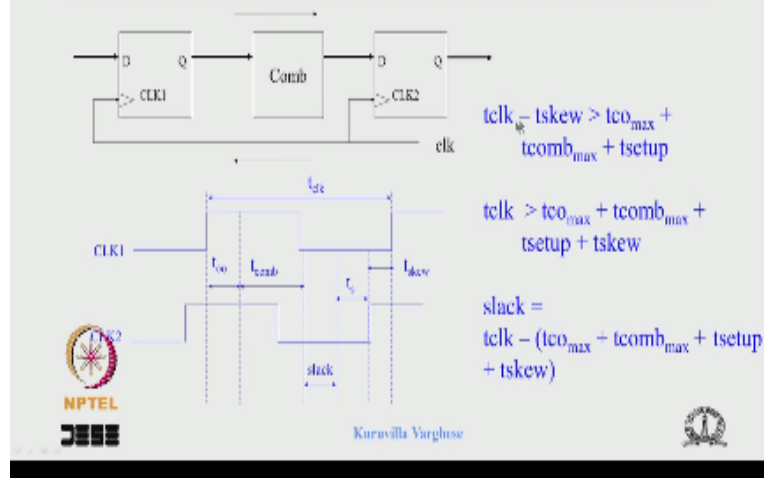


Kurusilla Varghese



And we have set 2 conditions or 2 scenarios where the clock and data flows in the same direction which is called a min path problem and where the clock and the data flows in opposite direction.

**(Refer Slide Time: 03:28)**

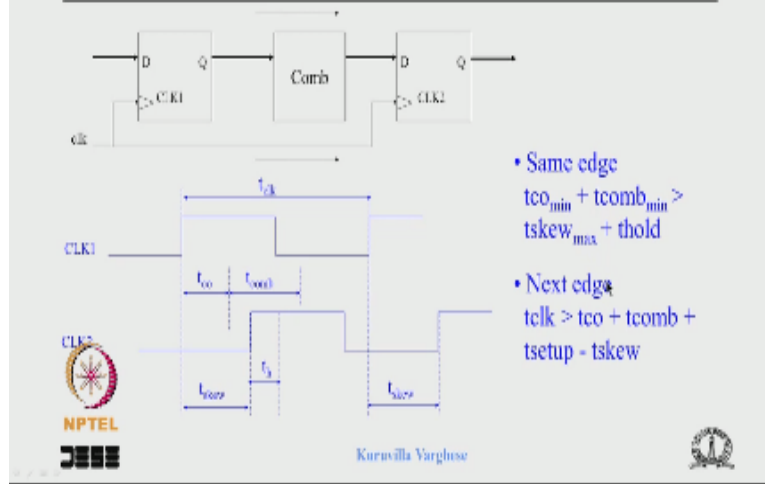


And we have looked at as a case where the clock and the data in opposite direction. Then in that case the source clock is kind of leading or the other destination clock is lagging. So the available time for a registered to register path is reduced by amount of skew. So  $t_{clock} - t_{skew}$  has to accommodate  $t_{com}$  and  $t_{setup}$ . So in a sense the clock period requirement goes up and the frequency comes down.

So because of this skew the frequency of operation is coming down in the opposite problem in the main part problem where are the other source and destination is receiving a clock which is delay version of the source register with respect to kind of you know which was I mean as far as the skew is concerned and in this case looks like the register to register path from 1 clock edge to the next clock edge.

We are in a better position because you have  $t_{clock} + t_{skew}$  is the kind of time available which need to accommodate this path delay ok. So if you analyse from 1x to the next edge things looks pretty comfortable because that is you know easy around but you know if you would have fix the clock period or something.

**(Refer Slide Time: 05:09)**



Then here is relaxation on the clock period for which is good in a way but the real thing as I said is that since the clock is getting skewed maybe this is the datapath is fast then it can violate the whole time because the clock is moving ahead and  $t_{skew}$  and  $t_{com}$  is min then it violate the whole time. So that is what is shown here  $t_{co} + t_{comb_{min}}$  should be greater than the  $t_{skew_{max}}$  and  $thold$ .

So it will not show it can violet and if this violation happens kind of reducing the clock frequency is no use ok, because in earlier problem be sorted out suppose we have a cock period which cannot accommodate this kind of skew then we can reduce the clock period but in a min but problem is not related to the clock period at all because we are talking about the same at the clock ok.

It between the same edge of the clock or between that edges which are delay same edges as delay by the skew, so the only way there is a hold time violation to avoid is increase the combination delay or logic delay. So in a sense clock when somebody is routing the clock in a check, if there is a relative delay between the endpoints ok whenever it is, that is troublesome that can reduce the clock frequency that can create hold time violation.

So there need to a mechanism to kind of round the clock to reset the relative delays between the end points of minimal. So that like that shows you know the with respect to earlier picture we cannot route like this you know that as a clock tree goes it gets in everything gets delayed because of the loading and from this to that end the delay is very high.

So we cannot have such kind of arbitrary scheme of clock routing. The ideal thing would be that from the clock pad to any end point that should be kind of equal length of the wire and equal number of buffers and this scheme does not even help even if we insert buffer here, buffer here, buffer here and all 2 kind of offset delays in offset the loading, that does not help because a buffer delays add to this skew.



So you should have at the buffers with a balance. So if it is ideally like you know in a simple way you can put a buffer then branch another buffer and another branch you know in from the main route one more branch with the buffer and so on.

**(Refer Slide Time: 08:03)**

### Clock Skew: Max path 87

---

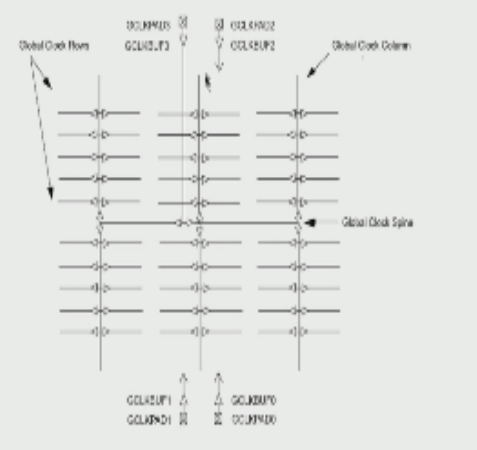
- Analysis for data path from first flip-flop to next
- We assume  $t_{co} + t_{comb}$  is greater than the hold time of flip-flop
- Hence, when a clock edge comes to both the flip-flops, new data from first flip-flop arrives at the second flip-flop after the clock edge, even after the hold time and 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 that the extension of that is a H clock tree where from the input point.

