

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

Lecture-41
Altera and Actel FPGAs

Welcome to this lecture on field programmable gate arrays in the course digital system design with PLDs and FPGAs. Last lecture, we have looked at the Xilinx Virtex configuration or programming, and we have also looked at certain features which are not available in the Virtex kind of device and what is the extra feature which is available in the new devices, citing Spartan 6.

Then we have looked at the issue of a kind of 1-hot encoding. And we have to kind of finish it, they were little detail left towards the end of the lecture, we will complete that, and in today's lecture we will look at similar device or FPGAs from other vendors like Altera and Actel not very kind of in depth, because we have already gone through the Virtex device in depth.

So, I presume that you will be able to understand that other architectures very easily you know you spend time, then you will be able to make out everything. So, before going to today's lecture part. Then we will look at the last lecture's portion briefly for continuity then we will continue with today's part. So, let us move to the slides.



(Refer Slide Time: 01:57)

1

Digital System Design with PLDs and FPGAs

Field Programmable Gate Arrays

Kuruvilla Varghese
DESE
Indian Institute of Science



Kuruvilla Varghese

(Refer Slide Time: 01:59)

2

Virtex Configuration

- JTAG: Prototyping (PC \rightleftharpoons 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

Kuruvilla Varghese

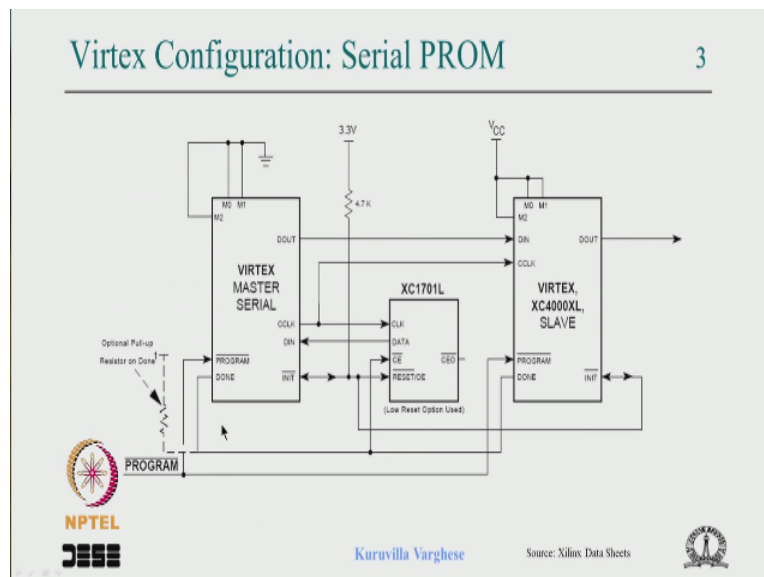
So, in the last lecture we have looked at the Virtex configuration one is through the JTAG port for prototyping anytime. Then master serial is where the FPGAs connected to a serial prom. And in the Virtex it was only 1 bit data. But in the current devices it can 1 bit, 2 bit or 4 bit serial data to enable to support you know larger device. Because for a single bit it takes configuration time will be quite high.

So, 4 bit should reduce it by 4. And the master serial would means that there is a master FPGA clocking the prom in a slave serial the FPGA expect the clock and the data. The data in

synchrony with the clock to be supplied to the FPGA. This is useful when in the serial mode, if you change multiple FPGAs a master will clock the serial prom as well as the slave FPGAs.

Then there is a select map mode which is a 8, or 16 bit wide mode. 8 bit in the Virtex, but 16 bit in the current FPGA. This also can be in the master mode of slave mode in the master mode . The FPGA give the clock in the slave mode the FPGA expect the clock again this can be chained in the current devices.

(Refer Slide Time: 03:37)



So, this shows a you know the this JTAG thing I am not showing, because it is simple, you normally a JTAG USB dongle is there connected to the PC which convert the USB to JTAG and JTAG is connected to the JTAG port of the board. And you can program using the programming tool anytime. Either you can program, the FPGA or the serial PROM even the serial PROMs have you know the JTAG port.

So, it is possible to program the FPGA in that case every power on it has to be program if you program the serial PROM. Then it is kind of permanent because you put the FPGA in the master mode. And if you program the serial PROM it is kind of non volatile each time at the power up FPGA clock is a PROM, elect the PROM and program itself.

(Refer Slide Time: 04:33)

3



Source: Xilinx Data Sheets

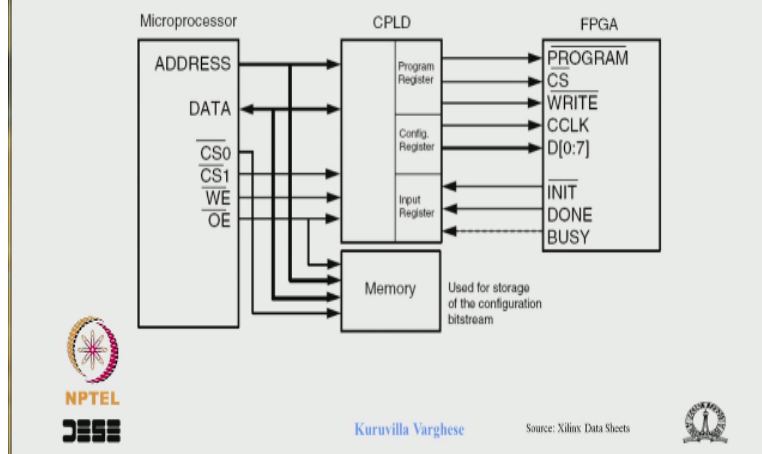


And you know the last slave programs last. So, this done pin is again wired and through the pull up with open drain and when it goes high the rest of the circuit knows that programming is done it can continue the operation while programming all the pins are tri stated and rest of the circuit has to take care. And all the flip-flops at the end of the configuration is reset by internally by the FPGA.

(Refer Slide Time: 06:09)

SelectMAP Scheme

5



And in the select map scheme the FPGA support a 16 bit or 32 bit parallel configuration interface. These are the additional signal and normally this does not match the protocol of the CPU. So, either you have to use a CPLD or use some parallel port to make control to make this particular interface. The only issue that, in the case of parallel port this can be pretty slow if it is software control.

So, it is worthwhile to have a CPLD kind of built in so, that this is done much more faster. And the configuration bit stream is stored in the CPU memory to save space and that is shows a timing.

(Refer Slide Time: 06:58)

Spartan 6: Configuration

8

- 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)



Kuruvilla Varghese



And in the current FPGAs as I said you have the boundary scan which can be single device or change as in the serial configuration the master serial it can be chain or ganged that means the serial devices identical devices can be parallel in gang chain is what we have discussed and the serial prom can be 1 bit, 2 bit, 4 bit. The prom itself can be programmed through JTAG port, and the slave serial again these devices can be 1 to 4 data bit.

The master serial the FPGA you know clocks a BPI flash with in 8 bit or 16 bit mode can be single device can be chain or gang, and there is a slave select which goes along with the master select which is where the clock comes from the external source.

(Refer Slide Time: 07:57)



Spartan 6: Bit Stream encryption 9

- 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

NPTEL JEE Kuruvilla Varghese

And the next thing we have discussed was that in this kind of scheme bit stream kind of open expose people can reverse engineer your design by copying the configuration bit stream. So, the current FPGAs have encryption that means the bit stream before programming can be encrypted with an AES algorithm with the 256 bit key. And FPGAs programmed through the JTAG port with that particular key.

It can be permanently fused or it can be stored in a battery backup RAM inside, where the battery backup is outside and once it is programmed certain configuration can be returned not to read but the configuration or even the key all that can be blocked okay. So, that is the encryption scheme. And there is a bit stream compression because only maybe a part of the FPGAs configured.



So, if the bit stream contains the information about the use of the resources it can be kind of removed from the configuration 2 reduces the storage requirement and the amount of time spending configuring. So, that is called bit stream compression it is useful where you have less memory space for storing the configuration stream or you want to configure at the power on much faster than what is usual.

