Overview of Digital Signal Processors

The programmable digital signal processors (PDSPs) are general purpose microprocessors designed specifically for digital signal processing applications. They contain special architecture and instruction set to execute computation -intensive DSP algorithms more efficiently. The programmable DSPs can be divided into two broad categories. They are i) general purpose digital signal processors and ii) Special purpose digital signal processors

General purpose digital signal processors: These are basically high-speed microprocessors with architecture and instruction sets optimized for DSP operations. They include fixed point processors such as Texas Instruments TMS320C5x, TMS320C5.4x and Motorola DSP563x and floating-point processors such as Texas instruments TMS3f.OC4x, TMS320C67xx and analog devices ADSP21 xxx.

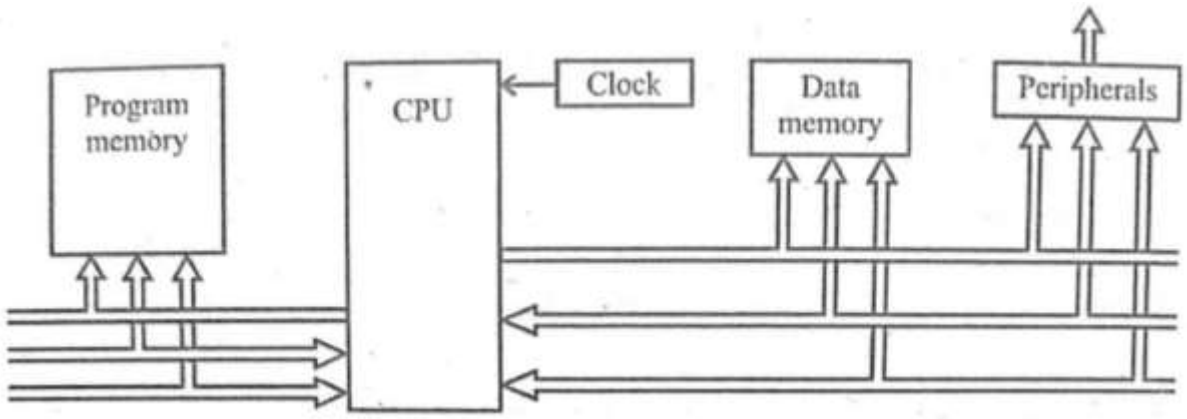
Special purpose digital signal processors: These types of processors consist of hardware i) designed for specific DSP algorithms such as FFr, ii) hardware designed for specific applications such as PCM and filtering. Examples for special purpose DSPs are Mi tel 's multi-channel telephony voice echo canceller (MT93001), FFT processor (PDSP 1651 SA,TM-44, TM-66) and programmable FIR filter (UPDSP 16256, Model3092).

Harvard Architecture

The term Harvard originated from the Harvard Mark 1 relay-based computer which stored instruction on punched tape and data in relay latches. The Harvard architectures physically separate men1ories for their instructions and data, requiring dedicated buses for each of them. Instructions and operands can therefore be fetched simultaneously.

Most of the DSP processors use a modified Harvard architecture with two or three memory buses· allowing access to filter coefficients and input signals in the same cycle. Since it possesses two independent bus systems, the Harvard architecture is capable of simultaneous reading an instruction code and reading or writing a memory or peripheral as part of the execution of the previous instruction. Since it has two memories, it is not possible for the CPU to mistakenly write codes into the program memory and therefore compute the code while it is executing.

However, it is less flexible. It needs two independent memory banks. These two resources are not interchangeable.



The modified Harvard architecture used DSPs multiport memory that has separate bus systems for program memory and data memory and input/output peripherals. It ·may also have multiple bus system for program memory alone or for data memory alone. These multiple bus system increases complexity of the CPU, but allow it to access several memory locations, simultaneously, thereby increasing the data throughput between memory and, CPU.

Pipelining

Most of the early microprocessors execute instructions entirely sequentially. After the execution of first instruction the next one starts. The problem with this is that it is extremely inefficient, since the second instruction must wait until all the steps of first instruction are completed. To improve the efficiency, advanced microprocessors and digital signal processors use an approach called pipelining in which different phases of operation and execution of instructions are carried out in parallel. That is in modem processors the first step of execution is performed on the first instruction, and then when the instruction passes to the next step, a new instruction is started. The steps in the pipeline are often called stages

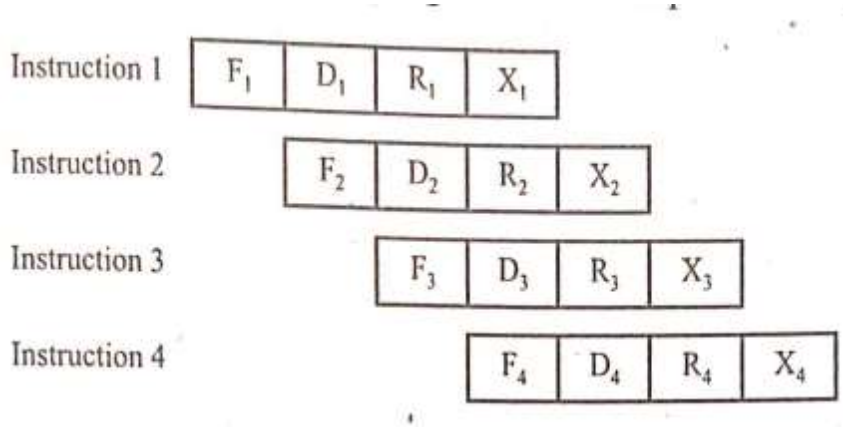
The basic action of any microprocessor can be broken down into a series of four simple steps. They are

1. **The Fetch phase(F)** in which the next instruction is fetched from the address stored in the program counter.
2. **The decode phase (D)** in which the instruction in the instruction register is decoded and the address in the program counter is incremented
3. **Memory read (R)** phase reads the data from the data buses and also writes data to the data buses.

