Introduction to Embedded System Design

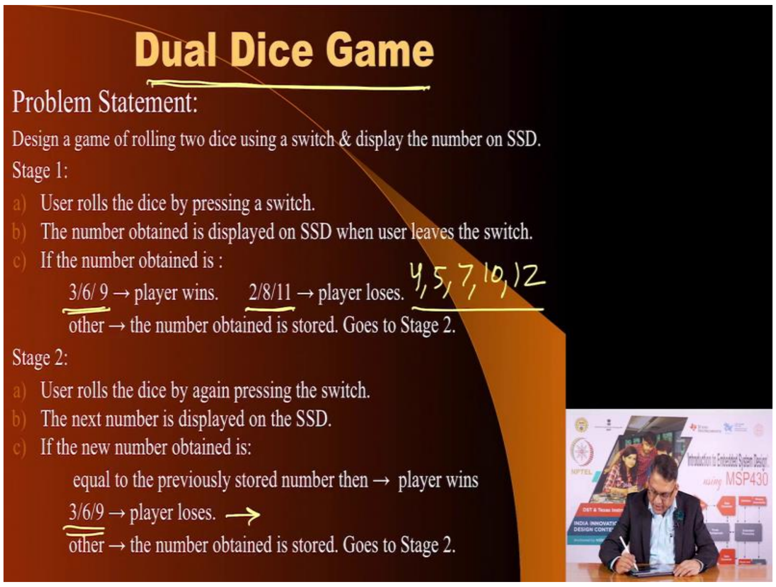
Professor. Dhananjay V. Gadre
Netaji Subhas University of Technology
Professor. Badri Subudhi
Indian Institute of Technology, Jammu

Module 12
Lecture No. 39
Single Purpose Computers- Continued

Hello, and welcome back to a new session and we are going to continue our discussions on Implementing Single Purpose Computers. I am your instructor Dhananjay Gadre. In the last lecture, we were we covered some ground to understand the components of a single purpose computer namely, we need a Sequencer and we need Data Path. The sequencer is nothing but a finite state machine, that finite state machine could be implemented either as a Moore machine or a Miley type of finite state machine it does not matter.

The data path elements will be determine by the requirements of the single purpose computer. So, to illustrate this idea I am going to present to you a single purpose computer which will play a game, which we call a Dual Dice Game. Before we start implementing the game as we always do whenever we have to implement something the first and foremost task is set down on a piece of paper the requirements unless you know what you require expect out of that implementation things will not move forward and so, here is the problem statement.

(Refer Slide Time: 1:33)



Dual Dice Game

Problem Statement:

Design a game of rolling two dice using a switch & display the number on SSD.

Stage 1:

a) User rolls the dice by pressing a switch.
b) The number obtained is displayed on SSD when user leaves the switch.
c) If the number obtained is :

3/6/ 9 → player wins.   2/8/11 → player loses. 4,5,7,10,12

other → the number obtained is stored. Goes to Stage 2.

Stage 2:

a) User rolls the dice by again pressing the switch.
b) The next number is displayed on the SSD.
c) If the new number obtained is:

equal to the previously stored number then → player wins

3/6/9 → player loses. →

other → the number obtained is stored. Goes to Stage 2.

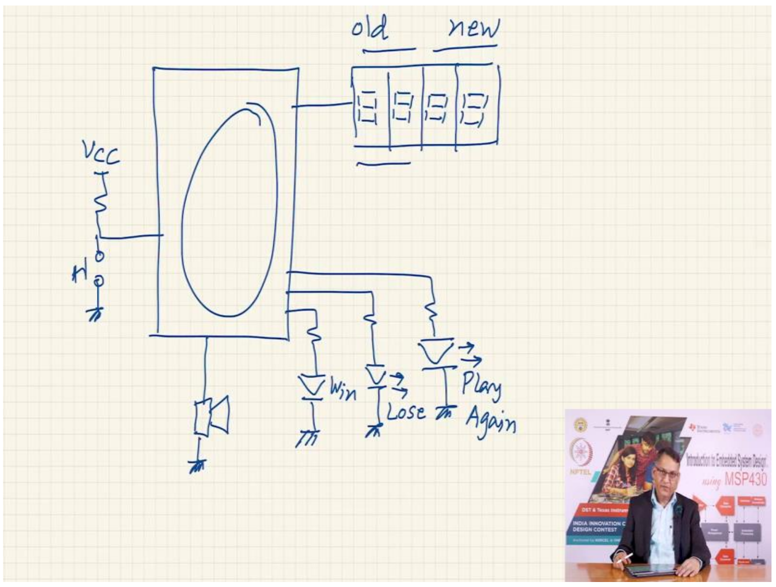
We are having, a game where there is a there are two dice that we roll now, when we roll a dice the number is between 1 and 6, when we roll two dice the purpose is actually to sum the output of these two dice and therefore, this dice will produce a numbers between 2 and 12. Based on this sum, we will take certain actions. So, when we roll the dice, the two dice the number will be displayed, the sum of these two dice will be displayed.

If, the sum is a value between 3, 6 or 9 that is the sum of two dice is 3, 6 or 9 the player wins and the game terminate there. If the sum is 2, 8 or 11, then the player losses and game ends again. In case, the number is anything other than these which means if it is 4 or 5 or 7 or 10 or 12, then the game proceeds to second stage, and before proceeding to second stage it stores the current number that was received which means, that number had to be one of these numbers.

In the second stage, the user is prompted to roll the dice again and the new number is displayed on the 7 segment display together with the old number you would like to know what your previous number was because the in the second stage you are going to take some decision based on the previous value of that number also and so, if the new number matches with the old number you win.

On the other hand if, the new number is 3, 6 or 9 then you lose else, you the current value of new number that you got becomes the old number you store it and prompt the user to roll the dice again and you stay in stage two till a decision comes out. Let me draw for end user what would the game look like.

(Refer Slide Time: 3:34)



So, we want a system which has mechanism to roll the dice and we do not have a physical mechanism where, we are shaking it and instead we have a switch with which we can simulate the act of rolling the dice. So, when I when I press the switch it is like I am rolling the dice. So, dice is being I am shaking the dice and when I realize the switch it means, that I have thrown the dice and you see the two numbers that come out and so, I would like to display them I need 4 7 segment because I have a number which can be more than 10 and I have to display 2 such numbers or the current stage and may be if, happen you go to the second stage.

So, I have four 7 segments displays here. Now, how does the user know that the game I have won or I have been lost or the game has gone to the second stage? So, we have to use some indicators to indicate. So, let us say that use a LED which is appropriately labeled so, this could be win, another LED could be to indicate that the user has lost lose, a third LED in case neither win nor lose then it must indicate that now the user has to press the switch again so, we call it play again or PA play again.

You could also, have a buzzer to sort of you know to indicate some result you may also have some LEDs to indicate are you in stage 1 or are you have gone to stage 2 so on and of course, these numbers could be old numbers and these could be new numbers this is the user interface to the system. Now, we have to design everything that goes with here with the help of logic circuits which will be portioned into two halve, one will be called the sequencer, what will be the sequencer do? It will essentially follow the rules of the game, it will prompt the user to press a switch.

