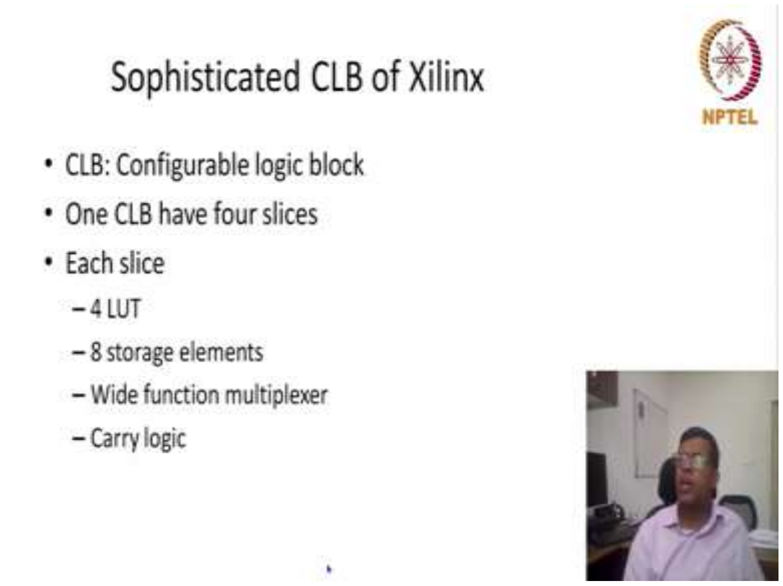
Digital System Design

Professor Neeraj Goel
Department of Computer Science Engineering
Indian Institute of Technology, Ropar

Xilinx CLB

(Refer Slide Time: 0:18)



Sophisticated CLB of Xilinx

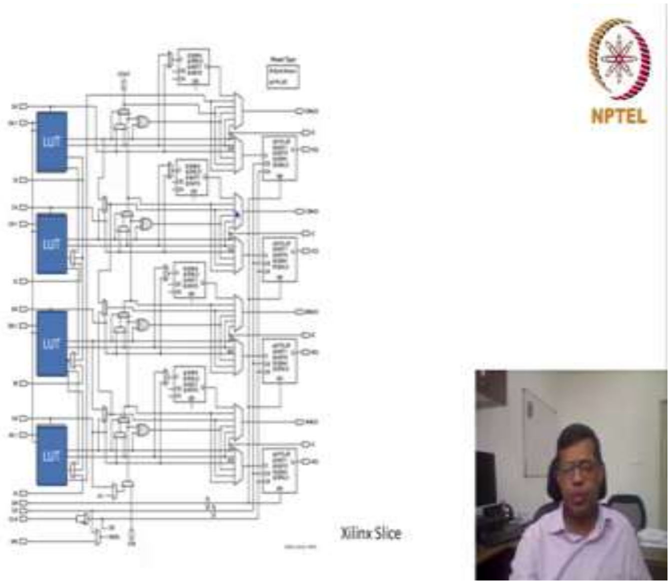
- CLB: Configurable logic block
- One CLB have four slices
- Each slice
  - 4 LUT
  - 8 storage elements
  - Wide function multiplexer
  - Carry logic

So, to still have more understanding we take a commercial logic block as an example. So, these days although FPGAs has been manufactured by different kind of vendors but the 2 vendors which are quite popular these days is 1 is Xilinx another is Intel, so there is to be an company called Altera which has been acquired by Intel, so the these 2 FPGAs Xilinx and Altera FPGAs are the leading FPGAs in industry, so we are going to see that our typical Xilinx logic block looks like.

So, this the reason of choosing or why we are understanding this logic block will also tell us that how a commercial FPGA would look like. So, here in Xilinxs the logic block or logic tile is called configurable logic block and each logic block, each CLB has 4 slices and each slice has further 4 LUTs, eight storage elements and a wide function multiplexer and a carry logic.

So, you can see that 1 CLB, 1 CLB 1 configurable logic block is a tightly connected so there are lot of hard wires it does not require so much of a configurable interconnect, so this CLB has essentially 16 LUTs and 8 cross 4, 32 logic elements and also a carry logic.

