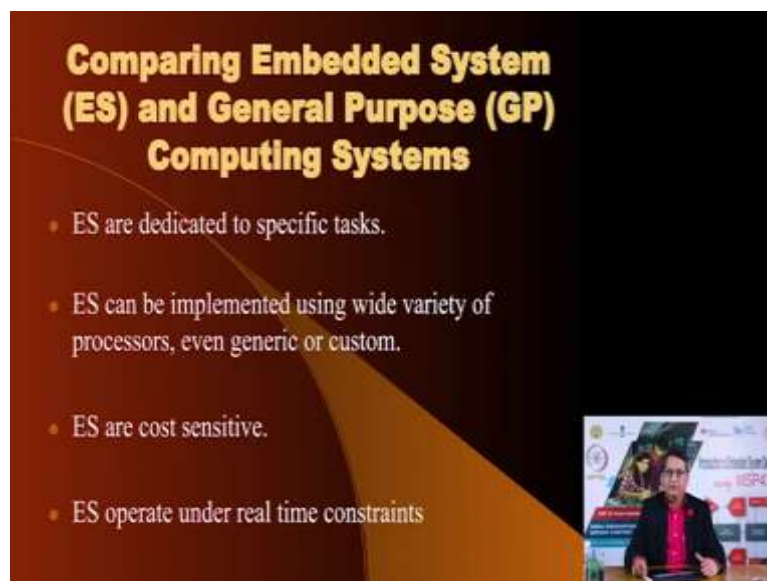


Introduction to Embedded System Design
Professor Dhananjay V. Gadre
Electronics and Communication Engineering
Netaji Subhas University of Technology
Badri Subudhi
Electrical Engineering Department
Indian Institute of Technology, Jammu
Lecture 02
Introduction Continued with Project Demos

Welcome back. So we are going to resume our discussions and in this session we are going to compare embedded systems and general purpose computing systems.

(Refer Slide Time: 00:33)



So the first difference is that embedded systems are dedicated for specific tasks. They maybe one task or there may be few tasks, there may be more than one task, but that number is fixed. As against general purpose computing system where you can put any applications, you can add applications embedded systems are different from such general purpose system.

As an example if I have a microwave oven the embedded computer in the microwave oven is designed only to cook food. I cannot reprogram it and expect it to wash clothes that is a kind of glaring difference that I have and between an embedded computer and a general purpose computer.

The second difference is that embedded computers can be implemented using a wide variety of processors. As you saw a processor microcontroller smaller than a grain of rice and on the other extreme big microcontroller more than an inch square, but when you look at general

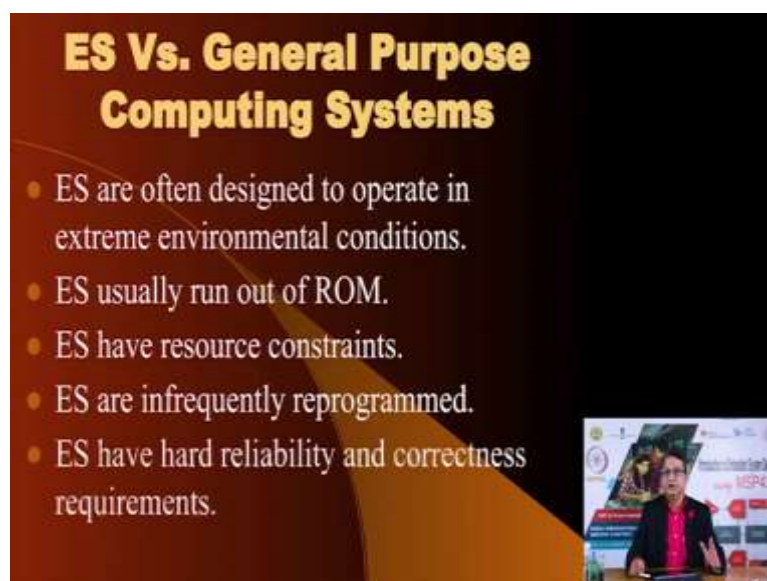
purpose computing devices usually you use the highest performance processor that is available. The third difference is that embedded systems are cost sensitive.

What is the meaning? Because you are buying a product you are not buying a computer. The embedded computer that form part of that embedded system gadget carry the small fraction of cost and therefore embedded computers, embedded systems are very cost sensitive and the fourth difference is that embedded systems are supposed to offer and work under real time constraints.

The meaning is that whenever there is a input change, environmental change, the embedded systems are supposed to react to that change and provide an output in a timely fashion. If there is delay in producing the output it may lead to fatality, it may lead to loss of life. As an example if the embedded computer which is flying an aircraft and the pilot wants the aircraft to climb and the embedded computer does not perform appropriately and keeps on flying at the same level it may lead to a crash.

Now you do not expect such a behavior out of a general purpose computing systems often time you may have noticed that you press a key and your laptop or desktop does not respond it is all right because it is not going to kill anybody and so there is vast difference between the expectation of an embedded computer in terms of requiring real time response.

(Refer Slide Time: 03:10)



ES Vs. General Purpose Computing Systems

- ES are often designed to operate in extreme environmental conditions.
- ES usually run out of ROM.
- ES have resource constraints.
- ES are infrequently reprogrammed.
- ES have hard reliability and correctness requirements.

The slide features a dark background with a light-colored diagonal shape. A small video inset in the bottom right corner shows a person speaking at a desk with a laptop and a screen displaying technical diagrams.

The fourth difference or fifth difference is that embedded systems are designed to operate in an extreme environmental conditions. Take a example of missile system that missile could be

launched from glacier environment or it could be launch from deserts and they must perform. Usually embedded systems run out of a ROM.

Meaning that the program that is stored in the embedded computer is stored in the ROM that is read only memory a permanent memory. But often times not often times most of the times the program that runs on your desktop or laptop computer is running in the RAM. Yes a desktop or a laptop system also has a ROM which is used at the beginning when you power on your device and that program is called boot up program call BIOS Basic Input Output System.

Once that completes execution the transfer the control is transferred to the operating system on the hard disk and then to the program that you would like to run. These programs are loaded in the RAM of the general purpose computing system so that is a vast difference between how an embedded system operates and how a general purpose computing system operates. Another difference is that embedded system has resource constraints. What are resources of a computer?

The computational capability is a resource the way the embedded system or the computer communicates with the outside world is resource and you see when you have a general purpose computing system such as your laptop or desktop it has many ways of communicating with the outside world. It has Ethernet, it has WiFi, it has FireWire, it has USB and so on.

But does your microwave have Ethernet connection can you control it with your phone through Bluetooth most probably not. Although, in recent times there are certain examples of such embedded applications with connectivity, but that connectivity is based on the requirement not because you want to flaunt that your computer has so many ways of communicating with the outside world.

So embedded systems have resource constraints, there are examples of for example a washing machine which is called IOT washing machine that washing machine can communicate with the outside world through internet and the purpose is not that you would want to control this washing machine sitting in your office.

But when the washing machine is operating at home and encounters loss in performance it can communicate that to the service provider the maintenance guy so that they can come and

provide preventive maintenance before the washing machine actually stops working. So this is an example where you have a communication protocol, communication link on your embedded device but it is out of necessity as a feature not because you have you want to say that your washing machine has so many ways of communicating with the outside world.

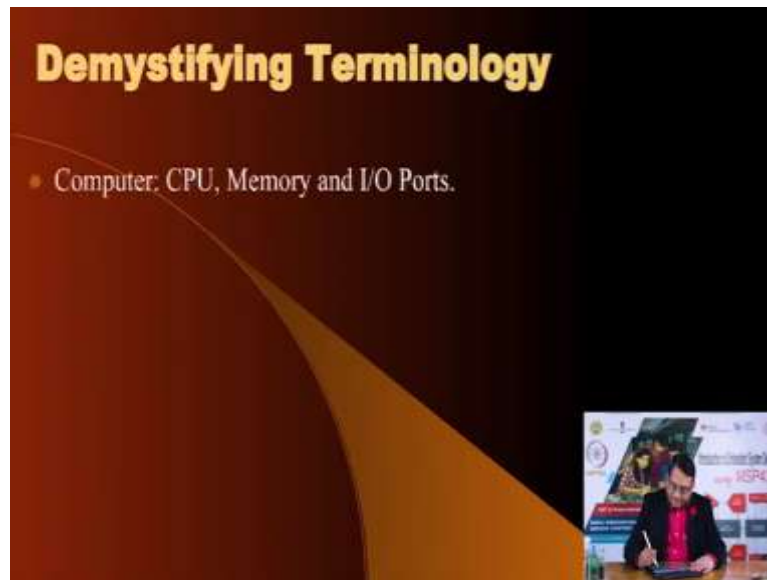
Embedded systems are infrequently reprogrammed. What is the meaning? That once you load a program in an embedded applications you do not ever change it, you do not change it most of the times and in most cases you do not ever change it, but in a general purpose computing device every time you run an applications a new application you are reprogramming that device.

