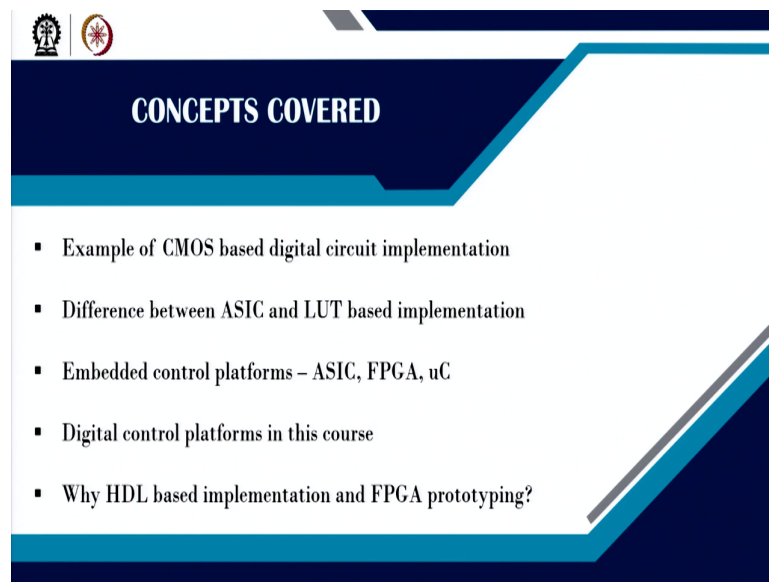


Digital Control in Switched Mode Power Converters and FPGA-based Prototyping
Prof. Santanu Kapat
Department of Electrical Engineering
Indian Institute of Technology, Kharagpur

Module - 01
Introduction to Digital Control in SMPCs
Lecture - 04
Overview of Digital Control Implementation Platforms

Welcome to this lecture we are going to talk about an overview of Digital Control Platform Implementation Platforms.

(Refer Slide Time: 00:30)



The slide features a dark blue header with the text 'CONCEPTS COVERED' in white. Below the header is a list of five bullet points. The slide is decorated with geometric shapes in dark blue and light blue, and includes two small circular logos in the top left corner.


- Example of CMOS based digital circuit implementation
- Difference between ASIC and LUT based implementation
- Embedded control platforms – ASIC, FPGA, uC
- Digital control platforms in this course
- Why HDL based implementation and FPGA prototyping?

So, here we will take an example of CMOS-based digital circuit implementation. Then we want to differentiate between ASIC implementation and Look Up Table-based implementation, then we want to talk about ASIC FPGA and microcontroller, and finally what are the digital control platforms that will be considered in this course, and finally why do you go for HDL-based implementation FPGA prototyping?

(Refer Slide Time: 01:01)

TWO-input AND Gate – An Example



$y = AB$



$y = AB = \overline{\overline{AB}} = \overline{\overline{A} + \overline{B}} = \overline{\overline{A} + \overline{B}}$

$y = AB = \overline{\overline{A} + \overline{B}}$

$x \triangleq \overline{\overline{A} + \overline{B}}$



So, first, we want to take an example of an AND gate ok. So, AND gate we are considering here is a Two-input AND gate where this A and B are the 2 inputs and y is the output. So, we want to implement this gate first we will go by you know we want to implement using CMOS; that means if we take y equal to AB. Then we can write AB then we can write A complement plus B complement overall and we are denoting x to be it denoted as A bar plus B bar.

So, why do we want to implement it in this way? Because we want to use CMOS architecture because almost all digital electronics are now nowadays it is implemented using CMOS technology. So, in CMOS we have a complementary MOS; which means, PMOS and NMOS.

(Refer Slide Time: 02:02)

CMOS Implementation of a TWO-input AND Gate

$$y = AB = \overline{\overline{AB}} = \overline{\overline{A} + \overline{B}} = \overline{\overline{A} + \overline{B}}$$

$x = \overline{A} + \overline{B}$

$y = \overline{x}$

$x \rightarrow y$

So, how do you implement this CMOS? So, if you take this x bar how do you implement this can be implemented by. So, this is like the sum of this PMOS; that means, A and B and then the product of AB ok.

Because if you see that here we are talking about x ; that means, this is the x and what is x ? x equal to A bar plus B bar and if you look at this A bar plus B bar; that means, if both are 0 then x will be 0. So, if you see both are 0 then if A and B both are 0, then what will happen? Sorry, 0 means, but that means, it will be 1 if either of them is 0 then also it is 1 and if both are 1 then it will be 0; that means, x is equal to 0 so it is clear from this diagram.

