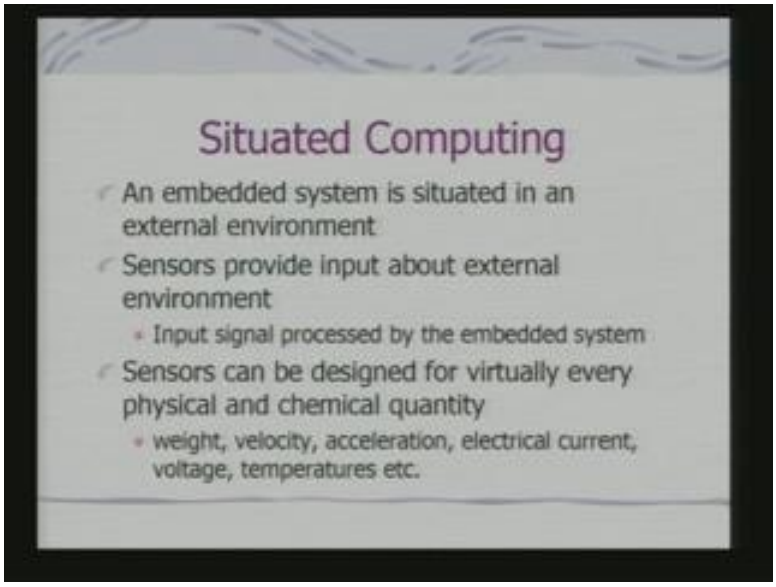
Embedded Systems
Dr. Santanu Chaudhury
Department of Electrical Engineering
Indian Institute of Technology, Delhi

Lecture No. # 8
Digital Signal Processing.

So, for we have looked at general purpose micro-controllers like PIC and ARM. And these micro-controllers in their instruction set as well as internal architecture there where enhancements to support signal processing operations. Today, we shall look at processors which have been designed specifically for signal processing applications. Now, why this signal processing processor is are important for embedded systems. In a way embedded systems are targeted for what we call situated computing.

(Refer Slide Time: 01:58)

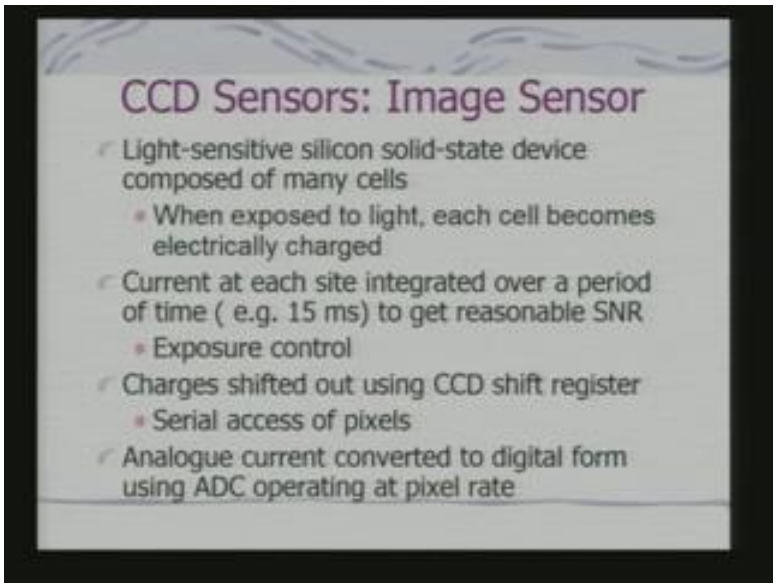


An embedded system is situated in an external environments it receives input from external environment via sensors. And the signals that it receives through sensors are processed for taking action via actuators or doing some kind of communication or data processing tasks.

Now, sensors can be designed for virtually every physical and chemical quantity and. In fact, today there are variety of sensors available which can be interfaced and used in your embedded systems. And all this sensors provide you with the set of discrete values which

needs to be processed. Although many of this sensors working analog domain I need to do what convert this data from analog to digital form to do any kind of processing. We shall look at some examples sensors.

(Refer Slide Time: 03:05)



A very common example is your CCD sensors which are used in camera for sensing images. In fact, I the CCD of SIMA sensors you will find in your common digital cameras which is very commonly used embedded system. The SIMA sensors you will find on your cell phones. So, these sensors of a sensing images we are looking at in particular CCD sensors which is a light sensitive solid state silicon device composing of many cells.

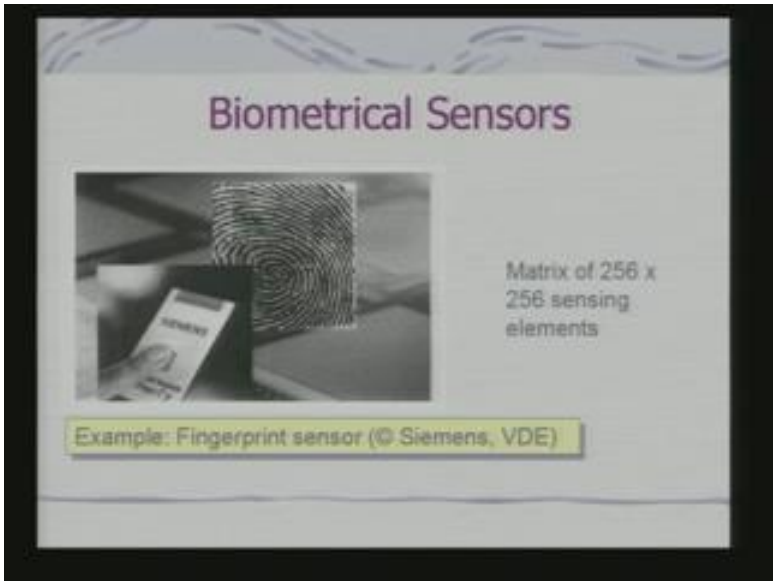
These cells are actually nothing but some form of capacitances which accumulate charge depending on the intensity of the incident light. And at each site the charges which translate to current is integrated over a period of time, to get the reasonable signal to noise ratio. So, you have really the data that you are getting represent external world and not the noise. And these charges are shifted out using shift registers they are not digital shift registers, but shift registers for storing charges. So, you have got something like charge buckets and charge is shifted out sequentially through this buckets.

Now; obviously, the output of this sensor is the, is analogue current and these has to be converted to digital form using ADC to get the digital data and these digital conversion has to take place at pixel rate. That means if you are taking frames at the rate of say 25

hertz and if you are talking about an image size of say 256 cross 256 then there is a definite timing window at which we have to obtain data for each pixel per frame.

So, analog to digital conversion has to take place at that rate and subsequently, I need to process this data for any kind of operations. Another example of a sensor today, which is becoming very common for security applications at these biometrical sensors.

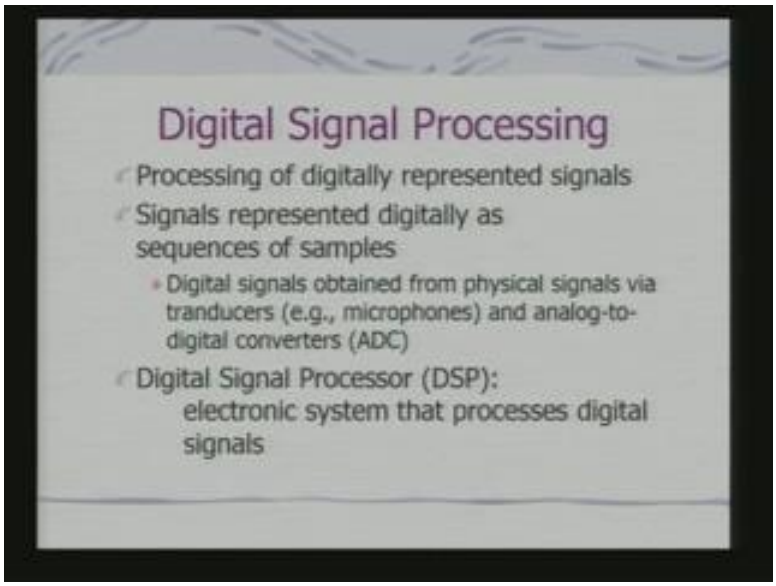
(Refer Slide Time: 05:34)



This is the sensor for taking fingerprints. Even today you will find products being announced that you will may have a cell phone with a fingerprints sensor. So that; it becomes a unique access. So various such products coming into the market. And these an example of a fingerprints sensor, which is got a matrix of 256 into 256 sensing elements which obtains the fingerprints and here this is a picture of the sensor and this is the kind of fingerprint image that this sensor provides

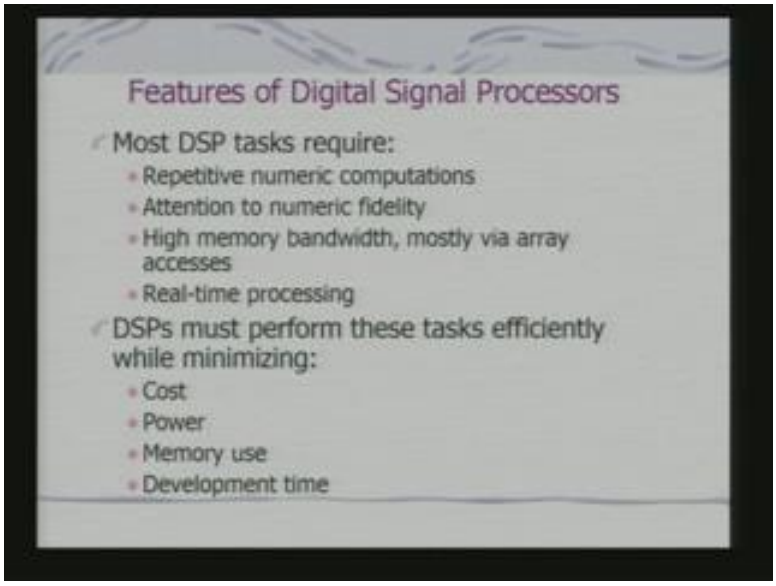
So obviously, to process these kinds of data which are coming from a variety of sensors, I have giving this two examples there may be hundreds of other sensors. We need special processing capability for the processors. So, we are doing digital signal processing on the processors for processing this data from the sensors.