How many times have you encountered a situation that somebody has come to your home saying that they would like to upgrade the firmware of your washing machine I have never encountered it, it does not happen and so that is the meaning of this point that embedded systems are infrequently reprogrammed. Embedded systems have hard reliability and correctness constraints.

All computers must operate correctly, but embedded systems have an additional responsibility that they must continue to provide service reliably not all most of them, many of them, some of them. Why because failure to do so may lead to loss of life. As an example medical devices like an x-ray machine you do not want the computer in the x-ray machine to fail and it may give needlessly high dose of x-ray to the patient under the x-ray machine.

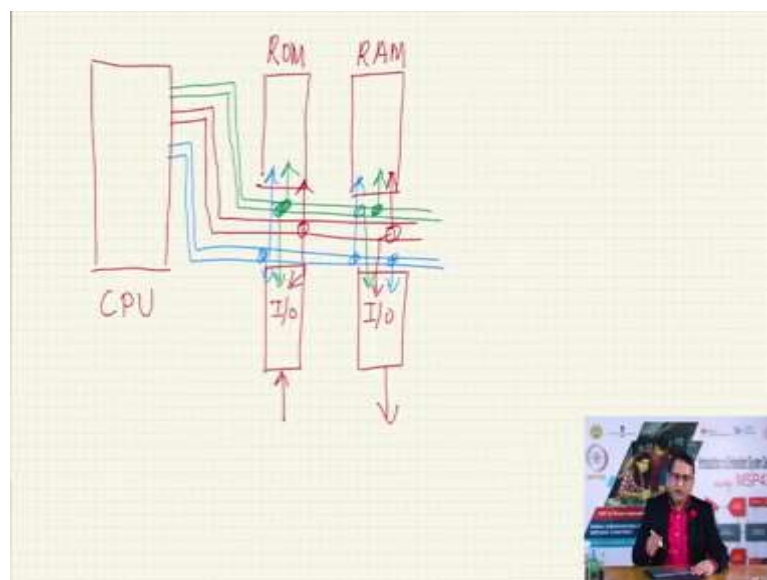
And in an aircraft you do not want the computer flying the aircraft to fail leading to loss of life and so embedded computers are expected some of the embedded computers are expected to have hard reliability requirements.

(Refer Slide Time: 07:50)



Now that we have seen major differences between embedded computers and general purpose computing devices it is time to go through certain terminologies that we use. And for many of you or some of you it may be a repeat of things that you already know, but it does not heard. So the first terminology is what is a computer? A computer is nothing, but a system which has a CPU that is a central processing unit, memory and input output ports. Let me draw a diagram to illustrate it.

(Refer Slide Time: 08:27)

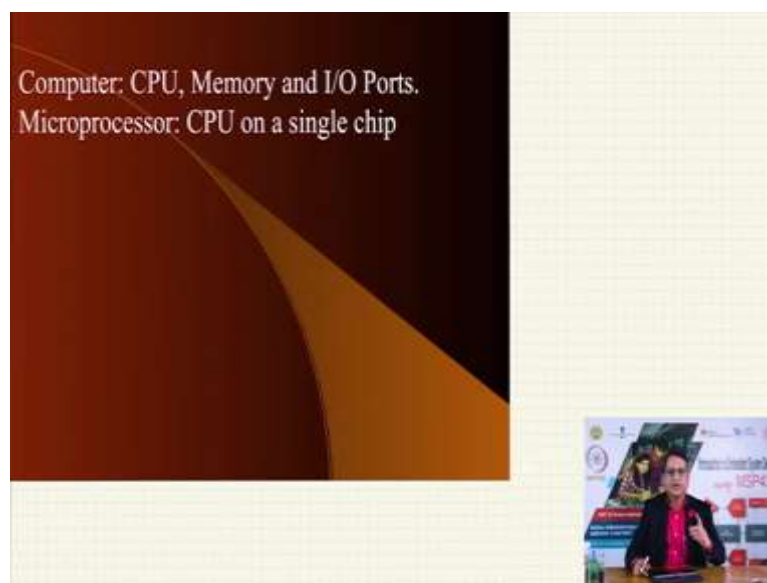


So this is your CPU, this is your memory and usually you would require two types of memories. One is the ROM which stores your program the other is the RAM which we use

for variables and then you would have certain IO devices maybe an input device and an output device let me erase this and write it here and how does the CPU communicate with these devices through buses.

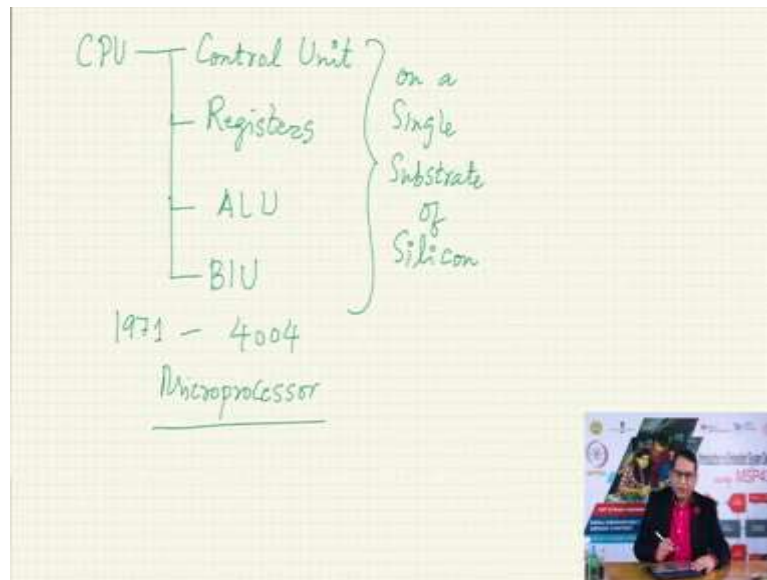
These buses are for data and for address and another set of bus is what we call as the control bus which tells whether you are reading information or writing information and all these. So this is the diagram for a computer here is the CPU, ROM, RAM that is memory and input output ports.

(Refer Slide Time: 10:18)



The second terminology is a microprocessor. A microprocessor is nothing but a CPU on a single chip. How did it happen? In the 50s and 60s the CPU was made with discrete components or discrete integrated circuits in the 60s. In 1971, a start up at that time that was called Intel. And today Intel is a giant made an interesting device what it did was it integrated the functionality of a CPU on the single substrate of silicon. What does a CPU entail?

(Refer Slide Time: 10:56)



A CPU has 4 components it has, so a CPU has control unit. It has registers in which temporary information is stored. It has a ALU through which it can perform mathematical and logical operations and it has a bus interface unit through which the CPU can communicate with the outside world to the memory and to the IO ports.

Now all these were originally in the 50s and 60s implemented using discrete components, discrete integrated circuits and what Intel did in 1971 is that it integrated all these on a single substrate of silicon and made a IC out of it and that IC will named 4004 microprocessor and the term microprocessor came about so that is the definition of microprocessor then what happened?

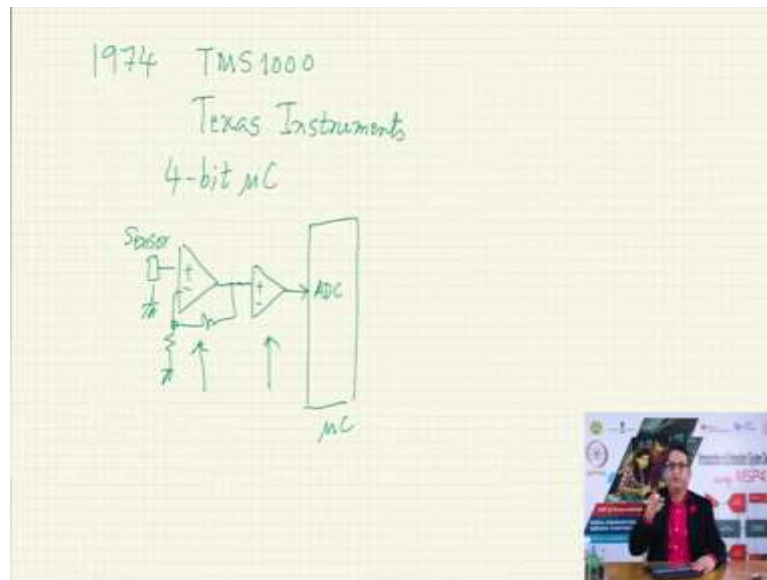
(Refer Slide Time: 12:25)



When people started using a microprocessor to create computers it drastically reduced the size of the computers. In the earlier days the computers would probably fill half a room, but now suddenly you could fit the entire functionality of a computer on a single printed circuit board and such computers were called microcomputers. So a microcomputer is nothing, but a microprocessor plus memory plus IO devices on a single printed circuit board.