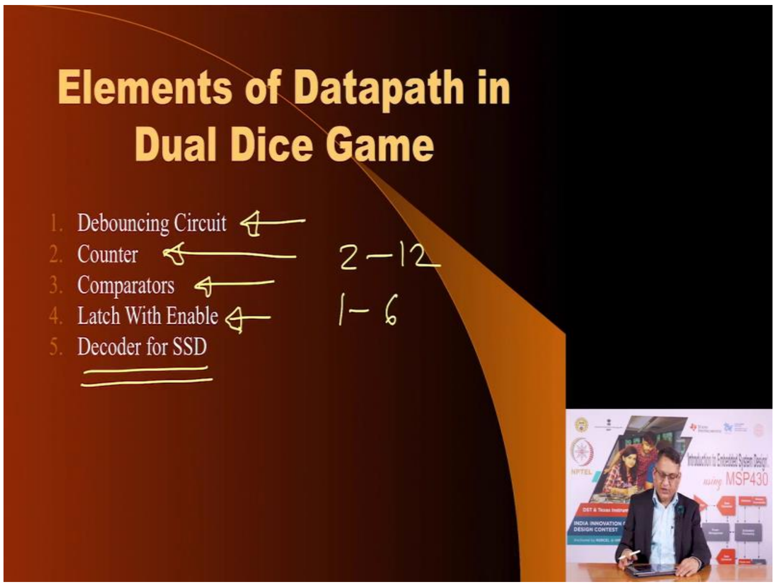
When, the switch is pressed will de-bounce the switch and then it will roll the dice when the switch is realized again it will de-bounce the switch and then it will stop the dice look at the numbers and there are many ways of processing that information may be, you have two dice and then use a arithmetic number adder to add the two numbers or you can take a alternative approach by saying let us make a counter, a single counter which counts from 2 to 12 so, it goes from 2, 3, up to 12 and go back to 2.

Now, of course there is a difference in the way these number will appear in this two approaches. If you use two counters which have a number between 1 and 6 for both of them and use adder to add these numbers the probability of some of the numbers will be more than the probability of other numbers.

In this case compare to a solution you can choose that you have a single counter which has a number between 2 and 12 because in this case the probability of each number is same where as in the earlier case of using two counters some numbers will have higher priority and there are again various methods of you know, improving the probability of certain numbers even in the single counter method.

So, this is the over-all block diagram. Let us look at what are the kind of elements of data path that we may require.

(Refer Slide Time: 7:31)



Elements of Datapath in Dual Dice Game

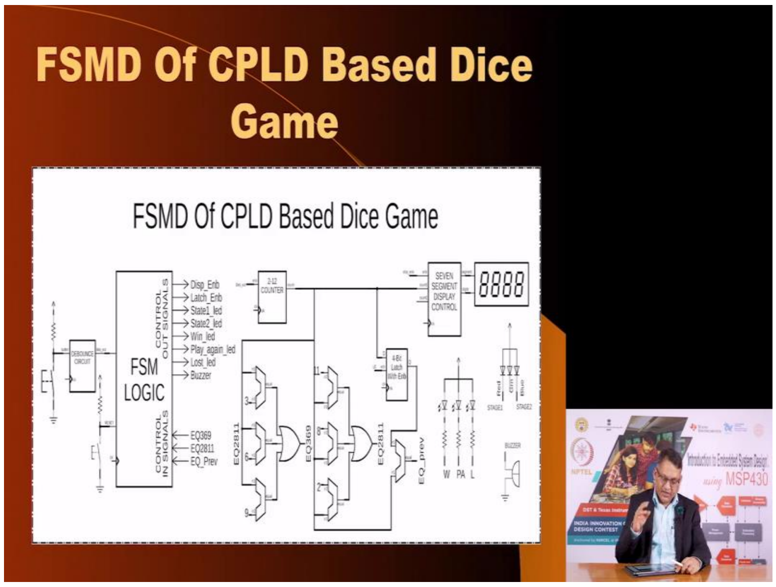
1. Debouncing Circuit
2. Counter
3. Comparators
4. Latch With Enable
5. Decoder for SSD

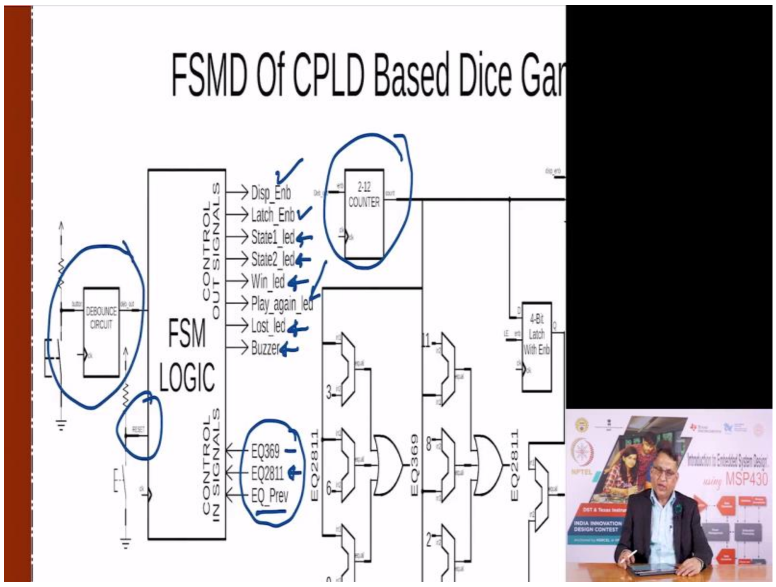
We would require, the de-bounce circuit because, there is a switch which mechanical switch in nature and when the user will press and realize they will produce switch bounce we need some counters to implement the random number generator as I mentioned it could a single counter to go from 2 to 12 or it could be two counters each of which have a number from 1 to 6 but, you will require another logic block to add these two numbers.

You may require comparators why because now you have to decide what is my number, what is the result of that rolling of the dice, is it 3 you will win, or if it 6 you win if, it is 2 you lose and so on. So, that is best done using comparators so you know what is the result of that. Then you may want if you in the stage one you do not win or lose either then you have to go to stage 2 you have to store the current number which will do using the latch with enable signal and of course you will require decoders to display the numbers on 7 segment displays.

So, broadly these are the building blocks the data path building blocks that you have require and let us look at the representation of these blocks with the interconnection to show, to evolve from our single block diagram user level block diagram to a more detailed interaction between the circuit elements.