**(Refer Slide Time: 08:06)**

### Virtex Clock Tree 92



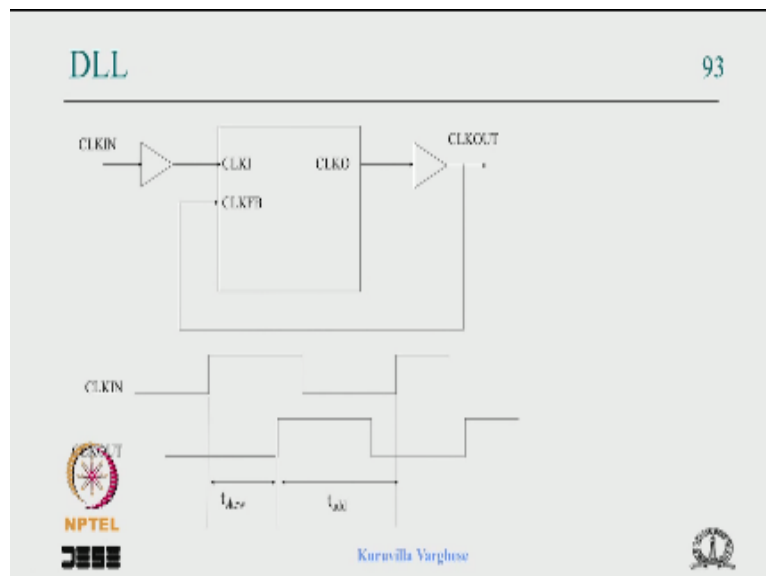

Kuruvilla Varghese
Source: Virtex Data Sheet


You know you go horizontally branch then vertically branch then branch wherever there is branching then you put a buffer ok. Now that does not mean that is just want any point where a single flip flop is connected you know that could be some kind of depending on the buffer capacity that could be 20 or 25 or so many depending on the buffer. So many flip flops are there in the in this branches in the last branches ok.

But like if you have order of magnitude increase in the number of flip flops a new branch vertically in each one from there we can go different vertical branches and so on ok and the idea is that if you taken from say take this end point and this endpoint and if you count the number of buffers up to the end point and take the length of the wires will be identical.

So the relative delays between the edges between end points will be minimal though from the kind of the input to the output that could be you know skew quite a bit of skew which is not much of a problem if that is a problem then can introduce a DLL which compensate for that kind of delay ok.

**(Refer Slide Time: 09:34)**



So that is what is a DLL is you suppose you ever clock tree which is getting loaded and because of the loading up the clock is delayed and there is a skew between them input clock and output clock then the DLL can avoid that because from the output there is a feedback to the input and there will be definitely delay a skew but how it is compensate is at  $t_{clock} - t_{skew}$  delays kind of further added into the tree.

So this edge will come in kind of sink with the next edge ok. So the delay locked loop you know kind of synchronize edge by further delaying already delayed edge you know that is that is how it works compare to PLL.

**(Refer Slide Time: 10:26)**

**DLL / PLL** 94

- 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.

Altera Virtex 5 has PLL blocks in addition to DLL in DCM.

NPTEL  
JEE  
Kurnavilla Varghese

The PLL essentially the scheme look similar you know you have the input clock and the feedback clock, but what is done as that the phase is compared and the phase difference is converted to a voltage with a low pass filter and or semi colon scheme ok and a voltage control oscillator you know synthesize a clock frequency which is not that kind of delayed version of this input clock frequency ask the keys of DLL.

But in a PLL a new clock is synthesize ok. In phase with the input clock and that are certain advantage in the sense that like if there is a given the input clock in a delay locked loop that will appear as a output without delay locked loop just delete the clock but in PLL since it is synthesize and there is a filter which kind of you know filters out the minute digitals.

And the clock is kind of stable okay it is a new clock, it would not have the distortion of the input clock or the detorted in the clock. So that PLL is better than you know that the PLL always act you know with kind of the range of there is a long range and there is time to lock and things like that in the beginning when you start up the PLLs to get in the lock so it takes for the PLL to come in to lock and so that is about the PLL.

**(Refer Slide Time: 12:12)**



- 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



The current FPGAs as DLL digital clock manager which is composed of DLL for disk price if the frequency multiplication, division and skills face like 90 degree, 180, 270 and all that. There are clock buffers, muxes for clocking you know clock glitch and that has to be glitch you know well it has to synchronise the edges, so there will be synchronising flip flops with synchronised you know 2 clock sources to each other. And all this can be connected in the clock or tclock.

**(Refer Slide Time: 12:50)**

- Resources
    - Buffers
    - DLL / PLL
    - Block RAMs
    - DSP Blocks
  - Usage
    - Vendor library components
    - Inferred by synthesis tool, when possible
- VHDL attributes with code



So that is what we have discussed and this can be special resources can be instantiated from the weather library using the code generator tool sometime synthesis tool will info from the code of driving up to write VHDL attribute along with the code to help the synthesis tool to sign up in for what is going on can do kind of inflow what is going on and kind of instantiate the correct component.

**(Refer Slide Time: 13:18)**

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

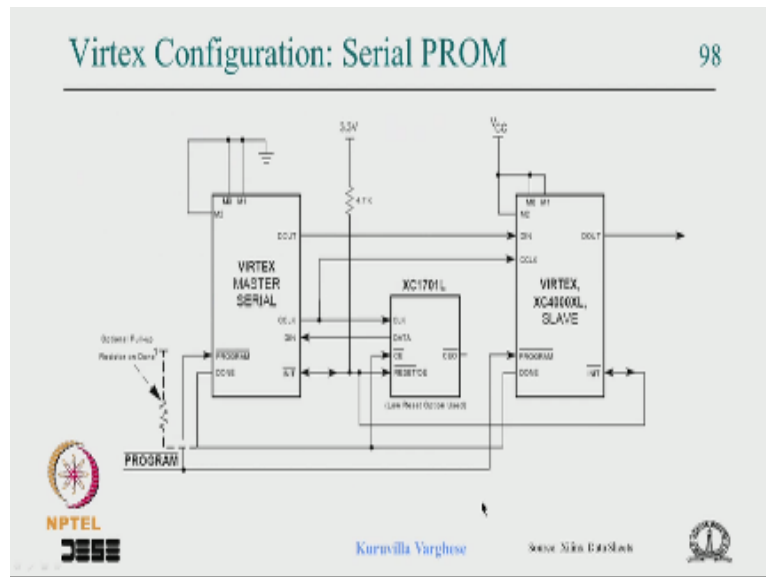
NPTEL JEE Kuruvilla Varghese

Then the last thing we have look that is was the configuration of programming of a FPGA and this is the first thing while prototyping the most suitable method is true asynchronous serial port call JTAG which is used for PCB testing, chip testing and all that which is used for FPGA configuration also. So there be a small dongle connected to PC which can program they FPGA and then there is a serial mode where the FPGA clock a serial PROM.

And get the data and programs at same serial exactly like master serials where in the slave serial FPGA expect the clock is input and the data in with the clock normally this watch in conjunction with master serial for a kind of chained programming of the FPGA and the select map is a bike ride or kind of programming 8/16 bit usually when there is a microprocessor to a parallel buss FPGA can be program.

And configuration bit wide can be stored in the memory of the processor okay wherever it is located maybe in a flash on the board and from there along with the frame where can be stored and at the start up the CPU can program the FPGA.

**(Refer Slide Time: 14:52)**



So this shows a master in serial configuration where 1 FPGA is a master and another FPGA is a slave which is configured by some more pin, 3 Mod pins are there, the current FPGA has 2 Mod pins and see this is in the master mode and this is in the slave mode and you can see there is a serial PROM. The clock for the serial promise given by the master FPGA.