In the same scalability fashion the entire functionality of the microcontroller people thought could be integrated on a single piece of silicon and what happened? It led to a microcontroller. So a microcontroller is nothing but a microcomputer on a single chip. Who did that?

(Refer Slide Time: 13:20)



It turns out incidentally that the world's first microcontroller was in 1974 it was TMS 1000. And this was by a company which is incidentally also supporting our program this course here Texas Instrument. This was a 4 bit microcontroller I am going to use this terminology Mu C to indicate that it is a microcontroller.

Now obviously you may have heard of Moore's Law that says that it is not really a law it is an observation that every 18 months that is one and half years the number of components, number of transistors that you could integrate on a piece of silicon would double.

Now what would you do with these increasing number of transistors ofcourse you could fit more and more functionality on a single piece of silicon and so this is the reason why you add microprocessor and from the microprocessor you had a microcontroller, what next? Now when you use a microcontroller in any control or embedded application.

Apart from the microcontroller here is your microcontroller. You still need analog components for example I may have a certain sensor and the output of this sensor may need to be amplified with the help of an operational amplifier and so I need to provide a certain amount of gain, so I am going to put components like this. This is going to, so this is my sensor here, the sensor the output voltage of the sensor is going to be amplified with this.

Maybe with another op-amp circuit I am going to amplify I am going to filter this signal in some way and eventually feed it to the microcontroller maybe to the ADC input, will come to more details about this ADC input, but you see if I have to design a system out of a

microcontroller apart from the microcontroller which has already integrated so many components on a single integrated circuit I still need these additional components.

These are analogue devices analogue components and it is very difficult to make analog device and program them meaning if I have an amplifier how do I write a program to change the gain of the amplifier, how do I write a program to change the cut-off frequency of a filter or the slope of that filter or the order of that filter it is very difficult.

(Refer Slide Time: 16:07)

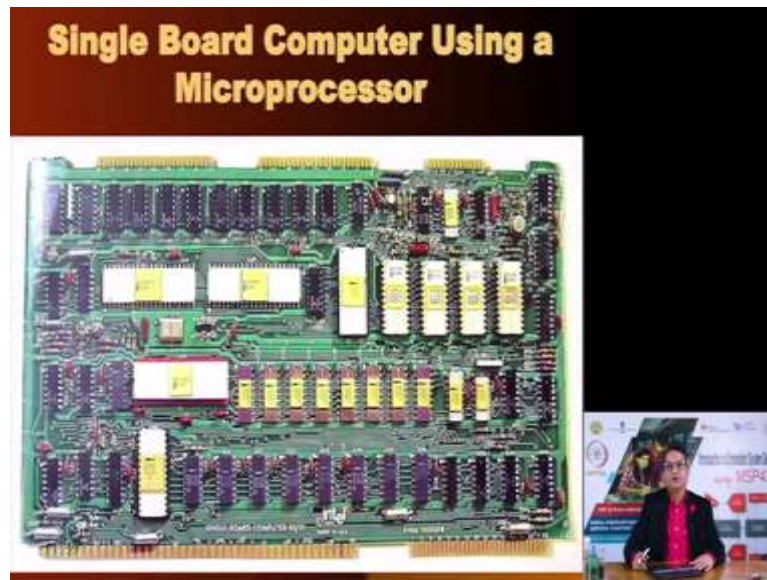


But interestingly that also happened and that led to a definition called system on chip. A system on chip in this context is a microcontroller plus analog components which are programmable and this was by company called Cypress semiconductors. What they did was they added microcontroller plus analog building blocks and you could change the functionality of these analog building blocks by merely writing a program.

So you could decide that today I want the analog building block to function like an amplifier with a certain gain or change the topology of the amplifier to be inverting or a non-inverting or a differential amplifier or you want that analog building block to be of, to provide functionality of a filter in which case what is the cut-off frequency, what is the topology of that filter do we want it to be of the low-pass, band-pass, high-pass, band reject, notch and so on and so this provides great functionality.

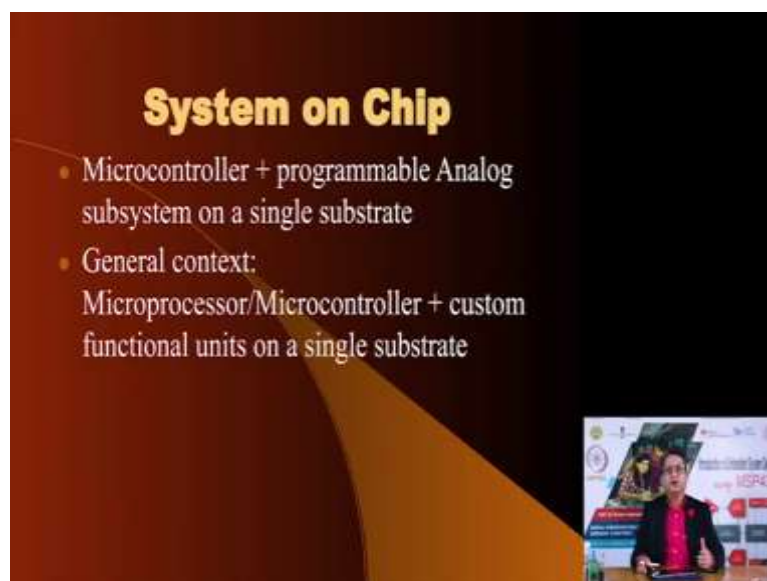
And so having a system on chip which integrates not only the microcontroller, but also analog building blocks will reduce the size of the eventual system to a single integrated circuit and that is a great improvement and we will see how.

(Refer Slide Time: 17:32)



Here is an example of a single board computer using a microprocessor you see a single printed circuit board has been used. This is just to show you how from the earlier times when the entire computer would be a big device has been shrunk to the size of a single PCB.

(Refer Slide Time: 17:50)

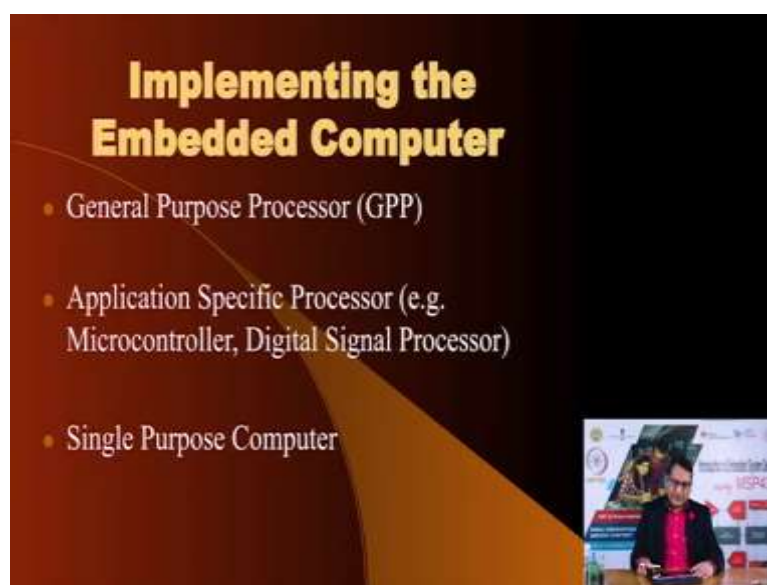


Anyway now in the context of system on chip I want to clarify that there are certain interpretations of this system on chip. One is that system on chip is nothing in our context of

embedded systems that system on chip is a microcontroller plus programmable analog components on a single substrate of silicon.

But in general when people refer to this term system on chip they mean probably a microprocessor or a microcontroller together with some custom functional unit or units on a single piece of silicon and that they mean the system on-chip and so it is very important to know the context is the person talking about an embedded application where it might mean a microcontroller and analog components or in a general context.

(Refer Slide Time: 18:42)



Now we come to this point we have mentioned this earlier that a microcontroller is a popular method of implementing the embedded computer, but what are the other methods we must be aware of them. So first method by which we can implement an embedded computer is through a general purpose processor meaning a microprocessor for example. Another method is to use what is called as application specific processor.