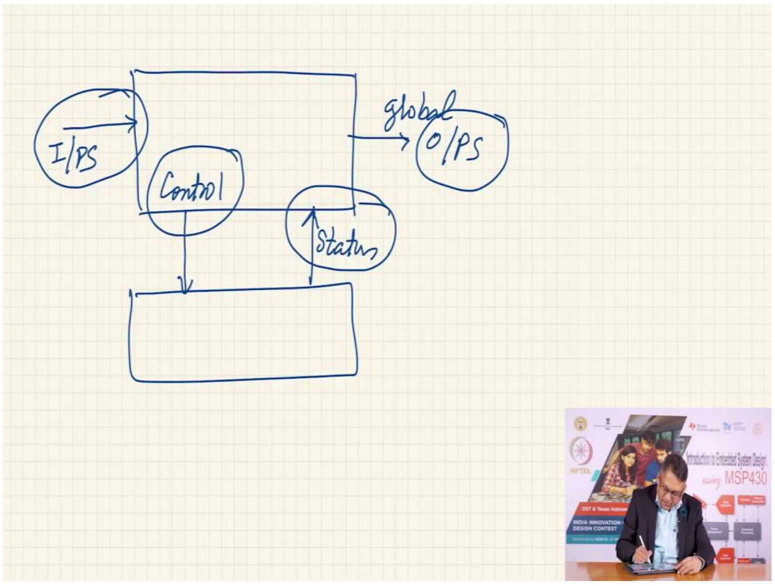
(Refer Slide Time: 8:59)


FSMD Of CPLD Based Dice Game



Here is the complete block diagram which involves the elements of the data path as well as single block which will represent our finite state machine. So, I am go to zoom on this part of the circuit to first talk about the finite state machine. So, here as you see the finite state machine produces two types of outputs if, you remember that let me draw for your easy reference.

(Refer Slide Time: 9:29)



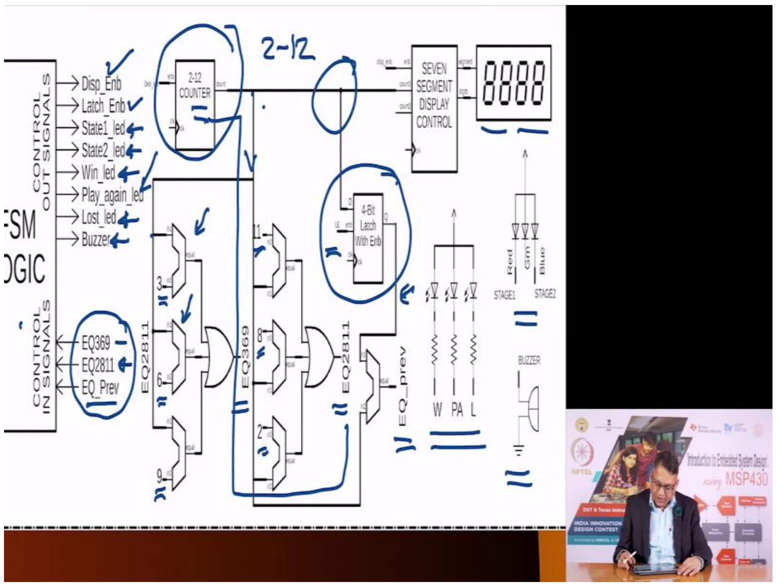
That a state machine will produce two sorts of outputs, two types of outputs, one which could be for global outputs so, these are global outputs and it could produce outputs which are control signals to the data path elements. So, these are control and it does that with the help of asynchronous inputs together with inputs coming from the data path elements and so, these are starters inputs.

So, you have to design a finite state machine were the inputs are the the asynchronous inputs as well as the starter inputs and the output such state machine will be two sets of outputs, one is the global outputs and the other will be control outputs which go to the data path and so, we must see the both sets of outputs together and this what is being done here if, you see if you look at this signal win, lose, play again and buzzer these are global outputs.

You also have, stage LED 1 and stage LED 2 this indicates whether you are in stage 1 or stage 2 and you have other control signals which is to display you want to enable or disable the display or you want to enable the latch and you have another signal which is the control signal for the counter, these are the global and intermediate outputs and then you have some inputs coming from the state data path and these are when the values are equal previous and new values are equal or if, the current value is equal to 2, 8 or 11 we have labelled it as single signal EQ2811.

And another signal which indicates whether the sum is 3, 6 or 9 and the other asynchronous input is the reset to reset the circuit to begin form scratch and the switch which is the actual switch used to interact with the game.

(Refer Slide Time: 11:48)



Now, let us zoom this part of the circuit to see how the various data path elements are required and what functions they serve. So, the first element of the data path is a counter which produces is a 4-bit counter, it produces numbers between 2 and12 this go into the 7 segment display control circuit it is a 4-bit binary number and it is converted into decimal number which could be anywhere between 2 and 12, and we displayed either these two digits or these two digits.

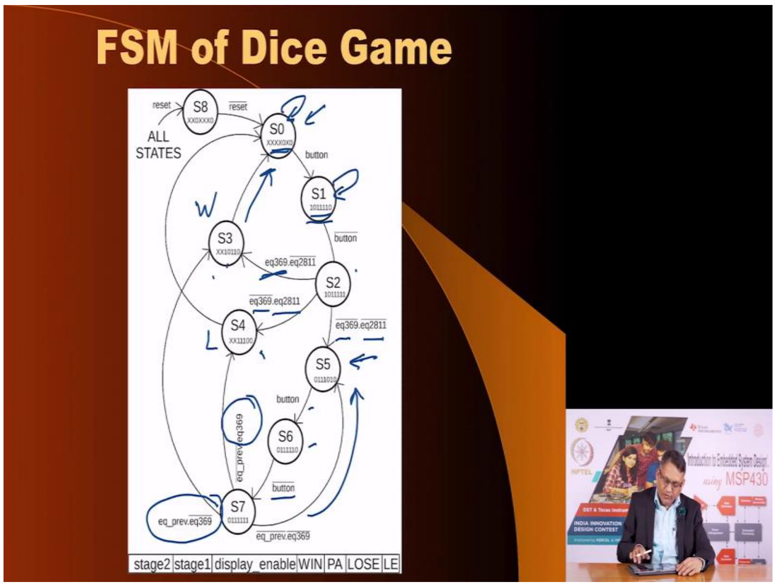
Then, in case you do not win or you lose it will go into stage 2 for before that you do that you have to store that number and it stored into the latch, it is a 4-bit latch enable signal and then if the number you have to find out whenever you rolled the dice is a number equal to 3 that you do with this comparator or if the number is 6 you do it with this comparator, two inputs of this comparator are 1 which is coming from this counter and the other which is fixed value of 3 here, 6 here and 9 here and these 3 outputs are ORed together to create single signal called EQ369.

So, if EQ369 is one that means your sum was either 3 or 6 or 9 and the state machine can appropriate action that is to declare either win if you win stage 1 or if, you lose in stage 2. Similarly, the same number is going to 3 comparators here, one where the input is 11, second input is 3 and third where this input is compared with the current output and if, this output which is a OR of these three comparators, if it is 1 it indicates that your number will either 2, 8 or 11 and again this signal is going to the finite state machine to indicate to the finite state machine the result of your of this roll of dice and to for it to take appropriate action.

In case, in stage one you neither win nor lose the finite state machine will generate a control signal to capture this input and it will be available at this point and it will be compared with current value when you rolled the dice again this current value is available here also and if these two numbers match then it is called EQ previous and this goes here again to the state machine as status signal and the state machine will decide what to do with that number.