Also the slave FPGA get the clock from the master and the data from the serial PROM is connected to the data input of the master of FPGA and there is a data output of the master FPGA which is connected to the data input of the second slave FPGA. Now if you are third FPGA what you can do is that the data out of this serial port can be connected to the data in of the third FPGA and so on.

And you can combine all the bit stream of all the FPGAs and store in the this particular PROM at the beginning what happens if the power on the FPGAs start programming and or if there is a program pin is there if the program when is made low and you know if you if you can I apply a negative you know 0 pulse then it starts programming anytime, one issue is that at the beginning these FPGA has to clear the configuration memory ok.

Because it could be not a power on programing, it could be a reprogramming of the FPGA while the power is on. So all the other program memory configuration memory has to be clear and depending on their FPGA type and the size it will take variable time. So the question is how do each FPGAs knows that the other is completed the initialisation. So there is a pin which is an open drain output which is also sample in that such an I/O pin which is open drain.

So and it just pull down, so it is like which forming a wired and ok, if everything is high this will be high you one of them is low which is full low ok that is a state of this kind of wired and connection and what happens is that suppose if the master clear the configuration memory and it has come out. Then it will drive the unit output high, so but if this slave is still in the any mode will be driving in low.

Because of this pull of process this still be in low and that is also go to this particular FPGA and that you know sample that input and if this still it is low it waits for other FPGA to finish. So this kind of the unit which is an I/O pin with an open drain which forms of wired and helps the units synchronization between the master and slaves. So ultimately everybody initialize the every FPAG initialise and come out initialize initialization.

And then the master FPGA you know start getting the data okay, you know it gives the clock, it enables the clock and you know you can see that the unit pin is connected to the output enable of the PROM as the unit clock is going unless it need to go high, this chip is not a chip output is not enabled show the data would come. So once the inner face is over on the FPGA start clocking in get the data.

So the first the master FPGA programmes while it is getting program this DOUT is made 1 ok, the 1 is going there, so the slave wait for the there is no starting pattern of the configuration since everything is one it will wait. So the master of FPGA configure itself once a configuration is over then the slave FPGA configuration with comes and what is bypass to the Dout and the slave gets you know program ok and so on.

Now if the second slave while the second slave program that leaves Dout of the 1 and the third sleeve wait, so one by one the FPGA gets program and this is a pin unpin which indicates that FPGAs kind of finished programming. Now again once again this is an open drain output, so which is pulled high, so it forms of wired and again. Unless all the FPGAs done pin is high this would not go high.

So this is an indication to the rest of the circuitry like you will have some like maybe a CPU which is working in conjunction with FPGAs. In that case is done pin is indicator CPU

saying that FPGA finished configuration and normal execution can kind of continuous. So if there is a CPU kind of samples done pin and wait for the FPGA configuration to be over.

Once it is done then will start you know enabling the normal operation or if there is another external circuit we have to make sure that the done pin kind of enables the rest of the circuitry otherwise there will be synchronisation problem with FPGA and the rest of the circuit. Because there is another circuit which gives input to the data, input to the FPGA and if FPGA is not configured yet those data will be lost.

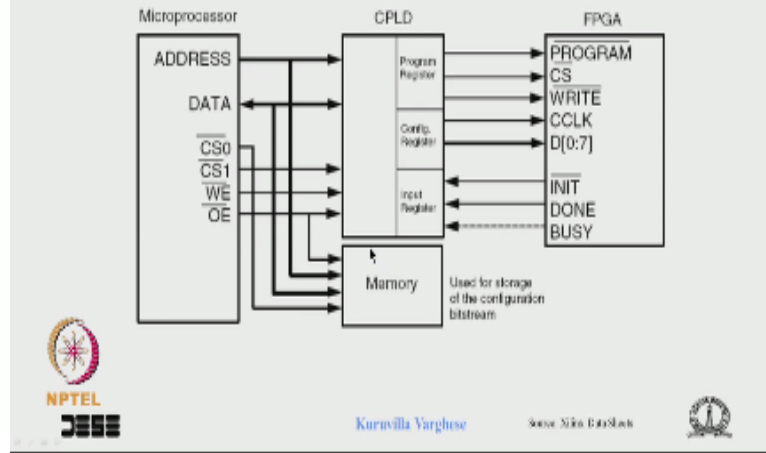
So this done pin has to be somewhere by the rest of the circuit and normally we can assume the rest of the circuit is maybe a CPU or another programmable logic based circuit whatever it is. This has to be taken care while designing. So that shows the kind of very important kind of mechanism to program the FPGA in a chain ok the multiple or this can be single it does not matter.

There is only single FPGA can forget about this and this works and nowadays SPI or a SPI based the PROM can use, so the FPGA offers instead of the custom serial port, SPI port and SPI can work with 1 bit data, 2 bit data and 4 bit data. In addition this Prom itself can be program through the data port permanently. So not only that while prototyping they FPGA can be program to JTAG.

This SPI PROM not this PROM, the SPI type PROM can be programmed to JTAG port which often when you buy an FPGA board. These options will be available you can program normally SPI flash should be connected in this way to the FPGA so that you can program the FPGA to a data port.

This flash through an data port and if you put appropriate either more pin FPGA can be configured from the flash PROM. All that is possible and that is what is written here, all what I have done is kind of elaborated.

**(Refer Slide Time: 23:05)**



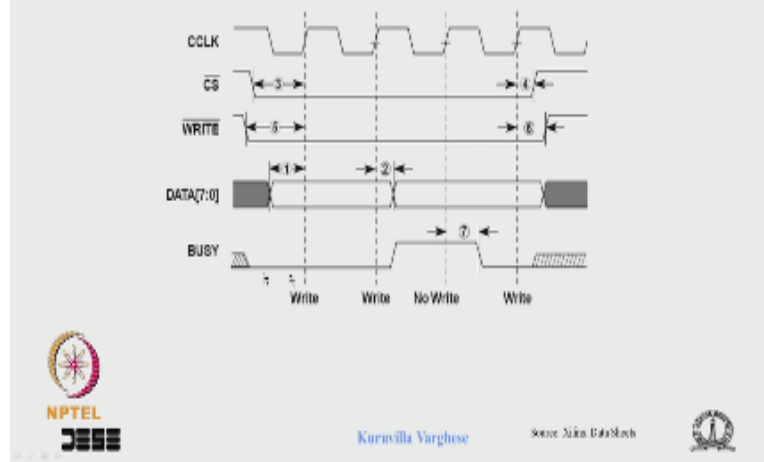
And the select map scheme where you have a CPU and you have a FPGA normally you have a synchronous bus or two bus data you know program the FPGA and this these are the pins you know program chips to select write clock and that is the synchronous bus, INIT DONE BUSY and all that okay. Now the issue with such a bus is that most processor bus are not sometime synchronous okay.

Like the microcontroller bus may not be synchronous, so you have to translate the bus protocol of the process to this synchronous protocol. So that can be achieved by CPLD coming in pin to do the protocol translation of the bus and we have discussed the CPLD in the CPLD part of the lectures and we have mentioned that CPLD is good for bus protocol translation.