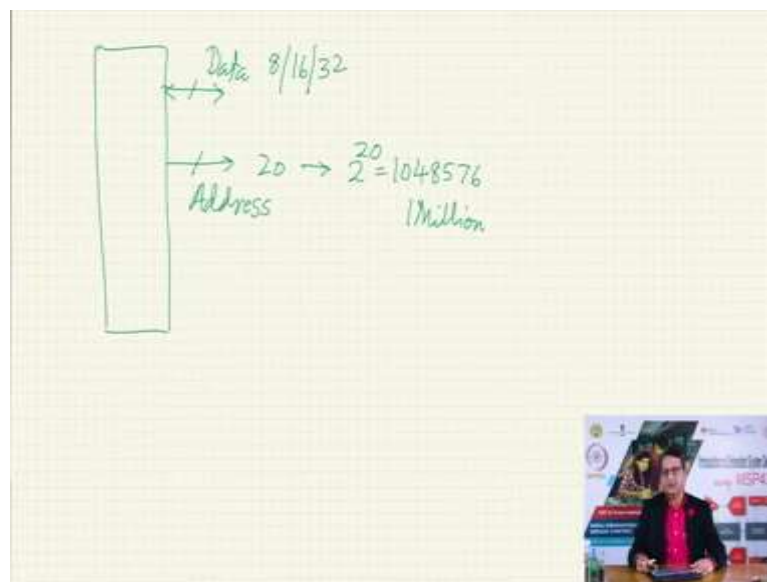
What are the examples of application specific processors? Well, a microcontroller is an example of application specific processor also a digital signal processor which is a specialized processor which can perform digital signal processing more efficiently than it is possible with a general purpose processor and the third method is what we call as single purpose computer.

I would like to elaborate on these 3 approaches. In a general purpose processor basically you are saying I am going to use a microprocessor, but a microprocessor is only one part of the

entire ecosystem of a computer. Apart from the microprocessor what you need is you need external memory of the type of ROM or of the type of RAM and then you are going to need ports.

And so it would lead to is that it would not remain a very efficient from a size point of view. It will not be a very efficient implementation because a microprocessor offer certain amount of memory that is if for example your let us write this down.

(Refer Slide Time: 20:08)



If my microprocessor has certain amount of data lines let us say this is data and common example are 8 or 16 bits or 32 bits. It also has this data is by the way bidirectional it would have address lines let us say it has 20 address lines.

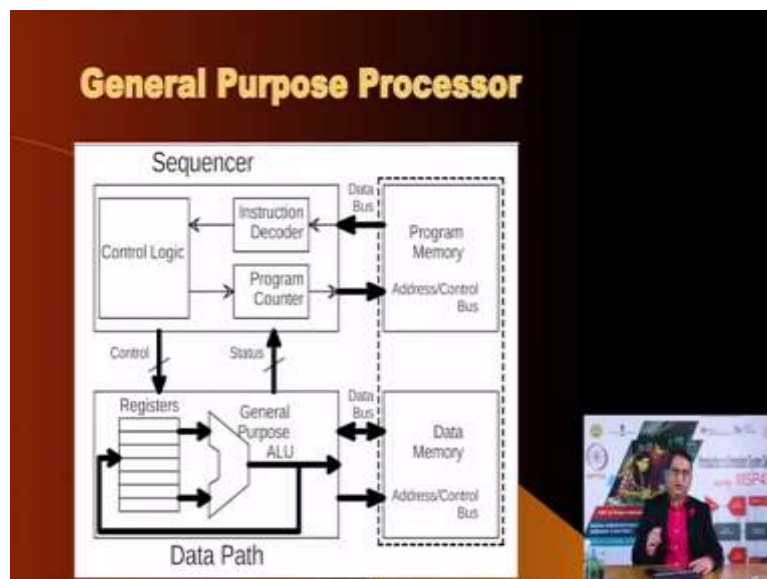
If it has 20 address lines that means it can access how many memory locations? 2 raise to the power 20 which is equal to 1048576 memory locations that is 1 million. Now it can communicate or it can talk to 1 million memory locations it is up to you to provide the requisite amount of memory maybe you realize or you anticipate that your application is going to need 16 kilobytes of ROM another 16 kilobytes of ROM.

So you have to add appropriate memory devices to fulfil the requirement. So this would lead to a physically large system as far as implementation is concerned. On the other hand when you choose to implement that application using an application specific processor such as a microcontroller, a microcontroller is going to have a microprocessor that is a CPU on that silicon substrate.

Apart from the CPU it is also going to have RAM and ROM in various types of a fixed amount maybe it has 8 kilobytes of ROM and 8 kilobytes of RAM. If you find that this amount of memory is suitable for a particular application it may make a better sense to use an application specific processor such as this microcontroller to implement that system because it would lead to reduction in size and reduction in cost.

It would improve reliability because there are less number of components to solder, so it would be a more reliable system. And so often times you would implement your application with a microcontroller as an example of application specific processor, but that is not all. You can also implement the same system with a single purpose computer and I am going to illustrate these 3 approaches using certain block diagrams.

(Refer Slide Time: 22:25)



Now when I implement an application using a general purpose processor here I have drawn a block diagram. This is an alternative view to view to understand what is the meaning of a microprocessor? Now one way to understand microprocessor which we have seen is that a microprocessor is nothing but a ALU, registers, control logic and bus interface unit on a single piece of silicon that is to indicate these 4 components as functional units.

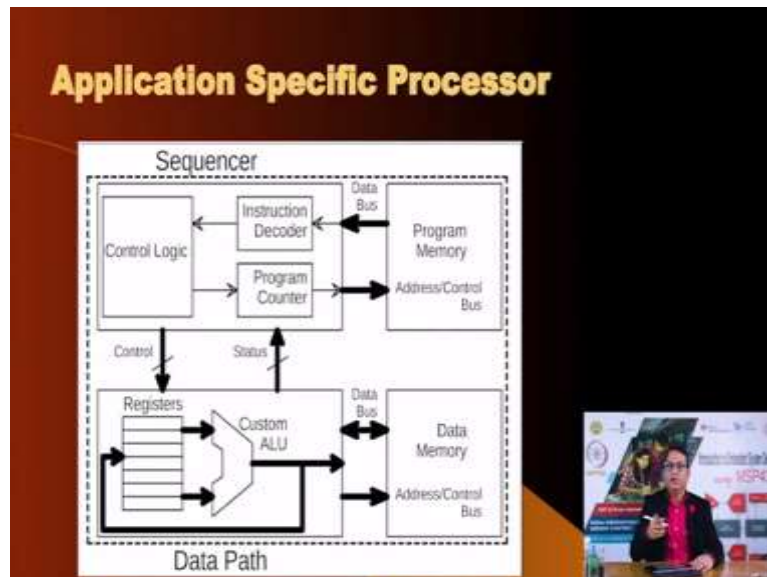
There is an alternative view to understand how a computer works and which is through this diagram that we have two blocks one is called the sequencer and the other one is called the data path. The sequence are consist of the control logic it has instruction decoder and a program router and the sequencer talks to an external memory device such as the ROM or EPROM or other types of memory devices.

And the reason why you can implement any different type of application is by changing the contents of the program. The sequencer understands the instructions and based on the sequence of these instructions which you as a user have to write it can implement a different application. It does that by providing inputs to the second part of the building block which is a data path and data paths consist of the registers for storing temporary values and the registers feed the arithmetic logic unit.

So, essentially the sequencer and data path form the CPU. This CPU could be in the form of individual ICs or it could be in the form of a microprocessor and that is the alternative view to look at a general purpose processor. You are going to need external program memory and data memory. Program memory feeds the sequencer, data memory is connected to the data path so that you can store variables.

So while this is the log diagrammatic approach of alternative approach of looking at the general purpose processor let us see if we were to implement an application using a general purpose processor what would it look like.

(Refer Slide Time: 24:37)



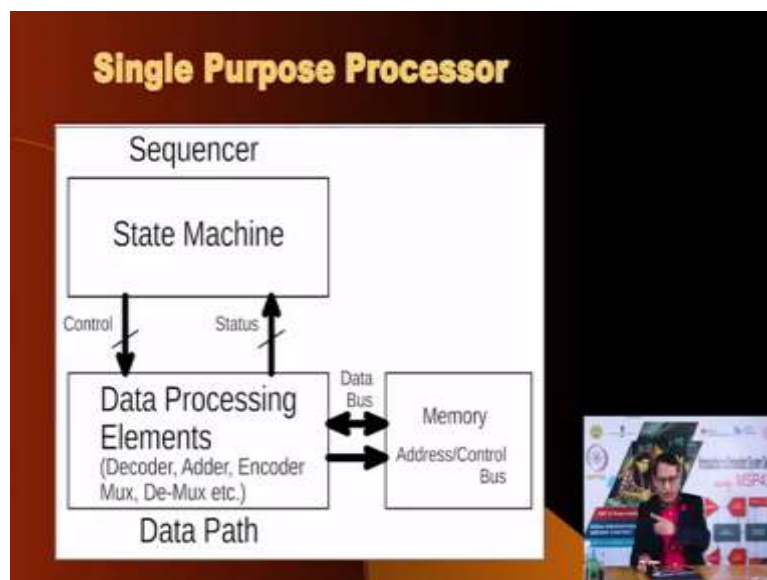
And for example we have a anyway before we come to that example let me show you the block diagram for a application specific processor also. This application specific processor also has a sequencer which consists of the control logic, it has the instruction decoder and the program counter which is connected to the program memory.