(Refer Slide Time: 06:33)



We are processing what digitally represented signals. And signals are represented digitally a sequence of samples. The sampling interval would be obviously decided by the properties of the signal. And so, digital signals typically obtained from physical signals via transducers and converted to a digital form using ADC. I have given you an example of how the image data get converted and then it is processed by a DSP, which is an electronics system that processes these digital signals.

(Refer Slide Time: 07:17)



What are the features of digital signal processors? If you have seen that nature of the data that has come in, they will be repetitive numeric computations. Because if you consider an image, there is a pixel coming from each site, sensing site. So, number of pixel is 256, if one operation has to be operated on a pixel, then I have to operate it 256 into 256 times. So, I have to do repetitive numeric computations.

High memory bandwidth is required to transfer this volume of data. Because it is effectively if you look at the image data 256 into 256, the standard representation is array accesses. Because I am just having a sequence of pixel data. The numeric fidelity is also of importance, because I cannot really have errors in the computation and the other requirement is real-time processing. And this real-time requirement is impose by the nature of the signal.

As I was talking about that, I have to obtain a pixel at a particular rate, satisfying the condition that I am taking frames at the rate of 25 hertz. And if I have to produce an output image at a particular rate there my processing has to take place consistent with the sampling time period available with me. So, external signal imposes real time constraints.

Now, all these things have to be done taking into account the overheads introduce by cost, power consumption, memory used and development time. You definitely like to have minimum cost. And that attempt to minimization of the cost, you will find today as resulted in having almost all cell phones coming with your image sensors; the camera. And your webcams available with your PC's at almost minimum cost.

Power consumption also becomes important. Because I won't like to do any kind of processing consuming a lots of power, because majority of my embedded appliances could work on battery and memory usage that is also restriction. The restrictions come from the constraints of power as well as that of the size. There are many DSP applications.

(Refer Slide Time: 9:49)

We have just listed some of them. And In fact, all of these applications are actually put into some kind of a dedicated system which are nothing but what we call embedded systems. Your audio coding decoding, surround sound, see your MP3 players are all basically doing what at decoding, decompression of the audio data; that means, it doing a signal processing tasks and there all embedded systems.

Cellular phones I have already talked about and teached to you which is targeted for communication, but it has to do signal processing because it is receiving your speech input then converting to an appropriate form. So that it can be communicated.

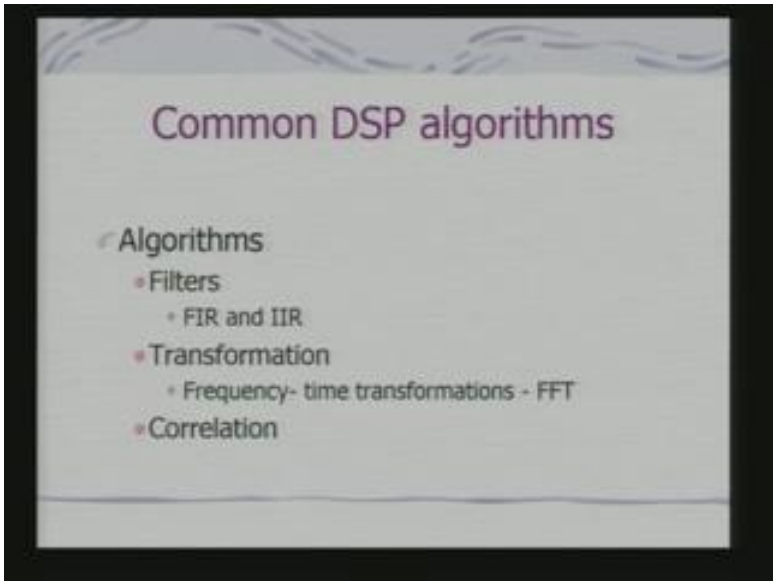
Robots, any kind of robo are example of embedded systems. Even your disk drive control which needs to do some kind of signal processing because it is sensing the position of that your head in the disk to get the data correctly. So, that is also an example of an embedded system because that is not doing a general purpose computing.

In the field of medical diagnosis, this diagnosis equipments; all of them are obviously, processing variety of signals obtained from biomedical sensors. Your ultra sound machine, your MRI imaging devices, your ECG devices they are all doing necessary signal processing.

And defense, your radar and sonar processing missile guidance are other examples which are really doing complex signal processing tasks. And In fact, a typical sonar processing

board may have a number of this powerful signal processing chip sitting inside doing parallel computations.

(Refer Slide Time: 11:55)

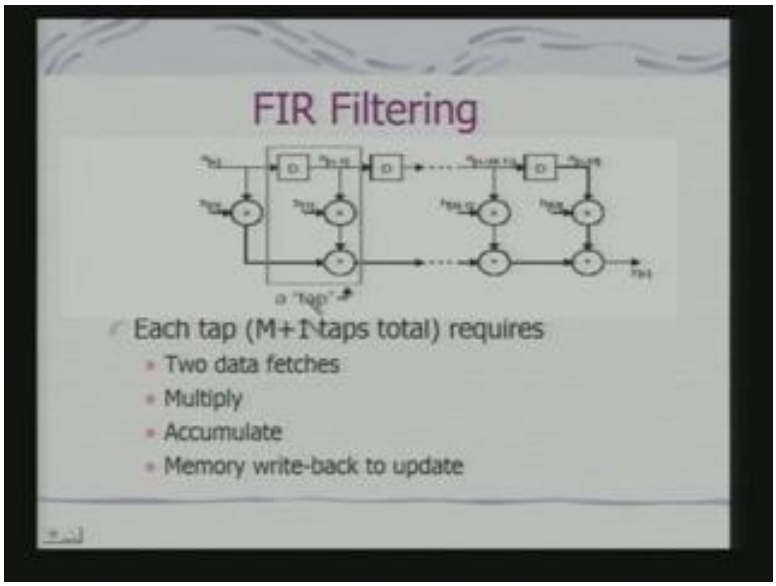


Common DSP algorithms will be filters, because we would like to remove noise that is one of the objectives. Other objective is to look at different components at different frequency bank. So, filters are very common application.

Transformation, that is taking the signal from time domain to save frequency domain incase of speech signal. In case of image signal we may like to transform from special domain to frequency domain. So, this transforms are also very important algorithms. Then performing variety of correlation tasks for doing signal classification it is also very common operation which is done

Now, these are the common algorithms then the basic question comes up is the DSPs that is digital signal processor must have dedicated hardware to facilitate implementation of this common algorithms. We shall look at the simplest of them and with respect to that try to understand the requirements of the architecture and see how that is really realized in different digital signal processors.

(Refer Slide Time: 13:18)

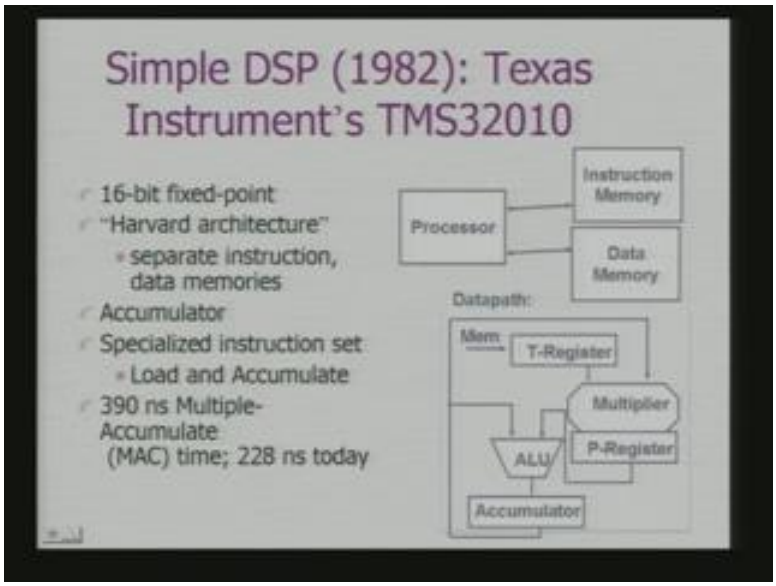


So, the simplest one is FIR filtering. So, FIR filter consist of multiple taps and what is happening here if you look into it these are nothing but signal samples and the D is just representing the delay element. And what is really happening is I am doing at each tap that if I look at this as a tap and each tap what is the operation.

I shall be doing two data fetches, multiplication and an accumulation and we would like to write back to the memory to do the subsequent computation. So, what is, what the requirement is therefore, in this case. The requirement is I need to have the ability to fetch data. In fact, if I can do multiple data fetches simultaneously that could help the implementation.

The next, what we are doing? We are doing a multiplication which is basically multiplying with the coefficient vector of the filter. And then we need to accumulate the data. So, what we are doing is this is a multiplication H one of these coefficients. So, you have multiplying with this and then you are accumulating with what has been already done earlier. So, what we are therefore, looking at we are looking at data fetch, multiply, accumulate memory write-back.

(Refer Slide Time: 15:03)



So, let us see how it can be done we are looking at one of the earliest digital signal processors. It was introducing in 1982 by Texas instruments. Now, here you will find that we are using Harvard architecture. In fact, there is a separate instruction memory and separate data memory. Why because obviously; I have require that for each tap I require data fetch and multiple data fetch. So, I would not like to have the bottle neck in memory data transfer. So, the option in the choice has been hardware architecture.

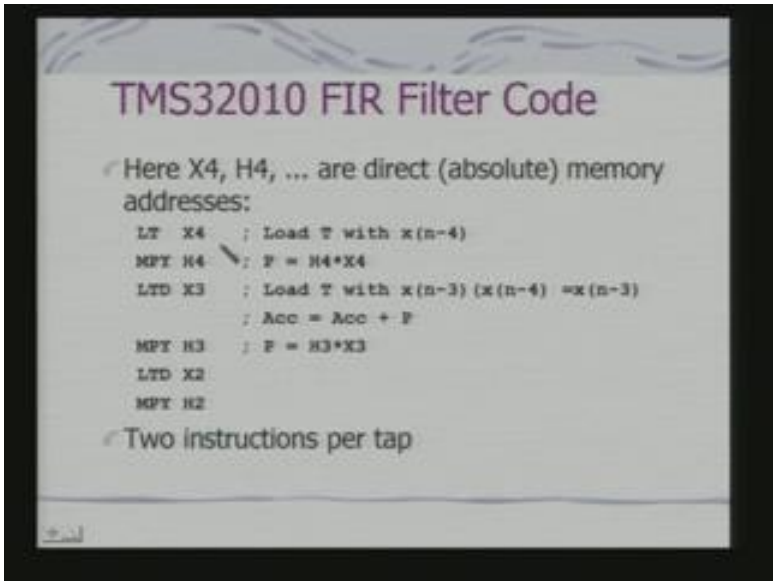
Now, if you look into here you have an accumulator, a classical accumulator which participates in these arithmetic operations. And specialize instruction containing load and accumulate.. So, here let us look at this internal organization and look at the data path. What you will finding here that you have got a dedicated multiplier. Because multiply has to be operated at each state in the FIR filter and it is to be used with the entire sequence of the data.