So that is the scenario which is shown here where the CPU has a memory which could be flash where they were restored. Now the FPGA configuration bit stream can be stored here, CPU can access that read, then you know maybe byte 1, 1byte is read here, then the bit is written there, again read and write no such a thing can be done using this kind of scheme and if possible the CPU can directly write if there is a synchronous port which matches the product.

Otherwise you will have use some kind of kind of parallel port in CPLD that can be done but which can be very slow sometime when we use parallel port between a CPU because you have to address each port you have to use the port for clocking which can be very slow sometime ok.

**(Refer Slide Time: 25:05)**



And that shows the timing like you know every clock edge the data is coming chip select and the write bar is low and anytime as a FPGA is not writing the data will indicate the busy signal in that case the data has to be kept for 1 more clock you know. So it is like a kind of extending the bus cycle if the peripheral is low, I am sure that you studied that scheme you know normally ready normally not ready kind of system.

So this is a kind of normally ready system if it is kind of nothing if the FPGA does not require more time then you keep clocking data every clock cycle. It requires then it indicate is not ready or it is busy, then you extend the bus cycle by adding extra delays and this is a kind of a very simple scheme with can be implemented in CPLD.

**(Refer Slide Time: 26:09)**

- While FPGA is being configured, its internal state is not defined and pins levels are also not defined.
- Xilinx FPGA has two internal signals to keep the FPGA state sane during and after configuration.
- GTS: This signal drives all FPGA outputs to tri-state
- GSR: This signal goes to all flip flop set/reset and keeps all flip-flops set or reset as reset state specified.
- Once FPGA is configured, these signals are released.



Use separate user resets, for normal reset operation.

So that is what is the kind of cracks of FPGA configuration, there are low current FPGAs are more detail. I will briefly mention it, but for Virtex these are the main ways of configuring it basically the JTAG or the master serials, slave serial and I mentioned about SPI port, then the select map which is byte white which is 8 bit or 16 bit one thing to remember is while FPAG is getting configure the pins will be in tri state mode.

So there are super circuit kind of has to make sure that while you are not been configured these pins the status will be tri stated and it has to be appropriately pulled up or pull down depending on your the requirement of the rest of the circuit ok, the rest of the circuit is sampling one of the output of the FPGA which normally in a default state or you are you are assuming it to be 0.

But while configuration rest of the circuit comes up before the FPAG configuration then assume that one of the input 0 is tri stated, it can create problem, so it has to be pull down and once FPGA is configured all the flip flops are reset using an internal reset line which the FPAG does not advise you to use as a reset you know. So in your circuit you want to search even the power on reset you should implement the separate power on reset than this internal reset signal.

There is a way of using that we set signal within your design it is a very kind of high, kind of lot of flip flops are connected to it is very heavily loaded. So it may be a better idea to reset it separately with you drive.

**(Refer Slide Time: 28:17)**

The slide is titled "Spartan 6: Configuration" and is numbered 103. It lists the following configuration options:

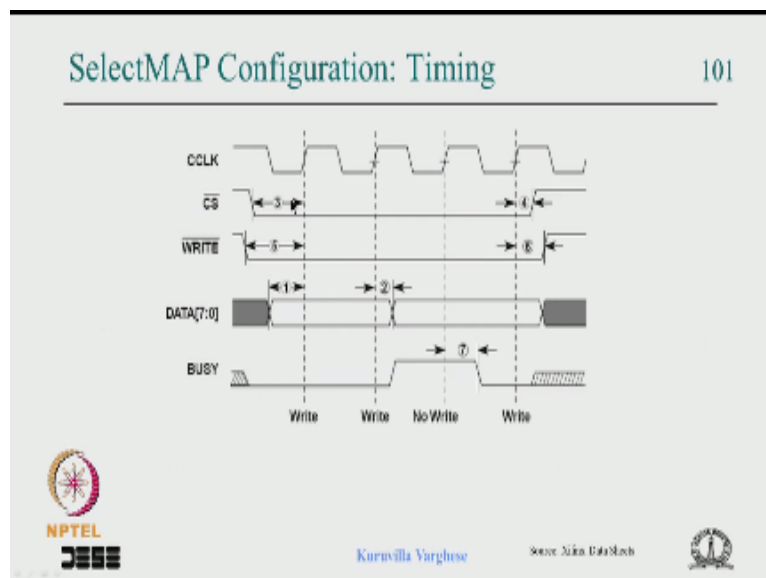
- Boundary Scan / JTAG / TAP / IEEE 1149.1
  - Single Device, Chain
- Master Serial (Chain, Ganged) (SPI: x1, X2, X4)
- Slave Serial (SPI: x1, X2, X4)
- Master SelectMAP (x8, x16)
  - Single Device, Chain, Ganged
- Slave SelectMAP (x8, x16)

At the bottom of the slide, there are logos for NPTEL and JEE, and the name Kuruvilla Varghese is mentioned.



Also this I just briefly mention because there are different you know kind of extended version of that Virtex configuration is available in recent FPGA, so I have taken an example of Spartan 6, it is true of Spartan 6 or kind of Virtex 6 or Virtex 7 or any of the seven series of FPGAs, so your boundary scan which a data port which can configure a single device or like we have seen in the serial mode we can change the multiple devices through the JTAG.

**(Refer Slide Time: 28:58)**



You know exactly like what we have seen here even the JTAG there is a type pin which take in the data TDO pin was take out the data, so TDO pin for FPGA can be connected to the TDI pin. So it can be changed ok. So that is possible so you can in a boundary scan you can have a single device or a chain you can program multiple device. In the master serial you can have chain you know along with slave you can have you know a chain or devices.

Sometime what is required is a thief same configuration has to be applied to all the FPGAs okay that means all the FPGAs are the same type and it is configured by the same device it can be connected in parallel and now in the serial mode if use SPI flash the data bit can be 1 bit, 2 bit or 4 bit. There are appropriate pins and similarly for the slave serial you know normally slave serial walk along with master serial for the chain.

And in the select map you have 8 bit and 16 bit configuration you can have a single device or you can have a chain of devices in select MAP, you can have slave select MAP work the clock is in here the clock is given by the FPGA but in a slave selectMAP if we expect the clock from outside along with the data okay. So it is little more elaborate than the Virtex, we

have studied, so I just mention so that the information is current whatever I have talked about the lookup tables logic.

And all that can be extended to these kind of new devices, but this is additional so I mentioned and another issue with the earlier FPGAs were suppose you come out with the proprietary design you put the configuration bit stream in this PROM ok. Now you send that in the field on a PCB or in a product what can happen is at the power on somebody can capture that is this data, it is very easy because the clock is coming in synchronous data is coming.

So this can be easily kind of reverse and you know somebody can read the bit stream and reverse engineer the complete system ok because in a FPGA the main thing is in this PROM and that is kind of the you know circuit on it which stream is coming as it is it can be easily read. So it be worthwhile if you can be kind of protected and because mostly this will be some intellectual property of the designer or the company which is doing the design.