(Refer Slide Time: 09:29)

Spartan 6: Bit Stream compression

10

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



Kuruvilla Varghese



Because, every device depending on the device complexity takes certain time so, when you compress it as a less configuration time.

(Refer Slide Time: 09:37)

Spartan 6: Multi Boot

11

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



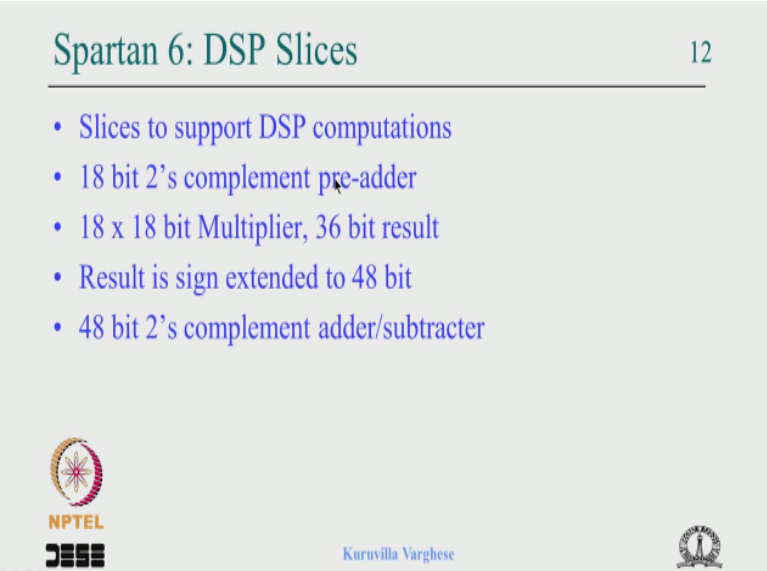
Kuruvilla Varghese

And the another option is that, if the bit stream you are programming if it is corrupted then one into reprogram it. So, it is possible to store a golden configuration in another place in the flash, and which works with the SPI flash at BPI, BPI is the platform flash with the 8 bit or 16 bit into face which also has JTAG so it can be program through JTAG. In that case if during the configuration any times as CRC error on bit stream occurs.

Then it can fall back to a golden configuration not a particular sight, particular location in the memory or if there is a sync word detection failure there is a watch track of which is watching and it times out. Then the FPGA try to configure from the golden bit stream so, normally in a whether it is embedded or FPGA golden would mean certain default very tested configuration which is not upgraded which is kind of gap from the beginning.

And the main configuration is the warn which gets up gradation in the field and so, there can be kind of corruption. Because you are writing to the flash quite often in that area and the bit stream itself could be kind of buggy or you know because it comes you know it gets downloaded through the net. So, it possible to of corruption so, this multiple takes care of that.

(Refer Slide Time: 11:28)



The slide is titled "Spartan 6: DSP Slices" in a teal font, with the number "12" in the top right corner. Below the title is a horizontal line, followed by a bulleted list of features in blue text. At the bottom of the slide, there are three logos: NPTEL on the left, the name "Kuruvilla Varghese" in the center, and a circular institutional logo on the right.

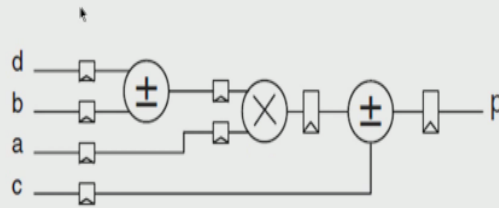
Spartan 6: DSP Slices 12

- 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

NPTEL
Kuruvilla Varghese

So, that is about the multiple these switches can be use in the current FPGA as DSP slice.

(Refer Slide Time: 11:32)



Kuruvilla Varghese

Source: Xilinx Data Sheets



I said a pre-adder which is 18 bit a multiplier which is 18 bit which is giving 36 bit result which can be sign x into 48 and add with the 48 bit. It is very useful in the DSP kind of algorithms where you are multiplier accumulate kind of operation. You can have various you know output of this bypass you know adders, cascade, the multiplier output taken out. All possible possibilities are there.

I am not showing it here, like the CLP you can go through this the detail schematic which should give you some good idea about it this helps in implementing the DSP algorithm very efficiently, and the tool vendors many a times so for a MATLAB and Simulink interface for such algorithm. So, implement those algorithms in the using signal processing tool box or Simulink.

Then that can be kind of equivalent VHDL code can be generated that essentially use as the DSP slice. And that can be kind of synthesise place and out in the FPGA to generate the bit stream.

(Refer Slide Time: 12:50)

- 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

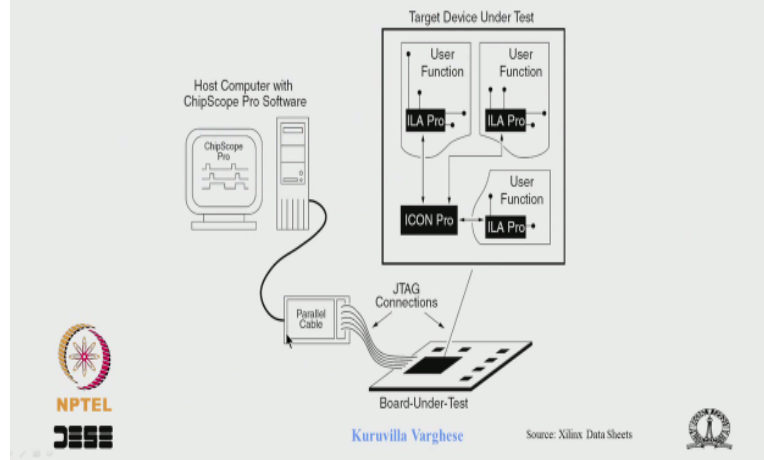


Another issue in a FPGA is that you program the FPGA then it is it is very difficult to kind of debug. You know you are program the FPGA and everything was you know timing simulated, but when you program it, it does not work as a it is intended, and then you have to debug. It is a big problem because only the external pins are accessibility you, if the something is wrong internally then it cannot be seen.

So, what is done is there a along with the connected to a JTAG there is a logic analyser IP, and the probes can be connected internally through your circuit interfaces inside. And you can check trigger saying that on this data boss on this particular signal lines if a particular value comes from there onwards, you capture or you capture before, and after sometime and transfer it through the JTAG port to the PC to do an offline analysis and debug it.

So, it is a very useful tool once you program the FPGA and that is called in the case of Xilinx it is called Chipscope pro and the Altera it is a signal probe

(Refer Slide Time: 14:15)



So, that shows the picture you know you have instantiate these kind of logic connected to the user function and capture it, and analyse it in change the code rebill and you know go on to iterate and very important thing to remember is that, it occupies certain areas. So, if you trying to use the chip scope pro, then you have to have some free area to put that IP. And sometime that can little bit mess up your timing.

Because unless you floor plan properly the presence of that circuit can upset the place and out if you are not doing anything. If you leave the place and route to the tool completely then it can kind of flatten everything. And all logic and get mixed up within the area you cannot say that particular you are one of the module will be very close together nothing like that, maybe the this particular Chipscope IP.

And you are logic everything can get mix so, whatever timing achieve maybe upset by the presence of that. But if you floor plan properly there should not an issue.

(Refer Slide Time: 15:33)

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.



That issue keep in mind and you have looked at the pins the clock pins in the programming mode pins. And few pins like the c clock program done INIT is not a dedicated pin. These are dedicated pin VCC ground the JTAG port rest are polar user define port. Even for the select map interface the user define pins are use, and after the programming.