Now once you have x which is a combination of the A bar plus B bar, then we can write y which is the x bar and this can nothing but if x is the input it will be an inverter simply an inverter that will be our y . And how do we implement inverter it is simply another like this architecture and here we have V_{dd} and this is our PMOS, this is our NMOS, and this is our x , and this point we are taking y .

So that means, we can implement this AND gate by this is what NOR gate is like no not NOR gate it is a NAND gate. So, it is the NAND gate because this will implement the NAND G gate ate followed by the NOR gate.

(Refer Slide Time: 03:51)

CMOS Implementation of a TWO-input AND Gate

Problems
- Fixed hardware

Number of transistor =6

So, the overall architecture of this AND gate using CMOS technology the first stage will be the NAND gate followed by the NOR gate, and overall this will represent the AND represents. So, in this architecture we need 6 transistors; that means, 3 PMOS and 3 NMOS transistors.

So, if we go for ASIC implementation this is the minimum number of transistors that are required to represent AND operation and that is hardware optimization as well as power optimization. But problem what is the problem what is the problem? So, this architecture is fixed hardware; that means, it is fixed hardware, fixed hardware architecture fixed hardware.

That means, we cannot use this hardware other than for AND purposes; that means, we have to only dedicatedly use it for any purpose any other 2 input logic functions cannot be implemented any other logic can be implemented cannot be implemented other than AND operation. That is why this is fixed hardware, but it requires a minimum number of the transistor.

(Refer Slide Time: 05:05)

Look-up-table (LUT) of a TWO-input Digital Logic

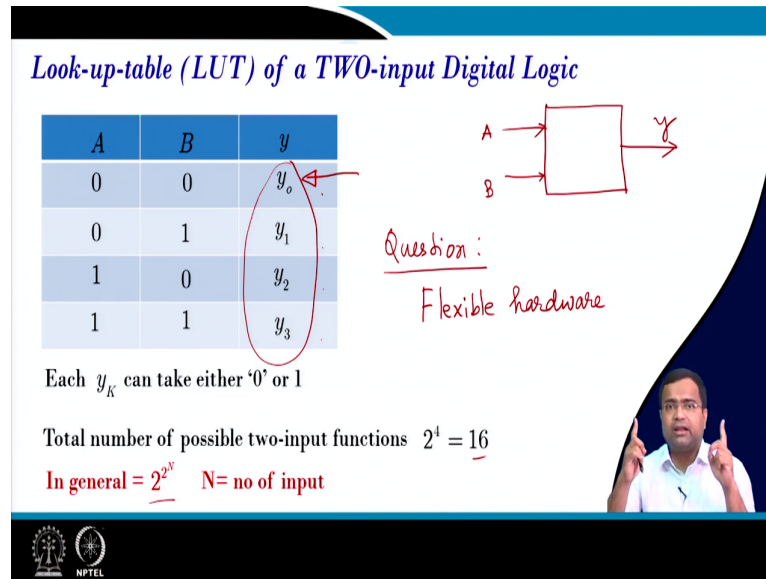
A	B	y
0	0	y_0
0	1	y_1
1	0	y_2
1	1	y_3

Each y_k can take either '0' or 1

Total number of possible two-input functions $2^4 = 16$

In general = 2^{2^N} N= no of input

Question:
Flexible hardware



Now, if we go for Look-up-Table-based implementation, suppose ultimately we want to write in terms of Look-Up-Table; that means if you take two-input logic. So, we are talking about a 2-input logic where A and B are the input and y is the output. And we know any Boolean algebra can be realized you know can be represented in this truth table and here A and B you know depending upon the A and B values.

So, if we can choose what is a y_0 to y_3 and that will give us what is the function that we are going to represent. That means, so that means in this architecture each of these output elements; that means, can take either 0 or 1 and there are such elements that are there 4 elements are there. So, you need a total of 2 to the power 4 possible function can be realized; that means, there can be 16 varieties of two-input logic functions that can be realized using this lookup table.

Now, if there are N number of inputs then we can have 2 to the power 2 to the power N number of functions that can be realized using Look-up-Table. Now the question is what is question can we make one hardware can we make flexible hardware, which can be used to realize any of these 16 functions?

Because earlier we saw for AND gate if we go for ASIC implementation and that is the way of ASIC which will optimize the number of transistors and as a result, it will also optimize the power consumption.

But if you want to use this hardware for any other of these 16 functions can you use a single hardware architecture or not?