So, you have got a dedicated multiplier, you have got a separate register where you are getting the data and putting it from the memory and output of the multiplier goes to something called a P register. Which can be used with the ALU to get the accumulator result in the accumulator itself. And they can be put back again with the multiplier if required.

So, these organizations, you can realize that this organization of the data path is primarily to facilitate multiply and accumulate operation. Which is the key operation

incase of any filter implementation and we have seen how it is critical for a FIR filters. So, let us look at the code of filter code with respect to this processor.

(Refer Slide Time: 17:17)



Let us consider this X4 and H4 are direct absolute memory addresses. And what we are showing here is I am loading T. T register, we have already seen with a particular sample I am multiplying H4 has already got a coefficient of the filter coefficient vector.

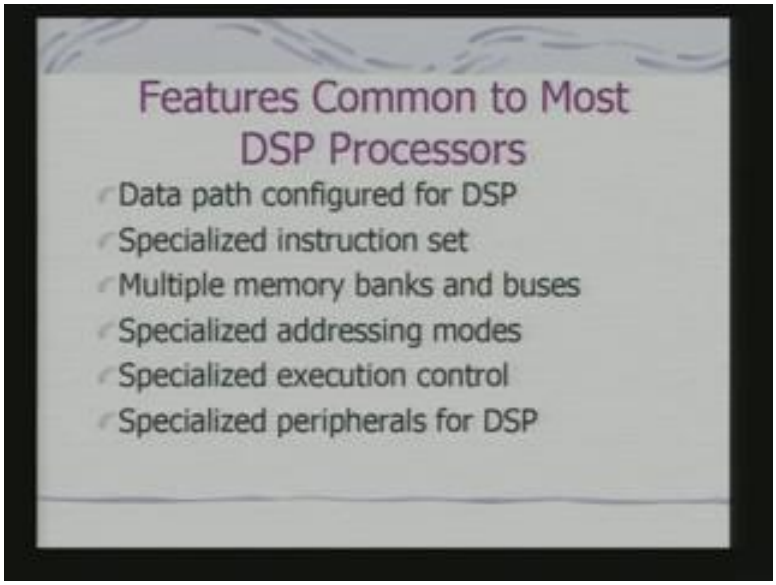
So, I can compute when I am multiplying I have got a P register below that of the multiplier block. So, that data goes into the P register and then have an instruction called LTD X3. So, which is loading T with the next sample and also loading my accumulator with accumulator plus P, these can be done because there are two different parts of the data path.

So, effectively what is happening is; what I require just two instructions for tap, effectively just two instruction per tap. And this is these has been possible and this is possible because the organization of the data path which has been provided for. You can see that I can do a register, this T register load as well as accumulation through ALU to the accumulator simultaneously.

So, you can understand therefore, the demands of the processing and how it has influence the design of data part design of the signal processors. So, with this motivation

on the background let us look at the other features which are common to most DSP processors.

(Refer Slide Time: 19:18)



These are the points which are listed and these are the features which are universally true for all this processors.

Data path is specially configured for DSP. It has got a set of specialized instructions. There will be multiple memory banks and busses because you would like to a multiple inputs simultaneously. For this purpose specialized addressing modes, some execution control techniques and specialized peripherals for getting inputs from the sensors. I may even have actually many of this signal processor you have onboard; digital to analog converters for the output and analog to digital converter for the input.

(Refer Slide Time: 20:13)

So, let us look at the data path; we have already seen some aspects of the data path and here we are trying to look at some of the other aspects. So, specialized hardware to perform key arithmetic operation in one cycle, and key operation is particularly multiply and multiply accumulate.

We have got hardware support for managing what are call shifters. In fact, we have already seen barrel shifter in the contest of ARM processors which facilitate multiplication. But shifters have also got another very important role in the contest of floating point numbers.

It can used very easily for adjustment of mantissas. Because, I have to make the exponents same by adjusting the mantissas, for doing any kind of floating point operations. Then many of these processors in their accumulators have got what are called Guard bit is. That means additional bit is, to increase the accuracy of the results. Because you can understand the precision of a computation depends on the number of bit is are located to store the data, each data element.

Now, if you can have additional bit is allocated then, what is the advantage?

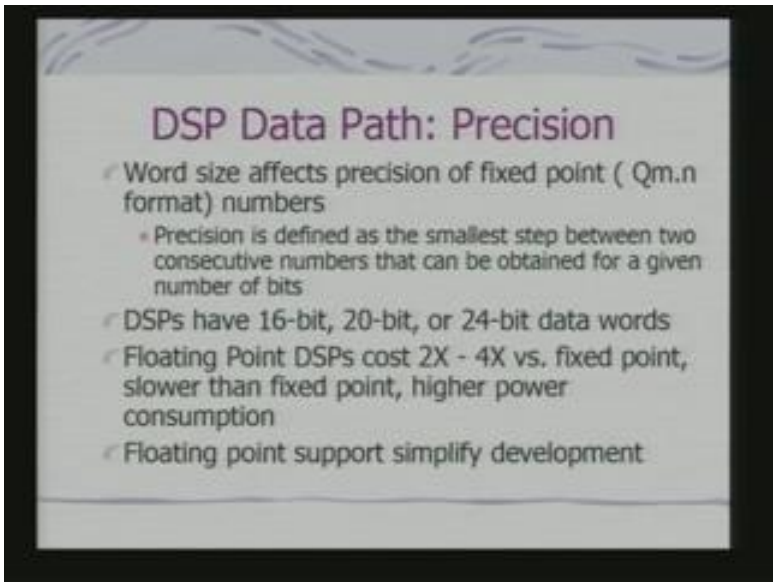
Advantage is the precision of the result increases. And if you are doing a multiple step operation, then the result because of truncation or rounding at a particular step will not

accumulate because I have allocated or I have got additional bit is to store the intermediate results.

So, you will find many of this digital signal processor, it may support say 32 bit data word. But accumulators maybe 40 bit and supporting 8 bit of Guard data to increase accuracy of computations. And these designs comes from the requirements of numeric fidelity.

Saturation we have already seen. The saturation instruction said, In fact, in the DSP extension of the ARM. So, that there is no wrap around. And this is important for delivering or managing the signals having logical, having values within logical limit is. Therefore what I was talking about with relation to the guard bit is is that of the precision.

(Refer Slide Time: 22:47)



Precision is an important component for representation of the fixed point data even for the fixed point processors. Precision is defined as a smallest step between two consecutive numbers that can be obtained for a given number of bit is.

And, In fact, you will find that this fixed point processors the use Qm.n format numbers. Where n is the number of bit is allocated for the fractional part, m for the integer part. And it is not really a floating point representation. It is a fixed point representation.

Depending on the number of bit is allocated there will be the range of numbers that can be represented is different.

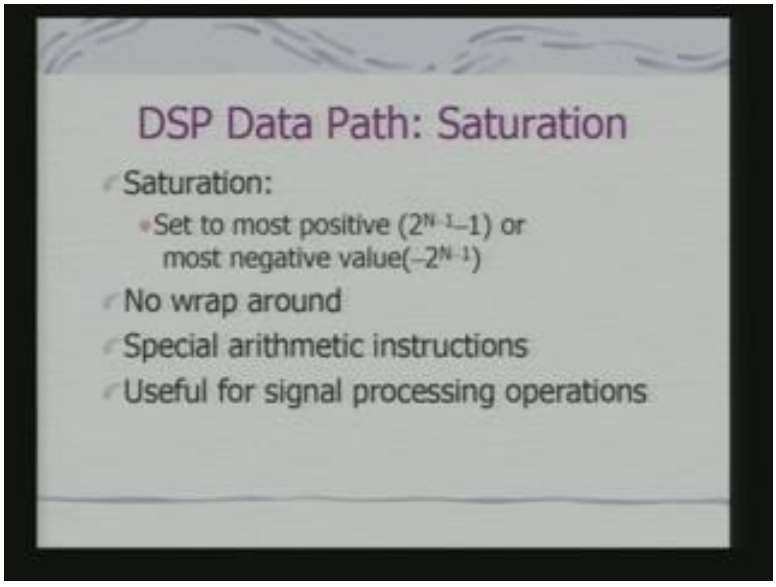
And the most significant bit that, if I am having a 16 bit number. So, the 16 bit can be used also as a sin bit. So, depending on the format the precision would change. Therefore you can realize that the word size is a critical component and you choose a DSP depending on the precision requirement of your application program.

I have shown some examples: The DSP's can have 16 bit, 20 bit or even 24 bit data words.

On the other hand you have got floating point DSP's as well. Obviously, floating point DSPs hardware will be more complex, their cost could be more and in many cases there also slower because of the clock speed restrictions and they will obviously, consume higher power. Because, you are having more hardware on the silicon real state.

But floating points supports simplified developments because why this issue comes in? Because if you have to realize say a FFT or DCT operation any kind of transform operation using a fixed point processor. Your software needs to take care of the the precision issues representation of the numbers and; obviously, that may result in more development, software development tag.