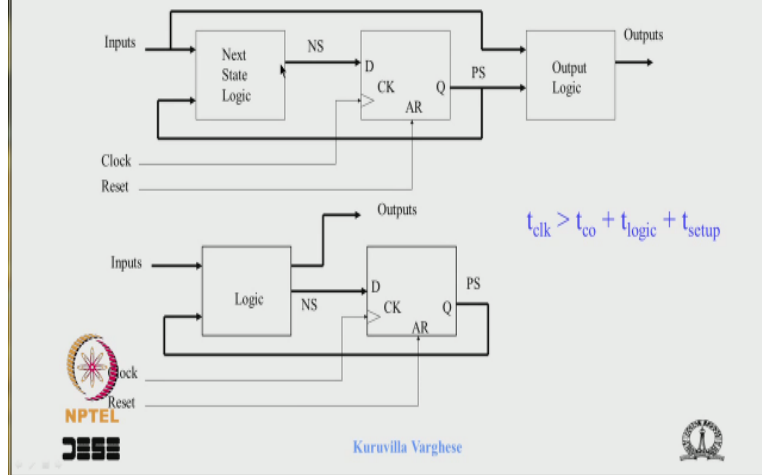
(Refer Slide Time: 16:04)

WRITE	No	Input	In SelectMAP mode, the active-low Write Enable signal. The pin becomes a user I/O after configuration unless the SelectMAP port is retained.
CS	No	Input	In SelectMAP mode, the active-low Chip Select signal. The pin becomes a user I/O after configuration unless the SelectMAP port is retained.
TDI, TDO, TMS, TCK	Yes	Mixed	Boundary-scan Test-Access-Port pins, as defined in IEEE 1149.1.
DXN, DXP	Yes	N/A	Temperature-sensing diode pins. (Anode: DXP, cathode: DXN)
V _{CCINT}	Yes	Input	Power-supply pins for the internal core logic.
V _{CCO}	Yes	Input	Power-supply pins for the output drivers (subject to banking rules).
V _{REF}	No	Input	Input threshold voltage pins. Become user I/Os when an external threshold voltage is not needed (subject to banking rules).
GND	Yes	Input	Ground



It becomes user defined pins depending on the programming at the power on, that, can depending on the mod pins. These pins will have the special programming features at the power on.

(Refer Slide Time: 16:18)



The next thing the last thing we have discuss was the one hot encoding. The issue is that if you encode do a binary encoding of the number of states say, if there are 5 flip-flops. And the flip-flops come here 5 inputs. In the worst case, when we say worst case because many a times in state machine you make decision on a particular signal or combination of 2 signal you do not combine all the signals.

But then let us assume there is a scenario where to a state there is transition from other state on different condition. So, that can happen, so when you develop the equation for that state there could be say 5 input conditions along with the state okay. So, such a thing in happen, so the effect is that this can become because you have 5 input, 5 flip-flop.

You have a 10 input logic implementations required for each flip-flop. And again once again in the worst case it might require a 10 input CLB which can SPAND quite a lot of logic blocks making this very kind of spread in the FPGA increasing the logic interconnect delay and reducing the clock frequency okay.

(Refer Slide Time: 17:45)

One hot encoding

19

- 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 \rightarrow 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.

Kuruvilla Varghese

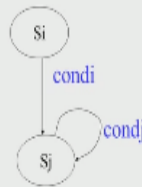


So, that is what I have shown here a particular case of an FSM, where the first case you can go to 16 CLB which can make T logic very high.

(Refer Slide Time: 17:53)

One hot encoding

20



$$D_j = \text{condi} \cdot Q_i + \text{condj} \cdot Q_j$$

NSL: $5 + 2$ inputs (Worst Case)



Kuruvilla Varghese



And we said the solution is to encode it is state in a flip-flop. So, that as for as, the state decoding is concern. There are number of inputs+the number of flip-flops wherein you know there is transition to that particular state you know. And that we have seen an example where Si on condition i transit to Sj, Sj on condition J remains there. Then the equation will be Dj will be condition i. Qi + condition J. Qj.



Q_j because S_i is kind of represented by single flip-flops Q_i , S_j/Q_j . So, this is 5 input condition. So, as I said less likely. But then mostly it will be 1. But then let us take the worst case it will be only 7 inputs not a big deal.

(Refer Slide Time: 18:51)

One-hot encoding Output logic

21

- Most Moore outputs are direct decode of a state or decode of more than one state
- If output is a decode of a single state, then that state flip-flop output is the output signal
- In case of multiple states produce an output, the output signal is the logical OR of all those state flip-flops
- Thus, one-hot encoding reduces the output logic also, at the cost of extra state flip-flops



Kuruvilla Varghese

And one more with this kind of one hot encoding is that many outputs should be Moore type output and in that CASE output will be high on a particular state many a time or in 1 or 2 states. So, the output logic will become just the q output of a particular state or of multiple Q s okay so, in one hot encoding not only the next state logic become simple.

The output logic also reduces, that should be kept in mind. The output logic will be or of 1 or flip-flops when the 1 in the case of 1 it can be taken out directly. So, even in timing wise it will be much faster than binary encoding. So, that is about the output logic.

(Refer Slide Time: 19:50)

- 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";



And the state encoding can be kind of manipulated not manipulate it can be change using the user define attribute. This I have briefly mention, when we looked at the state machine coding using VHDL okay. And this coding after you know specifying the state type this can be mention in the VHDL code. So, 1 example I am showing here say here there is a state type which is define as you know this particular name state type.

So, you say attribute state encoding of state type you say type is gray that means the gray code or type is one-hot-one or one-hot-zero okay. This is one-hot-encoding we will see, what is one-hot-one and one-hot-zero okay. And there is another possibility you say attribute enum encoding for a numerated encoding of state type you say type is and you literally say the encoding okay. Here we are assuming there are 4 states of 2 flip-flops.

And you are saying the first state is 00 01 11 10 which is nothing, but a gray code okay. Now mind you this is called a user defined attribute. So, it means that user would mean does not mean the designer here, it means the vendor okay, vendor of the tool okay. And so you have to check this it is not a standard VHDL. It is a user defined attribute. So, from vendor to vendor it can change it does not kind of you know guarantee that this will work for suppose if it is work for Xilinx it will work for Altera okay.

So, you have to check the tool vendors manual recommendation with regards to this attribute to change the state encoding okay. Suppose you are not able to okay now 1 little bit different between one-hot-one and one-hot-zero say.




(Refer Slide Time: 22:08)

One-hot one, One-hot zero

23

- One-hot one
 - 00001
 - 00010
 - 00100
 - 01000
 - 10000
- One-hot zero (Almost one-hot)
 - 0000
 - 0001
 - 0010
 - 0100
 - 1000

- Easy to initialize (reset all flip-flops)
- Starting state is never revisited



Kuruvilla Varghese

One-hot-zero, one-hot-one means true kind of one-hot-one like we have 5 states. Then you have 00001, because this flip-flop represent the first state, 00010 this one represent the second state and so on. So, you have 5 patterns represent the 5 states okay. But the issue is that sometime at the power on you have to come to starting state and if you are using resets for it it will be troublesome because you need a particular flip-flop to be 1.

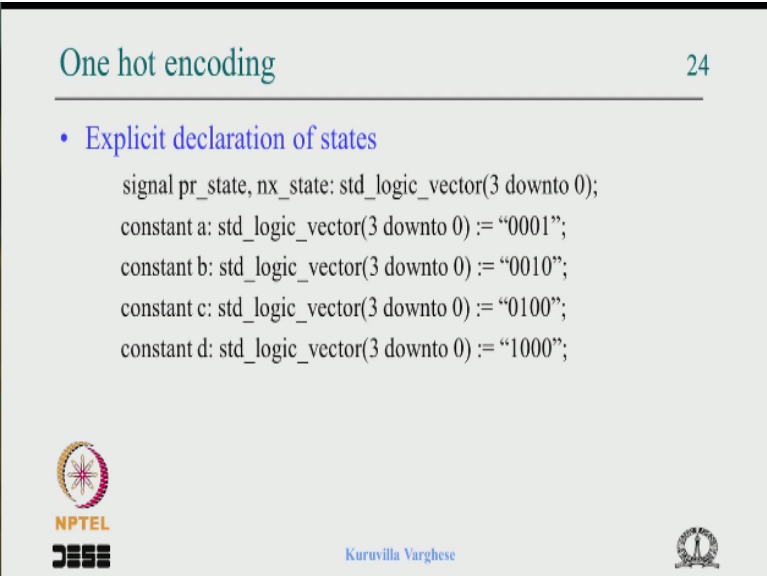
And reset is not there I mean set is not there you have to build otherwise into the next state logic, in that case the reset part of the flip-flop going waste. Sometime it is useful to start with all 0s. Then move to this kind of pattern okay. So, mind you that this is a kind of dummy power on state at the beginning it comes to this state, and but for all practical purposes it transit to the next state 001 then never goes back there.