So what is done in the current FPGAs are that this bit stream can be encrypted using the advance in script encryption scheme which is called ES using at 2 to 6 bit E, now through the JTAG port this FPGA can be program with that and FPGA can be told no read back that means once it is program this key cannot be read back, also the configuration can be read back okay. Otherwise it is possible to read back the configuration for the purpose of verification and so on.

So that cause can be disable and then you deploy that in the system that means you program this PROM with a protected bit stream, then what is going on this kind of this line is ES encrypted and without the knowledge of the key is very difficult break the stream and this can be kind of a key can be very specific to the device, otherwise if a company is making an industry, making as a 10000 devices each 10000 will have separate keys not the same key.

**(Refer Slide Time: 33:33)**

## Spartan 6: Bit Stream encryption

104

- Bit stream is AES encrypted with 256 bit key using BitGen tool
- Encryption key is programmed in to FPGA device through JTAG for decryption.
- Once programmed FPGA can be configured for no read back
- Configuration also can't be read back.
- AES key can be permanently fused in FPGA, Or in an SRAM with external battery backup



Kuruvilla Varghese



So that it can be very well protected. So that is available in the current FPGAs. So there is an ES encryption with the 2 specific key, so the bit stream is encrypted using the BitGen tool with the two specific key encryption key is programmed in the FPGA to JTAG code and once it is program it configured for no read back and the configuration also cannot be right back.

And AES key can be permanently fused like you blowing the fuse in FPGA or you can be programmed in an internal SRAM with a battery back you know with a battery backup which is connected, so one can choose those options if you permanently fuse it you cannot change the that AES key, so it will be permanently program will be forced use the same key for all the bit stream you program into the flash.

**(Refer Slide Time: 34:32)**

## Spartan 6: Bit Stream compression

105

- Bit stream can be compressed when there are lot of resources unused
- Less memory for storage
- Less configuration time



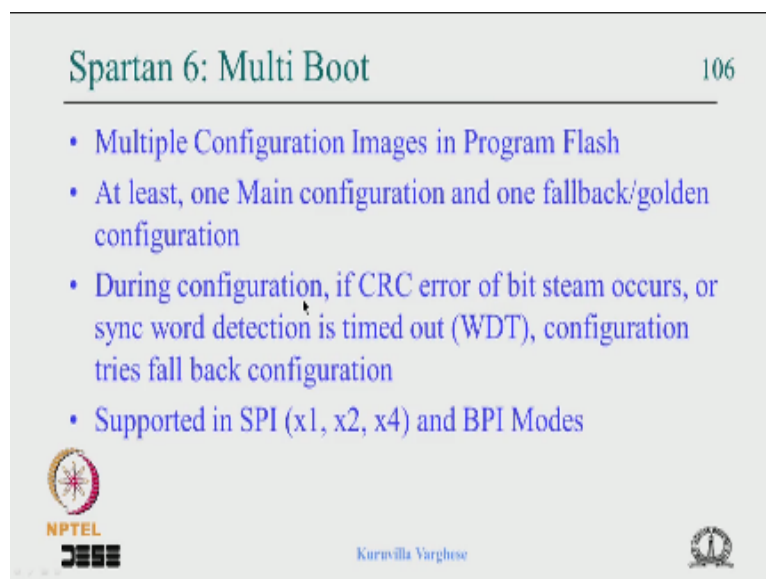
Kuruvilla Varghese



So that way the encryption kind protect the design another option available in the recent current FPGAs are bit stream can be compressed and which is bit older than you know not that this the Spartan 6 even before the FPGA is before have to add this option because there could be a lot of resources which are not use at all use, not configured at all. So there is lot of kind of redundant information that not being program.

So that information is removed and configuration bit stream size can come down, so that has 2 implication one is that you can store it in a lesser memory space and it can be configured very fast, the configuration will take lesser time. Otherwise for a fix device will take you know kind of fixed time for the device is logistics larger time, but even in a large device if it is not used only 50% is use it might take you know kind of less time than the full FPGA configuration.

**(Refer Slide Time: 35:42)**



The slide is titled "Spartan 6: Multi Boot" in a teal font at the top left, with the number "106" in the top right corner. Below the title is a horizontal line. The main content is a bulleted list in blue text:

- Multiple Configuration Images in Program Flash
- At least, one Main configuration and one fallback/golden configuration
- During configuration, if CRC error of bit steam occurs, or sync word detection is timed out (WDT), configuration tries fall back configuration
- Supported in SPI (x1, x2, x4) and BPI Modes

At the bottom left, there is a logo for NPTEL (National Programme on Technology Enhanced Learning) and JEE (Joint Entrance Examination). At the bottom center, the name "Kurusilla Varghese" is written. At the bottom right, there is a small circular logo.

Another possibility which is available is that suppose I know this address is suppose you are programmed a configuration here and the feel like if it is program in a flash and flash memory and sometime the flash can get corrupted you know suppose the flash get corrupted then the whole device may not work, the flash need to be reprogrammed okay. Nowadays it is possible you know you would have seen that earlier the you know the computers used to update you know through the internet.

So the drivers and the software and all that, but now you can see the set top box at your home connected with TV it can update through the cable you know through the cable the former get updated, the mobile phone can get updated with the former over there. So same thing happen

now the configuration through the internet or wireless network can get you know can get to the device.

And the device can be program itself. So that is a possibility, so but still you know in the field if the flash is getting corrupted it will be a good idea if you have a kind of a golden bit stream or a fallback bit stream which may not be the recent one which may be very old one which is stable okay it may happen that you update the configuration over the internet and some corruption has happened.

Then you can fall back on a very stable version which is which is there from the beginning ok so that is what is called multi boot and that is what is shown here in this particular slide so you can have at least one main configuration and one for back configuration and during the configuration if the CRC error like while the configuration with bit stream is read the CRC is calculated.

If there is any bit stream corruption CRC will give error, in that case for back on the golden configuration or if the sync word detection is timed out that means are the beginning there is a sunc word and if that is corrupted and they will be a Watchdog timer which is waiting for it and if it does not come that times out and in that case in for back it can fetch the golden configuration from a particular location and that can program this FPGA.

And can recover from the that particular error and this scheme of multi boot is available only in the SPI base slash form and the BPI base slash form okay, so in both it is available it is a very good option if you have kind of if are deploying FPGAs in the field we should think of you know the multi boot encryption and compression and all that which will kind of you to the reliability improve the security all that part ok.