4. **The Execute phase (X)** executes the instruction currently in the instruction register and also completes the write process.

Pipelining a processor means breaking down its instruction into a series of discrete pipeline stages which can be completed in sequence by specialized hardware. Because an instruction's Lifecycle consists of four distinct phases, the instruction execution process is divided into a sequence of four discrete pipeline stages, where each pipeline stage corresponds to a phase in the standard instruction Lifecycle. Note that the number of pipeline stages is referred to as the pipeline depth. So, a four-stage pipeline has a pipeline depth of four.

To understand the pipelining in a better way, let us assume that the number of stages is four and the execution time of an instruction is four nanoseconds. If we assume. the time taken for each stage in the instruction is equal, then the time - taken for each stage is one nanosecond. So, our original single-cycle processor's four-nanosecond execution process is now broken down into four discrete, sequential pipeline stages of one nanosecond each· in length. At the beginning of the first nanosecond, the first instruction enters the fetch stage. After that nanosecond is complete, the second nanosecond begins and the first instruction moves on to the decode stage while the second instruction enters the fetch stage. At the start of the third nanosecond, the first instruction advances to the ·memory read stage, the second instruction advances to the decode stage, and the third green instruction enters the fetch stage. At the fourth nanosecond, the first instruction advances to the execution stage, the second to the memory read stage, the third to the decode stage, and the fourth to the fetch stage. After the fourth nanosecond has fully elapsed and the fifth nanosecond starts, the first -instruction has passed from the pipeline and is now finished executing. Thus, we can say that at the end of four nanoseconds (= four clock cycles) the pipelined processor depicted below has completed one instruction. At start of the fifth nanosecond, the pipeline is now full and the processor can begin completing instructions at a rate of one instruction per nanosecond. This 1 instruction completion rate is a four-fold improvement over the single-cycle processor's completion rate of 0.25 instructions/ns (or 4 instruction every 16 nanoseconds).



Pipelining leads to dramatic improvements in system performance. The more stages that we can break the pipeline into, the more theoretical speed we can get from it.

Multiply Accumulate Unit (MAC)

The Multiply-Accumulate (MAC) operation is the basis of many digital signal processing algorithms, notably digital filtering. The term "digital filter" refers to an algorithm by which a digital signal or sequence of numbers is transformed into another sequence of numbers termed the output digital signal. Digital filter involves signals in the digital domain (discrete-time signals) and are used extensively in applications such as digital image processing, pattern recognition, and spectral analysis. In general FIR filters are preferred in lower order solutions, and since they do not employ feedback, they exhibit naturally bounded response. They are simpler to implement, and require one RAM location and one coefficient for each order.

For FIR filters the output of the filter is given by

$$y(n) = \sum_{k=0}^{N-1} x(k)h(n-k)$$

where x(n) is the input to the filter, h(n) is the impulse response of the filter and y(n) is output of the filter. The output of an FIR filter is simply a finite length weighted sum of the present and previous inputs to the filter. Hence to perform filtering through above equation, the minimum requirement is to quickly multiply two values, and add the result. To make it possible, a fast dedicated hardware MAC, using either fixed point or floating-point arithmetic is mandatory. Characteristics of a typical fixed-point MAC include

I. 16 x 16 bit 2's complement inputs ..

2. 16 x 16 bit multiplier with 32-bit product in 25 ns

3. 32140 bit accumulator

The Multiply-Accumulate (MAC) Function.