The program memory is populated is filled up with instructions by you as the designer. The sequencer reads the contents of program memory and executes whatever that program memory instruction expects you to do.

And then controls the operation through the data path which consists of registers and ALU. Now in the previous case the ALU was a general purpose ALU, but for a application specific processor it maybe a custom ALU meaning you would want to do certain operations more often. Therefore you provide that arithmetic logic unit with that additional functionality like in the DSP operation requires repeated multiplication and addition of numbers.

And so in a DSP you have a ALU which has a feature called MAC Multiply and Accumulate. Apart from that the program memory and data memory could be integrated on the same chip. If you see this block diagram I have drawn a dotted line around the components to indicate that these 4 block diagrams are actually could be integrated on a single piece of substrate to give you a more efficient implementation.

(Refer Slide Time: 26:11)



Now let us look at the block diagram of a single purpose processor which consists of a sequencer which is implemented as a state machine. State machine is another word for finite state machine or a straight machine. It is usually implemented with memory building blocks such as a flip-flop that sequencer communicates with data processing element.

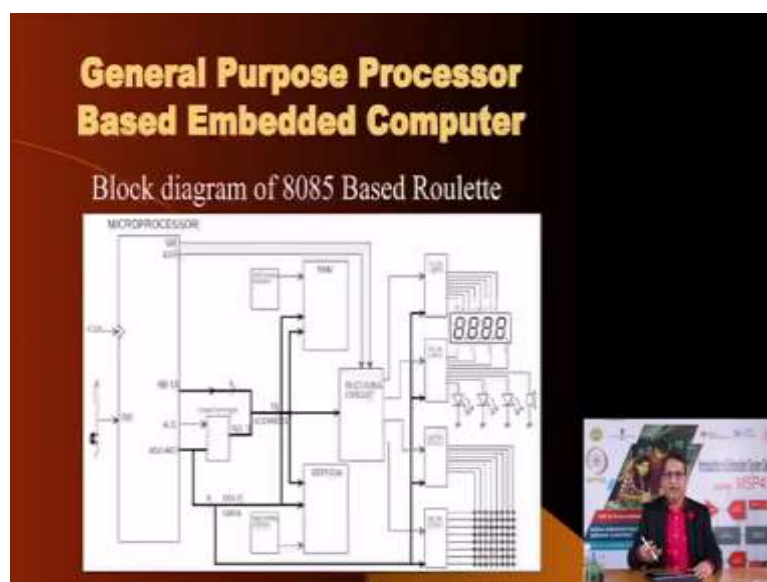
Now in contrast to the data processing elements that is the data path in a general purpose processor or application specific processor where you have registers and ALU to perform

those operations nothing is fixed for the data processing elements in a data path of a single purpose processor why because the purpose is fixed and it is only one purpose that you want to design it for.

You put appropriate data processing elements which could be decoder, adder, it could be an encoder, multiplexer, de-multiplexer and so on and so forth. And ofcourse the data path would want to store some information so it has memory usually in the form of a RAM and so when you design a system which is expected to perform only one task and you specifically designed this hardware just to do this and nothing else this would be called a single purpose processor and any application could be implemented in any of these 3 ways.

And it would depend on you as a designer of embed systems to decide whether you want to go through a general purpose processor route or you want to use microcontroller which is an example of an application specific processor or do you want to go single purpose computer. Now I have an example where particular application was implemented using these 3 approaches. The first one was implementation of a particular project which is a Roulette wheel game using 8085 microprocessor.

(Refer Slide Time: 28:05)

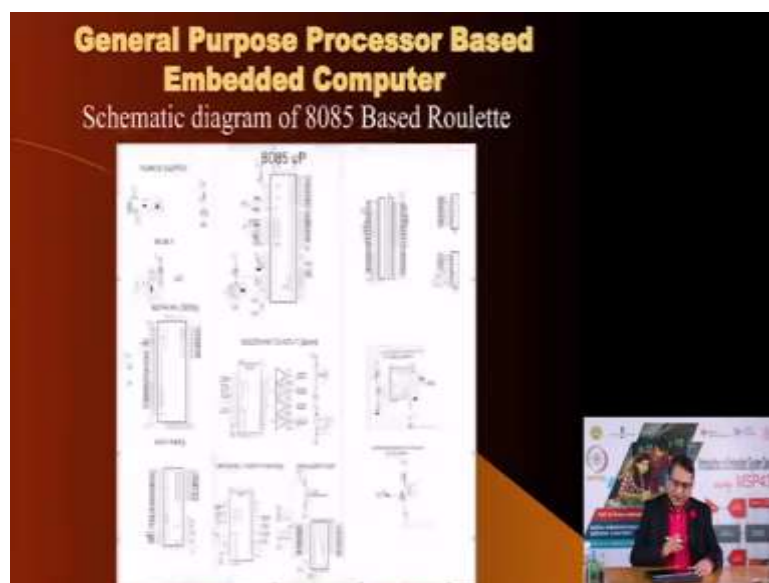


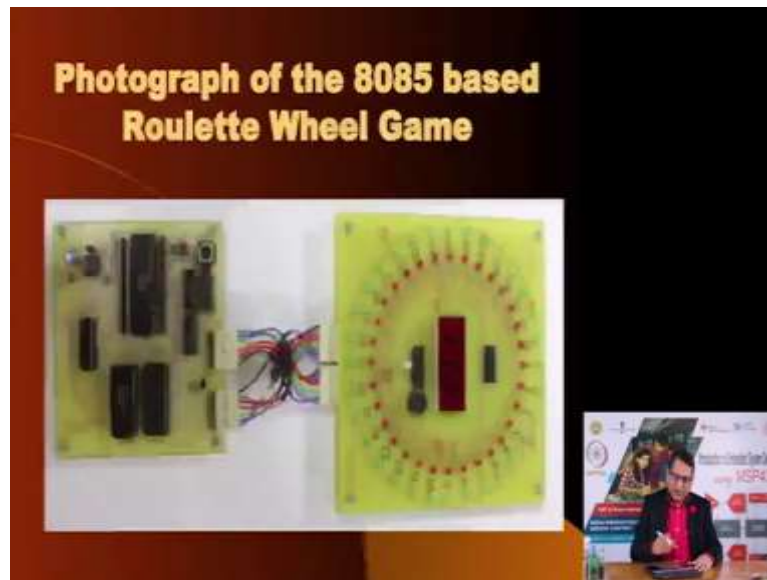
This is the block diagram if you see this it has so many building blocks one block let us go from the left you have the microprocessor which is 8085 in this case. Microprocessor requires RAM and ROM which are indicated on the middle blocks these have to be decoded the address of these devices have to be decoded.

So there is a decoding circuitry and then you need certain output ports to indicate the state of the game to the user and so for that we have 4 ports which is connected to 4, 7-segments displays, few discrete LEDs, a buzzer and a ring of LEDs, 2 rings of LEDs to indicate the game that you are playing that you want to place your bet and then you want to roll to let the Roulette wheel roll and then it will stop the speed of rolling will slow down.

And it will stop somewhere if that matches with the bet that you have placed you win otherwise you lose and this was implemented using 8085.

(Refer Slide Time: 29:02)

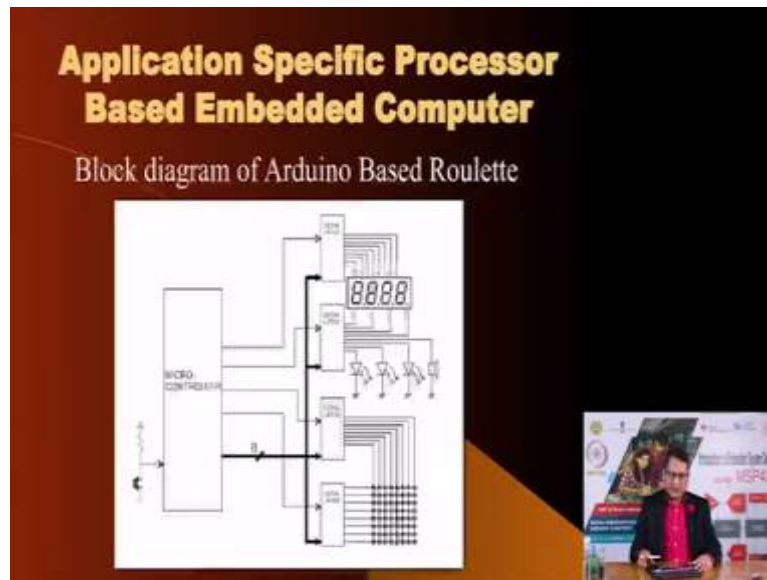




And here is a schematic diagram as you see it uses so many integrated circuits I also have a picture of the actual implementation on the left you have a PCB where the middle top IC is 8085 microprocessor at the bottom you have ROM and RAM.

