Signal Processing Algorithms & Architecture
Dr. Anirban Dasgupta
Department of Electronics & Electrical Engineering
Indian Institute of Technology Guwahati
Lec 20: Signal Processor Components

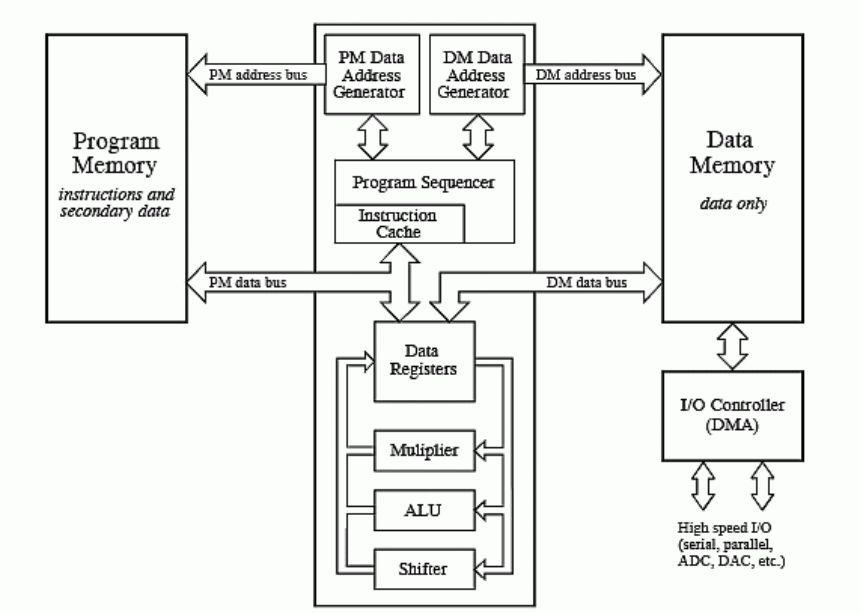
Hello everyone, welcome to a new lecture on signal processor components. This is Dr. Anirban Dasgupta and let us get started. So far, we have discussed the components in signal processing, including sensors, ADCs, DSPs, memory, and other elements. But here we are more focused on the components of the signal processor, such as what is inside this big signal processor, and you can see this diagram. So, there are a lot of things, and well, you may find this very complicated.

Those who have done a course on computer architecture might find this relatable but let us see a summary of what this is. So, the first thing is, like in Harvard architecture, as I said in the previous lecture. So, it has separate program memory and data memory. So, what is program memory? It is a set of instructions, such as "if I do convolution, it will be written as 'convolve this and this, then convolve this and this, then add this and this.

So, this set of instruction is in program memory. Also, you need to load the data from here and this is the data. This is actually where the signal segments are stored. Then we have buses. Now, what are buses? So, buses are basically what carry the data. Like in the real world, we say buses carry passengers. Similarly, here buses carry the data and address. So, you see, there is a data bus that is bidirectional, like from the processor to the memory as well as from the memory to the processor, the address bus is one way.

It is just giving the address location, and there are control buses also that are not shown here. Then there is something called data registers. So, data registers are somewhere we store intermediate results. There is something called an instruction cache, which will basically store the recent program to be executed or the recent instruction to be executed. There is something called an address generator. Here you see there are two address generators. Then there is a dedicated multiplier that will perform faster multiplication. Then we have split ALUs and DSPs, where each ALU can operate in parallel. There is a MAC unit that is not shown here, but the MAC unit typically refers to multiply and accumulate. There is a barrel shifter, which is like a circular shift.

There is a watchdog timer that will reset the processor if there are any problems or issues in the processing unit or operation. So, coming to the program and data memory. So, program memory is used to store the program, like when you write the code on the computer and then click on run or execute after you have connected your DSP. So, that code goes as binaries to the processor's program memory. And this is typically non-volatile; that is, I may want my DSP to work as a single kind of operation.



Say it will just filter the voice signal. So here my program is typically stored, and it should not be changed or erased. But now, these have often become volatile, and you can frequently rewrite the program and change it. But when you actually make a final product, like say this recorder which is recording this lecture, it is probably because the program is hard coded in a non-volatile ROM. Then data memory is for storing temporary data like signals that are coming and going and changing from time to time.

And this is typically volatile in nature, and it has to be because it is changing. So, whenever the DSP is powered off, you lose all the information unless you store it on a permanent drive. So, these are the three buses that I was discussing. So, the address bus is unidirectional from the processor to the memory or the peripherals, and it carries the addresses. And what is the address? Basically, if you want to find my space, that is typically an address.