**(Refer Slide Time: 39:01)**

- Slices to support DSP computations
- 18 bit 2's complement pre-adder
- 18 x 18 bit Multiplier, 36 bit result
- Result is sign extended to 48 bit
- 48 bit 2's complement adder/subtractor



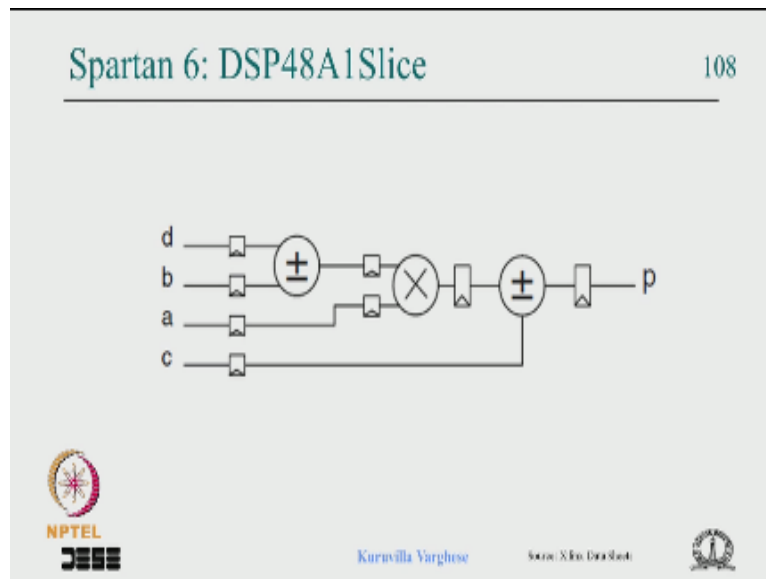
Now the current FPGAs in addition to the DLL, PLL block memory and all that you have the DSP slices which allows the designer to implement other DSP algorithms a very efficiently and the DSP algorithm normally used fix point computation 18 bit may be used, so there are and you know that in one of the major operation in signal processing algorithm is multiply and accumulate.

So which requires a multiplier and an adder ok, so the DSP box the Xilinx FPGAs DSP blocks give you this particular option you have a pre-adder that means there are 18 bit 2's complement pre-adder that means you can do signed addition 18 bit in the pre-adder and that added output can go to a multiplier which is an 18 bit\*18 bit multiplier. So a two 18 bit can be added and that can go to multiplier.

It can multiply another 18 bit which is coming from a separate port it produces a 36 bit result. Now this can be sign extended to 48 bit and it is followed with a 48 bit 2's complement adder subtractor and there are various you know if not done everything need to use you can use multiply along with the post order bypassing the multiplier you can take the multiplier result outside 36 bit result.

So all kinds of options are available in the DSP slice which really enables one to implement the DSP like filtering algorithm the encoders, decoders and so on. All that can be implemented very efficiently in the DSP slices and many times you just use the multiplier operator and then if the data size matches then it can implement in the DSP, slices can be instantiated and you can use it.

(Refer Slide Time: 41:16)



So this is a general architecture you have four port, all are kind of pipeline and to you can see two ports are added it was 18 bit result which is combined with 18 bit multiplier this output can be taken out or can be taken to in another order which is 48 bit or sign extended chat with you no sign extension that output is available. So this helps in implementing the DSP kind of algorithms.

And this is call DSP48A1 slice in the Spartan 6 and in addition I mention I have not put it in the in the slide but I should mention that the current FPGAs allows you to use the lookup table as shift register because look up table is normally suppose a form would look up table you have 16 flip flops inside serving as a location. So that is connected in a change and is available as a shift register.

In this case it is called SRL 16 and that can be changed for SRL32 and even higher that 64 bit shift register 2 fit in some kind of Spartan 6 slice and that can be in a change the cross. So that is you know you get a shift register implementation without using the flip flops of the slice because a number of slice limited and we have seen Virtex are only 4 and 2+ slice in a 4 CLD.

(Refer Slide Time: 43:21)

- Probing the internal signals in FPGA for debug.
- Signal Probe / Logic Analysis
- Use a Signal Capture IP
- Interface this IP to the JTAG port
- PC based software to configure signal capture IP and display the signal waveforms



Xilinx: ChipScope Pro

Altera: Signal Probe



Kuruvilla Varghese



So if you say implement kind of 16 bit shift register then you will end up using 4 CLB but using one lookup table, you can implement 16 bit shift register. So that is possible to implement and one other important very important thing with respect to FPGA design and kind of discussing all what is reminding you know one problem of the FPGA is that you design something in FPGA.

You know you have verified you know you have done behavior solution timing solution everything but the moment you put the FPGA for whatever reason is something go wrong, you cannot and of there is internal signal you cannot kind of debug it you cannot see what is happening inside, so what is done is the FPGA vendors give you a logic analyser circuit which is route to the JTAG.

Now you can instantiate is logic analyser it keep along with your design and the probs of the logic analyser can be connected to the internal signal and through the JTAG there will be a software which is available at the tools site and through the JTAG you can capture the waveform of the internal signal and you can analyse it and like logic analyser you can trigger say it will be kind of crazy.

Because suppose you have a some kind of 8 bit data line you want to monitor ok and your clock is 200 megahertz and in one second there will be kind of 200 megahertz kind of data passing over the data bus if it is clock at frequency then to analyse that you are to store all that there are not enough memory. So you can if you know that you are very kind of hint saying that what could be going wrong.

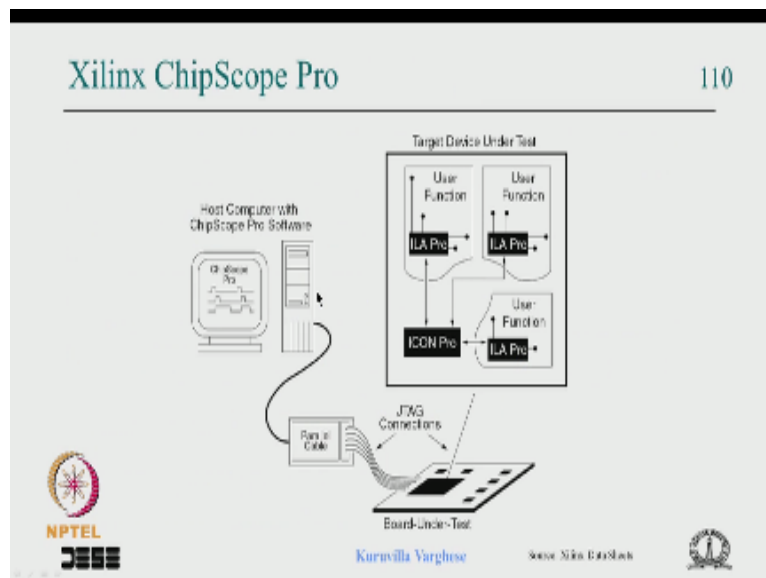


But on a particular data there happens then you can trigger on the data and capture some data around the trigger point or after the trigger point or before the to the trigger point, so you will use logic analyser you would have saying that pre trigger, post trigger, pre to the 50% and so on that, it depends on how much you capture before that how much time after that and analysis is always offline.

It is not that you are on a trigger point you capture some data and you analyse offline on the PC and try to debug it, so it is a great tool to it is like once you have some complex circuits going to the FPGA to debug this logical licence have to be used and the Xilinx call chip prob and the Altera call signal probe and it is very easy now over the years it has become a very nice tool, only thing is that it because is IP this circuit occupy some kind of space within FPGA.

So if you are not really floor planning in the rest of the circuit it can upset the timing performance of your design some time but then if you properly design floor plan that not very serious issue because it does not occupy much space.