Apart from that, here are 3 LEDs for win, lose or play again, here are 2 LEDs to indicate stage 1 or the stage 2 of the game, a buzzer to beep whenever certain action is taken. So, this is the representation. Now, we are still not anywhere in the completion we have to now decide what are the states required in the finite state machine. So, let us have a look on that here is the finite state machine.

(Refer Slide Time: 15:12)



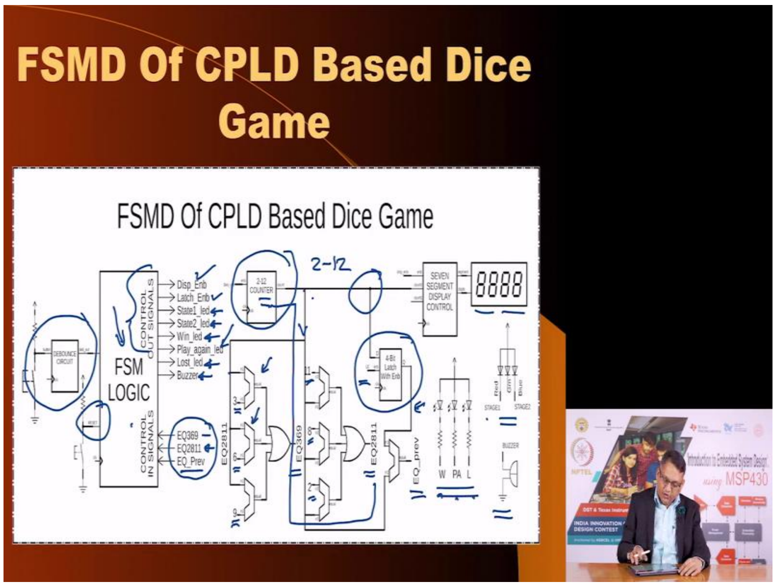
FSM of Dice Game

reset
S8
XX0XXX0
$\overline{reset}$
ALL STATES
S0
XXXX0X0
button
S1
1011110
$\overline{button}$
W
S3
XX10110
$eq369.\overline{eq2811}$
S2
1011111
$\overline{eq369}.eq2811$
S4
XX11100
L
$\overline{eq369}.\overline{eq2811}$
S5
0111010
button
S6
0111110
$eq\_prev\overline{eq369}$
$\overline{button}$
S7
0111111
$eq\_prev.\overline{eq369}$
$\overline{eq\_prev}.\overline{eq369}$
stage2 | stage1 | display_enable | WIN | PA | LOSE | LE

You start with S0 that is your reset state, you wait for a button if, the button is pressed you going to S1 if, the button is not pressed you remain in this state not mentioned here but, that is what is being done if, the button is not pressed you remain S0, if the button is pressed you go to S1.

In S1 now in each of the state certain outputs are being produced and I will show you the next slide where we have made a list of these outputs and we know these outputs, what are these outputs?

(Refer Slide Time: 15:45)



FSMD Of CPLD Based Dice Game

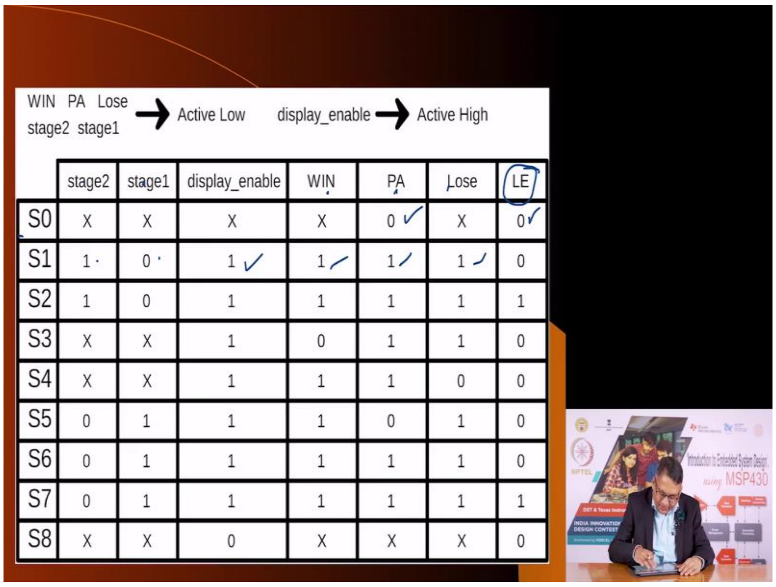
These outputs various values as well as which consists of control signals to the data path and 3 global signals which is win, play and lose and 2 more global signals which is to indicate whether you are in stage 1 or stage 2. So, all these outputs from the state diagram state machine are mentioned in each of the state S0 or S1. Now, these outputs are either 0 or 1 but, in some cases it is x means we does not matter which means it says do not care which would mean that is ok if it 0 or it is ok if it is 1 may be the previous state would decide if it will remain 0 in the state and so on.

So, those state are mentioned x, x here so, you go from are in state 0 waiting for button is pressed you come in state S1 that you start counter that will see when we look at these numbers here, you wait till the button is not realized when the button is realized it will going to S2 here, numbers are ready and they will be displayed if, the number is EQ2811 and not EQ369 you are go in the state as four which actually to indicate that you lost.

On other hand, if EQ369 is 1 I should not put a bar on it if, the state 369 is 1 and EQ2811 is 0 it means you are wining and state and appropriate output will indicate whether it win or lose and form there you will go into this waiting for button to be pressed else, if neither win nor lose you are going to state where you are request the user to press the button again by the help of LED called play again and once it is pressed you are going to the state here you are going to start the counter.

The counter, to produce second number wait for the button to be realized and then once the number is generated you are going to compare it with the previous number here if, it is equal to previous and not anything else you are going to win state, this is your win state, this is your lose state, this is your state to indicate your stage two if, the numbers are not equal then you go back if the number is equal to 369 then you lose else you go back and ask the user to press the switch again you will still in stage 2.

(Refer Slide Time: 18:31)



WIN PA Lose stage2 stage1 → Active Low display_enable → Active High

	stage2	stage1	display_enable	WIN	PA	Lose	LE
S0	X	X	X	X	0 ✓	X	0 ✓
S1	1	0	1 ✓	1 ✓	1 ✓	1 ✓	0
S2	1	0	1	1	1	1	1
S3	X	X	1	0	1	1	0
S4	X	X	1	1	1	0	0
S5	0	1	1	1	0	1	0
S6	0	1	1	1	1	1	0
S7	0	1	1	1	1	1	1
S8	X	X	0	X	X	X	0

Now, to know what outputs are produced here is the table, here is the table which indicates the outputs required stage 2, stage 1, display enable, win, play again , lose and latch enable. This latch enable signal is to generate the is to enable the latch to capture that number. In state S0, you want the play again since LEDs are connected in Active low mode which means the LED will light-up when the output is 0, you are only lighting the play again LED, you have disabled the latch you do want to capture the number and rest everything does not matter, the counter is free running so, it is going to keep on running.