(Refer Slide Time: 07:08)

LUT based Implementation of a TWO-input Digital Logic

A	B	y
0	0	y ₀
0	1	y ₁
1	0	y ₂
1	1	y ₃

$B = 0$
 $A = 1$

So, let us go again we write the truth table, and in this architecture, this is all 4 binary digits; that means, this bit has to be mapped here. That means, what is the value of y 0 you can simply store it here and this is an SRAM cell Static RAM.

So, you can represent each of these blocks; that means, you can either y 0 can be 0 or 1 y 1 can be 0 or 1. So, you can have such 16 possible you know a way we can store this element. Now once we store then we have to select line A and B, where A is represent the MSB and B represent the LSB. So, if you see the first mux; that means, in the first layer there are 2 mux, and in the second layer there is 1 mux.

So, in the 2 mux, the B is the common select line. So, now, if B equals 0 suppose you take B equal to 0, then it will take this top and this top channel. But then which one actually should be mapped to y that will be decided by what is the value of A. If A equal to 0 then this will be selected. So, ultimately this will map to this function; that means, y will be y 0 if both A and B are 0 and 0.

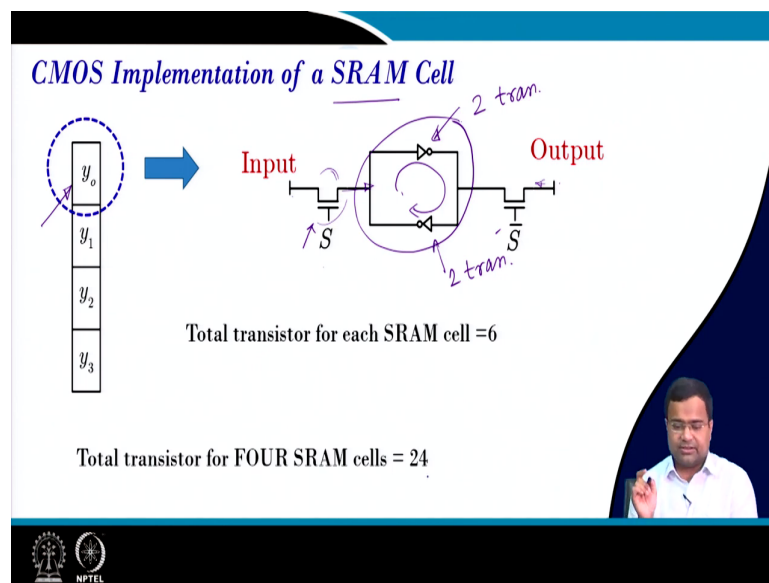
So, for the 0 0 conditions it will be y 0, now if you take 1 0 then it will route through this path this will route through this path ok. So that means, if I want if it's 0 0 that means if this quantity is 0 and this quantity is 0. So, this output y will be connected to this. Now if we

make it let's say let us use a different color if this is 0 and this is 1 then this will be selected like this it will be selected like this.

So, it will be y_2 and this is exactly what is the y_2 ; that means, the output will map to that particular stored value based on the status of the select line. Now in this architecture, you can realize any 16 logical functions using this hardware by suitably you know storing these 4 elements that are it. And this is very fast because it is just a multiplexer. So, it can be almost it can be a like parallel operation like a simultaneous operation and y can get the value of y_0 almost very fast.

But if you go to the earlier block there will be a propagation delay if you have more number of chain, so the propagation delay may increase. So, you may have a little longer propagation delay, but in this CMOS architecture mux architecture, your propagation delay can be reduced. But something there should be some penalty. What is the penalty?

(Refer Slide Time: 10:28)