It is only just for starting, then you are real state machine what you design start with 001. So, you have a dummy state which could be a replica of the starting state as far as the outputs are concerned. But then this is never kind of visited again. So, that is one-hot-zero which is most useful in the case of one-hot-encoding that you should keep in mind.

And suppose you are not able to control the state encoding for whatever reason using these attributes which is less likely n mostly the synthesis tool option will support this one-hot-one, one-hot-zero you do not even have to use the attributes many a times. But you have to check that, you have to check the tool how that this is handles. Mostly the synthesis tool options will help you in the case of Xilinx shave a synthesis tool called excess t Xilinx synthesis tool excess t.

And that has the command line option for these kinds of thing in the GUI there are pulled down properties synthesis properties where these encoding can be changed. And if you are not in, suppose you are somehow stuck with you want to kind of hot code the one-hot-encoding.

(Refer Slide Time: 24:53)



The slide is titled "One hot encoding" in green text at the top left, with the number "24" in blue at the top right. Below the title is a blue bullet point labeled "Explicit declaration of states". Under this bullet point is a block of VHDL code: `signal pr_state, nx_state: std_logic_vector(3 downto 0);`, `constant a: std_logic_vector(3 downto 0) := "0001";`, `constant b: std_logic_vector(3 downto 0) := "0010";`, `constant c: std_logic_vector(3 downto 0) := "0100";`, and `constant d: std_logic_vector(3 downto 0) := "1000";`. At the bottom left are the NPTEL and JEE logos. At the bottom center is the name "Kuruvilla Varghese". At the bottom right is a small circular logo.

Then you can explicitly define the state. Normally we use a enumerated data type for present state and next state. But then you cans say signal present state, next state is standard logic vector 3 down to 0. Because we have only 4 states, and you can literally say constant a is standard logic vector 3 down to 0. 0001 constant b is like that so on. So, we are defining the one-hot-encoding, and for that state you use a, b, c, d like that you know that is the basic idea.

So, we have kind of come to end of the FPGA I have in detail I have discuss the Virtex, and some of the features in the current FPGAs. Some issues like one-hot-encoding, now I am sure that you will be in a position to understand all the Xilinx FPGA, you spend some time looking going

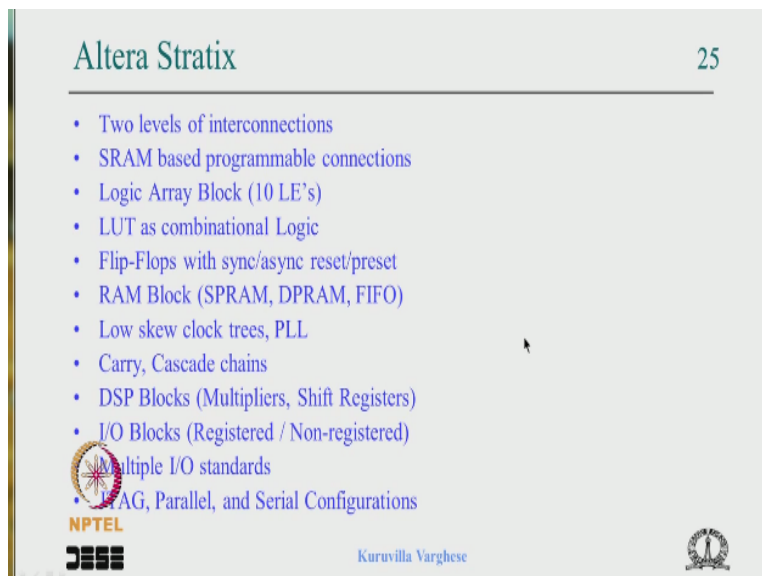
through the data sheets quite a lot of them are there. There are some part which I have not covered.

But you can look at it, there could be packages and package details so that you can like when you make a PCB there are guidelines how which package to use and how to do the lay-outing of the PCB how to route it properly, how to terminate certain high speed signal with termination all that you can refer to the manual. I am not detail you know discuss the basic electrical characteristics, timing characteristics.

All that can be looked at the data sheet. But internally the logic the routing special resources, configuration certain example of how these internal resources are used how it is map to the standard circuit all that I have covered. So, I am sure that you are in a good floating to now to understand other FPGA from Xilinx and from other vendors. But before closing down I want briefly very briefly look at Altera and Actel devices.

Because Altera is also a kind of commercially as a good share in the market and Actel has a very dedicated kind of technology for the space and high reliability application. So, very briefly we will look at it.

(Refer Slide Time: 27:27)

A presentation slide titled "Altera Stratix" with the number "25" in the top right corner. The slide lists various features of the Altera Stratix device in a bulleted format. At the bottom left, there are logos for NPTEL and IIT Bombay. At the bottom center, the name "Kuruvilla Varghese" is displayed. At the bottom right, there is a small circular logo featuring a person's silhouette.

Altera Stratix 25

- Two levels of interconnections
- SRAM based programmable connections
- Logic Array Block (10 LE's)
- LUT as combinational Logic
- Flip-Flops with sync/async reset/preset
- RAM Block (SPRAM, DPRAM, FIFO)
- Low skew clock trees, PLL
- Carry, Cascade chains
- DSP Blocks (Multipliers, Shift Registers)
- I/O Blocks (Registered / Non-registered)
- Multiple I/O standards
- JTAG, Parallel, and Serial Configurations

NPTEL IIT Bombay Kuruvilla Varghese

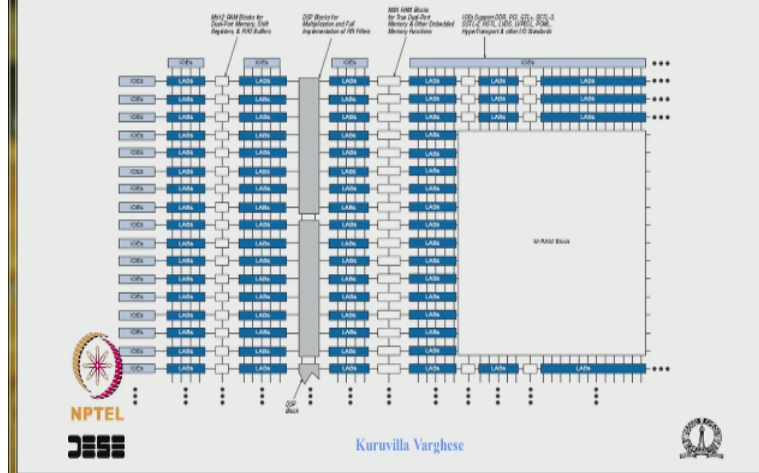
So, let us turn to that part, the Altera has there main stream high performance, FPGA cause Stratix series which is you know Stratix 2, 3, 4 I think 5 maybe the I am really not in kind of in touch with these devices. So, but you can refer to it pretty much everything is exactly similar to Virtex you know. If you look at the latest FPGAs the Altera and Xilinx will be kind of similar features they offer maybe 1 is better in some ways 1 is better in some other ways.

So, first thing to note is that, there is little bit difference in the interconnection very marginal difference which is 2 levels of interconnection. So, little difference you know in Xilinx FPGA you have interconnection metric and logic block. And we have seen that the adjacent logic blocks are interconnected you now directly. So, in the case of Altera this is little more extended.