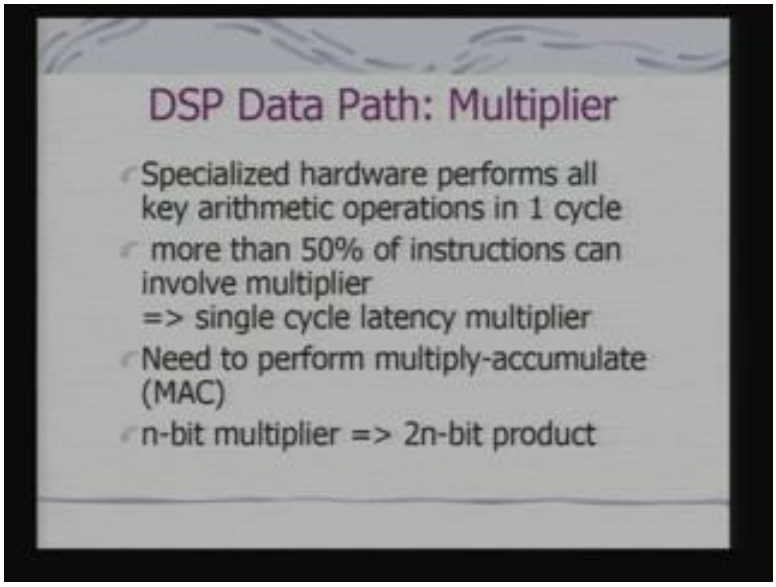
(Refer Slide Time: 25:30)



The saturation is we have already talked about, that in the data path you have got unit is. Specific unit is which sets a numbers to most positive value, prevents wrap around. And this is done using special arithmetic instructions.

And we have seen for signal processing operations like operations on the pixels. I would not definitely like a pixel value to become arbitrarily zero or beyond zero to wrap around. If it is maximum it should get stuck at the maximum. If it is minimum it should get stuck in the minimum.

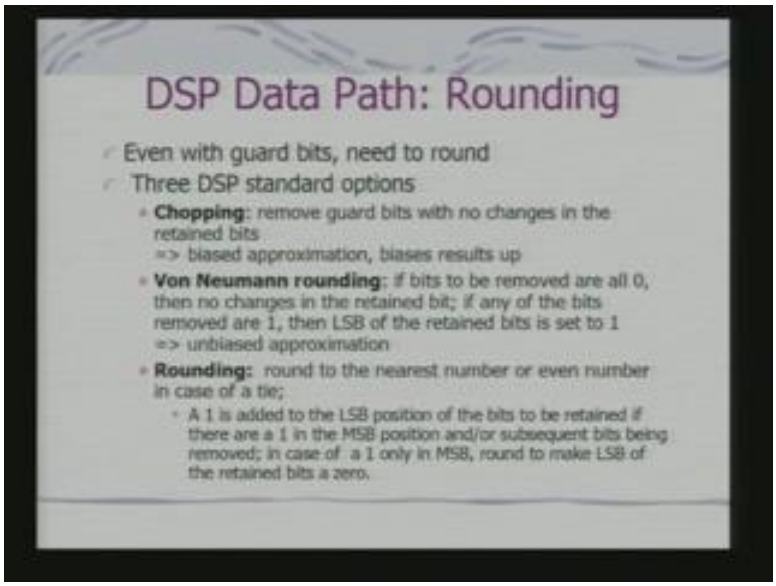
(Refer Slide Time: 26:08)



The multipliers is definitely a key component and a simple analysis of code have found that more than fifty percent of instruction can involve multiplier. And majority of this DSP's implement what are called single cycle latency multiplier. That means, you can do a multiplication using a single cycle.

And you have got multiply an accumulate instructions. So these are typically built as separate hardware blocks and not strictly part of the ALU.

(Refer Slide Time: 26:48)



Next issue is related to this is that of rounding. We have already talked about precisions.

Now, when we are actually doing this kind of real number arithmetic say multiplication. Multiply an accumulate, rounding becomes critical. Obviously when we use guard bit is you are increasing the precision.

So, when your accumulator is using 40 bit is. You are doing accumulate with the help of 8 bit additionally available as guard bit is and your precision increases. But when that result has to be stored in memory, this guard bit is have to be discarded because your memory is organize as a thirty two bit data word maybe. So, in that case you need to do rounding.

The typically you have got 3 schemes which are chopping, Von Neumann rounding and normal rounding.

In chopping you remove guard bit is with move changes in the return bit. So, it is a kind of a biased approximation because you have the result biased up to the positives.

The other thing is a Von Neumann rounding. What happens in Von Neumann rounding? If the bit is to be removed there all zero then there are no changes in the retained bit.

If any of the bit is removed are 1 then LSB of the return bit is is set to 1, And u can understand this pretty well, the whole motivation is that your, obviously, if are all zero then we need not do the change. If there is a some change we are setting the LSB to 1.

So, you are going to the, if the LSB is already zero of the return bit is by one we are moving to your positives. If it is already 1 we are not moving towards positives. So, their would be a negative error. So, it is a kind of a unbiased approximation.

But; obviously, the approximation error would be pretty high .Because what I am doing, I am changing just the last bit that LSB, depending on the bit is which are getting truncated.

The rounding, the basic philosophy of the rounding is round the nearest number or even number in case of a tie; is an approximation technique which reduces or which has got the minimum error.

So what happens here, a 1 is added to the LSB position of the bit is to be retained. If there are 1 in the MSB position and are subsequent bit is which are being removed have also 1s.

That means if I am removing three bit is. If the number is say 1 1 0 or 1 0 1 what shall I do? I add 1 to the LSB of the retain bit is.

What is the difference of the previous method? In the previous method, I just set it to 1, In this case I add 1 to LSB

And if I have the bit is which I am removing let us say three bit is and they are 1 0 0, then what do I do? I actually add one, if LSB is already one and otherwise I make it or keep it zero; that means, what I am doing? I am trying to make the LSB always 0 that is the even number.

So, effectively if I take a decimal case you will find that this is a kind of if it is I am adding one to the retain bit, when it is that is 1 0 0 and 1 0. I am not doing just for 1 0 0. I am doing it for 1 1 0 and 1 0 1 and I am adding 1 to the LSB.

Some approximating, the thrown away bit is by adding 1 to the previous retained bit

If it is 1 0 0, it is actually a tie. In the sense that if it is a 0.5. If I am just throwing away in case of 0.335 and if I am throwing away five. Whether to make 3 to 4 or keep 3 as it is. What we do? We take it to the even number.

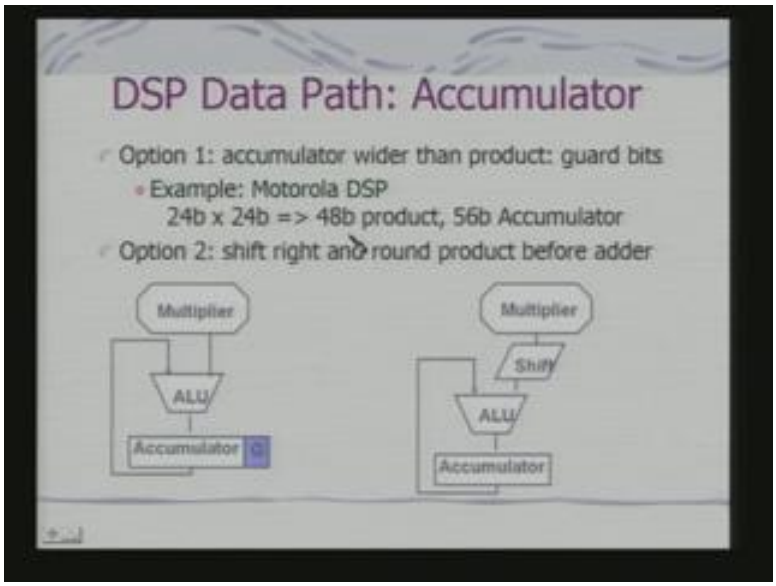
So, effectively I am resolving ties in favor of a even number. Otherwise I am adding one to the LSB, if I have got the bit is to be thrown off greater than 1 0 0 or 0 1, the condition wise 1 0 0 etcetera. If it is not. So, I am not adding anything to the LSB.

So, this is a rounding method and if I compare this. why I have talked about all these complete rounding methods? Simply because you can realize that if I am implementing this rounding, this introduces additional computational requirements. And I need to have additional hardware in the DSP to implement this rounding algorithm.

On the other hand, the other rounding; if you look at chopping. Chopping is a simplest with almost no hardware requirement for the purpose of doing the approximation.

So; obviously, these demands put in requirements of additional hardware on the processor. Which can be used depending on the kind of instructions you are using and also the kind of processors that you used determine what is the actual methodology may adopted

(Refer Slide Time: 33:24)



So, here what we are looking at is a kind of an organization in this case. So, I have got a multiplier and ALU and this accumulator has got a guard bit which is maybe an 8 bit.

So, I am just giving an example of a Motorola DSP. We shall see more such example: which is got if I am doing a 24 bit into 24 bit. I shall get a forty eight bit product, but I have got a fifty six bit accumulator to have the guard bit.

And if I am doing in a continual sequence of computations this guard bit is provide for high accuracy. And then when I am storing the value from the accumulator. I shall be using an appropriate rounding method and store the result in the memory.

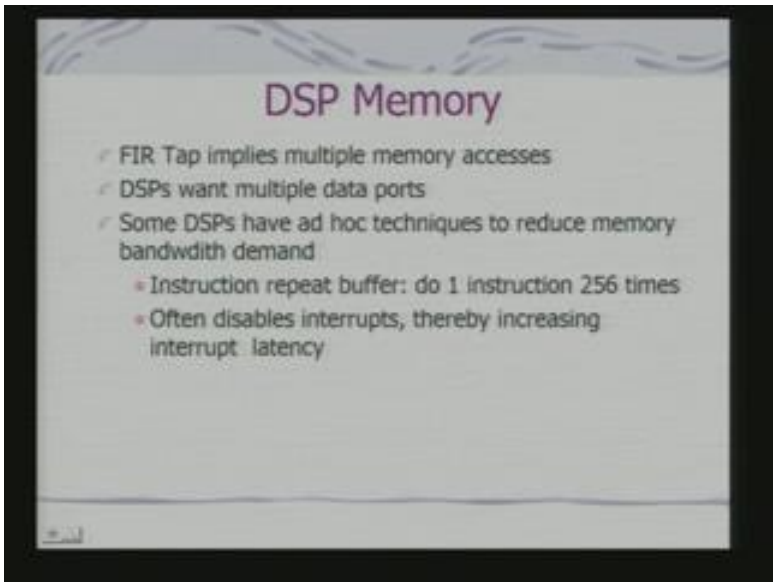
If I really do not have the accumulate accumulator guard bit is. What I shall do? Here I shall do the shifting and rounding. So, the round shift block which I am showing here, it is basically shift and rounding block.