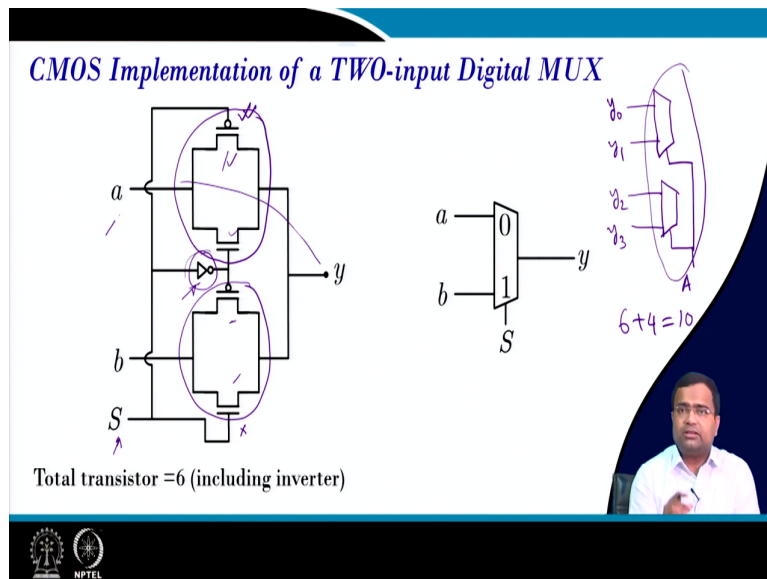
So, if we take this cell that means because we know in the earlier architecture for AND operation we need 6 transistors. Now, here we want to see how many transistors are needed. So, if we take this cell 1 cell; that means if we take this 1 cell this cell that is SRAM cell can be realized by this bistable latch.

So, this is the bistable latch and you need 2 such functions here. So, if you want to suppose you want to store 1, you turn on this switch and store the value of 1 here, then this 1 will be retained in this path and once it is stored then you disable this switch then it will behold.

And you can access this by the output side switch; that means when you turn on this output side switch the 1 data will be accessible so, you can take the data out. Again if you want to stay at 0 then again you turn off this, because they are complementary and you can again load another number here.

So, by that way, you can you know store either 0 or 1 in any of these SRAM cell. So, since such 4 SRAM cells exist. So, here we need 2 transistors for here 2 transistors are needed and here also we need 2 transistors ok. So, then a totally 6 number of transistors are needed for each SRAM cell, so for 4 such SRAM cells, you need 24 transistors.

(Refer Slide Time: 11:44)



Now, next in that architecture after the SRAM cell, we have a multiplexer how does the multiplexer work? So, these are Two-input mux. Now, this is a transmission gate CMOS transmission gate and this is a standard you know any digital circuit these 2 transmission gates are used to select either A or B. Now if the select line is 0 then this will be enabled and this will be disabled and this is a NOT function; so that means, y will be connected to A.

And you know why NMOS and PMOS are given in parallel? Because in the case of logic 1 and logic 0, either of these is best so, in the case of logic 0 NMOS is good logic 1 PMOS is

good otherwise, if you use logic 1 for NMOS there will be a drop and if you have such cell repeatedly then can we regenerative circuit.

That means the voltage can go below the threshold value ok. So that means, for logic 1 we should use that means the NMOS will be used, and for logic 0 if that means, y will be connected to A; but A can be either 1 or 0. So, depending upon their individual you know this transistor will be selected. So, by that way, if we enable and disable I mean select line if we choose S to be 1 then B will be connected to y.

So, in that way, we can implement a two-input digital mux. Now you can see for these 2 input mux you need 1 2 3 4 plus 2 transistors for this. So, you need a total of 6 transistors including 2 transistors for the inverter. So now we want to count, but suppose you have such 2 mux because we saw that in the first chain, there are 2 mux were there which was like a y 0 y 1 y 2 and y 3, but we have made this select line common.

So, since the select line is common. So, this inverter will be needed only 1 because it may not be needed individually. After all, there is a common line. So, that is why when you take the total 2 such mux connected in this fashion. So, you need a total of 6 into 2; that means, I can say so 6 for this plus 4 for the next plus 4 so it will be 10 transistors because these 2 will be common for both cells.

(Refer to Slide Time: 14:23)



LUT and ASIC Implementation of a TWO-input AND Gate

For 2 Input ASIC AND gate Total no of transistor =6 ✓

For 2 Input LUT based AND gate

- 4 SRAM cells → 24 transistors
- 2 2-input (input side) MUX → (12-2)=10 transistors
- 1 2-input (input side) MUX → 6 transistors

Total no of transistor =40 ✓




So that means, for ASIC implementation in AND gate we need 6 transistors for Look-up-Table based implementation we need how much.

For the SRAM cell, we need 24 transistors. Then, for 2 input mux, we need 10 transistors and there are 2 layers of mux the second layer of mux we need 6 transistors, so a total 40 number of transistors is needed. So, you can see it is almost 6 to 7 times more transistors. So, naturally, the number of transistors is increasing, so the power loss will increase, the area will increase and the cost may increase. So, that is why this is the tradeoff between flexibility and the hardware cost ok.