That is the basic difference here and SRAM based programmable connection no difference, logic array block is 10 logic element. So, you have that lookup table flip-flop such then 10 elements are connected together in the case Virtex you have 4 you know lookup table flip-flop combination there in a logic block lookup table is a combinational logic no difference. Flip-flop with synchronous/ asynchronous reset/reset DPRAM, SPRAM, FIFO all that is there.

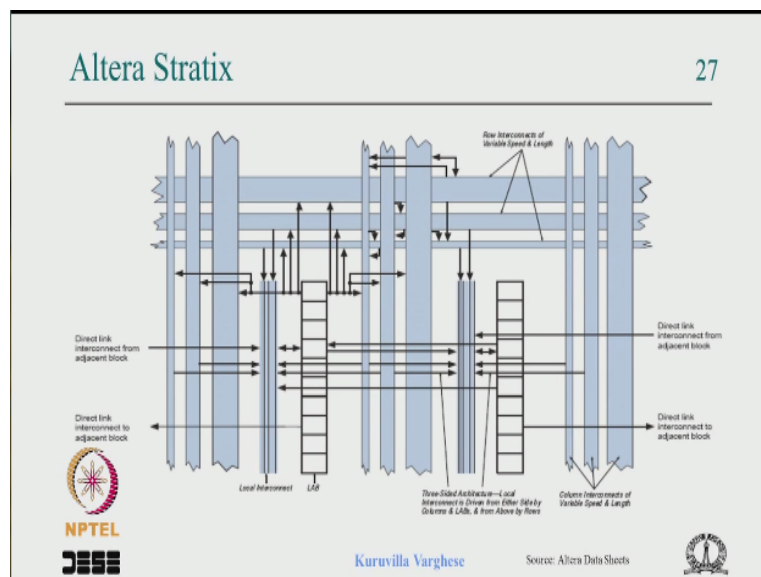
And the low skew clock trees and PLL the PLL of are there in the latest Xilinx FPGAs are other version of Virtex, carry, cascade chains same. DSP block, multiplier, shift registers again in the Virtex series it may not be there. But in the later Virtex series it is there in the Xilinx. I/O blocks, registered/ non-registered everything is same multiple I/O standard same JTAG parallel and serial configurations it same. So, it is very much identical if you know you know the other.

(Refer Slide Time: 29:52)



Just I am putting a very large you know high level block diagram which is this is taken from the Altera data sheet and you have the I/O pins, you have the logic array blocks you have the DSP blocks here I have know these are the memory blocks, and these are the DSP blocks and there is a big RAM block here. So, that is what is pretty much architecture which is very much identical to Virtex.

(Refer Slide Time: 30:23)

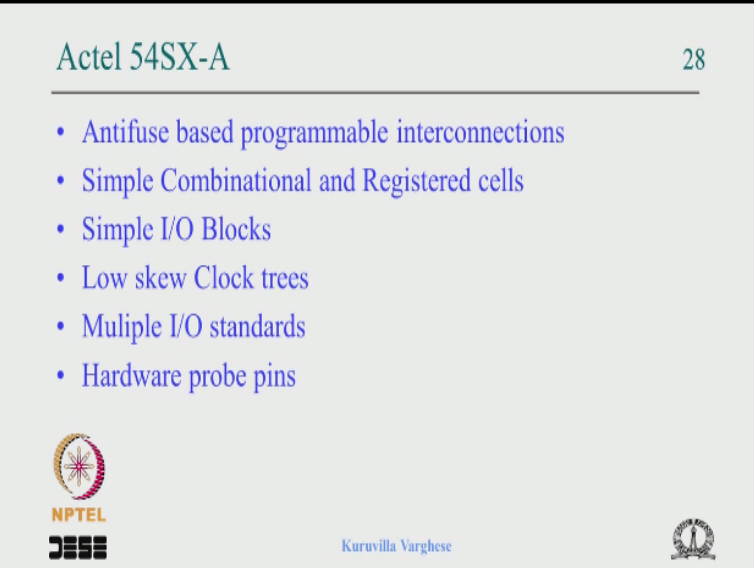


And when I say 2 levels of interconnection this is what I mean you know, you have vertical-horizontal metrics with interconnection. But 10 logic elements you can see that they are connected to adjacent logic element and between the logic element, there is rock kind of local

interconnection. This is the kind of maybe a slight difference between the Altera and statics with regard to this FPGA.

But the later FPGA one need to look at it how it is so, I think I will kind of will not elaborate more these are kind of taken from the Altera data sheet. These information you can go through there later, latest FPGAs but it is more or less similar to the Xilinx some FPGAs.

(Refer Slide Time: 31:16)



Actel 54SX-A 28

- Antifuse based programmable interconnections
- Simple Combinational and Registered cells
- Simple I/O Blocks
- Low skew Clock trees
- Multiple I/O standards
- Hardware probe pins

NPTEL JEST Kuruvilla Varghese

So, let us look at the Actel FPGAs. This is quite different you know Actel 45SX-A really old series. But very useful for certain application essentially, because it has the anti-fuse as a programmable interconnections. So, at the beginning we have looked at the general interconnection technologies, and we have said that, anti-fuse is an one time programmable solution.

So, if you send some something to space you know in a satellite then it is expose to radiation in the outer space. And the RAM or a flash can get corrupted you know. You cannot even think of even it is radiation hard and you a SRAM base FPGA. If it is send to the space there will be frequent corruption. It has to be kept on reprogram if somebody send it probably on an hourly basis.

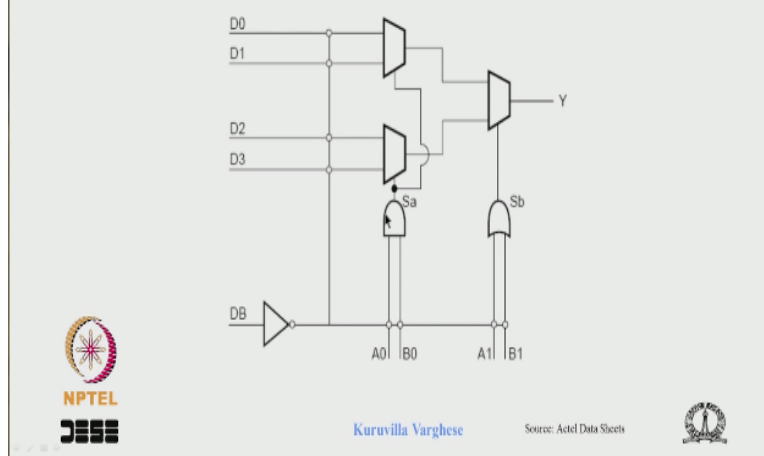
The bit stream has to be kind of put back you know, that is the only way to keep it same. But if you think of an Actel kind of FPGA. It is a fuse connection you know, it is a permanent connection nothing it get corrupted. So, the space and military basically anything to with the space where the radiation is there you can use it maybe it is used in some weapon technologies.

I am not very clear about it, but at least it is used in the space application. And the one issue we have discuss at the beginning is that. Since the anti-fuse technology take less space and less time with regard to the delay. There will be the simple logic blocks okay. So, there are combinational blocks and register blocks which is separate.

In the SRAM base FPGA everything is put together. But in a in Actel 54SX the combinational lookup table like thing is separate. And the flip-flop is separate, and you can have a kind of certain ratio by choosing some certain clusters okay. Very simple I have blocked low skew clock tree, multiple I/O standard and I must say that the Actel is the one which had hardware probe pins built into the chip okay. We talked about the internal logic probe or logic analyser IP connected to JTAG.

But Actel had 2 pins extra on the chip, and through JTAG one could connect the internal signal to this 2 probe pins, and capture the probe pins to debug. So, I think it was a kind of original idea at that point in time to provide 2 dedicated probe pins. That can be probed you know that can be. So, instead of taking the probe through the JTAG, it was taken through a pin. But the configuration was through the JTAG okay.