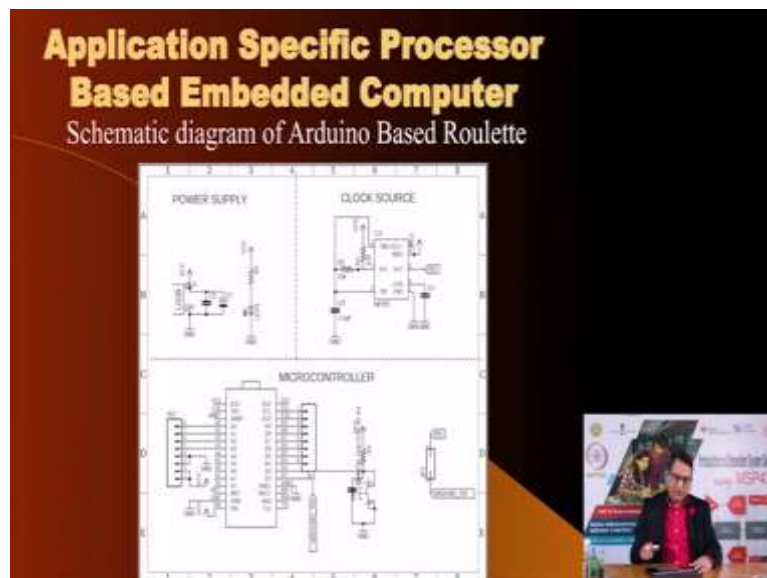
The ROM is in the form of a E-square PROM and you have these latches on the left and right and on your right is the actual input output board where you see two rings of LEDs, the red and the green and at the centre you have the 7-segment displays and obviously as you see it uses far too many ICs. My student implemented the same design, same application using a microcontroller.

(Refer Slide Time: 29:45)



This is the block diagram of a Arduino which is a microcontroller. Basically Arduino uses AVR microcontroller this is an application using Arduino and you can see these block diagram you see suddenly the number of blocks have reduced why because the microcontroller has integrated the microprocessor and memory and ports into a single chip therefore the size reduces.

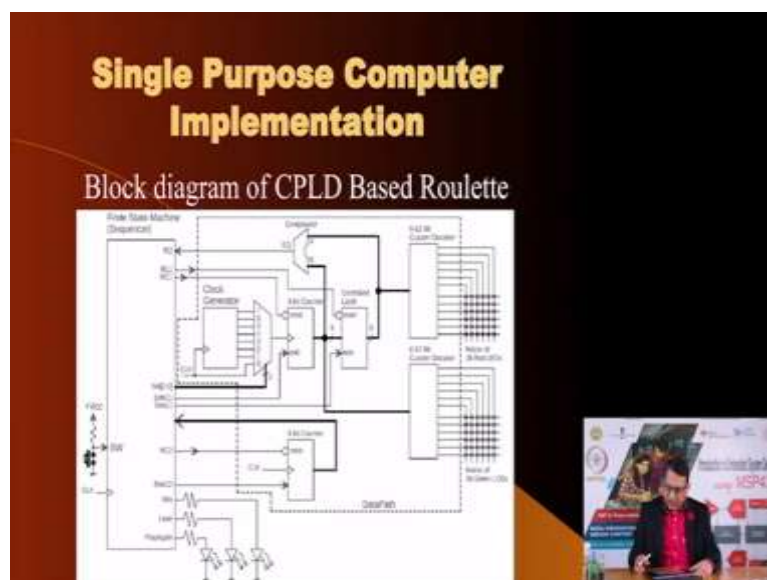
(Refer Slide Time: 30:08)





Here is the schematic diagram of the microcontroller based Roulette wheel as you see the components have reduced. And here is the photograph as you see you have very small PCB on your left which is just one microcontroller a switch and so on and on your right you still have the LED rings, red and green and the 7-segment displays. This is the second implementation of the Roulette wheel gain.

(Refer Slide Time: 30:39)

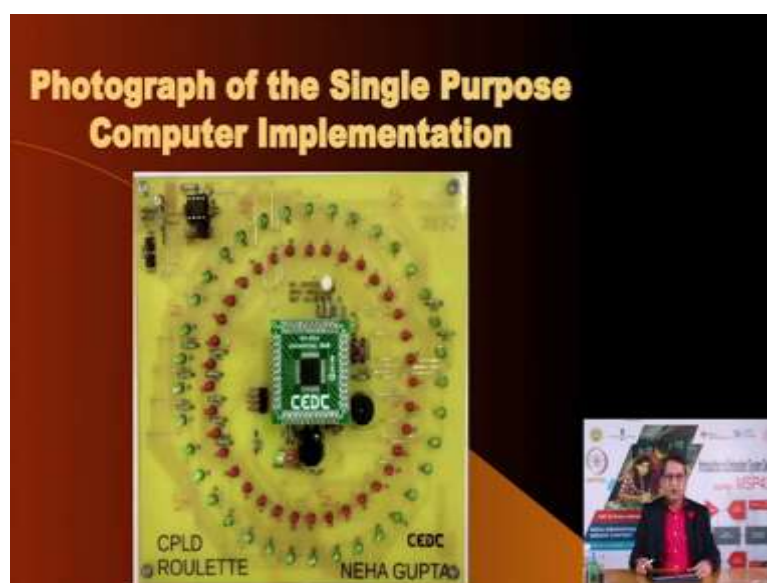
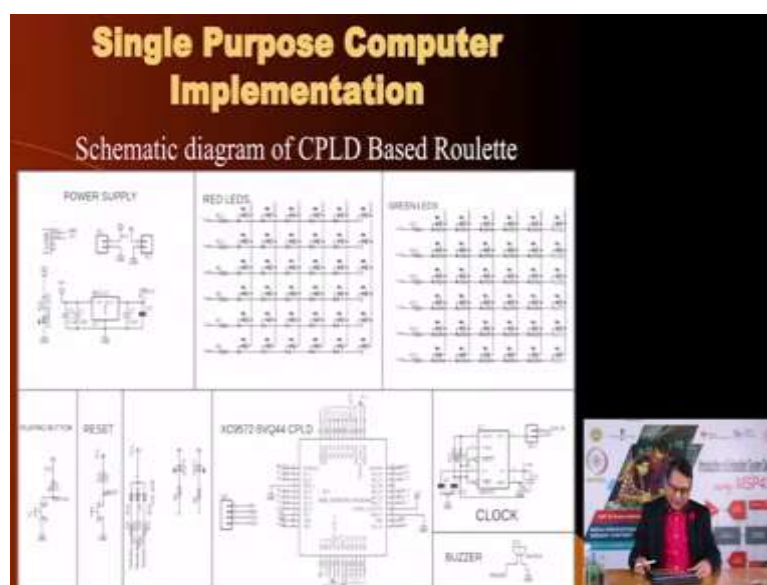


And then we also have the implementation using this single purpose computer implementation using a CPLD, CPLD stands for Complex Programmable Logic Device. This is a device that you can program to implement a custom circuit and this is the block diagram. You see this block diagram is split in two parts which is exactly what we had discussed earlier that it consists of a sequencer which is the finite state machine in this case. The finite

state machine controls the data path where the actual operations the processing of information, processing of data takes place.

In this case several building blocks of processing that data are being used for example this uses a counter it also has a comparator for comparing the numbers that are being generated and on the right most side you have decoders for controlling 36 red LEDs and another decoder for controlling 36 green LEDs which you form the inner and outer ring of the Roulette wheel.

(Refer Slide Time: 31:41)



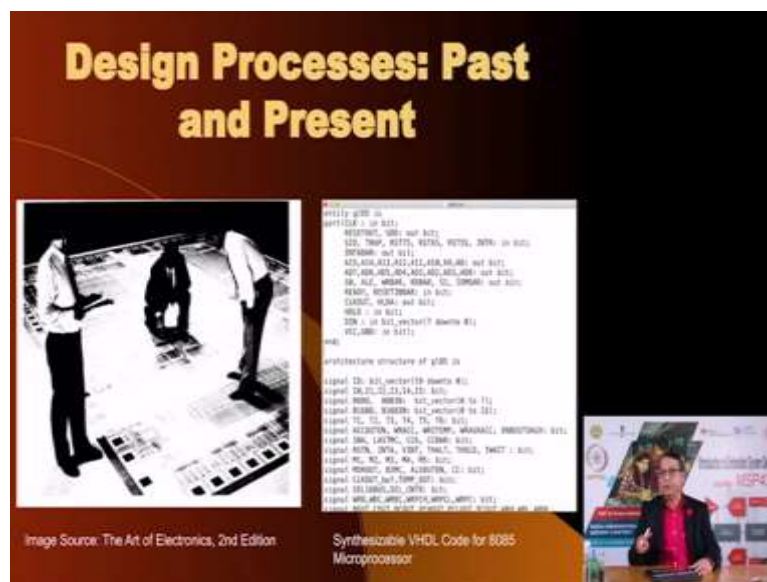
This is the schematic diagram as you see it has a CLPD and all these LEDs and all that and you see there are no almost no other ICs being used other than a clock generator based on a

555 timer IC and here you can see the actual implementation a single printed circuit board pretty much which has the LEDs on the outer periphery and at the centre you have the CPLD IC.