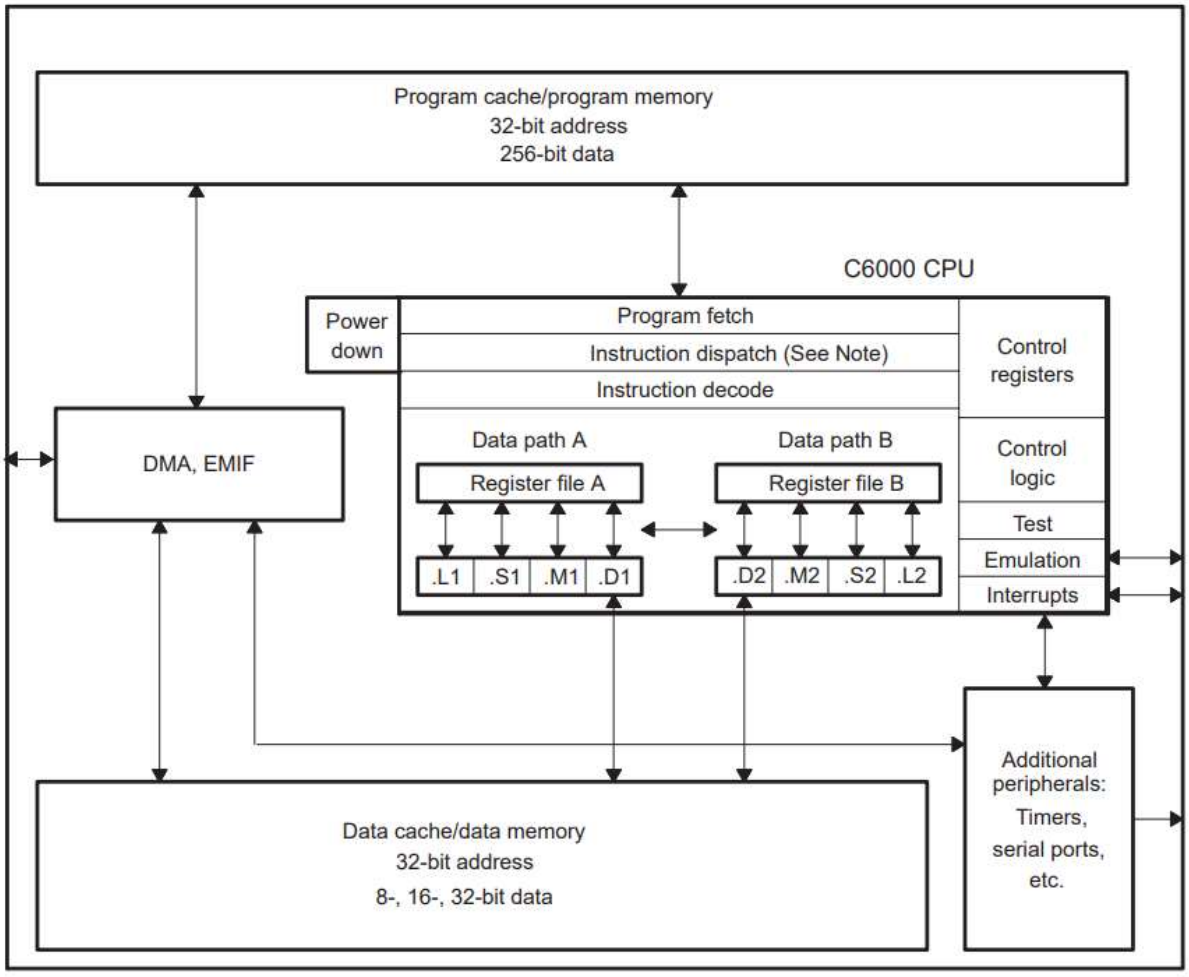
The MAC speed applies both to finite impulse response (FIR) and infinite impulse response (IIR) fi1ters. The complexity of the filter response dictates the number MAC operations required per sample period.

A multiply-accumulate step performs the following:

- Reads a 16-bit sample data (pointed to by a register)
- Increments the sample data-pointer by 2
- Reads a. 16-bit coefficient (pointed to by another register)
- Increments the coefficient register pointer by 2
- Sign Multiply (16-bit) data and coefficient 'to yield a 32~bit resu1t
- Adds the result to the contents of a 32-bit register pair for accumulate.

TMS320C67xx DSP Architecture

Given figure is the block diagram for the C67x DSP. The C6000 devices come with program memory, which, on some devices, can be used as a program cache. The devices also have varying sizes of data memory. Peripherals such as a direct memory access (DMA) controller, power-down logic, and external memory interface (EMIF) usually come with the CPU, while peripherals such as serial ports and host ports are on only certain devices.



Central Processing Unit (CPU)

The C67x CPU, is common to all the C62x/C64x/C67x devices.

The CPU contains:

Program fetch unit

Instruction dispatch unit

Instruction decode unit

Two data paths, each with four functional units

32 32-bit registers

Control registers

Control logic

Test, emulation, and interrupt logic

The program fetch, instruction dispatch, and instruction decode units can deliver up to **eight 32-bit instructions** to the functional units every CPU clock cycle. The processing of instructions occurs in each of the two data paths (A and B), each of which contains **four functional units** (.L, .S, .M, and .D) and 16 32-bit general-purpose registers. A control register file provides the means to configure and control various processor operations. To understand how instructions are fetched, dispatched, decoded, and executed in the data path, see

Internal Memory

The C67x DSP has a 32-bit, byte-addressable address space. Internal (on-chip) memory is organized in separate data and program spaces. When off-chip memory is used, these spaces are unified on most devices to a single memory space via the external memory interface (EMIF).

The C67x DSP has two 32-bit internal ports to access internal data memory. The C67x DSP has a single internal port to access internal program memory, with an instruction-fetch width of 256 bits.

Memory and Peripheral Options

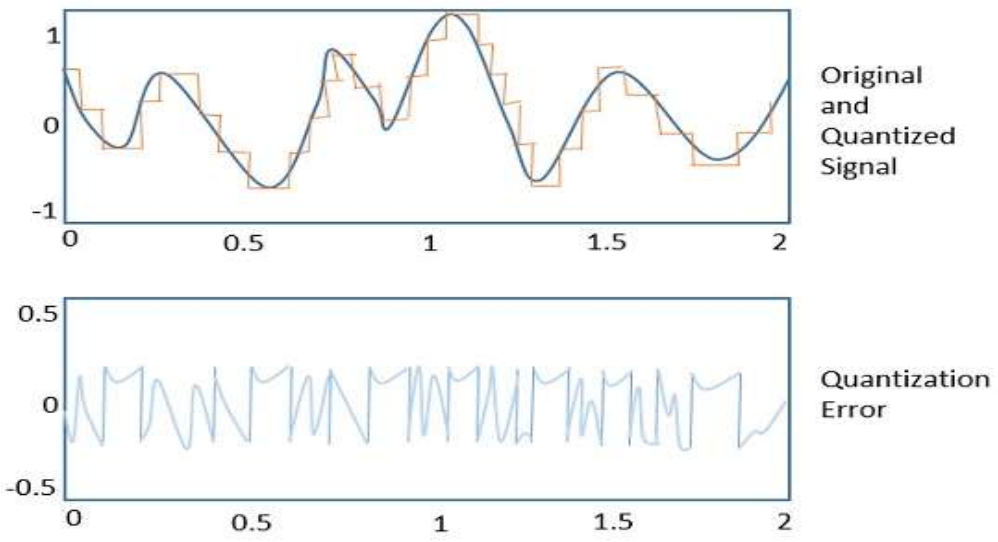
A variety of memory and peripheral options are available for the C6000 platform:

- Large on-chip RAM, up to 7M bits
- Program cache
- 2-level caches
- 32-bit external memory interface supports SDRAM, SBSRAM, SRAM, and other asynchronous memories for a broad range of external memory requirements and maximum system performance.