(Refer to Slide Time: 15:05)

TWO-input AND Gate – Comparing LUT and ASIC Implementation

Method	Description	Benefits	Limitations
ASIC	6 -Transistors ✓	Hardware optimized, power efficient	Fixed design, high NRE cost, long development time
LUT ↑	40 transistors ✓	Configurable hardware, low NRE cost, short development time ↑	Expensive hardware, power hungry, large component cost



Now, if you compare this ASIC versus LUT representation for this AND gate: So, we need 6 transistors for ASIC and LUT for 40 transistors, but remember when you develop an ASIC circuit it is an IC and LUT means there is flexible hardware where which can be used for any function. But if you want to develop various functions using ASIC you may need so many architectures.

So, for each function this ASIC has to go to IC, so the development time is pretty long, and if you want to develop the cost of hardware because of fabrication and transistor fabrication. So, this NRE cost can be very very high, but if you are going for mass production, that means if you are making millions of such ASICs then this cost per unit cost will be much lower than LUT-based implementation. But in general, the nonrecurring cost is high and the development time is also long because it is a transistor any IC development process takes

time there are multiple stages of development. But the benefit is that it is harder to optimize and power optimize.

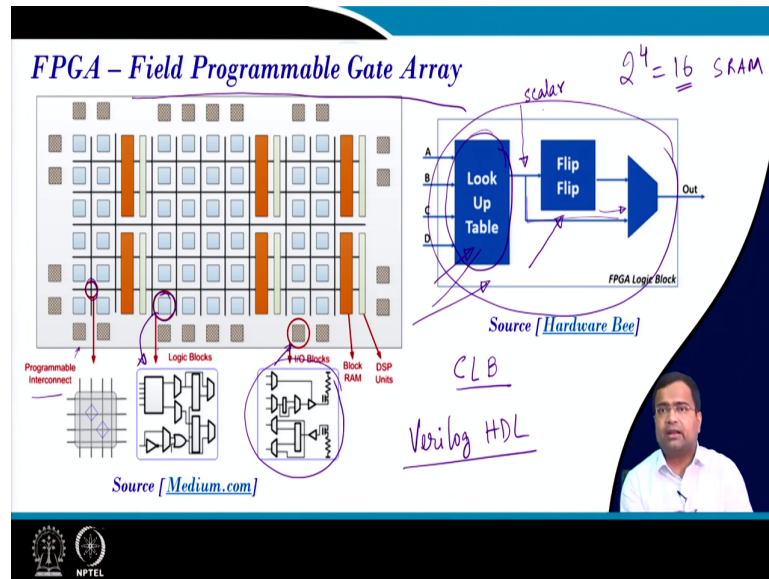
So, that is why if you go to the microcontroller where there are ASIC cells; that means, the microcontroller has dedicated hardware for addition block multiplication block those hardware's are dedicated which is only used for addition. So, there we have the scope for optimization in terms of ASIC. So, you can minimize the number of transistors you can make it power efficient, which means, this kind of ASIC base may be a microcontroller or any architecture that will be useful for low power consumption reduction in power consumption.

Whereas the Look-up-Table is very good if you want to prototype various algorithms. So that means, it is configurable hardware, but if NRE cost is also low and development time is also short because you can implement various. But it is very expensive in terms of hardware because silicon area increases power hungry. After all, there are too many transistors. So, it consumes a lot of power and a large component cost because of more transistors.

So, we have discussed this now this Look-up-Table is the building block for FPGA because when you say FPGA again as it is discussed here we want to have hardware where we want to test our HDL logic because we do not want to develop ASIC at the very beginning.

So, the best way to develop any ASIC is first to develop an idea test it and prototype using Look-up-Table based arrangement you know check the architecture functionality objective and whether all objects are made. Once it is tested by this hardware then one can go for ASIC because ASIC development time is long once it is kind of foolproof then you can go for ASIC implementation.

(Refer Slide Time: 17:56)



So, this way actually in this course we will be using FPGA so that we can implement the digital hardware, hardware level logic level and that can be realized using FPGA. What is FPGA? The full form is Field Programmable Gate Array. What is this?

So, this is one of the architectures of FPGA where you can see these blocks are logic blocks. What is a logic block? So, the logic block consists of Look-up-Table, and we just saw two-input Look-up-Table, which require 4 SRAM cells, then how many SRAM cells will be needed for 4 input? So, it will be 2 to power 4; that means, 16 SRAM cells will be needed. SRAM cell will be needed for the 4 input lookup tables.