So, whatever algorithms I have talked about rounding, that gets implemented in this block. Because the multiplier is producing the output that gets appropriately rounded and then through ALU it is added. And the result in stored in accumulator for multiply an accumulate sequence.

So, these are the two basic organizations, taking into account the Precision on the rounding in picture.

So, next look at the memory organization part.

(Refer Slide Time: 34: 55)



FIR tap implies multiple memory accesses. We have already seen that because I need the data,, I need the coefficient, the data samples there which is coming, as well as the coefficients.

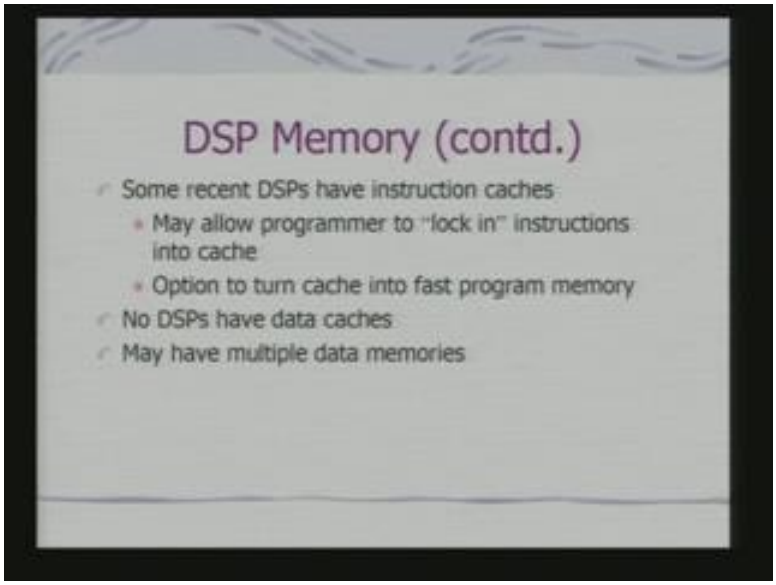
And DSPs want therefore, multiple data ports. And In fact, you will find many DSPs having multiple data buses and multiple data memories. And many DSPs it Implement what are call ad hoc technique to reduce memory bandwidth demand.

One is instruction repeat buffer. So, instruction repeat buffer is what? I can put a set of instructions. And if these instructions are repeated then I need not go back to the memory to fetch the instructions.

So, I already fetch the instruction put them in a buffer and execute them in a loop. So; obviously, this would reduce the loop overhead. And that this design issues comes up from the demands of repeated if numeric calculations of the DSPs

Often this disables interrupts, but and thereby increasing the interrupt latency. This trade of (Refer time: 36:01) is an important point to note.

(Refer Slide Time: 36:02)

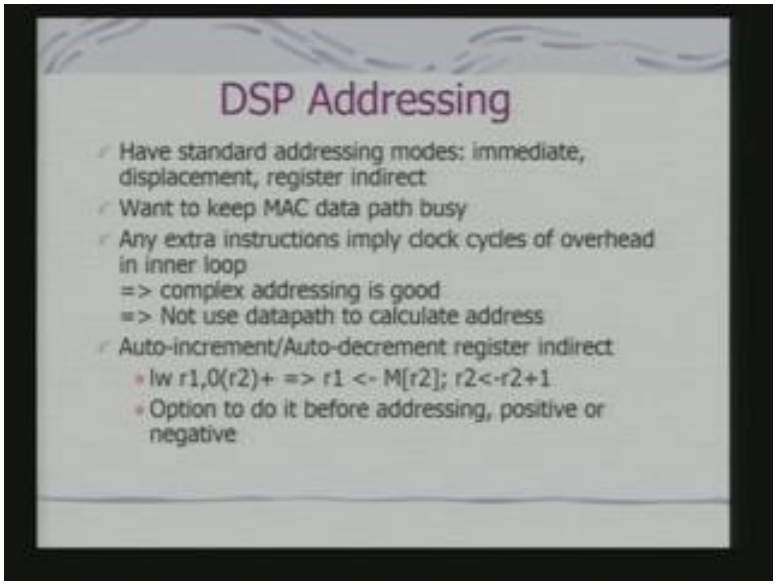


Many of the recent DSPs have instruction caches. And these caches also allow programmers to lock in instructions. That this instruction is never be remove from the cache, because there to be repeatedly used. And these cache memories turn into a first program memory.

No DSP's typically have data caches. Why? Because the data is coming in a sequence or it is already stored in a buffer. And to be applied in a sequence and same data is not expected to be use multiple times.

So, data caches you will not find commonly in the DSPs, but you will find multiple data memories.

(Refer Slide Time: 36:46)



So, what are the different addressing modes this DSP is use?

They have standard addressing modes: immediate, displacement, register indirect. And they would like to keep the MAC data path busy. So, that is MAC is the dedicated hardware and which is involved. So, there are there are data flow through this path.

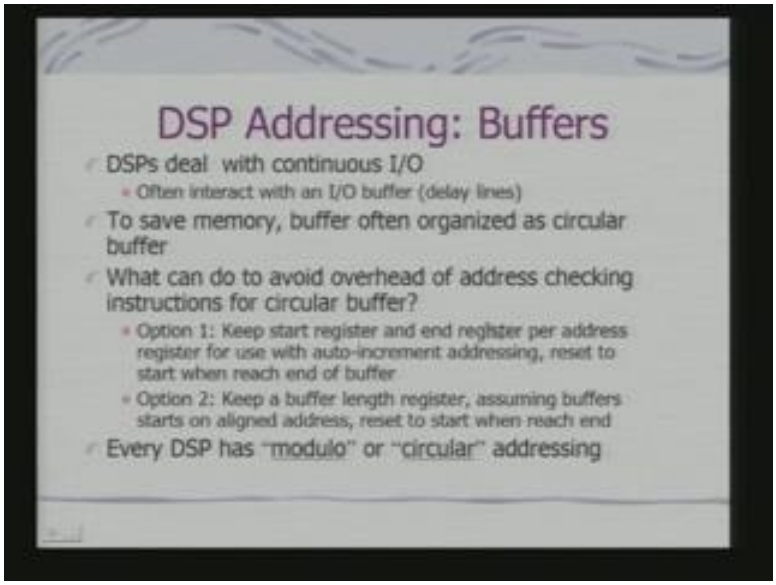
And any instructions imply clock cycles of overhead in a loop. So, that may reduce the efficient with which the MAC is used. So, you can use the complex addressing modes, but for complexing addressing modes, the basic tendency is not to use the data path.

So, the data path is primarily targeted for data manipulation. And address manipulation can be done through auxiliary computation unit is. This is a basic motivation in designing the hardware for implementation of addressing modes.

So, this kind of addressing mode are available: auto increment auto decrement. And you can do it before actual operations or after actual operations and you have discussed these kind of addressing modes in the context so far.

And you can realize why, I have implemented this to to facilitate this kind of sequence of data access? Because here what is happening, you are actually getting a sequence of data from a sensor. So, to access this data in a sequence these kinds of addressing modes are really useful.

(Refer Slide Time: 38:26)



You also have buffers.

So, when you are doing with continuous I/O, you would like to save the data samples in the buffer. And there has to be a logic implemented to avoid overhead of address checking instruction for circular buffer.

Now, why do you like the buffer to be organized as circular buffer? Because it is a continuous data flow. And you cannot have an infinite sequence of memory locations. You cannot have an infinite buffer

You have to simulate the infinite buffer. So, you use circular buffer to simulate an infinite and continuous flow in of data samples.

So, you keep start register and end register per address, using auto-increment addressing modes. So, if you have this then these can be very easily done. Easily help you in implementing the circular buffer. And what you do is other option. There is two options for implementation.

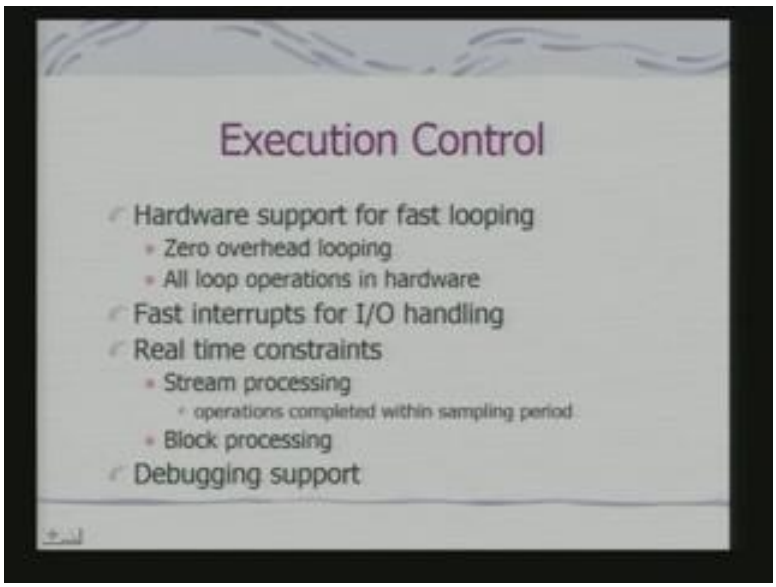
One is you keep start register and end register. And when you are using registers for auto increment addressing. So, automatically you check whether you have reached the end or not. So, how to deal with the next address and keep a buffer length register in that case starts. So, you have a start address and you keep a buffer length registers so, using this you can manage basically circular buffers.

So, that is why we say that each DSP has “modulo” or “circular” addressing. This is the distinct feature of the addressing modes of the DSP’s.

You will not find this kind of module or circular addressing mode available with general purpose processors. And these is implemented through these kind of options

Either having start and end register or having a start and length register. And all checks here in this case for implementation of circular addressing is implemented in hardware

(Refer Slide Time: 40:31)



For execution control you have hardware for fast looping and you would like to zero overhead looping. Because loops are Omni present. You have to do a number of numeric computations on the same set of data or on same set of operations on different data

So, for zero overhead looping you may have actually, I have already talked about the instruction buffers. You may also have a counter registers to keep track of how many times instructions or set of instructions are getting executed.