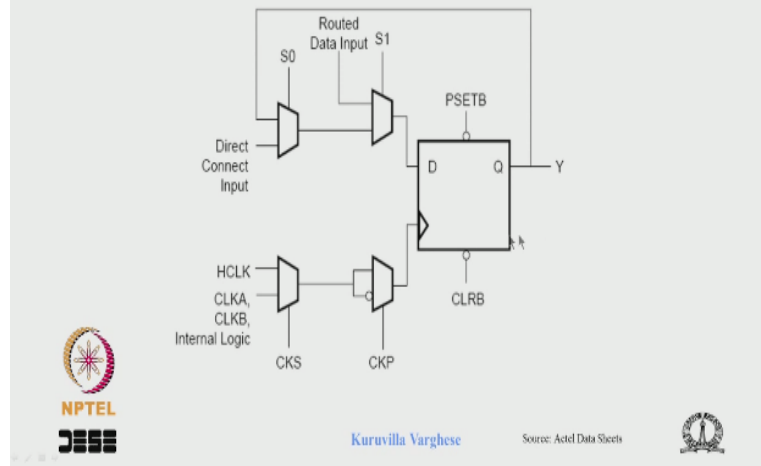
(Refer Slide Time: 34:50)



So, if you look at the combination cell essentially it is a 4 to 1 multiplexer in the case of like we have discussed the fine grain FPGA architecture. We have shown this as an example. So, 4 to 1 can implement 2 variables say, you connect a and b here, then all the mid terms are available here. You can select it by 1 and 0. You can implement a logic function but it is possible to connect the third variable at the input and implement 3 variables.

But because of the presence of this AND gate and OR gate. Instead of connecting you know kind of 1 signal you could connect 2 signals and some combination of the mid terms of 5 may because 2 here, 2 here and 1 here 5 inputs can be implemented in the combinational cell. And this goes to a flip-flop.

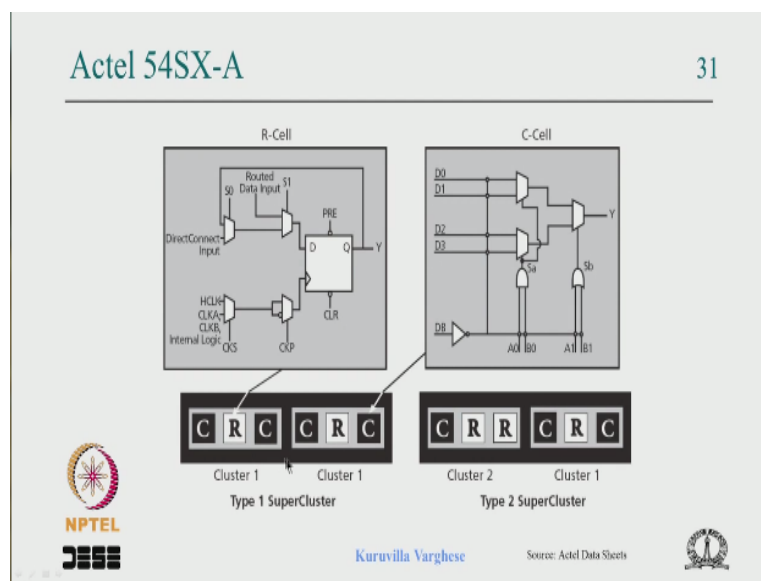
(Refer Slide Time: 35:48)



And the registers cell is nothing, but a flip-flop it set and reset you can choose the clock polarity you can choose from a hardware clock well low skew clock routing. And 2 different other clocks okay. And you see there is a re-circulating mux and this is connected directly to a combinational cell okay. So, the adjacent combination cell and this is for you know getting the wire connection with through the anti-fuse to here okay.

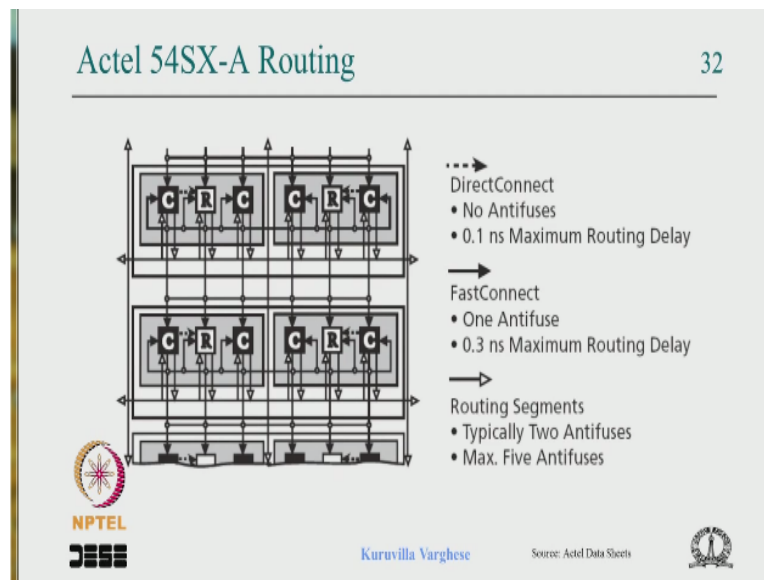
So, there is a port like this act as a clock enable to get this and this is also a kind of connections for the input from the previous c cell.

(Refer Slide Time: 36:35)



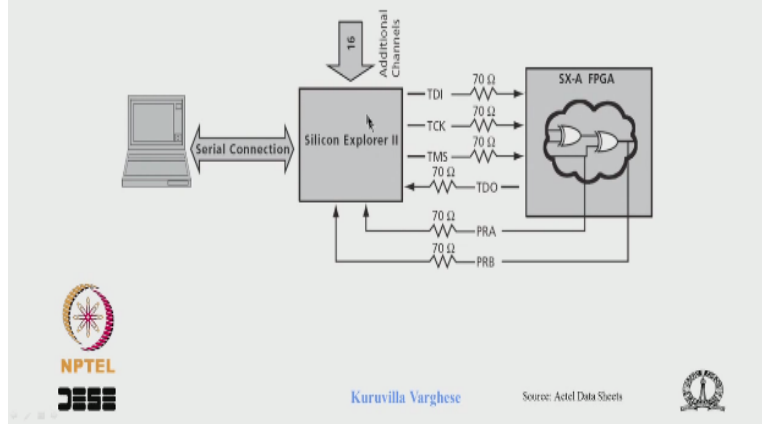
And normally what is done is that, they have a clusters 2 type of clusters 2 combinational and 1 register cell and 2 register cell and combinational cell. So, these clusters can be kind of mixed together to control the ratio of register to the combinational cell. So, maybe in certain device, there are more registers in certain device there are more combinational cell suited to different application.

(Refer Slide Time: 37:02)



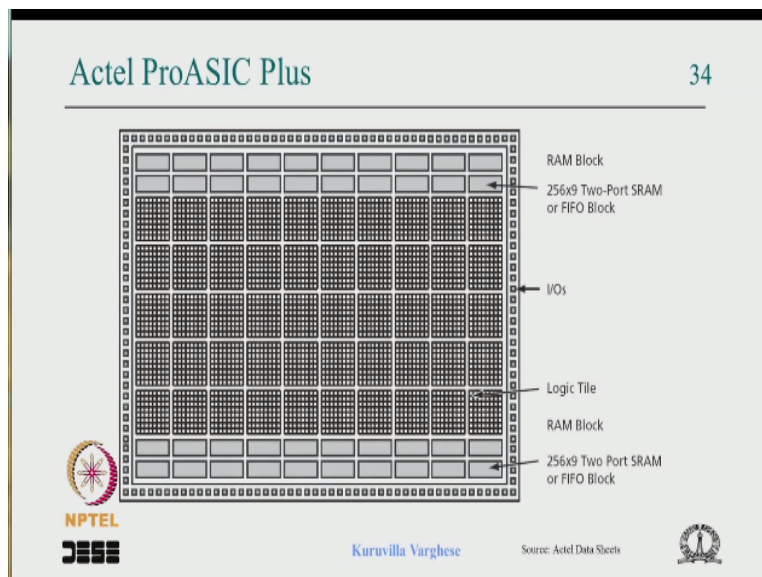
And the routing is that within say between a c and adjacent c and are direct connection within a cluster there is a 1 anti-fuse connection across the cluster 2 anti fuse connection what is specified is that the number of fuses used is minimal when the interconnection is close by. So, that you can get a fast interconnect. This is all taken from the Actel data sheet.