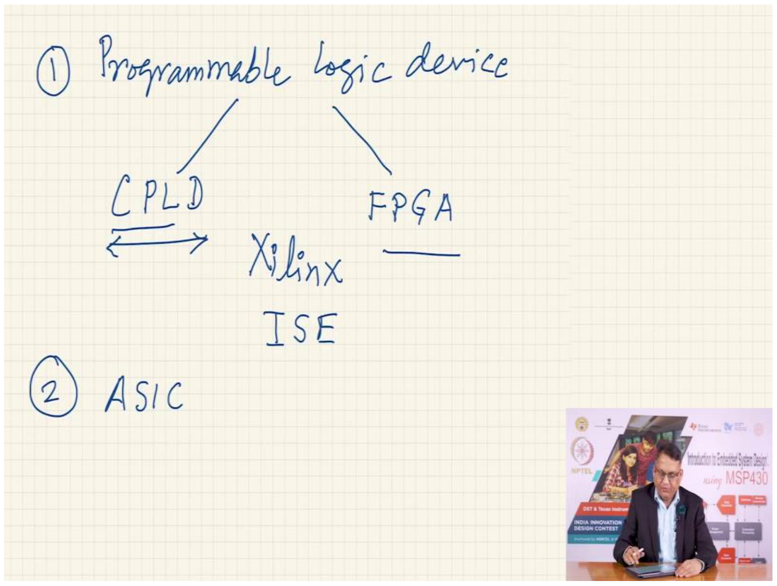
In S1, you are indicating that you are in stage 2, stage 1 because again its active low LED, you have enabled the display you are indicating that it is neither win nor lose nor play again and you have enabled the latch and from here you see each of the stages have indicated kind of outputs that will be produced some of the outputs go to the outside world in the form of win, play again or lose or stage 1 and stage 2.

The rest of the outputs go into the data path controlling those data path which we name is the counter and the latch and the display control circuit. In this table it is required. It will list all the states and the expected outputs and these numbers you see here x , x, x, 0, x, 0 will exactly be here x, x, x, 0, x, 0 and like-wise you see in each of the states we have mentioned what numbers what outputs are produced what state of those outputs.

Once, you decide this as I mentioned this is this finite state machine it can be implemented as a piece of HDL code all the elements of the data path whether it is counter or 7 segment display or these comparators can be again represented as the HDL code. So, now you create the single HDL code where you write down the code for each of these building blocks, finite state machine, the counter, the latch, and these comparators and so on and when you compile tha,t you will get a net list which will implementing in any form.

Now, what are the options we have? So, we can implement this net list at various in various ways.

(Refer Slide Time: 21:21)

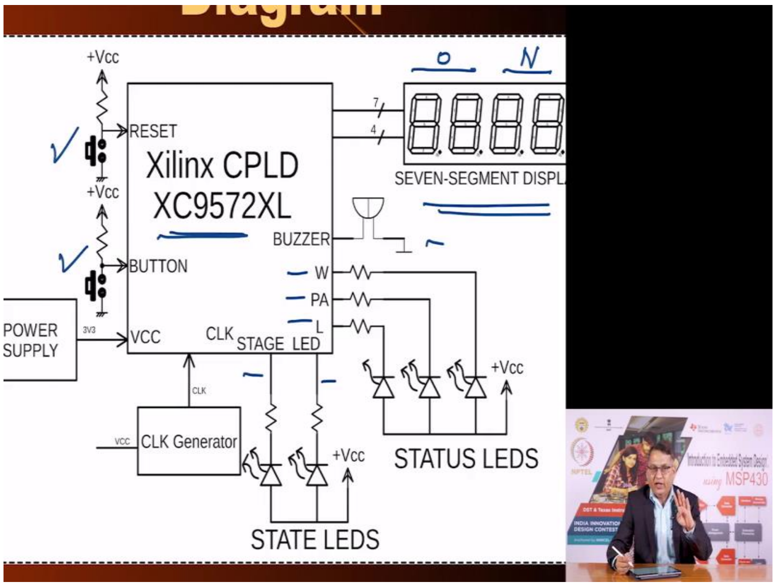


One common method is to use a Programmable Logic Device which are primarily of two types you either have a simpler, smaller device called Complex Programmable Logic Device or you have much more denser in terms of the number of gates you have a Field Programmable Gate Array. There are many manufacturers which make a CPLD and FPGA devices one very popular manufacturer is Xilinx.

They also provides a software for modelling codes and then targeting there products with they makes CPLD as well as FPGA they have they have a software called ISE which allows you to write and test, compile and download the HDL or long code into the IC which could be CPLD or the FPGA so, this is one approach.

So, second approach could be that given the net list you can get application specific integrated circuits that being made that is another possibility. Now, what we are going to show is to how to take the code that we have written compiled it by using an appropriate compiler in this the Xilinx ISE and we show implementation using the CPLD.

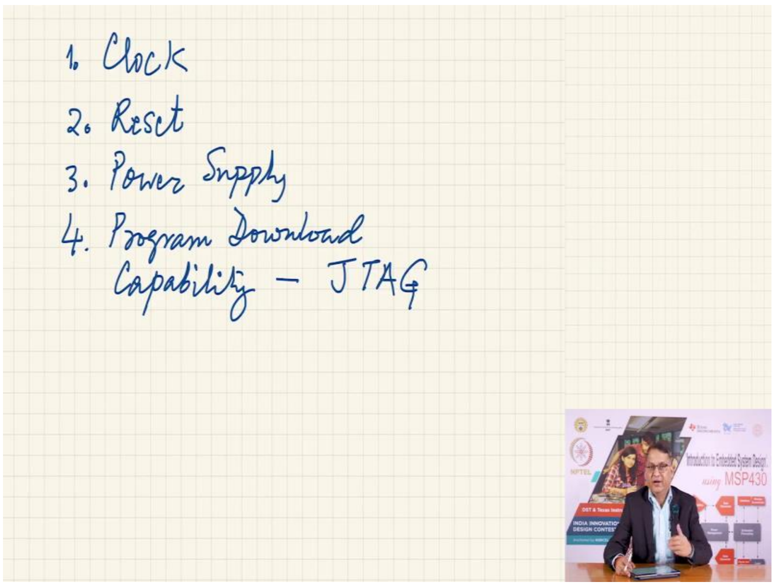
(Refer Slide Time: 22:58)



This is the block diagram of the implementation as you see here, here is the Xilinx CPLD XC9572. It has two switches reset and the button which is used by the user to interact it needs a power supply I am to going to come to that here is the display which indicates the new number here, and the old number here, these are the three starter LED which indicates win, play again and lose, two LEDs which1 is stage one and stage two and a buzzer.

Now, exactly like we talked about the Eco-system for a Microcontroller and we mentioned 4 elements of eco-system.

