**DSP Algorithms and Architecture**

**UNIT-1**

**Introduction to Digital Signal Processing**

## Syllabus:-

**INTRODUCTION TO DIGITAL SIGNAL PROCESSING:** Introduction, A Digital Signal- Processing System, The Sampling Process, Discrete Time Sequences, Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT), Linear Time-Invariant Systems, Digital Filters, Decimation and Interpolation. **5 Hours**

### TEXT BOOK:

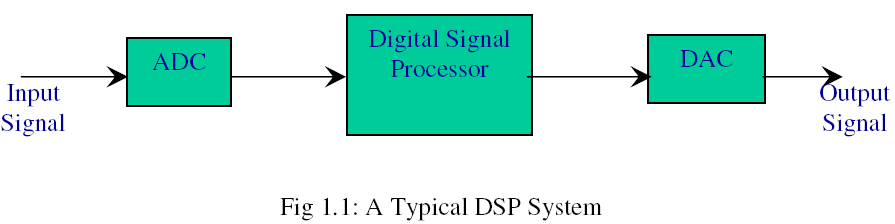
* **“Digital Signal Processing”**, Avatar Singh and S. Srinivasan, Thomson Learning, 2004.

### REFERENCE BOOKS:

* **Digital Signal Processing: A practical approach**, Ifeachor E. C., Jervis B. W Pearson- Education, PHI/ 2002
* **“Digital Signal Processors”**, B Venkataramani and M Bhaskar TMH, 2002
* **“Architectures for Digital Signal Processing”**, Peter Pirsch John Weily, 2007

### What is DSP?

DSP is a technique of performing the mathematical operations on the signals in digital domain. As real time signals are analog in nature we need first convert the analog signal to digital, then we have to process the signal in digital domain and again converting back to analog domain. Thus ADC is required at the input side whereas a DAC is required at the output end. A typical DSP system is as shown in figure 1.1.



### Need for DSP

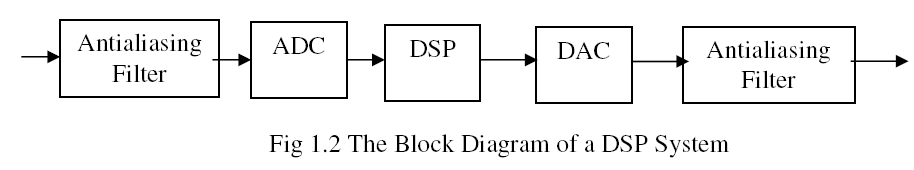
Analog signal Processing has the following drawbacks:

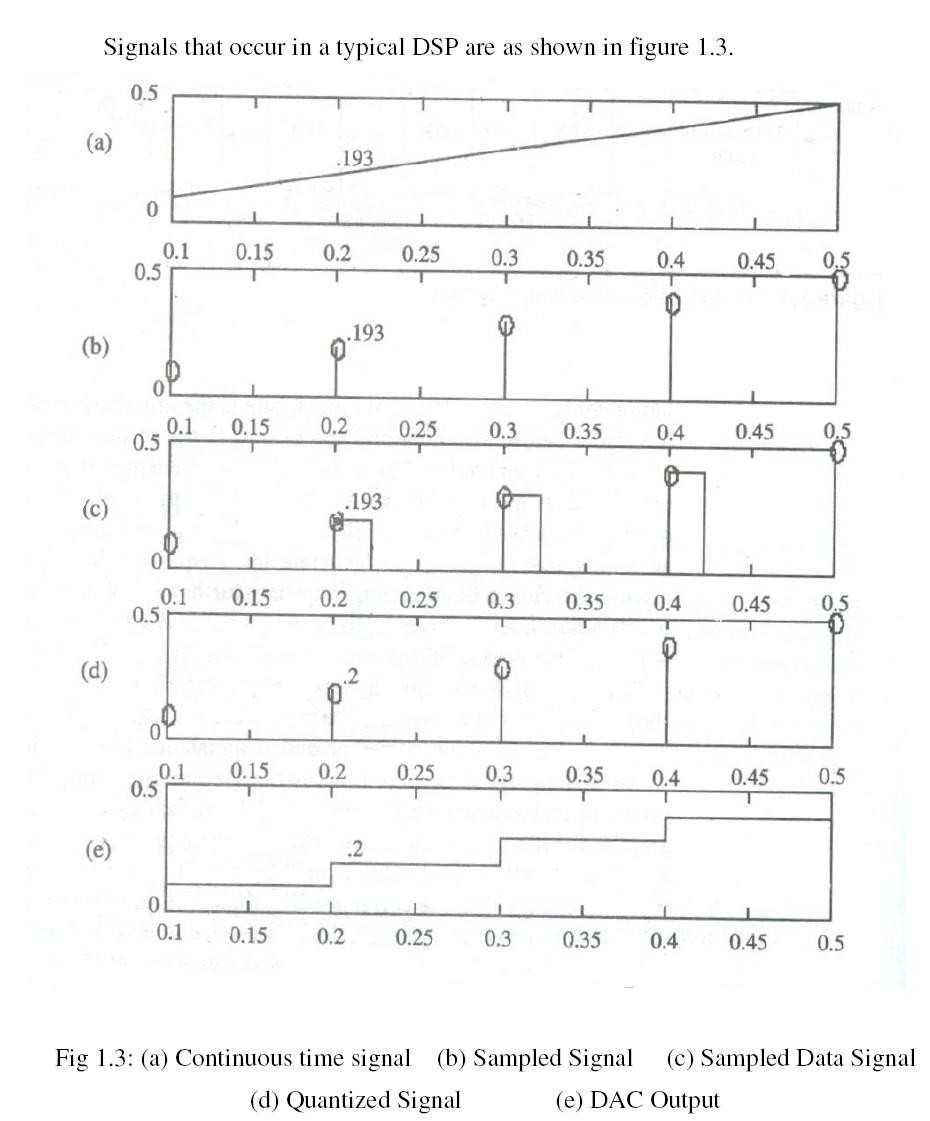
* + - They are sensitive to environmental changes
    - Aging
    - Uncertain performance in production units
    - Variation in performance of units
    - Cost of the system will be high
    - Scalability

If Digital Signal Processing would have been used we can overcome the above shortcomings of ASP.

### A Digital Signal Processing System

A computer or a processor is used for digital signal processing. Anti aliasing filter is a LPF which passes signal with frequency less than or equal to half the sampling frequency in order to avoid Aliasing effect. Similarly at the other end, reconstruction filter is used to reconstruct the samples from the staircase output of the DAC (Figure 1.2).