(Refer Slide Time: 37:35)



And this shows the idea of probing. There are 2 probe pins and this is the FPGA. So, there is a logic there is a hardware which connects to the JTAG port of the FPGA the probe pins. And this is serially connected to the computer and you can give command to connect internal signal to the probe pin capture it and view it and debug it. So, it does not require an addition logic to be put into the chip. And this was useful maybe at that time it was a good idea which stated by the Actel.

(Refer Slide Time: 38:14)

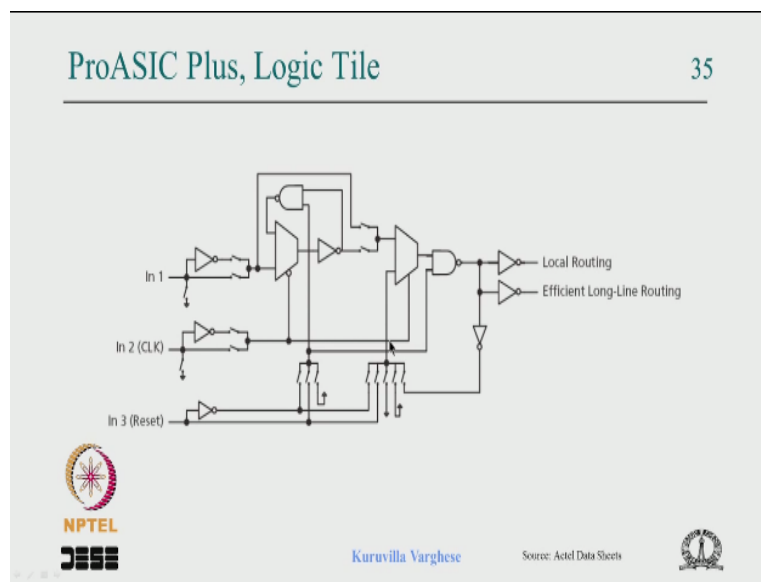


And the Actel also has a flash base FPGA which is essentially better than the SRAM in a way. Because even in a SRAM FPGAG you know that when you deploy something in the field, this

flash chrome is use okay to that extend only thing is that the beginning. There is a configuration time this say kind of copy from the flash to configuration memory which is SRAM.

So, additional configuration is involved. But here the programming memory, programming is done through the flash memory. So, it is non volatile, so it is already configured at the power on the chip is ready. So, that is a advantage, but otherwise the architecture is pretty much same there is a difference in the logic block. So, there is kind of memory, I/O pins, the logic blocks just for the kind of curiosity or.

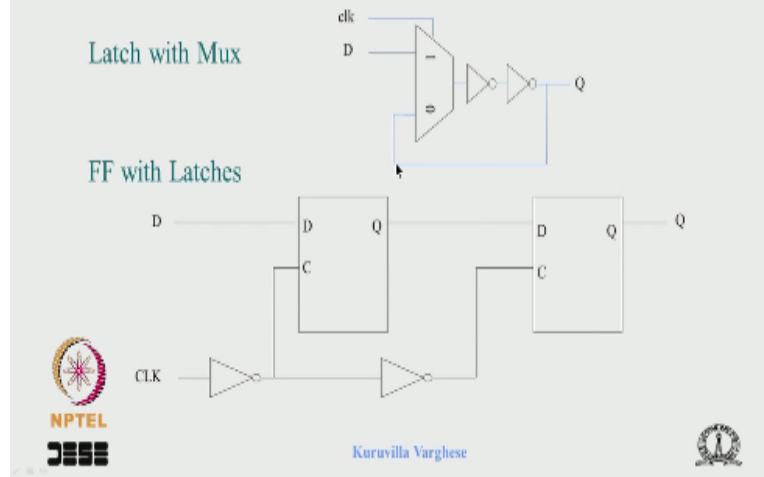
(Refer Slide Time: 39:18)



This is the logic dial or logic block of this ProASIC Plus this is what they have you know. This is what does the logic block first thing is notice that there is no flip-flops inside okay. That could be quite surprising, but if you look at the combinational circuit you have 2 to 1 mux. That means you can implement 2 variables like 1 in the select line, 1 in the input and there are such tool 2 to 1 mux that is all okay.

So, how do one implement an edge triggered flip-flop is a question okay. But if you kind of node the circuit then you can make out that, a 2 to 1 mux with this two inverters You know this can act as an inverter can act as a latch okay.

(Refer Slide Time: 40:19)



So, I am showing that in a picture here, this is the 2 to 1 mux this is the select line which is clock. This is D input and you say D is the clock is 1, D goes to the q, and if clock is 0 q is latched okay. So, that is what is here shown so, there is an input which goes through this NAND gate which can be by making this 1 this can be an inverter. And to the second inverter goes it act as a latch. Similarly look here you have an input say the input, then it goes through here it takes a latch okay.

Now you know that the master slave edge to the flip-flop, where there is a master latch, a slave latch. And there is an inverter in the clock, this act as an edge to that flip-flop. So, you see this is the clock line, which goes to the select of the mux, one is inverted, one is directly going. So, that is how this master slave latches implemented using this 2 to 1 mux and this particular.

You know cascading of both you can see the output is going to the input and this is the real thing. So, this can be very cleverly used as for combinational circuit as well as flip-flop. So, that you know it is a if I use a tile either to implement 2 to 1 say 2 such of 2 variable implementation or a flip-flop. It is a clever idea I have in used it, I do not know how effective the utilisation is and what is the kind of over red in terms of these kind of switches.



Because there are switches need a two configure inter connect you know chain it together and all that. I have absolutely no clue, because I am not played with this particular FPGA but one can look at it.

(Refer Slide Time: 42:28)

ProASIC Plus Routing

37

- Fast Connect
- Short Lines (1, 2, 4), Long Lines
- Clock Tree
- Pad Ring (Pin Locking)
- SRAM Blocks
- Programming Tech: Flash
- Non-volatile


Kuruvilla Varghese




And they have the fast connector you know between the adjacent blocks there are short lines of 1 logic block, 2 logic block, 4 logic block length long lines across the device clock tree pad ring there is so there is a wires around to do the pin locking SRAM blocks the programming technology is flash okay.

(Refer Slide Time: 42:57)

CPLD vs FPGA

38

Features	CPLD	FPGA
Logic	AND-OR	Mux / LUT / Gates
Register to Logic ratio	Small	Large
Timing	Simple	Complex
Architecture Variation	Small	Large
Programming Technology	Flash	SRAM, Anti-Fuse, Flash
Capacity	10 K	2 M LUT + RAM


Kuruvilla Varghese


So, that kind of winds up the look at the various other FPGAs the FPGAs from vendors like Altera and Actel, in the case of Actel we have looked at basically an anti-fuse technology and flash. Anti-fuse is very much useful in space application and flash is convenient. Because it does not have the power on with the configuration time, and that is at least the octal FPGAs have certain different architecture.

In the anti-fuse architecture is kind of totally different from the SRAM base architecture like if you are working in space application maybe you should look at the Actel FPGAs other FPGAs there are radiation add into devices available. But SRAM FPGAs PROM to corruption in the presence of you know kind of space radiation. So, that should be kept in mind so, very briefly.