- DMA Controller (C6701 DSP only) transfers data between address ranges in the memory map without intervention by the CPU. The DMA controller has four programmable channels and a fifth auxiliary channel.
- EDMA Controller performs the same functions as the DMA controller. The EDMA has 16 programmable channels, as well as a RAM space to hold multiple configurations for future transfers.
- HPI is a parallel port through which a host processor can directly access the CPU's memory space. The host device has ease of access because it is the master of the interface. The host and the CPU can exchange information via internal or external memory. In addition, the host has direct access to memory-mapped peripherals.
- Expansion bus is a replacement for the HPI, as well as an expansion of the EMIF. The expansion provides two distinct areas of functionality (host port and I/O port) which can co-exist in a system. The host port of the expansion bus can operate in either asynchronous slave mode, similar to the HPI, or in synchronous master/slave mode. This allows the device to interface to a variety of host bus protocols. Synchronous FIFOs and asynchronous peripheral I/O devices may interface to the expansion bus.
- McBSP (multichannel buffered serial port) is based on the standard serial port interface found on the TMS320C2000 and TMS320C5000 devices. In addition, the port can buffer serial samples in memory automatically with the aid of the DMA/EDNA controller. It also has multichannel capability compatible with the T1, E1, SCSA, and MVIP networking standards.
- Timers in the C6000 devices are two 32-bit general-purpose timers used for these functions:
    - Time events
    - Count events
    - Generate pulses
    - Interrupt the CPU
    - Send synchronization events to the DMA/EDMA controller.
- Power-down logic allows reduced clocking to reduce power consumption. Most of the operating power of CMOS logic dissipates during circuit switching from one logic state to another. By preventing some or all of the chip's logic from switching, you can realize significant power savings without losing any data or operational context.

Finite Word length Effects:

- In the design of FIR Filters, the filter coefficients are determined by the system transfer functions. These filters co-efficient are quantized/truncated while implementing DSP System because of finite length registers.
- Only Finite numbers of bits are used to perform arithmetic operations. Typical word length is 16 bits, 24 bits, 32 bits etc.
- This finite word length introduces an error which can affect the performance of the DSP system.
- The main errors are
    - Input quantization error
    - Co-efficient quantization error
    - Overflow & round off error (Product Quantization error)
- The effect of error introduced by a signal process depend upon number of factors including the.
    - Type of arithmetic
    - Quality of input signal
    - Type of algorithm implemented
- For any system, during its functioning, there is always a difference in the values of its input and output. The processing of the system results in an error, which is the difference of those values. The difference between an input value and its quantized value is called a Quantization Error.



Input quantization error

- The conversion of continuous-time input signal into digital value produces an error which is known as input quantization error. This error arises due to the representation of the input signal by a fixed number of digits in A/D conversion process

The quantization error arises when a continuous signal is converted into digital value, the quantization error is given by

$$e(n) = x_q(n) - x(n) \tag{7.37}$$

where $x_q(n)$ = sampled quantized value and
$x(n)$ = sampled unquantized value.

Depending on the way in which $x(n)$ is quantized different distributions of quantization noise may be obtained. If rounding of a number is used to get $x_q(n)$ then the error signal satisfies the relation

$$\frac{-q}{2} \leq e(n) \leq \frac{q}{2} \tag{7.38}$$

because the quantized signal may be greater or less than actual signal.

For example, let $x(n) = (0.70)_{10} = (0.10110011...)_2$

add

After rounding $x(n)$ to 3 bits we have

$$\begin{aligned} x_q(n) &= 0.101 \\ &\quad\ \ 1 \quad \text{add} \\ &= \underline{0.110} \\ &= (0.75)_{10} \end{aligned}$$

Now the error

$$e(n) = x_q(n) - x(n) = 0.05$$

which satisfies the inequality.

The probability density function $p(e)$ for roundoff error and quantization characteristic with rounding are shown in Fig. 7.4a and Fig. 7.4b respectively.

The other type of quantization can be obtained by truncation. In truncation the signal is represented by the highest quantization level that is not greater than the signal. Therefore, in two's complement truncation, the error $e(n)$ is always negative and satisfies the inequality.

$$-q \leq e(n) < 0 \tag{7.39}$$

The quantizer characteristicd for truncation and probability density function $p(e)$ for two's complement truncation is shown in Fig. 7.5a and Fig. 7.5b respectively.

From Fig. 7.4 and Fig. 7.5 it is clear that the quantization error means value is 0 for rounding and $-q/2$ for two's complement truncation.