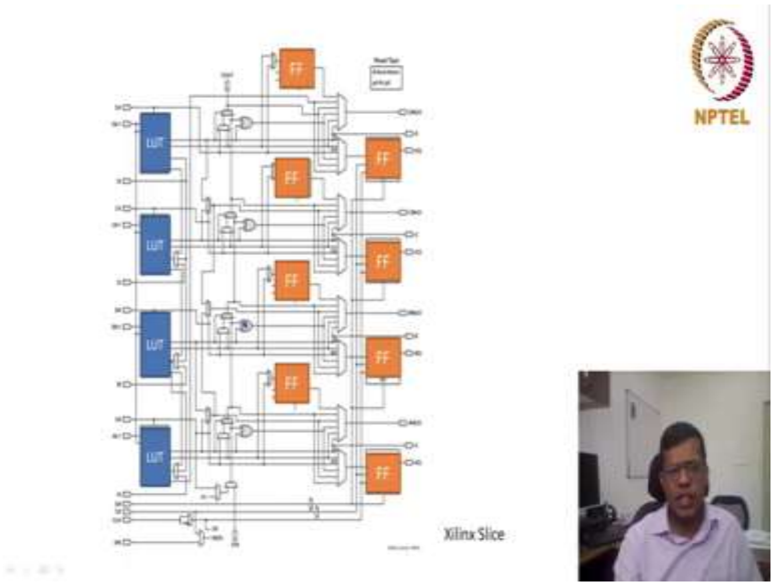
(Refer Slide Time: 2:08)



So, now this is a structure of 1 slice, so remember from our previous slide each slice has 4 LUTs eight storage elements and multiplexer and a carry logic. So, we see here it is hard to understand there are lot of but what we can see from this figure that there are lot of multiplexers, there are different kind of XOR gates, a lot of multiplexers are there. And now to understand this let us mark different kind of a logic. So, basically these are all LUTs.

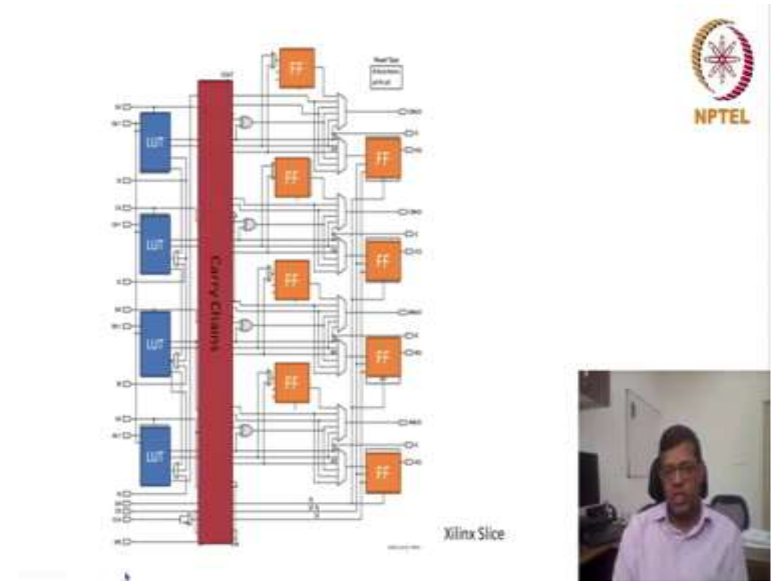
So, these LUTs take 5 inputs, d6 to d1, so these are all 5 input LUTs and their 4 LUTs all with sorry six input LUTs all the inputs with 1 to six, so there are six inputs to all of them and all the inputs are different, so this is a 1 to 6, this is b 1 to 6, this is c 1 to 6, this is d 1 to 6.

(Refer Slide Time: 3:17)



Now, there eight flip flops so basically each of the LUT can give 2 different outputs and both the outputs could be either latched output or known latched output. So, there are finally we will have this 4 different eight outputs here a and a0, so this a is a non- latched one and aq is a latched one, b and bq, c and cq, these flip flops are essentially internal which also are given so as a different output so called DMUX, CMUX, BMUX, and AMUX. So, there are 12 outputs for these 4 LUTs each LUT would have 3 outputs, one is multiplex output another is d, another is a latched output.