(Refer Slide Time: 23:43)



1. Clock
2. Reset
3. Power Supply
4. Program Download Capability – JTAG

Namely 1 was clock, 2 was reset, 3 was power supply and 4 was program download capability, these four eco-system elements that are required essential elements that are required by an Embedded Computer such as the microcontroller also required by programmable logic device such as the CPLD or FPGA and this previous logic diagram indicates that.

Here, is the power supply from the data sheet of this CPLD we found that the power supply required is 3. 3 volts it also mentioned the data sheet how much the current it will consume. So, the power supply has to be appropriately design so, that can provide that power, the source of the power supply important in this case implemented using USB input which is 5 volts you will require a clock.

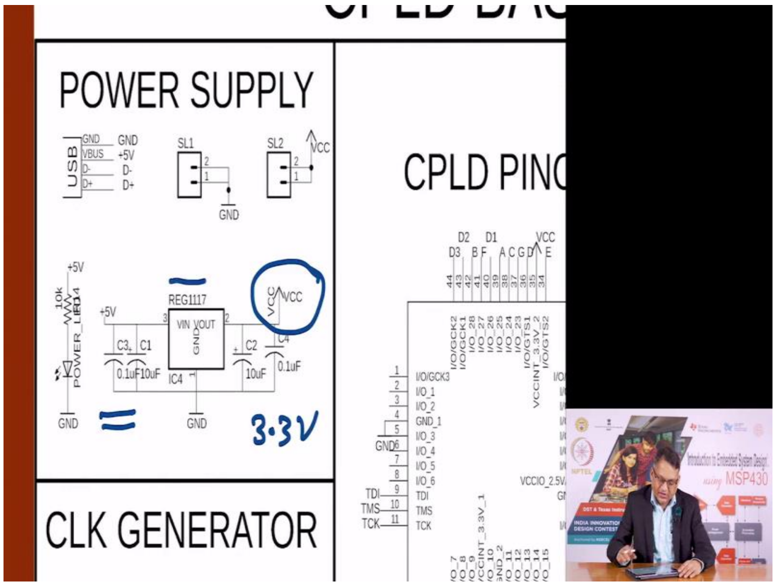
So, power supply and clock and this case the clock was implemented using a triple 555 a stable multi-vibrator because, the requirement was at very high frequency otherwise we could use other approaches and off course it requires a reset so, this is your reset switch.

So, in this case there is only 1 reset which is the user reset and power supply and the clock generator how do I download code into this? Most CPLD's and FPGA's are require JTAG, require a JTAG interface and so, whenever you want to download the code into your CPLD and FPGA you need a mechanism such as, JTAG correctors, JTAG interface to download the bit file.

Basically, when you compile the code, the compiler will produce what we called it as a fuse bit file meaning the fuses in your programmable logic device set on or off to create the circuit that what you want and this output of a compiler VHDL fuse bit file that needs to be downloaded into the target device and for that JTAG is used.

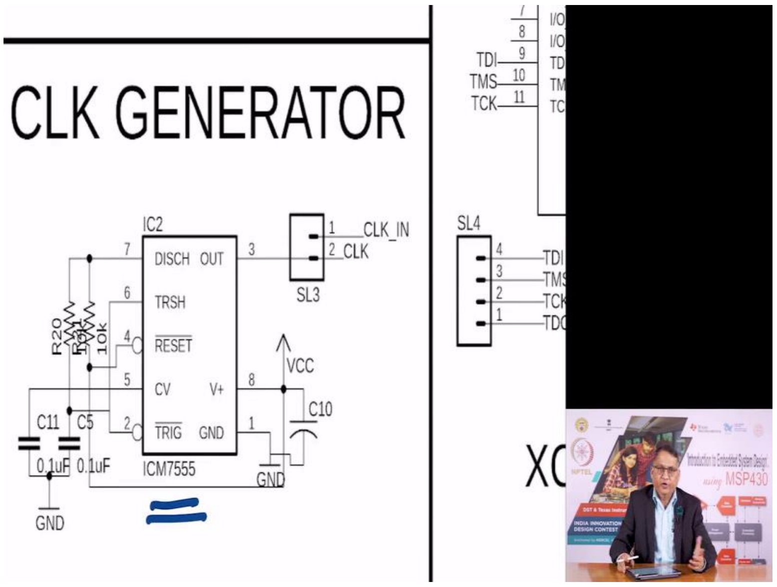
So, what we did was tested the circuit first in a evaluation kit like we have evaluation kits for microcontrollers, there are evaluation kits for programmable logic devices also and once the code was found to be working, then an entire Stand-alone implementation was created and let me show you that this is the systematic diagram of the CPLD.

(Refer Slide Time: 26:35)



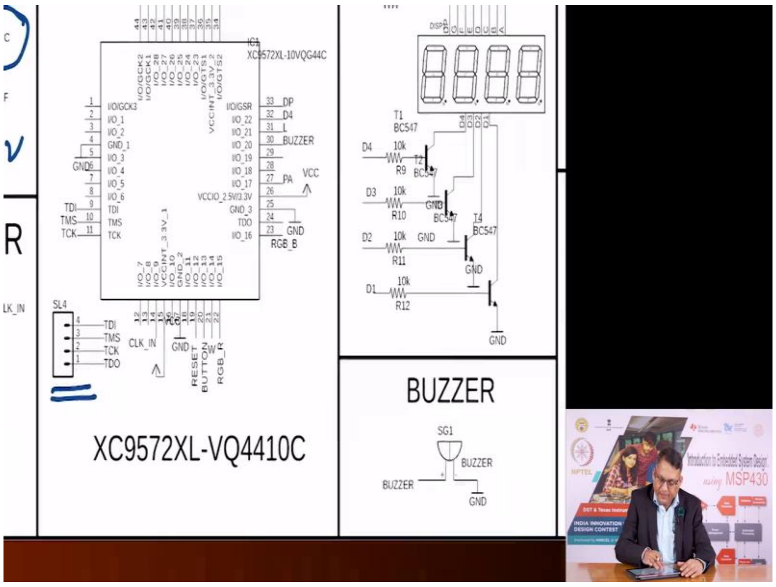
CPLD based dual dice game here is the power supply as you see the input is 5 volt source it produces VCC which is in this case 3.3 volts it use as the voltage regulator called triple 1117.

(Refer Slide Time: 26:51)



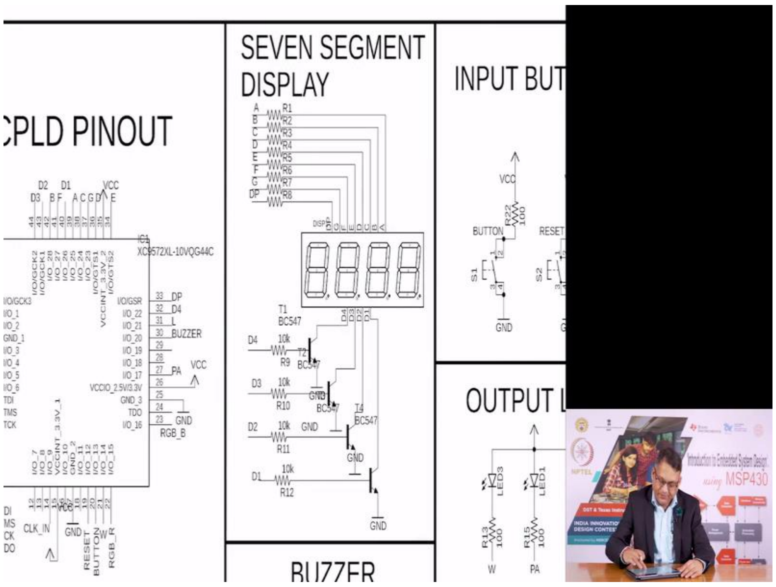
The clock generator is based on a 7 triple 555 a stable multi-vibrator produces about few kilo hertz of clock we do not need very high frequency clock.