You have got first interrupts for IO handlings. Because if you have really handling each sample. You are reading in each inputs sample via interrupt driven IO. You need to have first interrupts.

And this is related to the real type constraints that I was talking about. Because of the nature of signals that are coming in.

In fact, you have got basically two options available what is called:

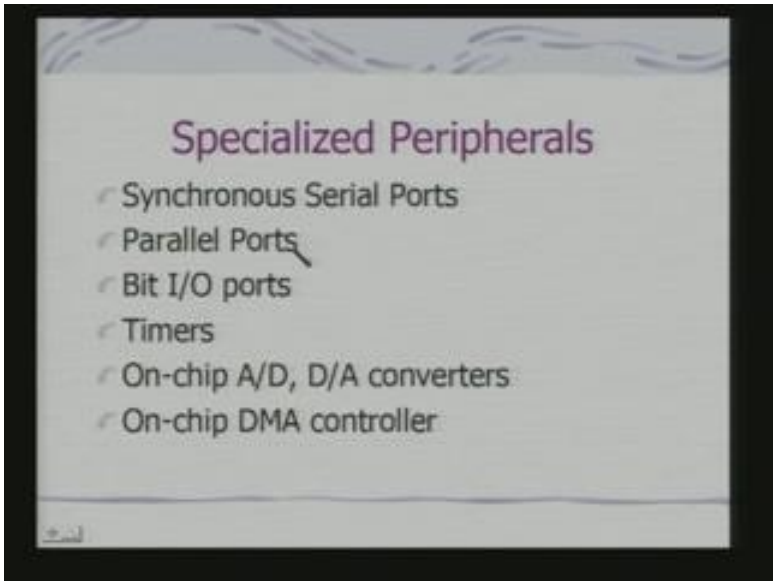
One is called stream processing, another is called block processing.

So, in stream processing the processor completes the operations within the sampling period itself. So, you make sure that you get the sample, do the operation, before your next interrupt comes in to reading the next sample and do the processing. So, this is your stream processing

And in a block processing you read the data, buffer the data. And use this kind of an auto increment addressing to get the data from the buffer and do the processing.

Also many of the cases you have debugging support in the hardware itself. In fact, this point you have seen with other processor also, which have been design for embedded applications.

(Refer Slide Time: 42:16)

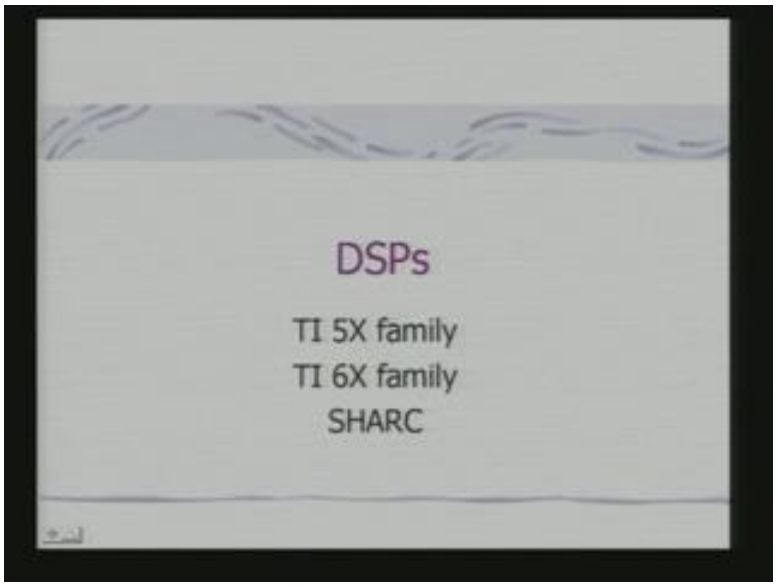


The specialized peripherals, you will find various kinds of serial ports and parallel ports for input of the data.

On-chip AD and DA converters, timers which have got the universal role on any kind of processors.

On-chip DMA controller for fast memory transfer of data, from one memory bank to another memory bank.

(Refer Slide Time: 42:41)

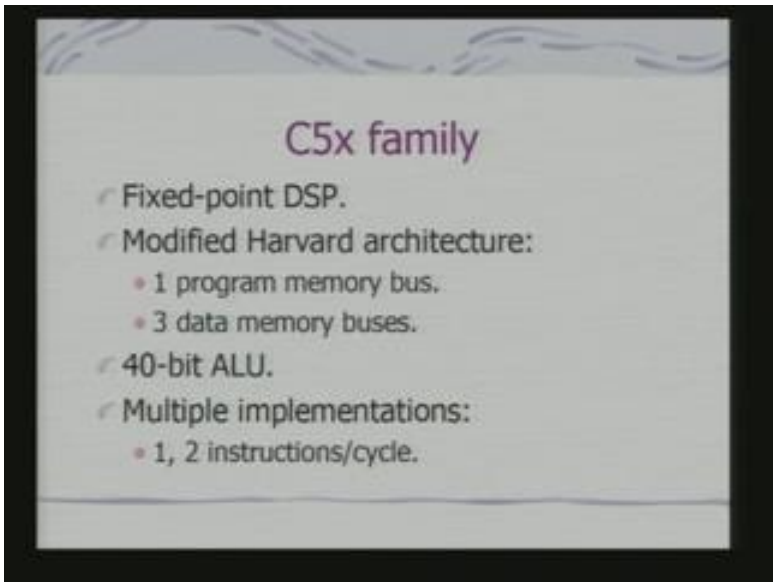


Now, with this coverage of the general features of the DSP. We shall see how this has been really realized in some of this specific family of the DSPs.

We shall look at TI as well as that of SHARC family. These are from two different manufactures.

Now, in case of C 5X this is a family of DSPs from TI. They are primarily fixed-point processors.

(Refer Slide Time: 43:11)

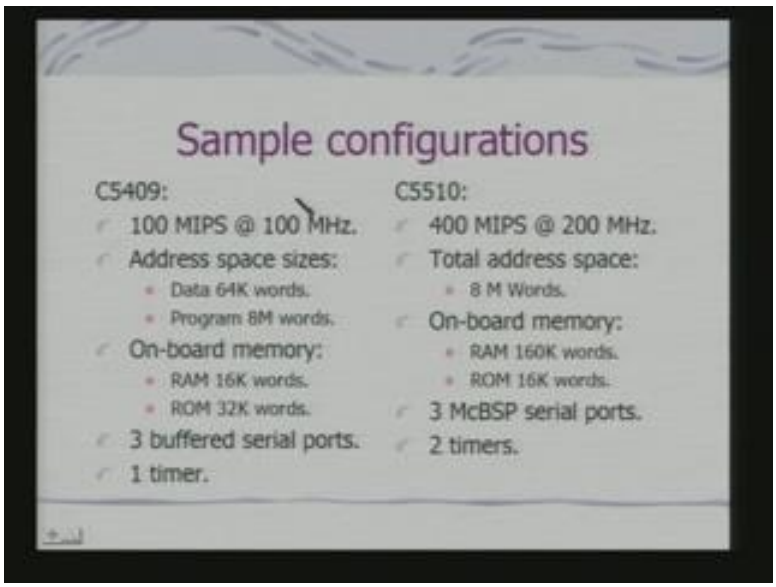


See if I want to deal with real data. I have to use QMN format and there is support for this QMN format data.

This is a Modified Harvard Architecture. There is only 1 program memory bus and 3 data memory buses. And this is the very interesting and distinct feature of this processor.

It has got 40 bit ALU. And multiple implementations, either 1 or 2 instructions per cycle it is not that all instructions are of the single cycle.

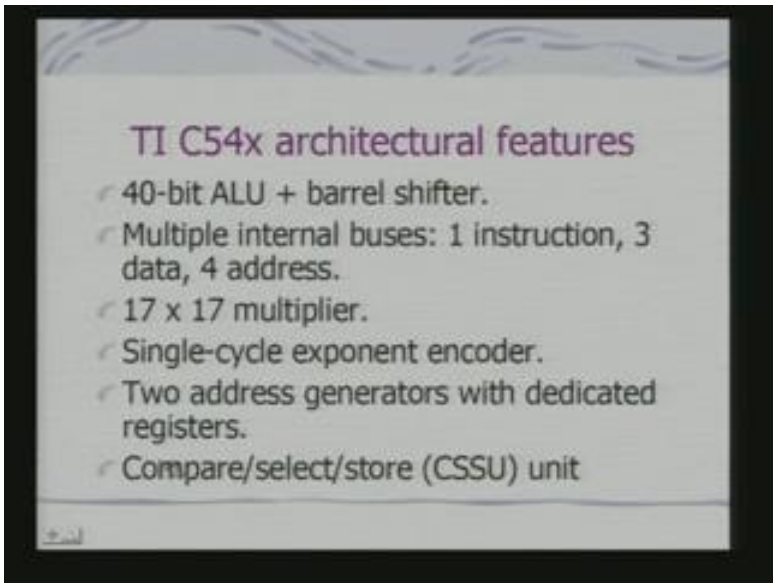
(Refer Slide Time: 43:43)



These are examples of two such processors: C5409 and C5510.

So, In fact, 54 and 55 again refer to two distinct families and variations in the architecture.

(Refer Slide Time: 44:00)



Let us look at now 54. So, 54x architectural features:

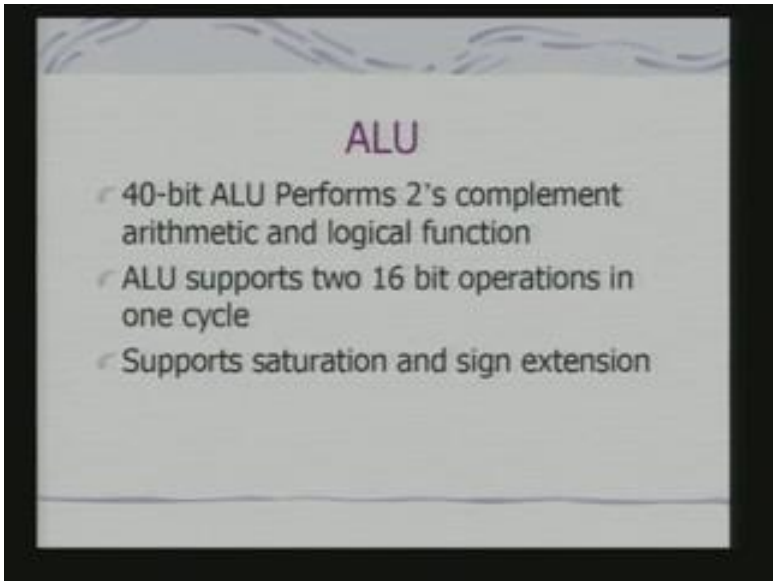
It has got a 40 bit ALU plus barrel shifter, multiple internal buses. 17 cross 17 multiplier. Two address generators with dedicated registers. Compare, Select, Store unit.