Then after that SRAM cell, there will be a mux then followed by you know after this mux you will get the output, but remember the output is a scalar. So, it is a scalar output even though you have 4 number of input, but the output is only scalar because it is y and it can be any you know if you draw the Look-up-Table it can take any value based on the status of ABCD, but it is a scalar quantity.

Now, once this output is developed then because this propagation delay can vary to avoid any data hazard generally we use a synchronous circuit that is a flip-flop that is for the storing element it is generally a d flip flop. And after that, we use a mux which means, that sometimes this can be used for storing purposes we can use you know for multiple purposes. So, you can store this value and then do the next operation you can do it pipeline operation, you can either pass the current value or you can pass the previous value.

So, this whole block is called logic block and that is the heart of FPGA. So, each FPGA can have thousands of logic cells you know that the logic cell configurable is called CLB. CLB Configurable Logic Block because this block can also be configured based on how are you storing the Look-up-Table SRAM values. And once this logic blocks then there each logic block can be connected by a programmable you can see programmable interconnect.

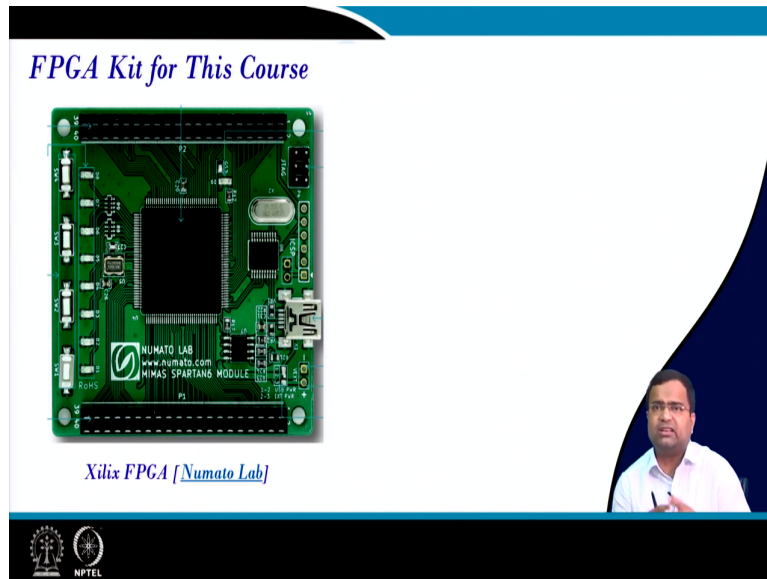
Because if you look at this the interconnects are also programmable there are switches and also at the terminal point of the IC we have IOPIN because outside there will be IOPIN and you can access the data and this IOPIN when it goes out. Then it has to have a buffer because it needs. After all, there will be a bond pad you know the conducting plate, which will behave like a capacitor and to overcome we need to provide sufficient current to charge and discharge the load capacitance.

So, the buffer circuit will be there and these blocks are also each of these pins are also programmable either you can use input or output. So, accordingly, the buffer arrangement will be readjusted. So, the bottom line is this and you can go through in detail how it is configured the buffer cell. So that means, what we learn the FPGA has many such configurable blocks and inside the configurable block there will be a lot of Look-up-Table and then the configurable blocks are also the interconnection that can be configured.

And also there is some RAM; which means, the memory storing element as well as DSP block for fast signal processing like a multiplier operation and all these. So, the dedicated hardcore you know a DSP unit, and they can be configured, and then IOPIN is also configured. So that means, none of the logic block output is connected to the input pin like physically it is through switches so that you can connect this kind and connect with any other logic block. Because you want to check the algorithm to different logic blocks it is possible.

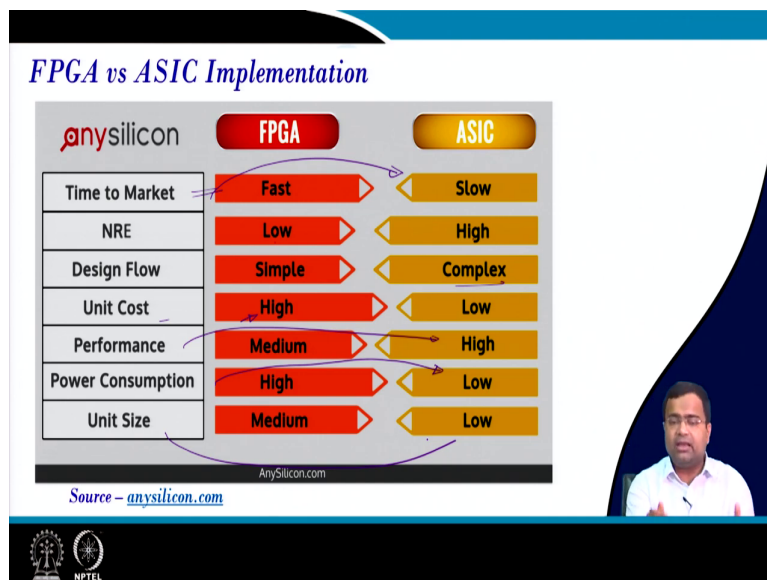
So, finally, when we learned HDL coding like a Verilog HDL that we will learn in our this code. So, very long HDL is a software platform language that will enable you to write your algorithm digital algorithm and develop it. Once you synthesize then this has to be implemented when it comes to implementation, either you can go for FPGA or you can go ASIC. So, accordingly based on the platform the implementation file will accordingly create a netlist and routing and then it will generate a bit file and will be dumped so, that you can make the interconnect accordingly.