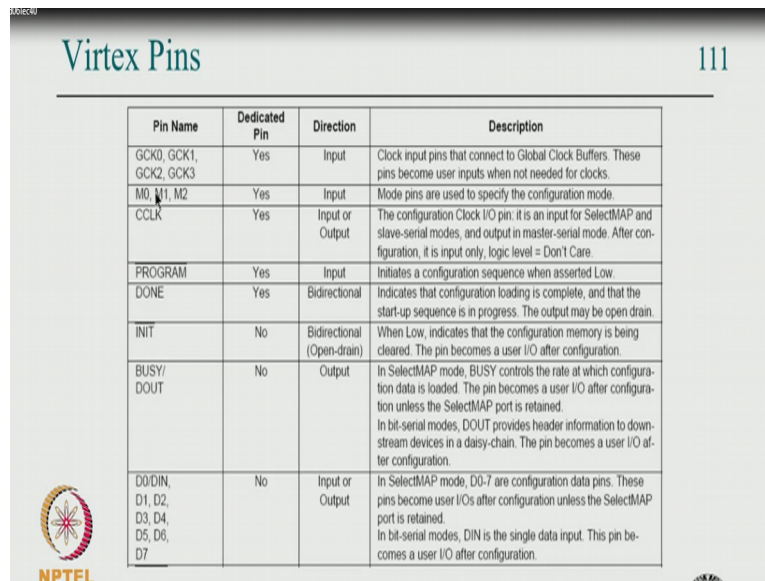
**(Refer Slide Time: 46:36)**



So that is shown in a picture here you have the FPGA board you have a FPGA what is done as that this is the kind of the blocks instantiate like you have a ILA pro it is a logic analyser Pro which is which has prob points which can be connected to the user port and this is the one which is connecting to JTAG and on the PC side you have a software which on the

trigger it captures data transfer it to the PC and you can analyse and take action, you know that is what is the signal probing does.

**(Refer Slide Time: 47:15)**



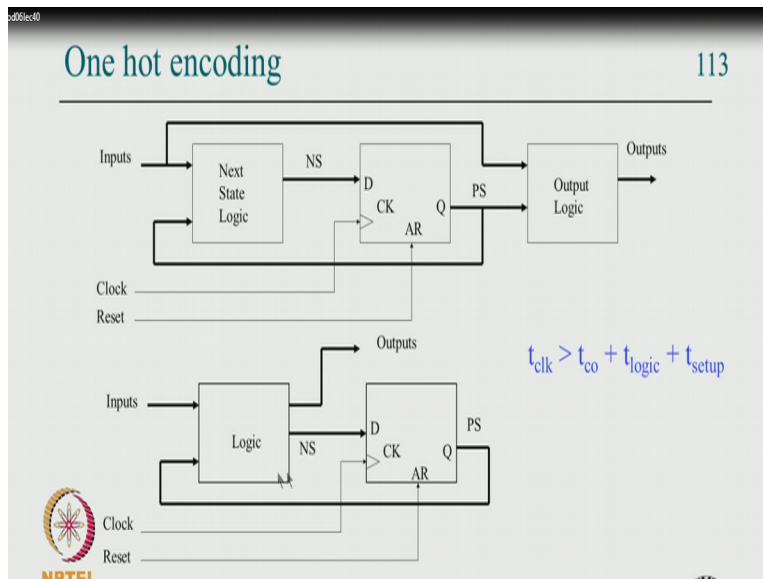
**Virtex Pins** 111

Pin Name	Dedicated Pin	Direction	Description
GCK0, GCK1, GCK2, GCK3	Yes	Input	Clock input pins that connect to Global Clock Buffers. These pins become user inputs when not needed for clocks.
M0, M1, M2	Yes	Input	Mode pins are used to specify the configuration mode.
CCLK	Yes	Input or Output	The configuration Clock I/O pin: it is an input for SelectMAP and slave-serial modes, and output in master-serial mode. After configuration, it is input only, logic level = Don't Care.
PROGRAM	Yes	Input	Initiates a configuration sequence when asserted Low.
DONE	Yes	Bidirectional	Indicates that configuration loading is complete, and that the start-up sequence is in progress. The output may be open drain.
INIT	No	Bidirectional (Open-drain)	When Low, indicates that the configuration memory is being cleared. The pin becomes a user I/O after configuration.
BUSY/ DOUT	No	Output	In SelectMAP mode, BUSY controls the rate at which configuration data is loaded. The pin becomes a user I/O after configuration unless the SelectMAP port is retained. In bit-serial modes, DOUT provides header information to downstream devices in a daisy-chain. The pin becomes a user I/O after configuration.
D0/DIN, D1, D2, D3, D4, D5, D6, D7	No	Input or Output	In SelectMAP mode, D0-7 are configuration data pins. These pins become user I/Os after configuration unless the SelectMAP port is retained. In bit-serial modes, DIN is the single data input. This pin becomes a user I/O after configuration.

So I am just showing the vortex pins you have the global clock pins which is dedicated which is used which is the input to the clock tree, the more pain for selecting the mode of programming dedicated pin, see clock which is the serial clock which is can be reused as a user IO program done in BUSY, the parallel port all that can be used once the programming is that it can be used as a kind of this program is dedicated.

But this pin is dedicated but these are I can be used as user IO and the rest all these you know this is a programming pin and there is a this is a JTAG which is dedicated, PMS is more select, TCK is tclock, TDI is data input and TDS data, output then you have the internal DCC, IO ground and so on, so normally the FPGA this is a scenario you will have clock pins, you have some dedicated JTAG port dedicated more pins and the programming pin some are dedicated most of it is kind of can be programming is over can be used by the user ok.

**(Refer Slide Time: 48:35)**



So there is one point I want to mention here is something about one hot encoding which is a state machine encoding which is used in FPGAs this is you see the this is a state machine block diagram we have in next state logic which is decoding the input and the present state and or a 2 block diagram you know enough of it but what can happen is that you have taken example to kind of in order to put the background for the need of one hot encoding.

**(Refer Slide Time: 49:15)**

One hot encoding 114

- e.g. FSM with 5 inputs, 18 states, and 6 outputs
- NSL:  $5 + 5 = 10$  inputs (worst case)
- For Virtex (Worst Case)
  - Basic block: 4 input LUT
  - 1 CLB → 6 input LUT
  - 16 CLB's for 10 input LUT
- NSL would be distributed increasing the delay bringing down the clock frequency of FSM.

Solution: one hot encoding, where each state is encoded using a flip flop.

NPTEL

Kuruvilla Varghese

Suppose you assume a finite state machine with 5 input 18 states and 6 output. So when you have any 18 states it means that you have you need kind of a binary encoding 5 flip flops greater than 16 less than 30 to 5 flip flops are required and there are 5 in boards so if you look at the next state logic and it will get 5 input and 5 state variable. So there are total 10 input to the next state logic ok.

But you know that there are 5 flip flops, so there will be D4 to D0 each of that may not use all the input, but we assume the all the in case all the 5 inputs are used less likely but then let us assume the worst case so and we assume that so there are 10input for next state logic and once again we assume the worst case in the Virtex that means we have seen that the five input function can be implemented by cascade or 6 input or 7 input can be implemented with two 4 input lookup table.

But we assume again worst case for the given for the sake of argument, so here we have 10 input next state logic. So one so for 1 CLB can implement 6 input lookup table by you know two 4 input make a 5 input, two 5 input make a 6 input so on ok. Now when you have a 10 input lookup table that means that you need 16 CLBs because you can implement 1 CLB I can implement 6 input look up table.