And these are, you will understand what, why these things are obviously provided. And 2 address generators. So, that you can have multiple operands coming in to the for execution.

Compare, Select, Store unit is you can compare basically the words; high words with the low word and you can select the higher one to store if it is, so decide. This is required for some of the signal processing applications particularly applications for beta decoding.

I shall not go into details of beta decoding, but this is a dedicated hardware targeted for those applications.

(Refer Slide Time: 45:03)



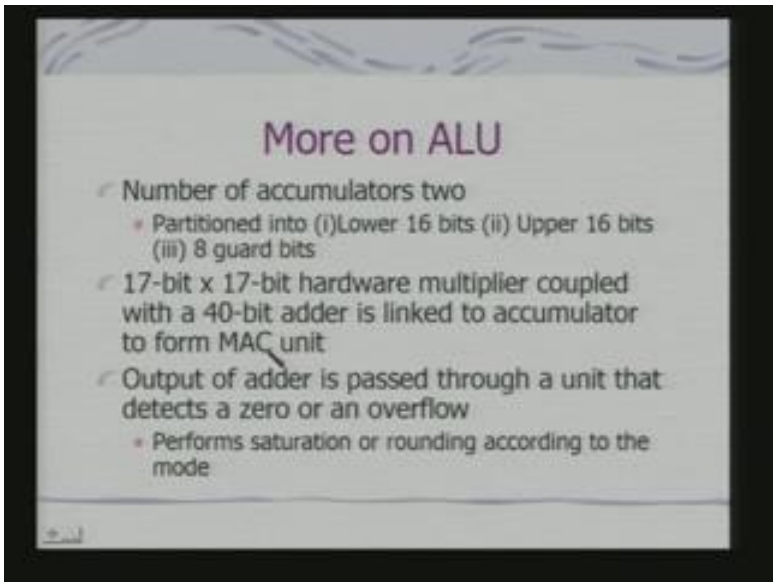
In fact, 40 bit ALU performs 2's complement arithmetic and logical function. And it supports two 16 bit operations in 1 cycle.

In fact, your ALU is basically targeted for 32 bit operations. And 32 bit word. In fact, 40 bit ALU which is provides for 8 guard bit is. But this 32 bit ALU itself can be considered as two 16 bit ALU's. So, you can have two 16 bit operations in parallel.

Supports saturation and sign extension. Saturation is requirement for all signal processing operations.

In fact, this accumulators has therefore; partitioned into lower 16 bit is, upper 16 bit is and 8 guard bit is.

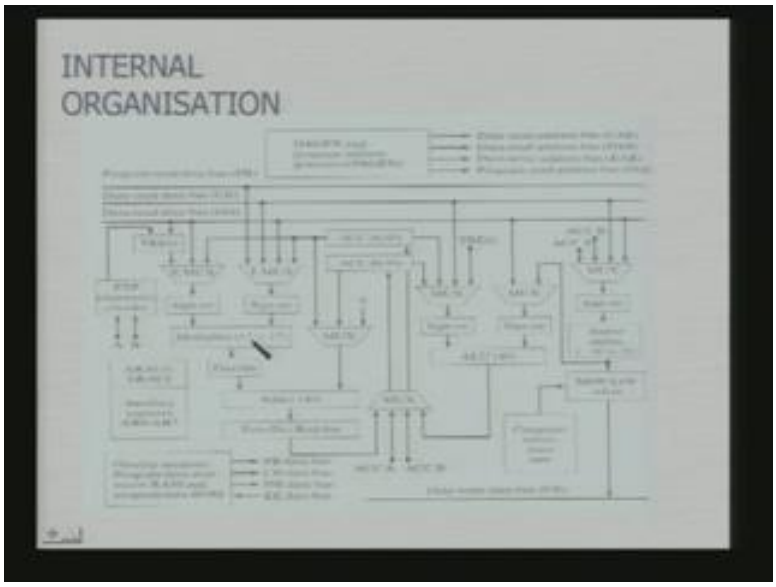
(Refer Slide Time: 45:52)



And this 17 cross 17 bit hardware is associated with the 40 bit adder and this 40 bit adder is distinct from ALU. So, that you have got an independent MAC unit, which is different from that of your ALU.

This output of adder is passed through a unit that detects a zero or an overflow. And so that it can perform saturation or rounding according to the mode of the instruction that is being operational.

(Refer Slide Time: 46:19)



So, this gives you a diagram and internal organization. And you will find that the complexity in the data path which is there.

It has got 2 accumulators and you will find that. Let us look at this part where you have got this multiplier. So, you have got this 17 cross 17 multiplier. And this multiplier on it is part itself has got an adder. These adders is distinct from that of your ALU

And these results the other operand for the adder can come from any of the accumulators.

.This accumulators, accumulator A and accumulator B. There is a two accumulators and then this adder. Output of the adder passes through zero detect or saturation or rounding block.

In fact, the hardware that I was talking about this is a special hardware block, which passes through. And then it can be stored back to an accumulator or it can be stored back to the memory also.

This is a special compare, select, store unit which can look at the data and select MSW or LSW. Which for, what we select depending on the size.

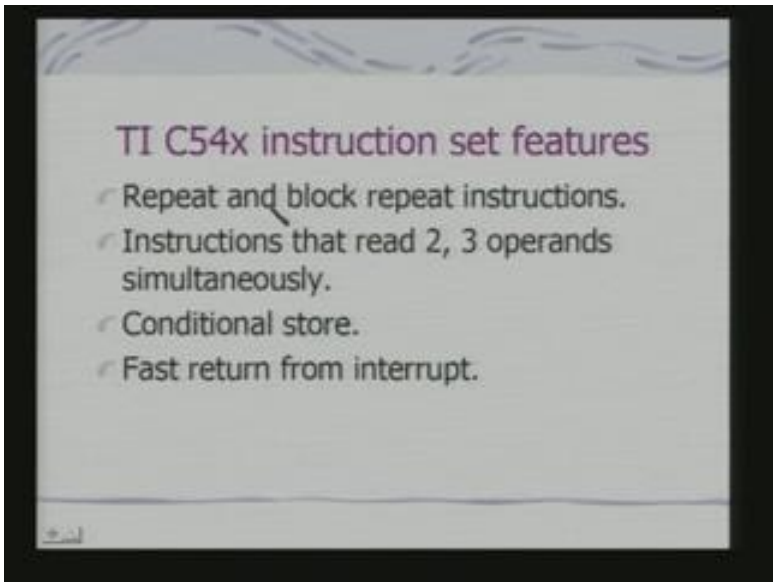
I said that I can compare the most significant word or the least significant word. Find the one which was the bigger and I can store accordingly.

This is the barrel shifter which can do the basic operations with multiplications and other things. So, you will find that it has got a pretty complex data path and it is address generation unit is different. Which we call DJEN. So, this address generation unit generates the address and it is flows to a number of buses. And here I have just shown the 2 bus data read bus.

And there are other buses. In fact, if I have a multiple data memory is there be multiple buses available for this operations.

The instructions sets; obviously, would support MAC and other operations arithmetic operations, but what is interesting is this unique feature.

(Refer Slide Time: 48:43)

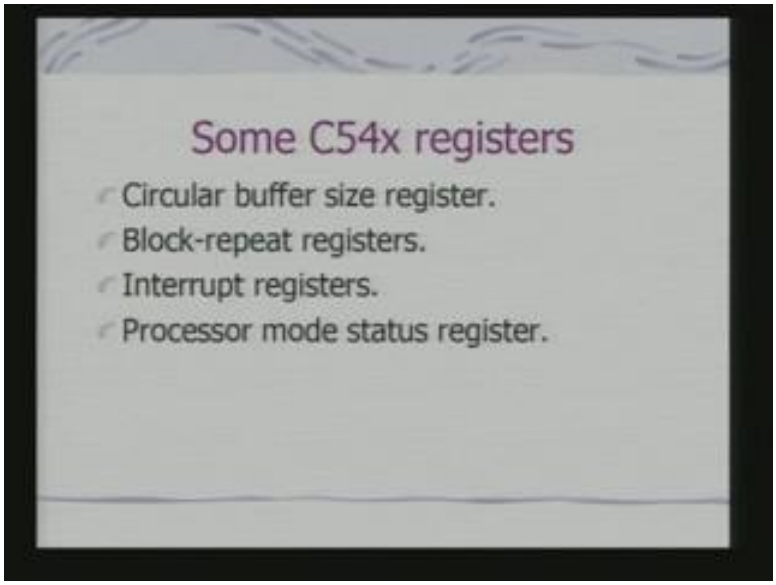


Repeat and block repeat instructions. That is I can actually repeat and instruction and that repetition is completely supported in hardware. Instructions can read 2 or 3 operands simultaneously this kind of thing is not typical of any of ARM. Because it was primarily designed as a RISC processor.

You can do a conditional store. Depending on a condition where and how to store the data that can be specified as part of the instruction.