(Refer Slide Time: 4:36)



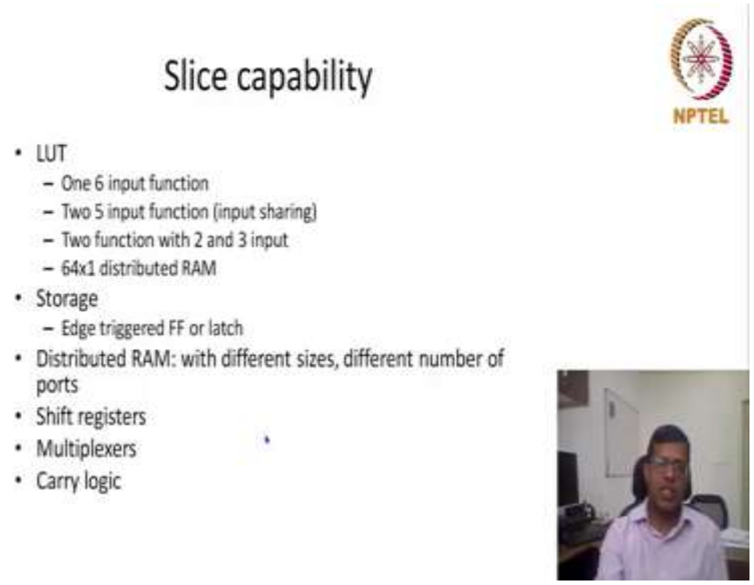
Further because as we have pointed out earlier the addition is a quite a popular circuit but we never know that what is the size of addition we require so that is why there is an internal carry chain circuit which is there so you see Cin as a given as an input and so the output whatever is generated by this LUT is also given as a carry into the second, so this carry chain circuit help us to propagate the carry, so this is a ripple carry which is going from Cin here this output can also be would give some as output as well as carry as output and that carry could be given as the input to the next carry and so on so forth.

So, this carry chain which is a hardwired logic can help us to boost the performance of at least addition with the with this additional course of the hardware. So, one can argue that this additional cost of the hardware this will get unutilized if this CLB is not implemented as adder that is true.

Similar is the fact that if this the output is not latched so in that case all these eight flip flops will also be not used but that is the idea of a configurable logic block or that is the idea of a logic block that there could be circuit which will not be used but we would like to have some circuit which would be fast enough when it is configured like that logic.

So, there are certain inputs you see which are given as a universal inputs like you have clock, chip enable, SR and AI, WE, so these 5 inputs are essentially the global inputs given to all the slices or all the inputs. So, this way this one slice can work like a different logic units or different kind of, it can be configured as a different kind of logic.

(Refer Slide Time: 6:35)



Slice capability

- LUT
  - One 6 input function
  - Two 5 input function (input sharing)
  - Two function with 2 and 3 input
  - 64x1 distributed RAM
- Storage
  - Edge triggered FF or latch
- Distributed RAM: with different sizes, different number of ports
- Shift registers
- Multiplexers
- Carry logic

So, you see one slice can, 1 will have 4 LUTs and each of the LUT can implement 5 input functions because there are two outputs so it can have 2, 5 input functions with all the 5 inputs sharing the same input, the yeah, the sixth one could be different so because there are six input one of the function could be of two input, another could be of three inputs, so different combinations are there for different kind of a functionality.

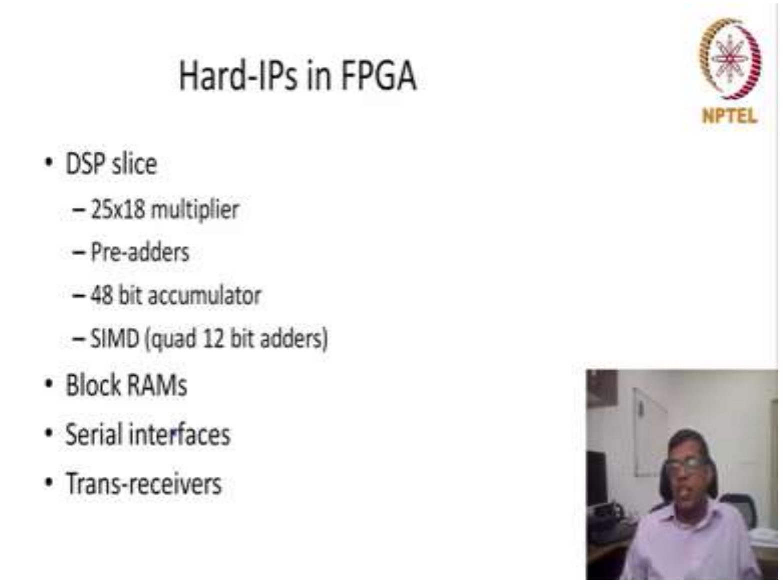
So, from this specification it appears like the internally the LUT is broken into 2 different LUTs with 5 input each, so that is why it could work as a one 6 input function or two different 5 input function while the 5 inputs are same for both of them, and two different functions with 2 and 3 inputs. Why 2 and 3 input? Because input if the inputs cannot be shared then the LUTs would have different functions, so different basically configurations.