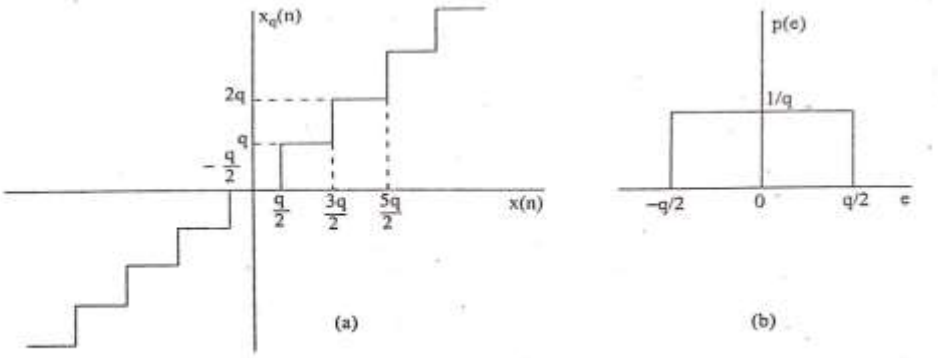


Fig. 7.4 (a) Quantizer characteristics with rounding (b) Probability density function for roundoff error.

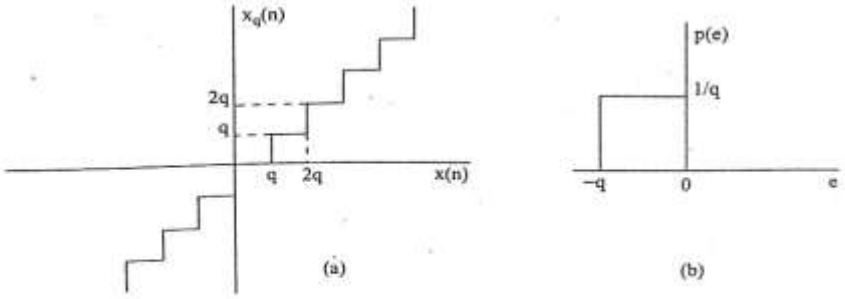


Fig. 7.5 (a) Quantizer characteristics with two's-complement truncation (b) Probability density function of truncation error.

Product Quantization error.

In fixed-point arithmetic the product of two b bit numbers results in numbers $2b$ bits long. In digital signal processing applications, it is necessary to round this product to a b-bit number, which produce an error known as product quantization error or product roundoff noise.

The model for fixed point roundoff noise following a multiplication is shown in Fig. 7.9. The multiplication is modeled as an infinite precision multiplier followed by an adder where roundoff noise is added to the product so that overall result equals some quantization level. The roundoff noise sample is a zero mean random variable with a variance $\frac{2^{-2b}}{12}$ where b is the number of bits used to represent the variables.

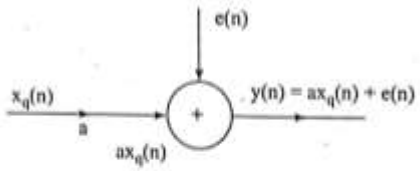


Fig. 7.9 Fixed point product roundoff noise model.

In order to model the effects of rounding due to multiplication in digital filter, certain assumptions must be made.

1. For any n, the error sequence $e(n)$ is uniformly distributed over the range $\frac{-q}{2}$ and $\frac{q}{2}$. This implies that mean value of $e(n)$ is zero and its variance is $\sigma_e^2 = \frac{2^{-2b}}{12}$
2. The error sequence $e(n)$ is a stationary white noise sequence.
3. The error sequence $e(n)$ is uncorrelated with the signal sequence $x(n)$. Thus each noise source is modeled as a discrete stationary white random process with a power density spectrum of $\frac{2^{-2b}}{12}$.

Based on the above model the block diagram of a single pole, IIR system represented by the following difference equation.

$$y(n) = a_1 y(n-1) + x(n) \tag{7.52}$$

can be drawn as shown in Fig. 7.10 a). In this diagram finite precision multiplier has been replaced by an ideal multiplier and additional roundoff noise $e(n)$.

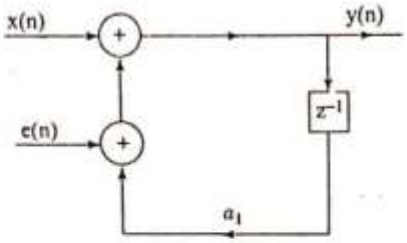


Fig. 7.10 a) Quantization noise model for a first order system.

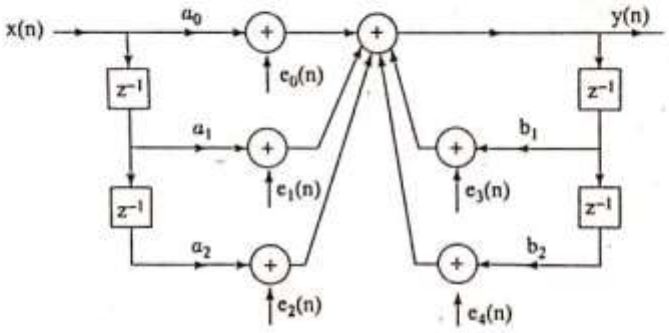


Fig. 7.10 b) Quantization noise model for a second-order system with five noise sources.

Fig.7.10 b) shows a block diagram for a second order direct form digital filter where each finite precision multiplier has been replaced by an ideal multiplier and an additive roundoff noise.

Since all the noise sources are added at the same point in the filter, all these sources can be replaced by single noise source $e(n) = e_1(n) + e_2(n) \ldots + e_4(n)$ with zero mean and variance $\sigma^2 = \sigma_0^2 + \sigma_1^2 \ldots + \sigma_4^2 = 5 \times \frac{2^{-2b}}{12}$ as shown in Fig. 7.11.