(Refer Slide Time: 22:53)



So, the FPGA kit that will be used here is a Spartan 6 FPGA we are going to use, but yeah one can use any other FPGA.

(Refer Slide Time: 23:00)



So, now if you go for FPGA versus ASIC implementation we have realized the time to market; that means since FPGA you can use it for various algorithms. So, it is a very fast time to market you can develop very quickly any algorithm, but actually, if you go to ASIC solution and the hardware HDL code will learn in this course may be helpful to develop ASIC.


So, that development time is longer, but NRE cost is also high for ASIC and low for FPGA, but the design flow also in FPGA is simple it may be complex because it has to go through simulation routing you know layout and many other things then it will go to the fabrication. But the unit cost here is high here unit cost is low if you make it for mass production the performance will be very high.

Because they are dedicated hardware you can optimize delay and another thing, you can optimize power consumption low that is why such ASICs are you are the internal structure of or microcontroller and other architecture which has dedicated hardware and you can reduce the power consumption. So, for low-power applications, they are very useful, but they are dedicated hardware.


But the unit size is also low ASIC can be very low whereas FPGA can be high. Because it has to accommodate more number of transistor than we saw for AND gate almost 6 to 7 transistors are needed naturally the size will be very high for FPGA.

(Refer Slide Time: 24:34)


FPGA vs Microcontroller vs. ASIC Solutions



ASIC



MCU



FPGA


Source - Octopart.com


Specific ←

→ Generic

Source - fpgakey.com

Micro	FPGA
Write software.	Design hardware.
Typically executes one instruction at a time.	All parts of your circuit can operate independently.
Fixed maximum clock speed.	Maximum clock speed is dependent on your design.
A handful of I/O pins that can be accessed in small groups (typically eight) at a time.	Many I/O pins that can all be accessed simultaneously.
Usually store programs in nonvolatile (persistent) memory.	RAM based and needs to be programmed after power on (the Mojo does this automatically).
Often very power efficient with advanced sleep modes.	Power usage depends on your design but typically requires more than a microcontroller.
Fixed peripherals that limit the devices you can connect.	Can interface with virtually any digital device.





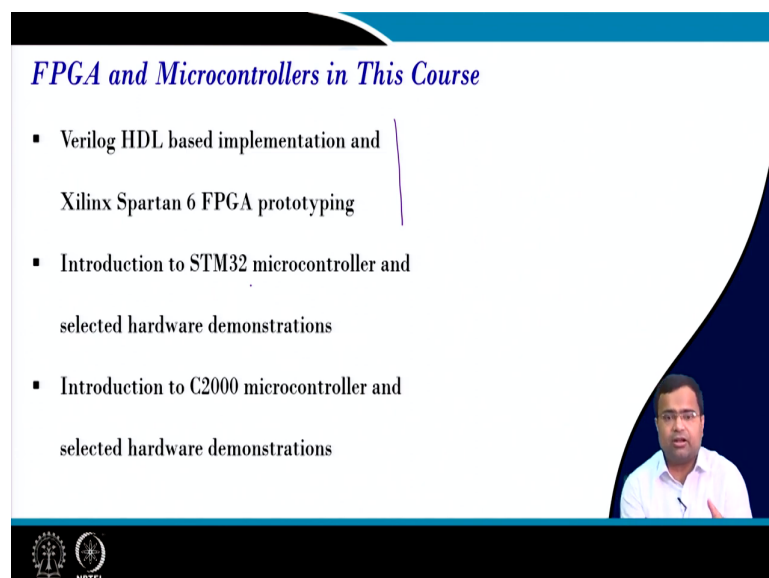
Now, if you are going for this comparative study ASIC microcontroller FPGA. So, you know the ASIC is the building block and inside the FPGA the hardware can be ASIC; that means, we have millions of transistors inside the microcontroller, and the microcontroller will have a dedicated digital functional block which will be harder optimized.