So you see it is possible to implement any embedded application in 3 ways: the general purpose processor approach which usually involves lot many components. There may be some good reason for you to use that approach, but from the point of view of cost and reliability and economics maybe you would want to implement the same application using embedded computer.

Using a microcontroller and the third approach ofcourse is the single purpose computer and there are many-many applications which are done using this approach.

(Refer Slide Time: 32:38)

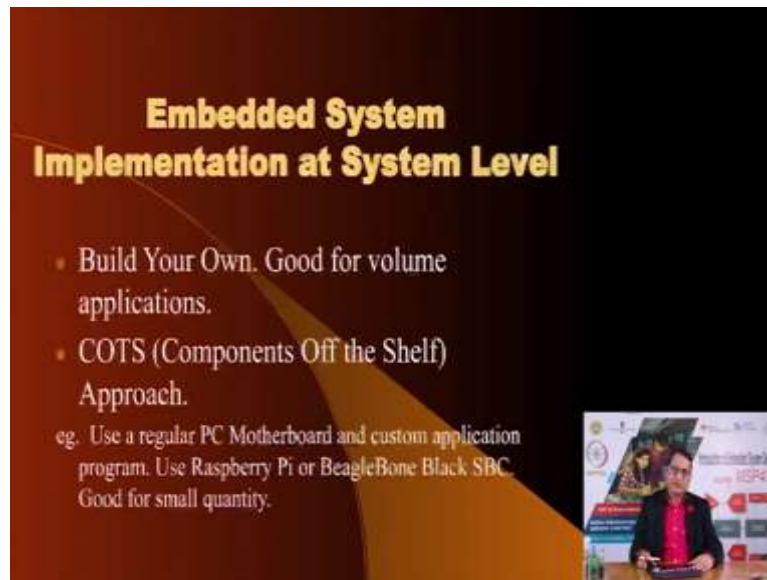


I want to show you how things used to be designed earlier in 60s and 70s integrated circuits. On your left is a picture of design engineer inspecting the layout of an IC. You see the IC itself is very small probably a millimetre square or couple of millimetres or thereabouts.

But you cannot inspect the design on such a small piece of substrate and so you blow it up by several magnitudes and then it occupies almost big room and you see these design engineers are inspecting the design and today you do not have to do that at all because you are using CAD tools and languages which are created for implementing and designing hardware and so on the right you have the synthesizable code to implement a 8085 microprocessor.

Ofcourse this is just an example and today you would not implement 8085 microprocessor. You would implement whatever fancyes you whatever is in your mind and you can use language such as VHDL or Verilog to implement that circuit.

(Refer Slide Time: 33:52)



Till this time we have discussed implementation of the embedded system from a processor that is the embedded computer point of view. But that is not the complete story because we would like to build a complete system. So at a system level what are the various approaches it turns out that there are two popular approaches. One is that you build everything from scratch on your own meaning you decide what embedded computer you are going to use for that what kind of circuitry is required, what kind of additional components are required.

You would need a printed circuit port for that, you design it you get it fabricated and then you integrate all the components on that PCB and usually you take this approach when you are expecting large volumes because you would benefit from economies of scale, but when your applications are not when your requirements are not that large volumes are not that large there is a alternative method and I call it COTS approach that is Components Off the Shelf.

Meaning you buy things that are available and you quickly integrate it and implement your design and these days you could implement this COTS approach using popular single board computers which are very powerful computer today such as Raspberry Pi or Beagle Bone Black single board computer. In the earlier times or even today if there is such a need you would use a regular PC motherboard to implement what you want.

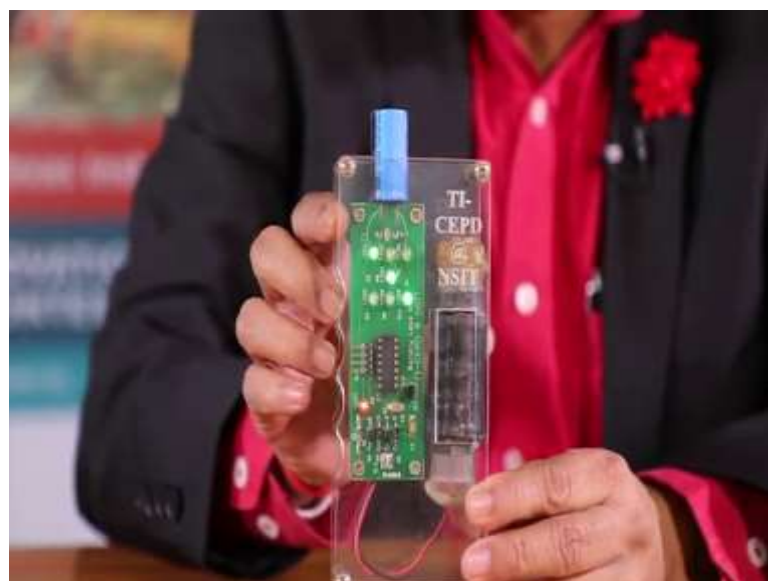
So, at this point we are at a stage where we have seen all these things you can do with embedded systems, how you implement the processor, how you implement the system.

(Refer Slide Time: 35:29)



And I would like to now give you a small demonstration of MSP-430 based projects that have been designed in my lab. So that you the student, you the participant can feel confident that if my students can do it in my lab you sitting wherever you are sitting can also design such systems. So let us have a look at some of these projects.

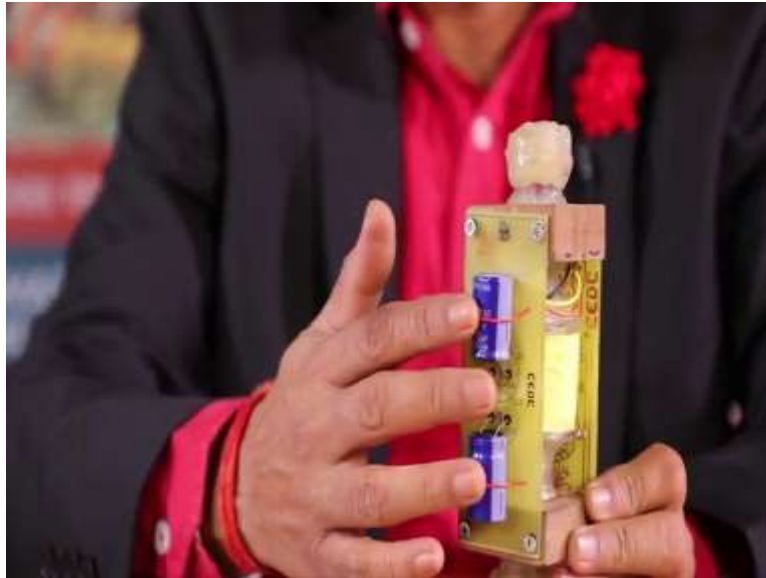
(Refer Slide Time: 35:58)



For that let me start with simple project first this is a project which is a battery-less electronic dice, but it has only one display often times you play a game where you need 2 dice.

(Refer Slide Time: 36:09)

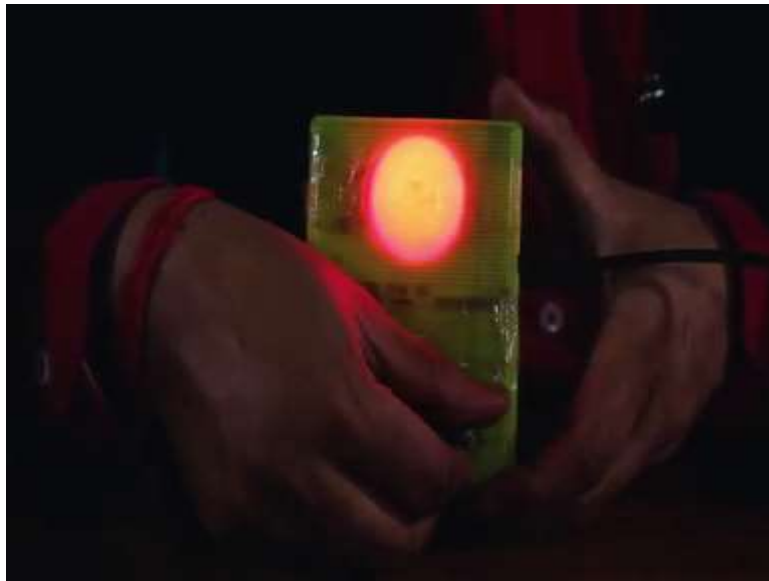




And so here is another implementation this has two displays and use it again this is battery-less so I will shake it and when I stop you see 2 numbers, 2 random numbers are being printed here. Again I shake it when I stop it produces two random numbers. What does it use? It uses MSP430 microcontroller in the middle you have the tube which produces which converts mechanical energy into electrical energy for the circuit to operate and on this side you have capacitor which store the voltage that is produced so that this circuit can operate and you see this entire system has been fabricated.