And you can also see from this specification the internally the connections are fixed so the 5 inputs are going to both the 2 internal LUTs, the 5 inputs are going to 2 both the internal LUTs, yes. So, because it is a 6 input function internally there will be 64 elements, so that way this LUT can also work like a 64 into 1 RAM.

So, internally we also seen the slice as storage as triggered flip flop or there also level triggered flip flop like latches and using the slice people can implement distributed read only memory with different sizes, different ports, all of these possibilities are there. There are eight flip flops inside so they can also effectively work like a shift registers, so their internal connections like that they can work like a shift left registers or shift right register so all those multiplexers are already built in or implemented as a hard logic.

So, these slices can also effectively work like a multiplexer and yeah as I said earlier so there is a carry chain which is there so they can effectively work like a adder or ripple carrier. So, now you see within the slice or within a configurable logic block also there are good amount of hard connection as well as programmable connection, so the same logic block can work like a different logic based on the requirement, though we also see that lot of redundancies there, lot of logic would be there which will never be used or which would rarely be used.

(Refer Slide Time: 9:44)



Hard-IPs in FPGA

- DSP slice
  - 25x18 multiplier
  - Pre-adders
  - 48 bit accumulator
  - SIMD (quad 12 bit adders)
- Block RAMs
- Serial interfaces
- Trans-receivers

So, at the larger scale there are more fixed logic which is present in some of the FPGA but that all depends on the requirement. So, for example, if we see a particular FPGA we would like to, we would like to have an FPGA which would be used for signal processing applications. What are signal processing applications, like which is used for speech recognition, for image processing, for yeah so for audio video communications, so these are called signal processing applications, so these signal processing applications people have observed that it require a specific unit which is which would be invariably required in all different kind of FPGAs or different kind of circuits which is called multipliers or multiplier and accumulator.

So, because of that because multiplier is a very commonly used function in DSP kind of applications so those FPGAs has many of the multipliers instance as a hard or a fixed logic, so although that fixed logic can be connected using programmable interconnect and it could also be programmed like a different interfaces. So, the size of multiplier could be variant or yeah so there could be variations in multiple sizes.

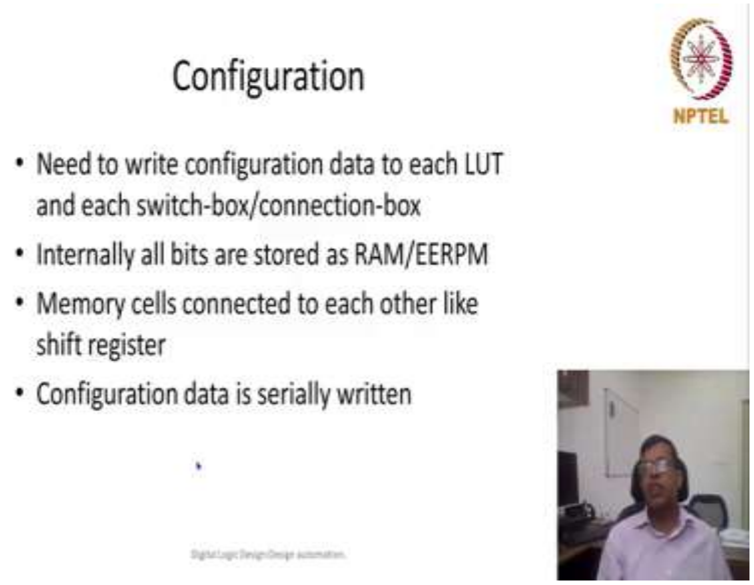
So, the same multiplier can also be used as accumulator, can also be used as SIMD means that for example this 48-bit accumulators this is an adder which could be adding multiple inputs. So, this 48 bit accumulator can be used as 4 12-bit adders which can be done in parallel. So, these DSP slides are there if you would like to, we choose an FPGA which will be used for DSP application.