The second order filter of Fig. 7.10 b) can be realized as a cascade of two first order systems as shown in Fig. 7.12.

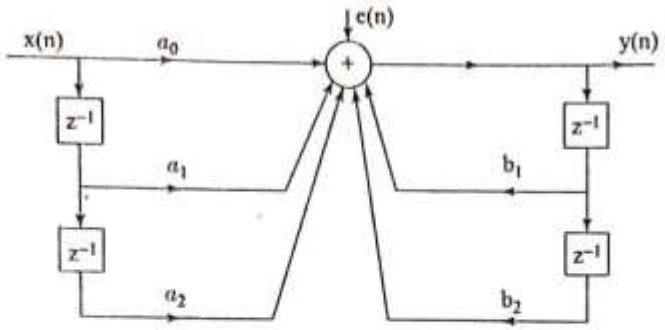


Fig. 7.11 Quantization noise model for a second order system with a single noise source.

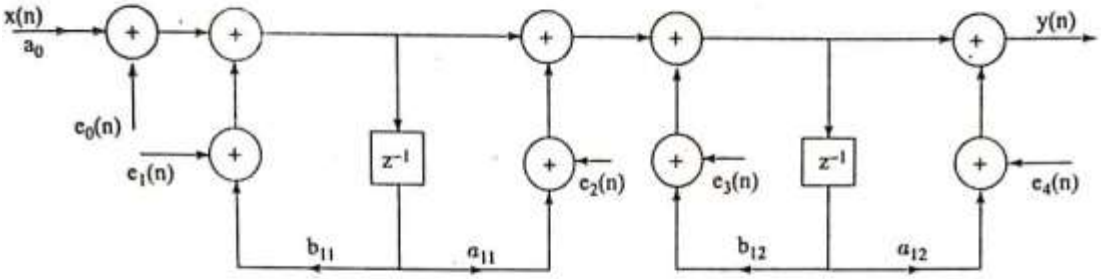


Fig. 7.12 (a) Quantization noise model of a cascade section.

There are five noise sources in this realization. The noise sources are added at different points and they do not see the same noise transfer function, that is the transfer function from the noise source to the filter output. Using assumption 2, the total variance can be obtained by adding the individual variances.

Consider the kth noise source $e_k(n)$. If $h_k(n)$ is the filter's impulse response from the noise source to the filter output, the response due to noise source $e_k(n)$ can be obtained by convolution as

$$\epsilon_k(n) = \sum_{m=0}^{n} h_k(m) e_k(n-m)$$

The variance of $\epsilon_k(n)$ is

$$\sigma_{0k}^2 = E\left[\sum_{m=0}^{n} h_k(m) e_k(n-m) . \sum_{l=0}^{n} h_k(l) e_k(n-l)\right]$$

$$= \sum_{m=0}^{n}\sum_{l=0}^{n} h_k(m) h_k(l) E\left[e_k(n-m) e_k(n-l)\right]$$

(See example 9.3)

$$= \sum_{m=0}^{n}\sum_{l=0}^{n} h_k(m) h_k(l) \delta(l-m) \sigma_e^2$$

$$\delta(l-m) = 1 \quad \text{for } l = m$$
$$= 0 \quad \text{for } l \neq m$$

$$= \sigma_e^2 \sum_{m=0}^{n} h_k^2(m) \qquad (7.53)$$

where $\sigma_e^2 = \frac{2^{-2b}}{12}$.

For the IIR filter, the impulse response approaches to zero as m tends to infinity. Then the steady state noise variance can be obtained by tending m to infinity.

$$\sigma_{0k}^2 = \sigma_e^2 \sum_{m=0}^{\infty} h_k^2(m) \tag{7.54}$$

Now the total steady state noise variance σ_0^2 is

$$\sigma_0^2 = \sum_k \sigma_{0k}^2 \tag{7.55}$$

The Eq.(7.54) can be written as

$$\sigma_{0k}^2 = \sigma_e^2 \frac{1}{2\pi j} \oint_c H_k(z) H_k(z^{-1}) z^{-1} dz \tag{7.56}$$

where $H_k(z)$ is defined as the noise transfer function, that is, the transfer function from the noise source to the filter output.

Coefficient quantization error

In the design of a digital filter the coefficients are evaluated with infinite precision. But when they are quantized, the frequency response of the actual filter deviates from that which would have been obtained with an infinite word length representation and the filter may actually fail to meet the desired specifications. If the poles of the desired filter are close to the unit circle, then those of the filter with quantized coefficients may lie just outside the unit circle.

Example 7.8 Consider a second order IIR filter with

$$H(z) = \frac{1.0}{(1 - 0.5z^{-1})(1 - 0.45z^{-1})}$$

find the effect on quantization on pole locations of the given system function in direct form and in cascade form . Take $b = 3$ bits.

Solution

Direct Form I

We can write $H(z) = \dfrac{1}{(1 - 0.95z^{-1} + 0.225z^{-2})}$

$$(0.95)_{10} = (0.1111001...)_2$$
$$(-0.95)_{10} = (1.1111001...)_2$$

After truncation we have $(1.111)_2 = -0.875$. Similarly

$$(0.225)_{10} = (0.001110...)_2$$

After truncation we have $(0.001)_2 = 0.125$

$$\text{So } H(z) = \frac{1}{(1 - 0.875z^{-1} + 0.125z^{-2})}$$

Cascade form

$$H(z) = \frac{1}{(1 - 0.5z^{-1})(1 - 0.45z^{-1})}$$
$$(-0.5)_{10} = (1.100)_2$$
$$(-0.45)_{10} = (1.01110\ldots)_2$$

Types of number representation:

There are two common forms that are used to represent the numbers in a digital or any other digital hardware.