You have to find my room number, and then the department and something like that. So, that is the address and that is the role of the address bus. The data bus is bidirectional. So, because I can send data back and forth, I can take the parcel and I can also return the parcel. So, that is a data bus that is bidirectional.

And then there is a control bus, which can be bidirectional as well as unidirectional, and it is basically managing the control signal that you now have to write or read, something like that. So, reading is one way, writing is another way, but the control bus will basically coordinate or manage the control signal. So, this is the data register, which is very small, and these are fast storage locations. Now, a register is for storing immediate results, and for example, if I want to do convolution. So, I require immediate data.

So, I will pick up the data from the data register because it is very near and the access time will be less. And in this way, it will not take much time to fetch data from somewhere in memory that is quite far away from the processor. So, data registers can be of different types, like general-purpose registers, and these are used for temporary operations, such as storing input values, output values, and immediate results. For example, if this is a filtering operation, the filter coefficients and your input signal value that you are going to convolve can be stored in your general-purpose register. Then there is an accumulator.

The accumulator is basically storing the output of the arithmetic operations. So, in one way, it is behaving like a GPR, like for convolution. So, you have to multiply two things, then add, then multiply again, and add. So, you can store these immediate results in the accumulator. Then there are some special-purpose registers, like the program counter, which holds the address of the next instruction that is going to be fetched.

These instructions are fetched sequentially typically. Then there are status registers that contain flags that represent the results of the most recent operation. And then, let's say there is an output of 0, so you can have a flag of 0, which is 1, that means there is a 0. Then there can be a divide by 0 flag, which can also indicate 1; there can be a carry flag and an overflow flag. Now, if you want to read more about this, you can go and check out some microcontroller or microprocessor courses where these will be given in detail.

Then there are pointer registers, which are used for managing pointers to memory locations, like the stack pointer and base pointers. Then index registers, which are used for storing the index in addressing modes. So, I can access the indices of arrays or buffers. Then there are shift registers which are used to perform bit shifting operations. This is very interesting, which is the instruction cache.

Now, cache, or instruction cache, is a small, fast memory located within your DSP. Now let me give you a very good analogy to understand this concept. So, say you have a book rack or bookshelf that many people have, which is very nice, and you have all your course books, like your signal processing books, your control system books, your embedded system books, and, what else, image processing books. And say you are studying, and when you want to study signal processing, or say you have your signal processing exam tomorrow. So, will you take out all the books from the rack and keep them on your table? No, but you would prefer to have all the signal processing books on your table because that is what you will need most.

So, if you put them back on the shelf every time, and then every time you want to read again you take them out, that will waste a lot of time and energy. So rather, you keep it on the table because you know that you will be frequently accessing that book, and that table is basically your instruction cache. So, it will store the recently fetched instructions

and frequently used instructions. So, in your phone, you have often seen the terms cache memory or clear your cache. So that is basically your cache.

Now that your table is filled, you can clear your cache so that you can have more space to keep more things in the cache memory. So, this is how you improve speed and efficiency by reducing the time it takes to access instructions from slower main memory, which is your book rack. And now, when the cache becomes full, you basically have to clear the cache; otherwise, you replace it with a new one, and the DSP will decide what to do. Then address generator, now the address generator is a specialized hardware that is responsible for calculating the memory address dynamically. Now one is memory allocation; suppose you make a big building or a big flat and you want to number them, so you can number them sequentially and then, when allocating addresses, say you have a department and some offices are filled while others are empty, so you want to allocate; therefore, you have to search for which addresses are empty. So, that is how you can allocate this address. So, then the addresses can be mapped in a much smarter way. and in a faster way. So, if you have multiple address generators, each data unit needs to be stored in some location, right? But there has to be a clear location where it has to stay.

Like students who are staying in a hostel, each of them is allotted a room, right? And how this allotment is done, maybe in a lottery system or in some sequential manner, but somebody has to do that allocation and that if the number of rooms is high, like 500, it will be very difficult for an individual to locate all the rooms. So, if you have multiple such allocators, that will be faster, and many DSPs have multiple address generators. So, it will receive some Bates address as the starting point in memory, and then it can compute the offset or index, like this is the first room, the base room, and then there are these many already allocated. So, what should be the next address to which I can send it? And it will incrementally calculate the next MRE address once you have been allotted a space.