(Refer Slide Time: 27:04)



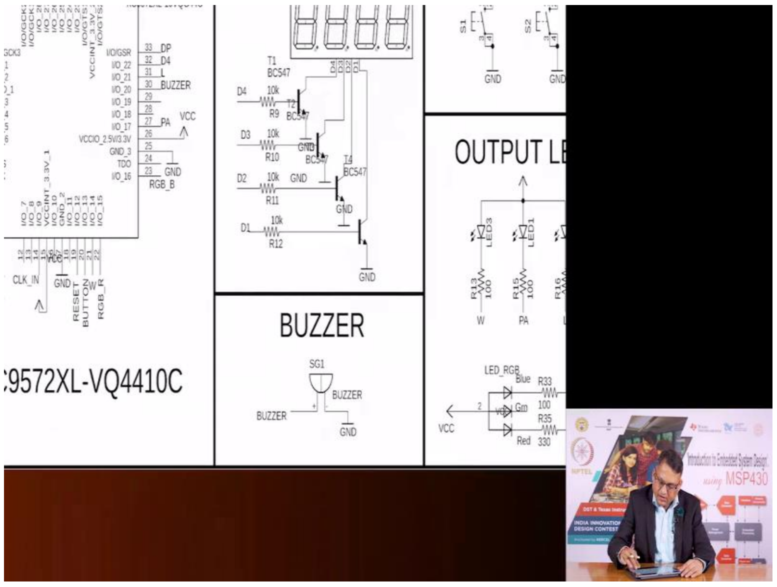
Here is the CPLD which is a XC9572 various outputs of the CPLD are connected to various peripheral devices such as displays, and buzzers and LED's and here is that JTAG collector which is TDI, TMS, TCK, and TDO this is an external collector external interface.

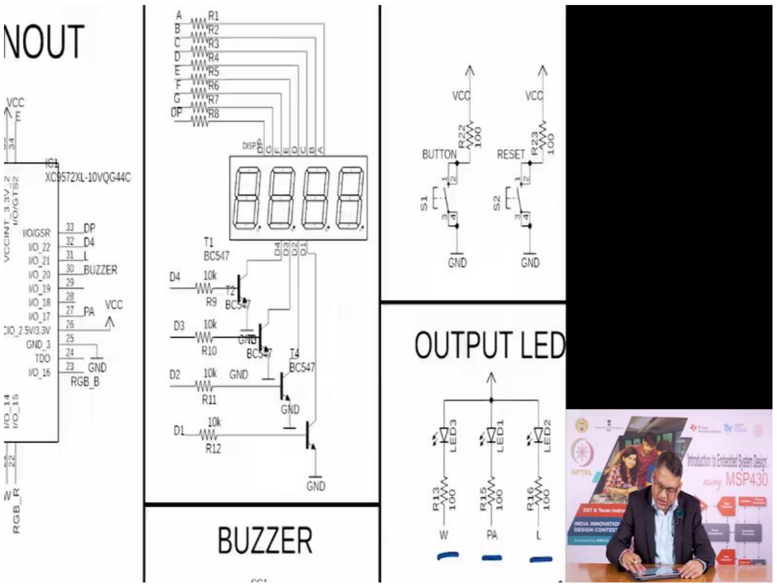
(Refer Slide Time: 27:26)



Here, is the display which is a four digits 7 segment display.

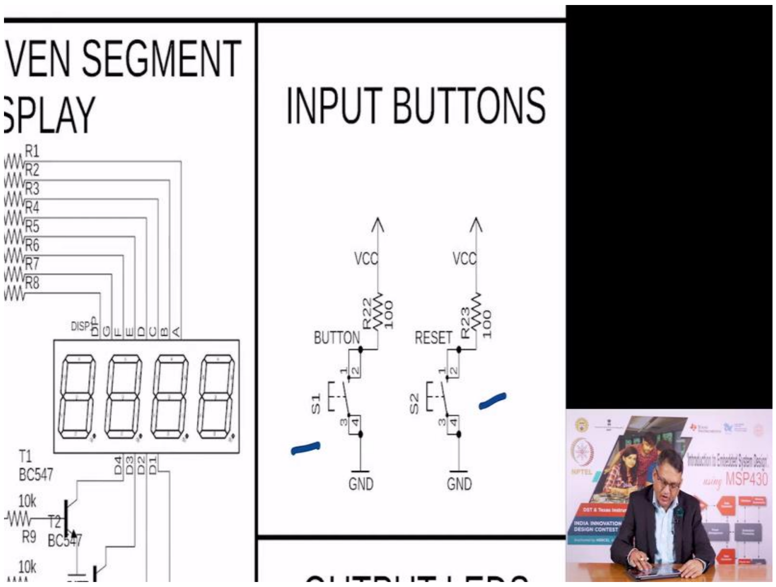
(Refer Slide Time: 27:30)





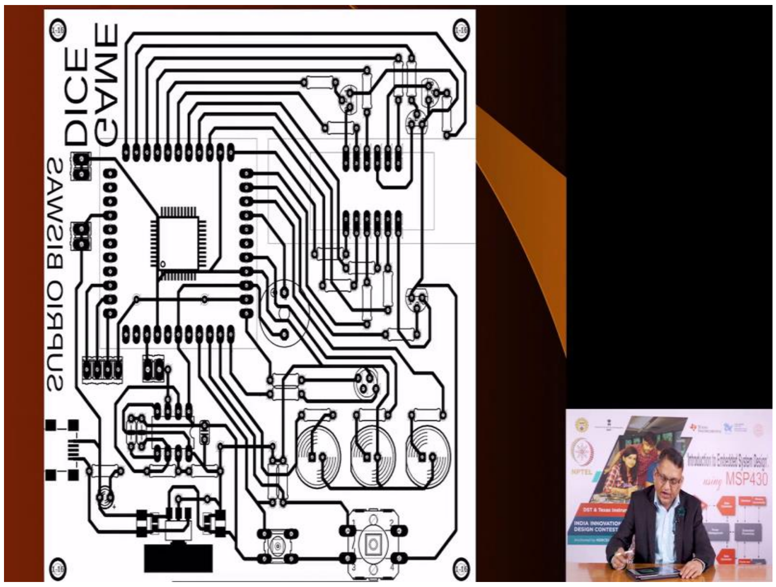
Here is the buzzer. These are the output LED's for win, play again and lose and also stage 1 and stage 2.