Similarly, many FPGA also have fixed processor inside this. Now, again quickly why do we require this hard IPs or fixed IPs? This fixed IPs, would if they are programmed or they are basically configured using CLBs then the performance would be below power, the area required so the number of CLBs required would be huge they would be slower in nature to make things faster, there would be some fixed IPs or hard IPs which would be instantiated.

Similarly, at, for memory block RAMs, block RAMs are read only memories or RAM units, memory units which are used as a fixed logic which is this block RAMs are typically used in all kind of applications, so that is why invariably all Xilinx FPGAs will have certain amount of block RAM which is present could be if we are using this FPGAs for serial interconnections like for example USB interfaces, et cetera then there are some fixed serial interfaces as well as the protocol which is used for those interfaces is also implemented as a fixed logic.

For communication FPGAs trans receivers are hard IPs which are instantiated inside the FPGA, so this way various hard IPs or fixed IPs, fixed hardware would be there to make performance of the FPGA better also to do this area versus delay trade-off to more sustainable. So, also many a times processors are also used as a fixed hardware in some of the FPGAs.

(Refer Slide Time: 13:50)



Configuration

- Need to write configuration data to each LUT and each switch-box/connection-box
- Internally all bits are stored as RAM/EERPM
- Memory cells connected to each other like shift register
- Configuration data is serially written

So, now the next question which I would like to take up is that how the configuration is done. So, what we have seen so far that if we would like to configure an LUT we have to fill in that for example if you would like to configure a 6 input LUT we have to configure those 64 cross 1 memory locations. Similarly, to configure switch boxes, to configure connection boxes also we would require some memory elements.

So, these memory elements are internally stored as either RAM or EERPM, so the RAMs are used mostly wherever we are trying to do some sort of a prototyping experiments, lab experiments so that we can use the same FPGA for different purposes, we, these internal bits are stored as a ROM or known programmable memory elements when we would like to use this FPGA as a product.

So, for example, you would like to build a new washing machine or new air conditioning unit where you would like to see a new feature, so there we can implement configure this FPGA as a using huge interconnections or using read-only memory so that once it is configured it can be used at the customer and the way people would like, so there people also use EERPM so that if at all need arise then they could be reprogrammed using some other way.

So, usually in the lab whatever FPGAs we are using they are, they have all the configuration data stored as a RAM not as a ROM so that we can reprogram every time we give a power. So, now the question which you have to understand is the question which is before us that you have let us say thousands of configurable logic block, you have so many thousands of switchboxes and interconnects, how do we configure all of them?

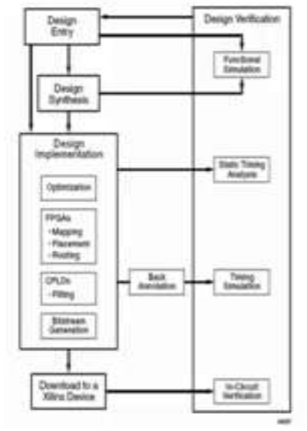
So, they have to be configured may be using hardware or software but all of them need to be accessed via some external link. So, to program all of them effectively internally all of these memory cells are connected to each other. Let us say there are thousands of CLBs each of the CLB memory location is connected to another CLV memory location in a serial in serial out fashion.

So, it is internally a serial in shift register, so that from the output so from the external interface let us say your configuration is there in a flash drive or in a USB drive or it is there in your host computer then using a USB interface or using a serial in interface all of the bit data could be given 1 by 1 and since internally during the configuration time our whole logic FPGA logic works like a single shift register.

So, because everything works like a single shift register so bit by bit all the bits are reached to correct configurable logic block as well as correct switch box or correct connection box. So, all of these things are connected one by one and finally once this configuration is done then FPGA can be used as a standard device or as a as the application which you have configured it onto. So, this is this is the overall flow of the configuration.

(Refer Slide Time: 17:50)

FPGA Design Flow



Source: Xilinx.com

Now, let us quickly see if we have a design how do we implement it using an FPGA. So, a design entry is usually done using a VHDL or very long so that is a hardware description languages and from the hardware description languages then synthesis would be done. So, the synthesis is usually done again using the software or EDA provided by vendors.