In my lab these PCB have been fabricated locally and the entire system has been integrated here, this is the second project.

(Refer Slide Time: 37:03)



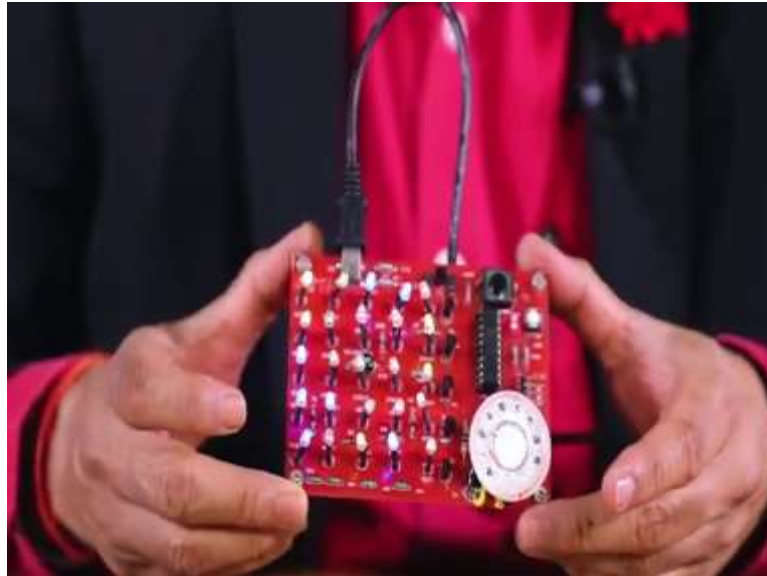


Continuing with the LED projects here we have another system we call it the colour mixer because it is going to mix 3 primary colours green, blue and red and create a custom colour. I have the power supply for this comes with the help of in the form of this power bank I am going to connect it to show to you.

Now this is how it works I am going to reduce the colours so this is green 256 levels of green, blue colour again 256 levels here again 256 levels of red and I can add two certain levels of red I can add certain amount of blue you see I get various shades of magenta or instead of blue I can add green and I get various shades of yellow or into this now I can add blue and I get various shades of white and so on.

And this user MSP-430 microcontroller it controls these 3 LEDs using a concept called pulse width modulation that we are going to cover and because each primary colour has 256 levels the total number of colours that can be made out of this colour mixer are 256 cube which is 16 million colour this is the number of colours that can be made with this.

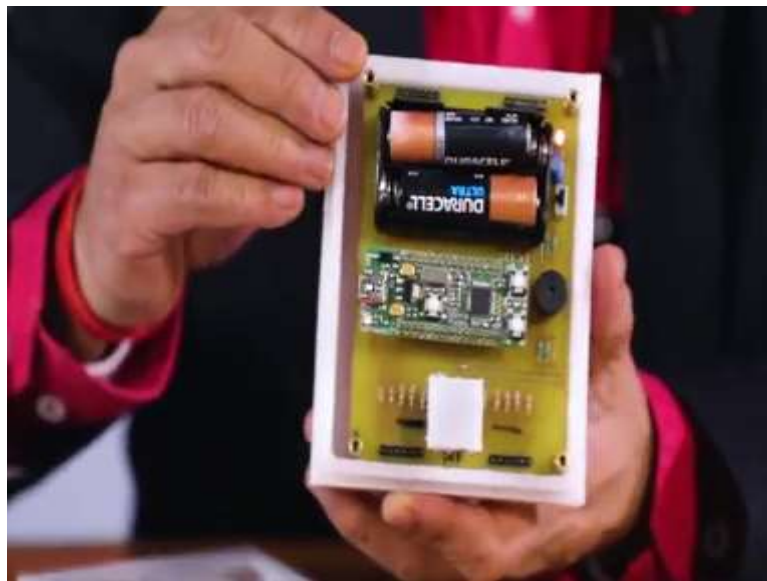
(Refer Slide Time: 38:39)



Then I have this project which we call the electronic birthday blow out candles. These are supposed to be birthday candle so that they function without using real candles, real candles with small kids can lead to a fire hazard and here so we have done away with real candles and instead used LEDs to mimic the function of a candle. Now, if these are candles or if these are mimicking the operation of a candle then I should be able to blow them off like I can blow off normal candles.

And that is the magic here we have used a sensor thermistor which will detect air blow when I blow on it. The microcontroller here is going to sense the air blow on the thermistor and will randomly turn these LEDs off. So let me try and as you see it is also making a sound it is playing happy birthday sound. So this works like a replacement to traditional birthday candles is perfectly safe to use any number of times you or your kid would want to.

(Refer Slide Time: 40:06)



Let me show you another project now and this is electronic sandglass. So instead of sand falling from one part of the sand glass on to the other side here we use LEDs as you see as I am holding it this is the LED indicate as if sand is falling. Now if I turn this around this sand should come to the neck and you see now it is falling in the other direction and normal sand glass when it empty the sand it does not say anything.

But because this uses a embedded computer it will tell you that the time has elapsed. Now if I turn it around again it is falling on the other side and this has been program to empty the so called sand in about a minute of time. These has also uses another MSP430 microcontroller and the sensor for the orientation is also integrated in this project and in fact it was conceived and designed in the lab.

And this project actually I will show you this is the enclosure which was made in the lab using a CNC machine this is the embedded computer here, this is the sensor part here and it is being powered with 2 AA sized alkaline batteries.

(Refer Slide Time: 41:21)



Let me show you another project and in this case this is a LED Kaleidoscope, normal Kaleidoscope has glass pieces in the form of bangles and 3 or 4 mirrors where the external light is reflected, refracted through the glass pieces, glass bangles and you see multiple reflection through the mirrors.

In this case we have done away with the natural light and which means you can use it anytime you want whether there is outside light or not inside there are RGB LEDs several of

them and their lighting pattern changes as you move it around why because we have integrated certain sensors in this.

And of course there are mirrors also and embedded computer MSP430. As you can probably see as I am turning the Kaleidoscope tube around the lighting pattern inside has changed and if you have to look through this you would see interesting patterns. In fact what I will do is I will share pictures with you of the closer pictures looking through this hole so that you can see how the lighting patterns look like.

(Refer Slide Time: 42:34)



Now I have another project this is slightly advanced project based on MSP430 and this is called Talking Tom and this is the hardware implementation of a Talking Tom application

you may have seen on a mobile phone what this does is on what that application does is that it will record your sound then it will play back at higher pitch. So this project we call Talking Tom.

We actually call it Talkative Tom because we did not want to have issues of copyright names this does that and it also adds another features that in the second mode it can play back the recorded sound and it will sound like an alien. So let me give you a demonstration of that I have two selected switches in which I turn it off and on and with the other switch I can select the mode in this case I have selected the high pitch mode.

Hello this is the project demonstration of Talkative Tom. Let me change the mode and make it the alien mode. Hello this is the project demonstration of Talkative Tom in the alien mode. The entire enclosure was fabricated in the lab as you can see the circuit board was also fabricated and soldered in the lab. It has integrated battery source, charger and audio amplifier and speakers.

(Refer Slide Time: 44:12)



And last but not the least the project I am going to show you is what we call as the levitating doll, I am going to power it with the adaptor 12 volts adaptor. In my hand is a 3D printed doll and in the hat of the doll there is a small magnet there is a microcontroller on top and here I have a electromagnet. At the bottom of the electromagnet there is a hall effect sensor which senses the position of this magnet when I hold it there and the microcontroller will maintain the position of this doll suspended in the air let us see how it works.

It is floating in the air why it is sensing the position if because of gravity the doll tries to fall, it turns on the electromagnet if the doll tries to come too close it turns off the electromagnet and this operation it is doing 1000 of times the second to maintain the position at a given location and all these are implemented by my students using MSP-430 microcontroller and so my students can do it I am very sure you as my remote student can also do it. Thank you.

We are going to continue our next lecture and talk about other issues related to embedded systems, but at this point I would like to summarize what we have discussed here. We have gone through various definitions of embedded systems, we have seen application areas of embedded system. We have seen how embedded systems could be implemented using a general purpose processor or application specific processor or using a single board computer.

We have seen what are the characteristics of embedded systems and at the end we have seen some applications which I hope you found to be very encouraging. Thank you very much and I will see you in the next lecture, bye.