1. Fixed point representation
2. Floating point representation

Explain the various formulas of the fixed-point representation of binary numbers.

1. Fixed point representation

- In the fixed-point arithmetic, the position of the binary point is fixed. The bit to the right represents the fractional part of the number and to those to the left represents the integer part.
- For example, the binary number 01.1100 has the value 1.75 in decimal. $(0*2^{1)} + (1*2^{0})$ + $(1*2^{-1})$ + $(1*2^{-2})$ + $(0*2^{-3})$ = 1.75

In general, we can represent the fixed-point number 'N' to any desired accuracy by the series

$$N = \sum_{i=n_1}^{n_2} C_i r^i$$

Where, r is called as radix

- If r=10, the representation is known as decimal representation having numbers from 0 to 9. In this representation the number

$$30.285 = \sum_{i=-3}^{1} C_i 10^i$$

$= (3*10^{1})+ (0*10^{0})+ (2*10^{-1})+(8*10^{-2})+(5*10^{-3})$

- If r=2, the representation is known as binary representation with two numbers 0 to 1.
  - For example, the binary number

110.010 = $(1*2^{2}) + (1*2^{1}) + (0*2^{0}) + (0*2^{-1}) + (1*2^{-2}) + (0*2^{-3})$ = 6.25

Examples:

Convert the decimal number 30.275 to binary form

0.55 * 2 ➔1.10 ➔1
0.10 * 2 ➔0.20 ➔0
0.20 * 2 ➔0.40 ➔0
0.40 * 2 ➔0.80 ➔0
0.80 * 2 ➔1.60 ➔1
0.60 * 2 ➔1.20 ➔1
0.20 * 2 ➔0.40 ➔0

2	30	
2	15	--0
2	7	--1
2	3	--1
	1	--1

$(30.275)_{10} = (11110.01000110)_2$

0.275 * 2 ➔0.55 ➔0

In fixed point arithmetic =, the negative numbers are represented by 3 forms.

1. Sign-magnitude form
2. One's complement form
3. Two's complement form

Sign-magnitude form:

- Here an additional bit called sign bit is added as MSB.
  - If this bit is zero → It is a positive number
  - If this bit is one → It is a positive number
- For example
  - 1.75 is represented as 01.110000.
  - -1.75 is represented as 11.110000

One's complement form:

- Here the positive number is represented same as that in sign magnitude form.
- But the negative number is obtained by complementing all the bits of the positive number
- For eg: the decimal number -0.875 can be represented as
  - $(0.875)_{10} = (0.111000)_2$
  - $(-0.875)_{10} = (1.000111)_2$

0.111000
↓↓↓↓↓↓ (Complement each bit)
1.000111

1.3 Two's complement form:

- Here the positive numbers are represented as same in sign magnitude and one's complement form.
- The negative numbers are obtained by complementing all the bits of the positive number and adding one to the least significant bit

$(0.875)_{10} = (0.111000)_2$
↓ ↓↓↓↓↓↓ (Complement each bit)
1.000111
\+ 1
1.001000

$(-0.875)_{10} = (1.001000)_2$

Examples:

Find the sign magnitude, 1's complement, 2's complement for the given numbers.

1. -7/32
2. -7/8
3. 7/8

1. $-\frac{7}{32}$

0.21875 * 2 → 0.43750 → 0
0.43750 * 2 → 0.87500 → 0
0.87500* 2 → 1.750000 → 1
0.75* 2 → 1.50 → 1
0.50* 2 → 1.00 → 1

$-\frac{7}{32} = (-0.21875)_{10} = (1.00111)_2$

Sign magnitude form = 1.00111
1's complement form = 1.11000
2's complement form = 1.11001

2. $-\frac{7}{8}$

0.875 * 2 → 1.75 → 1
0.750 * 2 → 1.500 → 1
0.500 * 2 → 1.000 → 1

$-\frac{7}{8} = (-0.875)_{10} = (0.111)_2$

Sign magnitude form = 0.111
1's complement form = 1.000
2's complement form = 1.001

3. $+\frac{7}{8}$

Sign magnitude form = 0.111
1's complement form = 0.111
2's complement form = 0.111

Addition of two fixed point numbers:

- **Add $(0.5)_{10} + (0.125)_{10}$**

$(0.5)_{10} = (0.100)_2$
$(0.125)_{10} = (0.001)_2$
$(0.101)_2 = (0.625)_{10}$

➢ Addition of two fixed point numbers causes an overflow.

For example
$(0.100)_2$
$(0.101)_2$
$(1.001)_2 = (-0.125)_{10}$ in sign magnitude form

Subtraction of two fixed point numbers:

➢ Subtraction of two numbers can be easily performed easily by using two's complement representation.