So it can also support indirect addressing. Now, what is indirect addressing? Suppose you want to find, say, an address; I want to find that, okay, this villa, this Prakash villa, for example. So, where is this Prakash Villa? So, you ask someone, and that person will say, "Okay, after four blocks, that is Prakash Villa." So that is direct address.

You are finding the location directly. Indirect addressing means saying you do not know the location but you know who knows the location. So, you should say, "Sorry, I do not know where Prakash Villa is, but if you go to Raman Villa, say the guy in Raman Villa knows where Prakash Villa is because that Raman Villa guy is a friend of Prakash." Raman is a friend to Prakash. So, if you go to Raman Villa, I can show you that it is two from this row. So that way, you are doing indirect addressing.

The next is dedicated multiplier. Now, of course, you know what multiplication is? Well, you studied in your very young days, or childhood days. I would say that multiplication is repeated addition. If I want to multiply 16 by 16, I add them 16 times, right? So, add, then store in the accumulator, then add again, and do this until I reach the figure 16 and that will take 16 clock cycles if I do it sequentially. So, a dedicated multiplier does something in parallel so that you get the result in 1 clock cycle, and that is this specialized hardware for doing multiplication efficiently at high speed.

And this typically consists of an array of logic gates that can handle fixed-point and floating-point operations depending on the DSP design. So, the DSP can be a fixed-point design or a floating-point design. If you are not sure what a fixed point or floating point is, say that if you have an 8-bit register, you can store 0 to 255, considering all are integers. But if I say there is something negative, like I have something at 0.5, then in the integer part I can store something from 0 to 127, and the last bit I can use for a fractional part, like whether it is 0.5 or not. So, this point can be placed, so this is called Q format, Q 7.1; then if I use 2 for decimals, then Q 6.2. So, these are Q-formats. Now, if this is fixed like always, it will be 7 places of integer and 1 place of decimal; then this is called fixed point.

And if I can adjust or vary this point based on my operational needs, that is a floating point. The next is a split ALU, and the split ALU says that ALUs are basically what it is doing. It is performing the arithmetic and logical operations. So arithmetic operations you know are plus, minus, multiply, and divide. And I would say this is divided, and then you have your logical operations, which are "OR," "AND," "NOT," and also comparisons like "greater than," "less than," and "equals to.

So ALU performs arithmetic and logical operations and if you have multiple ALUs, then your processing can be sped up provided the operations are not sequential; they are independent of each other. So that is the work of a split ALU. So, a split ALU can perform arithmetic operations and then bitwise operations like AND, OR, NOT, and XOR, and in this manner, it can efficiently handle different data streams, which is beneficial in multi-threaded environments. Then the MAC operation, or the multiply-accumulate operation.

So, what it does is basically combine this multiplication and addition. So, in many operations in DSPs, first you need to multiply and then add to the accumulator. For example, if I have 1, 2, 3, and 4, 5, 6, and if I want to do a convolution or, say, a dot product kind of thing, what will you do? You have to multiply 1 by 4, then add this result to $2 \cdot 5 + 3 \cdot 6$. So first I will calculate this and say that this is my accumulator.

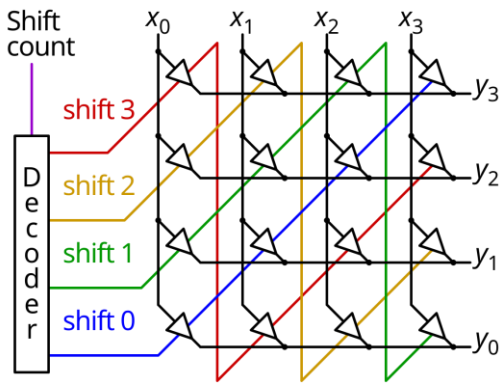
So, multiply and accumulate. Then I will multiply this and add it to the accumulator. So, this is

$$A + new\ A,$$

and then this whole thing becomes my accumulator now. And then I multiply this, and then this whole thing is added, and this becomes my new accumulator. So, any dot product operation can be performed in a MAC format. Now you see that multiplying and then accumulating 2 clock cycles, multiplying and accumulating 4 clock cycles, and multiplying and accumulating 6 clock cycles are wasted.