So, it is similar to ASIC and once you put in more and more flexibility you go to FPGA. So, from specific to generic the flow is like this way ok. And if you go to microcontroller you need software here the design hardware FPGA; that means, you can write directly. HDL code typically in microcontrollers in most of the earlier microcontrollers is a sequential operation. So, execution happens 1 by 1, but in FPGA you know concurrent operation parallel operation.

So, that is why the FPGA are popular for signal processing applications where you need to do a lot of mathematical computation like FFT and other algorithms which require parallel operation so; that means, but if you are talking about power consumption this power consumption will be high whereas, the microcontroller will low. So, it all depends on what application you are targeting.

But for power converter of the self, application let us say for high power applications when you want to use a dedicated digital control card, then microcontrollers are generally a very popular solution. But in this course, we are going to have some demonstration microcontrollers, but FPGA will be used as a platform where you can also go for ASIC solutions. That means you can make a digital ASIC IC the digital IC.

(Refer Slide Time: 26:19)



FPGA and Microcontrollers in This Course

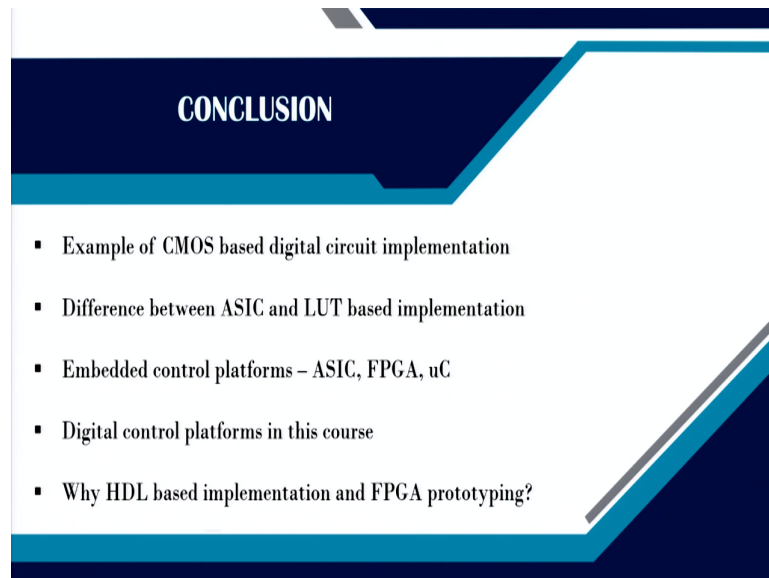
- Verilog HDL based implementation and Xilinx Spartan 6 FPGA prototyping
- Introduction to STM32 microcontroller and selected hardware demonstrations
- Introduction to C2000 microcontroller and selected hardware demonstrations

The slide features a video inset of the instructor in the bottom right corner and logos for IIT Bombay and NPTEL in the bottom left corner.

So, in this course, we will be using FPGA that which is what I told and that will enable we will use HDL-based coding we will also show STM microcontroller STM 32 and C2000

series. Where you will get to know a little bit of detail about the capability of individual microcontrollers from the design expert from SI microelectronics and Texas instrument.

(Refer Slide Time: 26:41)



CONCLUSION

- Example of CMOS based digital circuit implementation
- Difference between ASIC and LUT based implementation
- Embedded control platforms – ASIC, FPGA, uC
- Digital control platforms in this course
- Why HDL based implementation and FPGA prototyping?

So, in summary, we have given an example of CMOS-based digital circuit implementation using an AND gate and we saw that using ASIC requires 6 transistors using Look-up-Table which requires 40 number transistors. So, we can understand the difference between flexibility and hardware resource as well as the power consumption point of view.

So, ASIC is very good for low power as well as specific hardware, but Look-up-Table may be good for flexible hardware we have also understood some aspects of ASIC FPGA and microcontroller and what are the digital control platform that will be used in this course.

We have discussed and have also discussed touched upon you know why we are using HDL-based implementation and FPGA prototyping. So, I think we got an idea about various platforms of digital controllers and we will be using FPGA and 2 different microcontrollers. I hope you will get to know more detail about this in a future lecture. So, I want to finish it here.

Thank you very much.