Subtract 0.25 from 0.5

0.25 * 2 ➔0.50 ➔0
0.50 * 2 ➔1.00 ➔1
0.00 * 2 ➔0.00 ➔0

Sign magnitude form = $(0.010)_2$
1's complement form = $(1.101)_2$
2's complement form = $(1.110)_2$

$(0.5)_{10} = (0.100)_2$
$-(0.25)_{10} = (1.110)_2$ ➔Two's complement of -0.25
$(10.010)_2$

Here the carry is generated after the addition. Neglect the carry bit to get the result in decimal. $(0.010)_2 = (0.25)_{10}$

Subtract 0.5 from 0.25

0.5 * 2 ➔1.00 ➔1
0.00 * 2 ➔0.00 ➔0
0.00 * 2 ➔0.00 ➔0

Sign magnitude form = $(0.100)_2$
1's complement form = $(1.011)_2$
2's complement form = $(1.100)_2$

$(0.25)_{10} = (0.010)_2$
$-(0.5)_{10} = (1.100)_2$
$(1.110)_2$

Here the carry is not generated after the addition. So the result is negative.

Compare floating point with fixed point arithmetic.

Sl.No	Fixed point arithmetic	Floating point arithmetic
1	Fast operation	Slow operation
2	Relatively economical	More expensive because of costlier hardware
3	Small dynamic range	Increased Dynamic range
4	Round off errors occurs only for addition	Round off errors can occur with addition multiplication
5	Overflow occur in addition	Overflow does not arise
6	Used in small computers	Used in large general-purpose computers.

Fixed Point Digital Signal Processor

In a fixed-point digital signal processor, every number can be specified through a minimum of 16 bits, even though a different length can be utilized. The number can be represented with different patterns. The fixed-point means that the fractional point position can be assumed to be fixed and to be identical for the operands as well as the operation result.

Fixed point processors are used in different flexible embedded applications because it uses low power and less cost. The fixed-point digital signal processor are; TI's TM320C54x, ADI DSP BF53X, TM320C55x, TM320C64x, TM320C62x and Motorola MSC810x.

Floating Point Digital Signal Processor

Floating-point digital signal processors mainly use a minimum of 32 bits to store every value. The distinct feature of floating-point DSP is that the signified numbers are not spaced uniformly. Floating-point digital signal processors can simply process the fixed-point numbers, a requirement to implement counters & signals which are received from the analog to digital converter and transmitted to the digital to analog converter.

For both the operations of fixed-point and floating-point DSPs, SHARC DSPs are simply designed, optimized & executed with equivalent efficiency. As compared to fixed-point DSPs, the programs of floating-point DSPs are simple, however, they are normally very expensive and power consumption is also more. The types of floating-point DSPs are TI's TMS320c67x and ADI ADSP 2116x/2126x.

Difference between Digital Signal Processor and Microprocessor

Digital Signal Processor	Microprocessor
It is a specialized microprocessor chip	It is a computer processor
DSPs are extensively used in telecommunications, audio signal processing, digital image processing, etc	Microprocessors are used in PCs for text editing, computation, multimedia display & communication over the Internet.
In DSP, instruction can be simply executed in a single CLK cycle.	The microprocessor uses several clock cycles for one instruction execution.
Parallel execution can be achievable	Sequential execution is possible.
DSP is suitable for the operation of array processing.	It is suitable for general-purpose processing.
Addressing modes used in this processor are direct & indirect.	Addressing modes used in microprocessors are direct, immediate, register indirect, indirect register, etc.
Address generation can be possible by combining program sequencers & DAGs.	The program counter or PC can be incremented to produce an address sequentially.
It includes three separate computational units: MAC, ALU & Sifter.	It includes simply the main unit like ALU.
The program flow can be controlled by an instruction register & program sequencer.	Program counter can control the execution flow.
It includes separate data & program memories.	It does not have separate memories.
In DSP, several operands are fetched at once.	In a microprocessor, the operand can be fetched serially.
In DSP, address & data bus are multiplexed	In a microprocessor, address & data bus are not multiplexed.

Applications Using Digital Signal Processor (DSP)

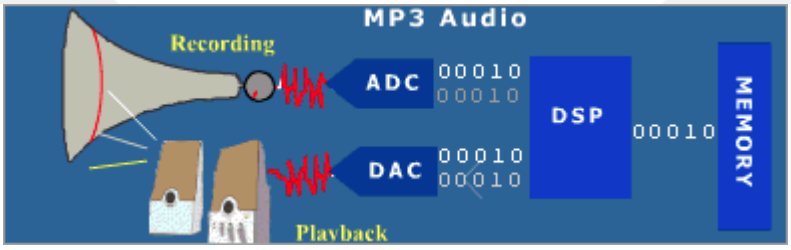
DSP is used in many modern applications. In today's world, digital devices have become indispensable as almost all our daily life gadgets are run and monitored by digital processors. The ease of storage, speed, security, and quality are the main value add.

MP3 Audio Player

Music or audio is recorded and the Analog signals are captured. ADC converts the signal to a digital signal. The digital processor receives the digitized signal as input, processes it, and stores it.

During playback, the digital processor decodes the stored data. DAC converter converts the signal to analog for human hearing. The digital processor also improves quality by improving volume, reducing noise, equalization, etc.

MP3 Audio player working model:



Computers and Laptop

The latest computers and laptops with digital processors are more flexible, faster, of better quality, and better portability. The digital signals from a computer are sent to the graphics card and are transmitted through a cable to a digital display. The graphics card converts the digital signals to analog signals and transfers them to an analog display for human viewing.

Smart Phones

The smartphones, IPAD, iPods, etc. are all digital appliances that have a processor that takes inputs from users and converts them to digital form, processes them, and displays the output in a human-understandable form.

Consumer Electronic gadgets

Gadgets like washing machines, microwave ovens, refrigerators, etc are all digital appliances that we use in our daily lives.

Automobile Electronic gadgets

The GPS, music player, dashboard, etc. are all digital processor dependant gadgets that are found in automobiles.