Since we have looked at the CPLDs and FPGA basic you know comparison we have mention this at the beginning so, the logic is in a CPLDs and AND and OR, and most of it wasted. Because there if very wide and decoding and sometimes the product number of product terms quite high the FPGA essentially uses look up table. And there are few FPGAs are mux and few gates the register to logic ratio is small very very less number of flip-flops.

But FPGA as lot of flip-flops and even we have seen that look up table itself can be used as memory it can be used as shift registers we are mention this. And so, there are quite a large registers and dedicated memories available so, for all practical purpose, this a huge advantage FPGA not only in terms of complexity. The register to logic ratios is quite huge in the case of FPGA, and the timing is very simple.

Because there is a cross bar central switch interconnecting the whole logic block so, the timing is simple you know you have 1 or 2 inter connection. But in the case of FPGA can be very complex, because from one end to other end, there can be quite a lot of switches program. And that all to add to delay unless one fit in your design and floor planet properly would not be able to kind of estimate the timing appropriately.

You can say that at the beginning for a complex design how much performance in case of in terms of delay you can get it, and the architecture variation in the CPLDs very small like you

take the CPLD from Altera or Xilinx are say Athmel the architecture will be identical. But if you look at the FPGA it can be quite different at least the SRAM FPGAs look similar.

But then you can have the anti-fuse FPGA which has quite a different architecture and the programming technology. In the case of CPLDs always flash it is good in a way non-volatile, but in the case of FPGA it is SRAM base mostly a SRAM base. The some flash and some anti-fuse with regard to capacity let us look at the slide see it is limited CPLDs as a kind of 10k blocks but here there are kind of 2 million look up in the largest FPGA available.

Now + lot of memory dedicated memory not we are not talking about the look up table being used as memory the dedicated RAM is quite large. So, that is the comparison between CPLD and FPGA, FPGAs very much useful for prototyping. So, by a large the use of FPGAs in prototyping that means you want to develop an ASIC you implement that first in a FPGA okay that is why high and FPGAs are used.

There are Virtex 7 very complex FPGA, which cause quite a lot is there. But then that enables, one to test an ASIC before going it to the boundary Because, ASIK the capital investment or the non-recurring engineering cause is very high. And if you go with the only simulation the timing simulation to the boundary then it is a huge risk is involve fabricating it. There is a bug, you have heard about the internal bug in you know floating point unit.

The famous bug you know the chip manufactures after call it back then correct the issue. So, huge trouble you know all the production is stop it affects you know the lot of money is wasted lot of time is wasted so, all that can be avoided by you know putting the design in FPGA. And, when say this CPU vendors put there design in a FPGA.

They have to put it across multiple FPGAs okay so, like if you take an Intel CPU even the core functionality cannot be put it in a single FPGA. It might require multiple FPGAs to put everything together but once it is tested for long time the confident and go for ASIC one more thing is that many a times at clock speed testing may not be possible like a if there is a CPU which clocks at 1gigard.Or 2 gigahertz.

There is a less very less chance that you know it can be done in I mean it is not less chance. There is no chance at you can clock and FPGA when the latest one 22 nanometre can clock at 2 gig or 2.5 gig it is impossible. It will be in the order of mega hertz 200 to 400 mega hertz. But it is you can run it for a long time so, mainly the FPGAs are use for prototype checking.

There are lock of FPGAs like Spartan which can be used in a mass produced item, but it is all about the economy that you make a 10 dollar FPGA. And you put it in a consumer device maybe in a set top box, maybe in a TV the, but always the people think of integrating that logic into part of an ASIC. Because there will be some CPU related SOC which is working.

So, why not put this whatever logic which was there in FPGA into that ASIC. So, whatever even if the locus FPGA at used there is a great chance that the logic within that FPGA get absorbed into the SOC which is being built. So, by and large I would say the use of FPGA is in the IP development. And in prototyping and which is which a huge role you know you should not underestimate saying that you do not see an FPGA in a commercial device which is mass produce.

But there is kind of the particular place for FPGA in this particular design space. And one can play with different high performance architectures like accelerators in you want to speed up something and you can use FPGA okay. And that once you know you can try out various algorithm when it is successful you can make an intellectual property, that can be later translated to an ASIC you know.

So, and the high performance we know that the FPGA clocks at a very clock frequency compared to ASIC. But you still you can get very high performance out of FPGA. Because you can parallelise computation okay, so, assume that you are doing some packet processing in FPGA say you want to do say a networks you know kind of security device say intrusion detection device where you have to analyse the packet okay.

Now when you get a packet you get it serially, you convert into parallel okay. Now maybe you have to look at the addresses okay. The destination IP, source IP, the port number of tcp. The basic idea is that once parallelise it, you can do a parallel processing okay. You can lookup say you do not have look at sequentially, you do not have to say first look at the destination IP address.

Then source IP address, then the protocol number everything can be looked up parallelly you know you put while reading when you converts serial to parallel you store this bit stream in a wide memory like maybe 128 bit wide memory and read the 128 byte all together sorry 128 bit you know all together and do the parallel processing, you know kind of simultaneously look at the source IP address, destination IP address, source probe port address, destination port address.

Even if you want to look at the higher layer, suppose you are doing some http kind of protocol processing or parsing say you want to look at the sql queries. Because, there could be attack on through sql. So, these are deep inside the packet the tcp within the tcp pay load and so that can be parallelly looked at. So, that way the performance will be quite high So, that how I achieve high performance using the FPGA.

So, you do a parallel you know you make a wide data bus. And wide memories and you do the parallel processing. Sometime you may have to use multiple memories like you have the say the maximum possible memory width is 128 bit. So, what to do is that you put 2 such memories and the memory output can be kind of parallelly feeding your processing logic okay.

And sometime you use multiple memories okay say you have a 4 port memory, where 2 ports are writing, 2 ports are reading different location then doing parallel processing. So, all these techniques can be use to extract high performance. And 1 thing I have not touch because this was a kind of a basic course I wanted to kind of take you from with a minimal background to a good level that was intention of the course.

But there many things you know, I have not seriously discuss the synchronisation or these high performance techniques. One thing I have not mentioned is the pipe lining, so wherein a data

path operation can be kind of sequence through registers. The idea is that say you are adding very simple you have a ripple adder okay. And normally you give an 8 bit ripple adder all the 8 bits are added together.

But it is possible that you introduce registers such that. The when too least significant bits are added, and then in the second stage the next 2 bits are added okay, and while the first data second bits are added. The new data can come in and the LSP of the new data can be added okay. So, there are these possible that 8 additions can happen parallel it involves lot of memory to store the previous results and so on okay.

But that is essentially pipelining you may have kind of studied that in the contexts of CPU. But pipelining is a general technique, which is not only in hardware. It you know it kind of it stems from the factory assembly line automation, and if you want to look at the hilarious background you can look at the Charlie Chaplin's modern times how the automated factory line works.

So, basic principle is that, so the through pipelining when the data is streaming okay. pipe lining is useful only if the input data is coming continuously. In such cases you can get very high through put by pipelining okay. So, that is another way of getting high performance using FPGA. So, I think I have briefly mention how to performs, you know high performance computation within FPGA by parallel processing by replicating the computational engines you know.

You have a basic code doing some basic computation, you can replicate it to do things parallelly. Then you can do pipelining to get high through put, so these are the different method of achieving very high performance, and you can use like the multi port memories which aid in kind of parallel computation. Because you can write through 1 port, you can read through 1 port, or you can read through multiple ports from multiple location for the processing to proceed parallelly.

All these techniques can improve very high through port, and we have implemented very high through put kind of processing which is some time kind of quite surprising that with the very low

clocking like 200 mega hertz. You can achieve such high frequency that is one reason the FPGAs give very high serial ports. You know there are serial port which clock at gigabits okay gigahertz.

So, that you can get in the data very fast then paralyse it though the internal clocks are low, you can do a parallel computing. So, that is the basic idea about high performance design using a FPGA so, please go through the FPGA lectures I have given try to understand it, and you will be able to do good design and I wish you all the best and thank you.