What if I can do it in one clock cycle? So that is the specialty of a MAC operator, which can perform this MAC operation in one clock cycle, or it is like a parallel dot product. Now, why is this used in DSPs? Because in signal processing, we are doing a lot of operations that are in the dot product form. For example, convolution—so if you see convolution, what are we doing? We have a kernel and a signal that are just shifting; then, say this is my kernel and this is my signal, and if I shift this, then next I shift this, next I shift this, next I shift this. So efficiently, if I do zero padding, what am I doing? I am doing the dot product or even if you see this as a matrix operation that I had shown, you multiply this row with this column, then this row with this column.

So, each is a dot product operation. By using the MAC operator, you can do this faster. So, convolution and correlation are very similar. FFT also, if you see, I have this DFT matrix and this signal, so it is also using this dot product kind of thing. And coming to the barrel shifter. So, first, if you recall your digital electronics class, you have studied shift registers.



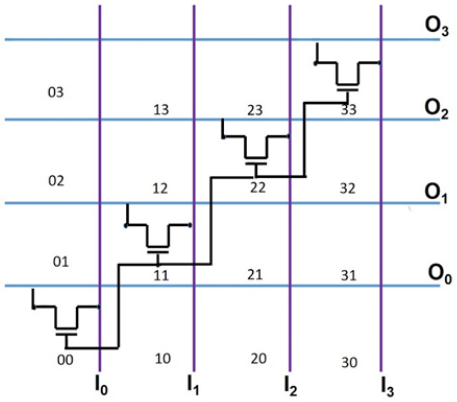
What is a shift register? It is a sequential circuit, and in one clock cycle, it will shift one bit. But a barrel shifter is a combinatorial circuit, and in one clock cycle, you can get your desired amount of shift, something like this. So, if you look at this truth table, you can understand. So, if I want a shift of 3, this will give me a shift of 3 directly in 1 clock cycle.

If I want a shift of 1, it will give a shift of 1. So, this is how the bits change. This is a combinatorial circuit. This is nothing but a decoder circuit where I can provide the shift count, and that shift count will be decoded, resulting in the shift you will get. So, this is

4X4 Barrel Shifter

Shift	O_0	O_1	O_2	O_3
0	I_0	I_1	I_2	I_3
1	I_3	I_0	I_1	I_2
2	I_2	I_3	I_0	I_1
3	I_1	I_2	I_3	I_0

Shift	O_0	O_1	O_2	O_3
0	1	0	0	0
1	I_3	I_0	I_1	I_2
2	I_2	I_3	I_0	I_1
3	I_1	I_2	I_3	I_0

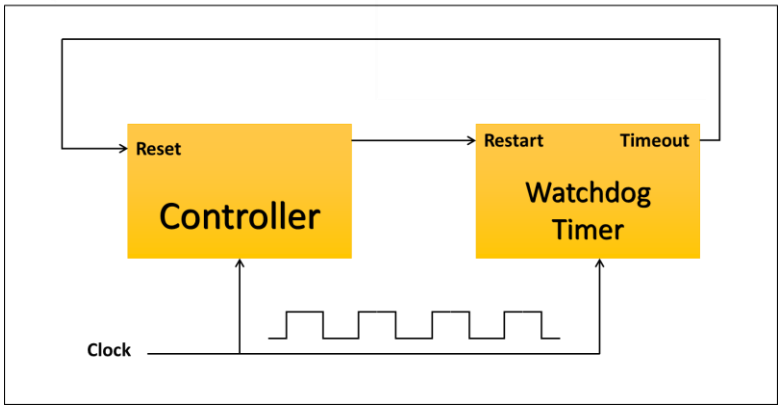


like shifting a barrel. So, just so you know, this is like a circular shift kind of thing.

So, this is basically about the barrel shifter, and then finally we will discuss another component, which I said is the watchdog timer. So, the watchdog timer is like a watchdog or a security guard, which will monitor if the processor has any malfunction or not. How will it check for a malfunction? So, whenever so it knows that at this time it should have a send me a result. If it does not send me a result, then there is a problem. It is like you tell the security guard that if I do not come out in, say, one hour, then there is some problem.

So, you just check up on me to see whether everything is fine or not. So, it does similar thing like it will periodically send a signal to the processor to indicate whether the system is working properly or not. If it is not receiving any such signal, that means there is a problem, and it will send a reset signal. So that the processor is reset within a set time frame and the operation will resume as if it had received an automated reboot. This often happens to our computers when we get an automated reboot, often because there is a system crash or something.

So, this is the pictorial representation of a watchdog timer.



Thank you so much. So, we will meet again. Have a nice day.