So you need two 4, 7, four for 8 and so on ok so you will end up with 16 CLB I definitely have kind of excited because is not the case at all the input, all the next state decoding will require all the external input and it is even if there are really 10 input for some kind of decoding then that may not require the lookup table which is 10 input lookup table, one can cast, but assume the worst case it happened one can kind of come out with such a scenario.

Then what happens is that this next state logic is spread in multiple CLBs and the interconnection of that make it slow. So this clock frequency which is  $t_{co} + t_{logic} + t_{setup}$  and we are talking about the logic delay of the next state logic and that can become very high and it can bring down the clock frequency of the state machine and we have analyse data path and we have tried to kind of be very aggressive in the kind of the timing of the data path.

And suppose we have you know designed a high-performance data path but if the state machine is low then nothing can work, so it is not a good solution you know the problem here is made by the next state logic which is very complex ok, so the question is how to kind of reduce the complexity of the next state logic. So if you look at it again you come back to the slide we are 5 input, 5 state variables.

So the question is can we reduce the number of state variable, so and we have some kind of binary encoding witnesses e a 5 flip flops for encoding 18 states. So why not encode 18 states

in 18 flip flops ok. So why do coding decoding S states you know when decode a state you do not need 5 bit processing that 1 state you can take 1 bit presenting that is basic idea.

**(Refer Slide Time: 53:24)**

The slide, titled "One hot encoding" (slide number 115), illustrates a state transition between two states,  $S_i$  and  $S_j$ . A transition from  $S_i$  to  $S_j$  is labeled "condi". State  $S_j$  has a self-loop labeled "condj". Below the diagram, the logic equation is given as  $D_j = \text{condi} \cdot Q_i + \text{condj} \cdot Q_j$ , and the next state logic (NSL) is noted as "5 + 2 inputs (Worst Case)". The slide also features the NPTEL logo, the IIT Bombay logo, and the name "Kunavilla Varghese".

So you have their each state is a flip flop ok. So you take this kind of state transition like  $S_j$  get 2 transition, one from the previous  $S_i$ , one condition I and  $S_j$  there is a self loop which is condition j. So the  $D_j$  because this is nothing but  $A_{sj}$ ,  $D_j$  corresponding to this stage a is nothing but condition I and  $S_i$ , but  $S_i$  is just a single flip flop. So we say is  $Q_i$  or conditionJ which is composed the worst case 5 input.

And  $Q_j$  because this state is represented by a single flip flop  $Q$  ok,  $Q_j$ , so you have the worst case 5 inputs specifying this in good condition and 2 inputs for  $Q_i$  and  $Q_j$ , so you have 7 input next state logic which require only maybe 2 CLB which is less spread and then we have less logical delay and the timing the clock frequency becomes a manager ok. That idea one hot encoding.

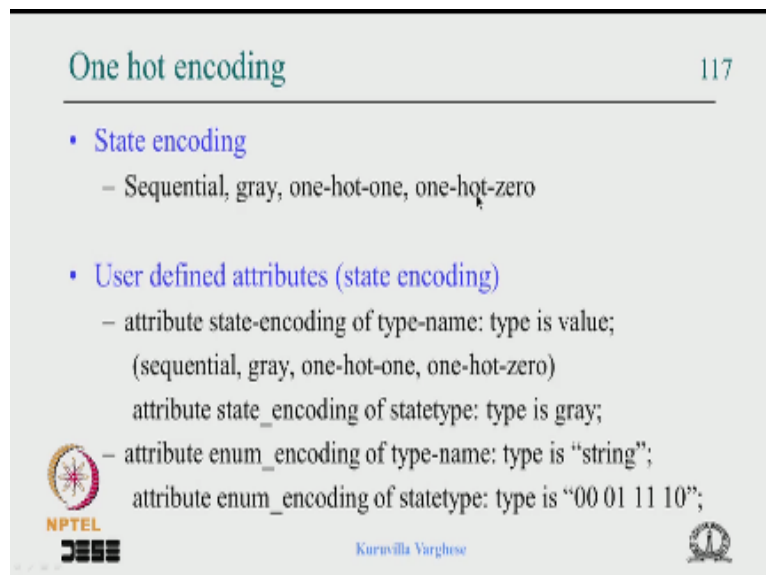
So there is many a times a people blindly like when you see a FPGA you just say ok but just be one hot encoder is not required, suppose you have four state machine with 2 input and 2 output absolutely there is no need to go for one hot encoding because this can be encoded used 2 flip flop and with 2 input the next state logic can go in one lookup table 4 for particular flip flop.

And there is absolutely no need of going for one hot encoding. So but when the number of states are more, number of inputs are more then you can go for one hot encoding which eats

of the flip flop but the timing become better which definitely is at the cost of XR flip flop but the timing is better. So this can be kind of control using some kind of attribute in VHDL code you can have a state in coding like sequential which is by binary like the state 0 will have 00 state 1 is kind of 1 and the grey code.

And this is one-hot-one and one-hot-zero which is a single flip flop for a particular states, you know that is one-hot-one and one-hot-zero and this can be controlled using some attributes like saying suppose you can say suppose you have to find a state type called you know enumerated state have caused a type can say attributes state encoding of state type, type is grey for one-hot-one and one-hot-zero and things think like that.

**(Refer Slide Time: 56:27)**



The slide is titled "One hot encoding" and has the number "117" in the top right corner. It contains a bulleted list of state encoding options and user-defined attributes. The list items are:

- State encoding
  - Sequential, gray, one-hot-one, one-hot-zero
- User defined attributes (state encoding)
  - attribute state-encoding of type-name: type is value; (sequential, gray, one-hot-one, one-hot-zero)
  - attribute state\_encoding of statetype: type is gray;
  - attribute enum\_encoding of type-name: type is "string";
  - attribute enum\_encoding of statetype: type is "00 01 11 10";

At the bottom left of the slide, there are logos for NPTEL and IIT Bombay. At the bottom center, the name "Kunivilla Varghese" is written. At the bottom right, there is a small circular logo.

We can specify attribute you know encoding of state type, type is you can say S0 is this is S1, this S2, S3 can literally specify the state encoding ok and now this is the vendor dependent this is not a part of the VHDL, this is a user defined attributes, so you have to refer to the vendor tool manual whether they support it okay. So we are coming to the end of the lecture. So I will complete this part in the lecture.

Maybe another 15 minutes I will be able to complete the FPAG part then we will look at some remaining VHDL part, so today basically we have completed the configuration have looked at what is implemented in the current FPGA in terms of the configuration we have looked at the bit stream compression, bit stream encryption, multi boot, then we have looked at this one hot encoding where the next state logic complexity can be reduced.

So that the state machine works with at a faster clock or so that the data for it is in sync with the data path and that one-hot encoding can be control using some attributes that is what is we have discussed today and what is remaining is very minimal we will look at complete this part. And we will look at some FPGAs from other vendors very briefly one from the Altera and one from the Actel and wind it up. Please review the portions I have covered, so that you are sync and I wish you all the best and thank you.