(Refer Slide Time: 27:42)



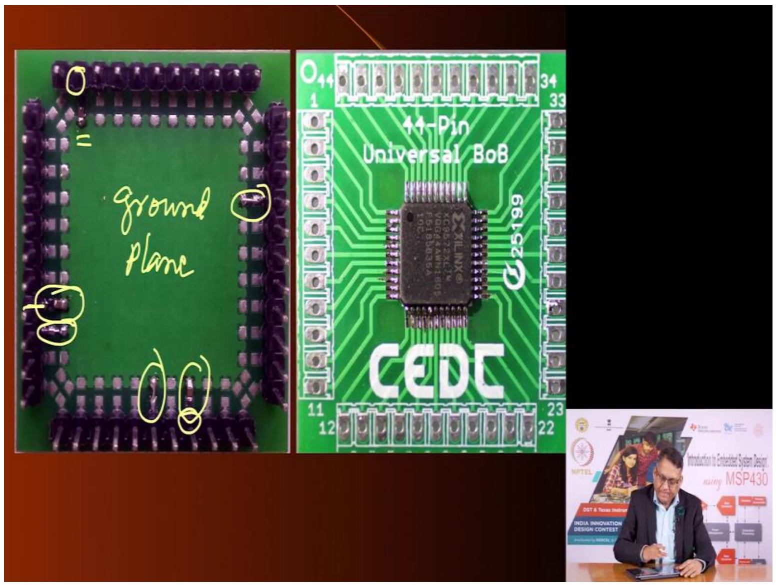
And these are the two switches for playing you the user button and the reset switch this is what we have, here is the broad layout.

(Refer Slide Time: 27:52)



This was fabricated in my lab on a single sided broad. Now, the CPLD itself is consist of ASMD IC which has very fine pins and it is impossible to fabricate a circuit to accommodate such a fine pin IC an interesting approach as I mentioned earlier was taken to create a break-out broad.

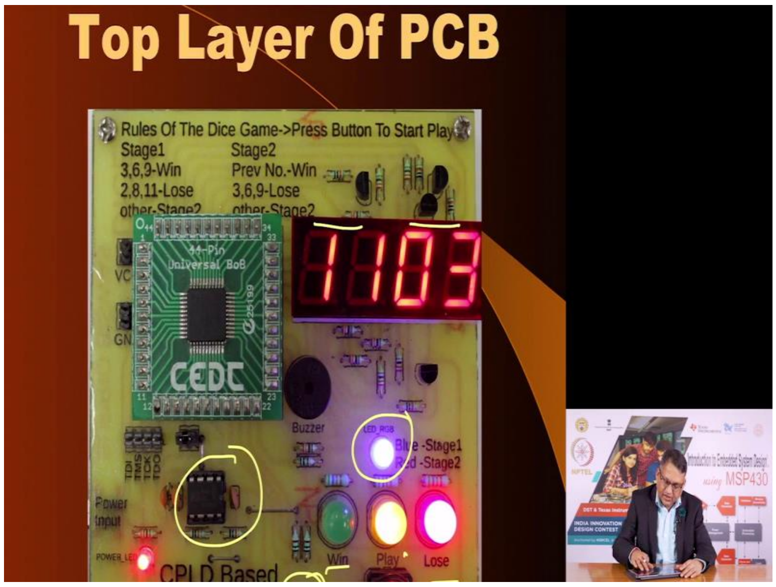
(Refer Slide Time: 28:19)



This is the break-out broad, this was fabricated using outside fabrications facilities and on that this is actually a 44 pin Universal BoB which means any 44 pin IC can be sold it onto this and at the bottom appropriate grounds planes this is going to be ground plane, this is ground plane.

So, appropriate depending upon which pins of IC that you are soldering it to be grounded here, here, here and somewhere here, some of the pins are VCC so, they need capacitor to be connected to the ground you see a capacitor here, you see a capacitor here, and you see a capacitor here, thess this pin must be VCC, VCC, and VCC and this is what its looks like at the top and the bottom.

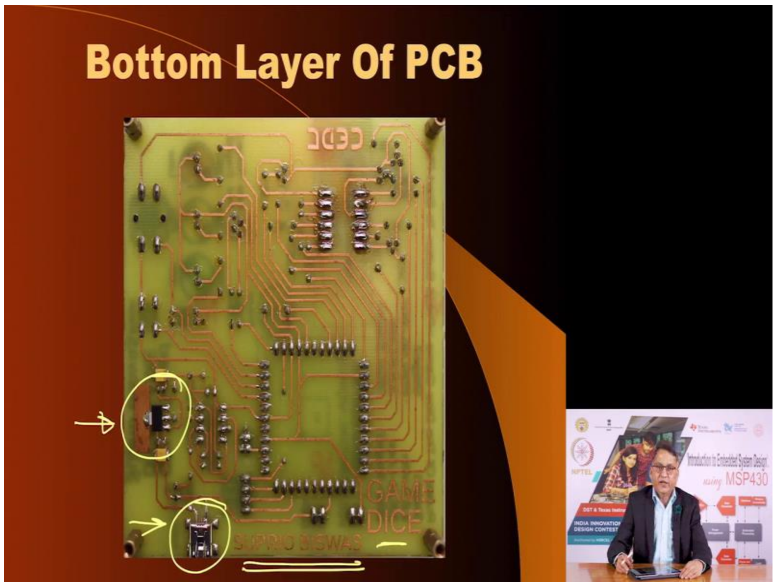
(Refer Slide Time: 29:14)


Top Layer Of PCB

Rules Of The Dice Game->Press Button To Start Play
Stage1
3,6,9-Win
2,8,11-Lose
other-Stage2
Stage2
Prev No.-Win
3,6,9-Lose
other-Stage2

This is the actual implementation. This is the top layer, here are the win, play again, and lose LED, this indicates the stage you are in, this is old number, this is new number, this is the clock generator, this is reset switch, and this is the user switch which you press, these are the rules which the game are mentioned.

(Refer Slide Time: 29:34)



So, this is the bottom layer of the PCB which has the power supply regulator, this is the USB connector and labels have been labels have been put to indicates it is, who implemented this is name of my student who implemented this, it is your provocative what you want to print.

The PCB technique, offers you that flexibility and I hope you enjoyed this lecture. It gave you inside how to design and implement single broad computers on programmable logic devices. A CPLD is typically, has much less resources and so, you have to really struggle with the logic to optimize it if, you have access to FPGA which have 10 of thousands and lakhs of gates.

You will be able, to implement larger designs but, the approach still will be same, it is an approach which has a sequencer, it has data path elements and it is left to you to decide what elements of data path are required, how are they controlled by the sequencer, what outputs do they produced, what kind of global outputs they produced, what kind of status outputs they produced, once you integrate two building blocks and you model it using the HDL in a single program file to integrate everything, you will be able to implement your own single purpose computers.

I hope you found it very useful to, I hope you will be able to learn from it and implement your own single broad computers. Thank you, I will see you very soon.