And there are provisions for instructions. So, that you can return first from the interrupt. If I am returning first from the interrupt; obviously, my interrupt latency and the accuracy of the timing behavior of the interrupt routine is corrected

(Refer Slide Time: 49:29)



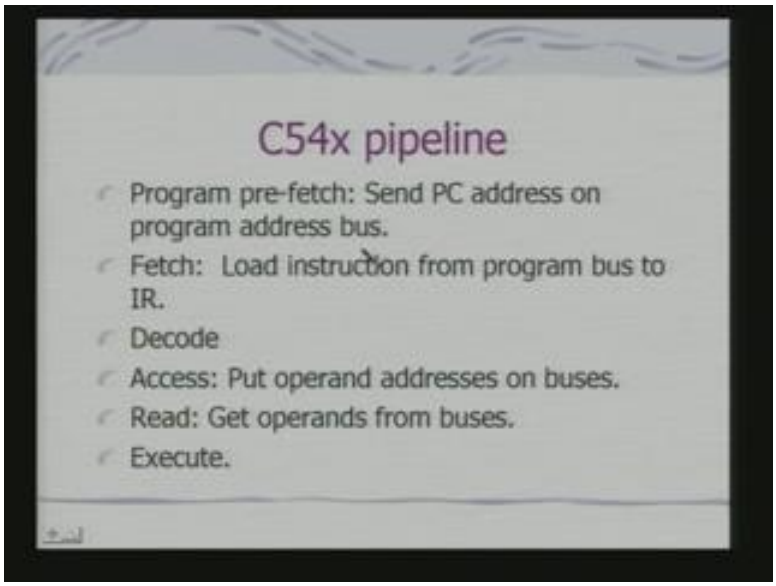
There are some registers I am talking about. There are number of registers to support arithmetic operations and other memory access.

But the interesting thing is there is a circular buffer size register, which enables you to implement circular addressing that I was talking about.

Block- Repeat register which tells you that the instructions and how many times they are to be repeated. Obviously, interrupt registers are there to take care of the control and masking conditions.

And processors mode status registers. It is a pipeline processor because; obviously, any todays modern DSPs. Just like your general purpose micro controllers which are targeted for embedded systems. Here also I shall have a pipelining, because pipelining increases the instruction throughput.

(Refer Slide Time: 50:20)



In fact, it has got this what we call a program pre-fetch. The send PC address on program address bus.

Fetch is load instruction from program bus to instruction register.

Decode.

Then you have got an access block. Put operand address on the buses. Because it is a operand are also obtain from memory it is not just there inside the register. And then you read, get operands from buses and execute.

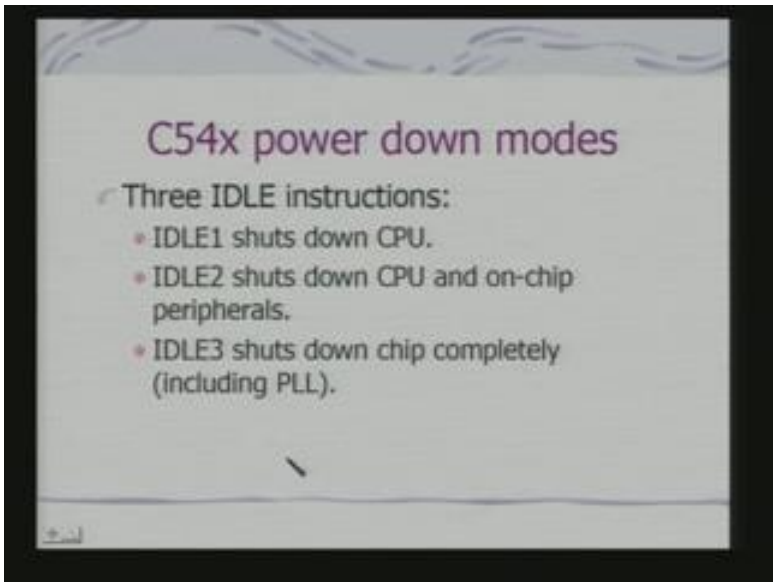
You can understand the basic difference with the say ARM pipeline. In the ARM pipeline the operands were typically as part of the registers. So, there was no pipeline stage for getting the operands from memory.

In this case you have got, because you will getting a sequence of data values maybe in a block processing mode. Which is stored in different data memories. So, you put operand addresses on bus.

And then get operand. So, this stage, these two stages comes into the pipeline

It also provides for power efficiency, what are called power down modes

(Refer Slide Time: 51:23)



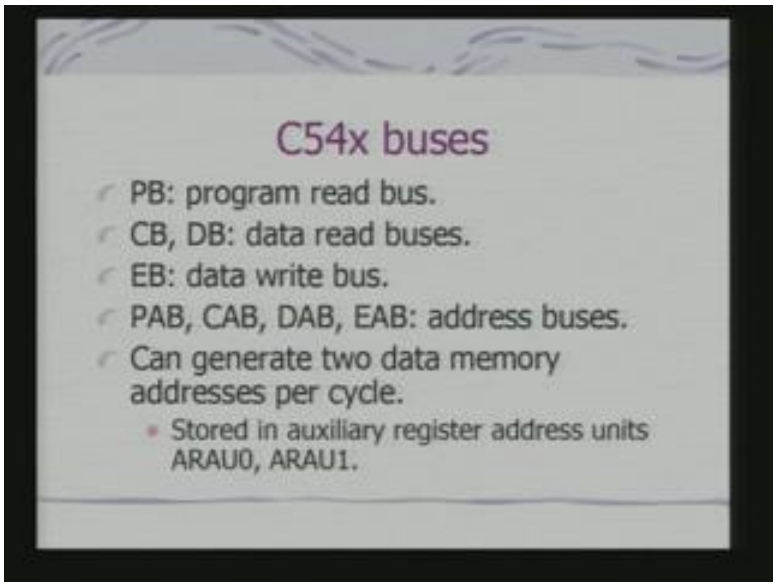
In fact, we shall discuss about this power down modes in more detail when we look at the power management aspects of different processor. Subsequently this is just listing that it is supports three power down modes.

IDLE1, IDLE2 and IDLE3.

In fact, this shuts down the CPU and on-chip peripherals and shuts down the chip completely.

I hope you remember the clock example we discuss with PIC. Where processor CPU can be shutdown, for the peripherals can work. So, similar kind of facilities are there with this processor.

(Refer Slide Time: 51:59)



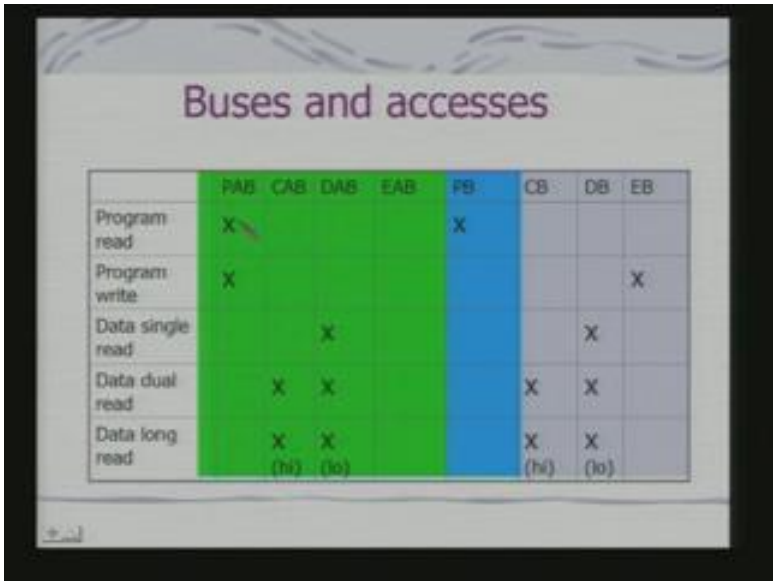
The most interesting feature of this is buses. The number of internal buses that it supports for accessing data in the program.

So, I have got program read bus, data read bus, data write bus. And these are all address buses. So, you can see that since I have got these addresses, address buses. All this operations can be actually overlapped in time.

And that is why if you have, if you remember the pipeline stages. I can have instruction pre-fetch, fetch as well as data fetch, data address. And then data fetch can be a different stages of the pipeline. Because I have got distinct buses to transfer the data and provide the address.

And this is a very interesting aspect of these DSP architectures, the number of internal buses that have been supported.

(Refer Slide Time: 53:00)



So, what we have got is if you look at the interestingly, when I am doing a program read. I have to provide address on this program address bus.

And I read the data on the program bus, this is what the instruction is coming along this bus.

Program write is again providing the address on the same bus because; obviously, will not be reading and writing on the same time. And if I am writing on to the program memory and I am using this bus.

Data single read is here the address and here I get the data. And data dual read is I can provide the address in these two buses. And I can get data in both these buses. And I can also have; so, this is what I am getting two operands on two buses simultaneously.

And I can also get the data long read. So, there are other bus operations and modes as well.

What I was interested to show you is that, how the different buses the processor uses.

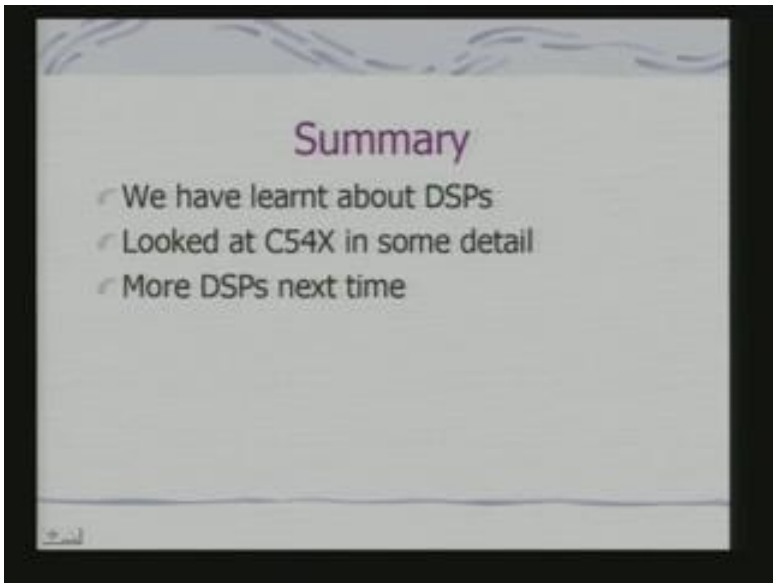
Because it gives a pretty efficiency in supporting what is called multiple memory accesses. Because you would like to have multiple data points, multiple data stored. Brought in multiplied and accumulator and this whole operation you would like to do in a proper sequence.

And because of your demands of the data you have got multiple buses over which these accesses can take place.

So, you have separated out the program access, you have separated out the two data accesses. And so, effectively you can have two distinct data memories and one program memory.

So, this more or less have brings as to the end of today's lectures we have learned about the basic features of digital signal processors.

(Refer Slide Time: 55:04)



We have looked at one of these chips C54X in some detail. And we shall look at other DSPs, some of the other DSPs in the next class.

.