So, during the course what we have done, we have taken a design entry in (())(18:23) and we were doing only the simulation, functional simulation part, so this functional simulation was checking whether the design is correct or not. So, the synthesis part we have not done at least in this course.

So, after doing synthesis the finally design is represented using basic AND OR gates or basic gates, basic multiplexers or basic elements then after that those basic gates need to be mapped onto there would be first of all some optimizations would be done, optimization means the optimization logic optimizations or state machine optimization, so logic optimizations we have seen that how we can reduce the logic into minimum number of gates and in state machine optimization we have seen how we can reduce the number of states.

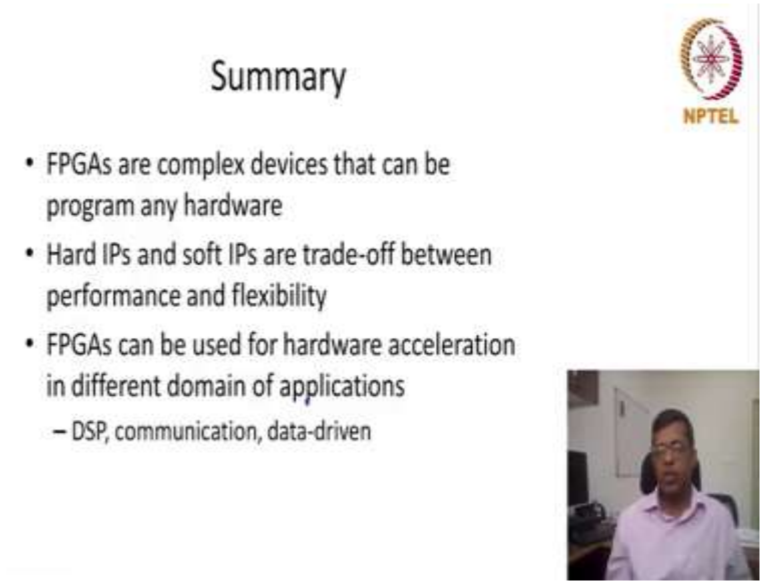
So, after that we have to map that which logic has to go to which particular logic tile or configurable logic block and how many logic tiles or how many logic blocks would be required and after that after knowing how many then we have to also fix that which particular physical block would be required for one particular logic so that is called placement.

So, once placement is done then we have to do the interconnection, the output has to be connected to the other input of the output and inputs need to be connected to all the logic blocks, so this is called routing. So, mapping placement and routing all of these things would happen for FPGA.

Similarly, the other kind of programmable logic is called CPLD, their fitting would happen instead of having all these three functions. So, after completion of this then a configuration file would be generated, configuration file is also called bit stream because it is a stream of bits, so that bit stream is generated and that bit stream is downloaded to a Xilinx device, so this downloading or we can also say is my Xilinx device or an FPGA would be programmed using this bit stream.

In parallel we also do design verification that means first functional simulation is done using (())(20:39) and then after the implementation has been done then we also see that how much what would be the delays what would be the latency that is called static timing analysis and timing simulation is done so that we can check whether a clock period requirement would be met or not and once it is downloaded onto FPGA device then we can also see in circuit verification whether it finally works as the circuit which I implemented or not. So, this is the overall design flow. So, in the next lectures we will try to elaborate onto this design flow also.

(Refer Slide Time: 21:17)



Summary

- FPGAs are complex devices that can be program any hardware
- Hard IPs and soft IPs are trade-off between performance and flexibility
- FPGAs can be used for hardware acceleration in different domain of applications
  - DSP, communication, data-driven

So, with this let us conclude today's lecture. What we have seen that although initial idea of FPGAs was a simple gate array but over the time it turned out to be a complex devices which can program almost any hardware. And the complexities is partially due to having some hard IPs or fixed IPs as well as some soft IPs so programmable IPs which can be where soft IPs programmable interconnection or programmable logic gives us the flexibility while the fixed logic gives us the performance.

Similarly, a fixed interconnections give us the performance and programmable interconnections give us the flexibility, so there is a trade-off between flexibility and performance in FPGAs also and they are used for hardware acceleration basically making some particular function quite fast in different domain of applications. So, the application domain where these FPGAs are typically used is DSP signal processing essentially, communication and sometimes data driven. Today in machine learning also they are quite heavily used. So, with this I would like to close. Thank you very much.