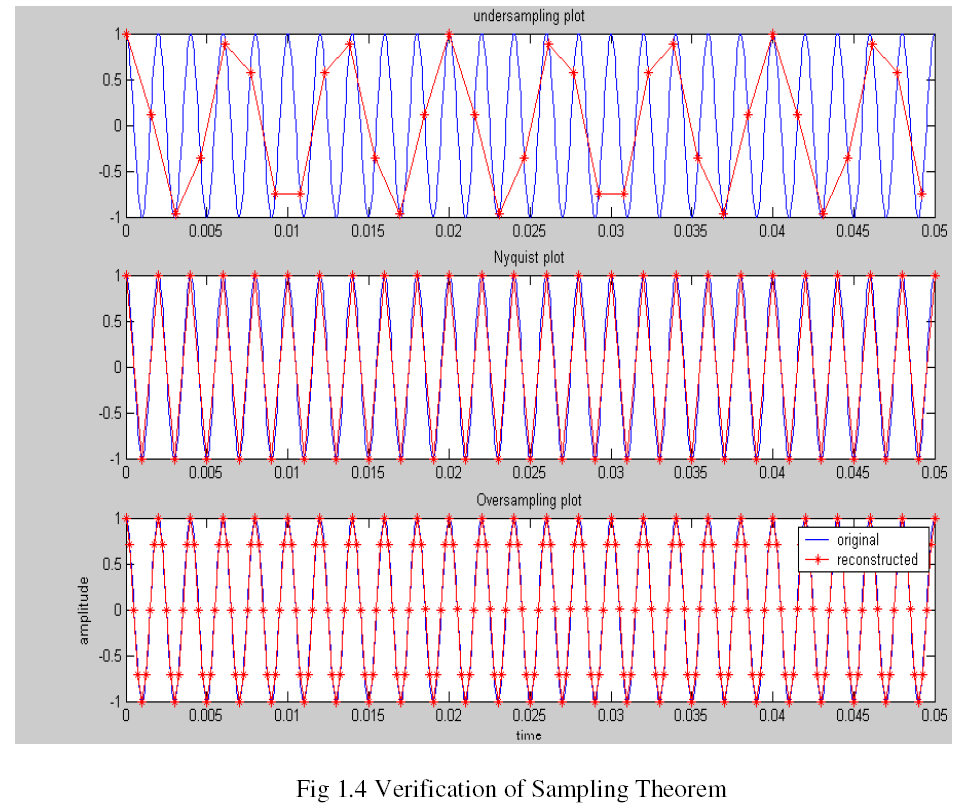




### The Sampling Process

ADC process involves sampling the signal and then quantizing the same to a digital value. In order to avoid Aliasing effect, the signal has to be sampled at a rate at least equal to the Nyquist rate. The condition for Nyquist Criterion is as given below, fs= 1/T  2 fm

Where, fs is the sampling frequency, fm is the maximum frequency component in the message signal. If the sampling of the signal is carried out with a rate less than the Nyquist rate, the higher frequency components of the signal cannot be reconstructed properly. The plots of the reconstructed outputs for various conditions are as shown in figure 1.4.



### Discrete Time Sequences

Consider an analog signal x(t) given by, x(t)= A cos (2 ft). If this signal is sampled at a Sampling Interval T, in the above equation replacing t by nT we get, x (nT) = A cos (2 fnT)

where n= 0,1, 2,..etc

For simplicity denote x (nT) as x (n)

* + - x (n) = A cos (2πfnT) where n= 0,1, 2,..etc We have fs=1/T also θ = 2ΠfnT
    -  x (n) = A cos (2πfnT)= A cos (2πfn/fs) = A cos πn The quantity  isθ called as digital frequency.

θ = 2πfT = 2πf/fs radians

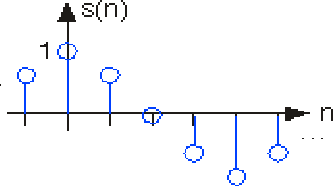


Fig 1.5 A Cosine Waveform

A sequence that repeats itself after every period N is called a periodic sequence. Consider a periodic sequence x (n) with period N x (n)=x (n+N) n=……..,-1,0,1,2,……..

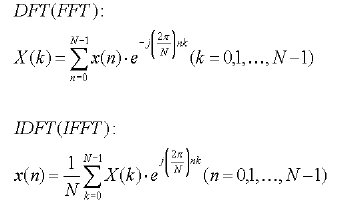
Frequency response gives the frequency domain equivalent of a discrete time sequence. It is denoted as **X(ejθ)=∑x(n) e-jnθ**

Frequency response of a discrete sequence involves both magnitude response and phase response.

### Discrete Fourier Transform and Fast Fourier Transform

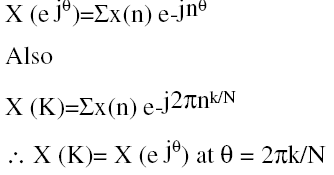
#### DFT Pair:

DFT is used to transform a time domain sequence x (n) to a frequency domain sequence X (K).The equations that relate the time domain sequence x (n) and the corresponding frequency domain sequence X (K) are called DFT Pair and is given by,



#### The Relationship between DFT and Frequency Response:

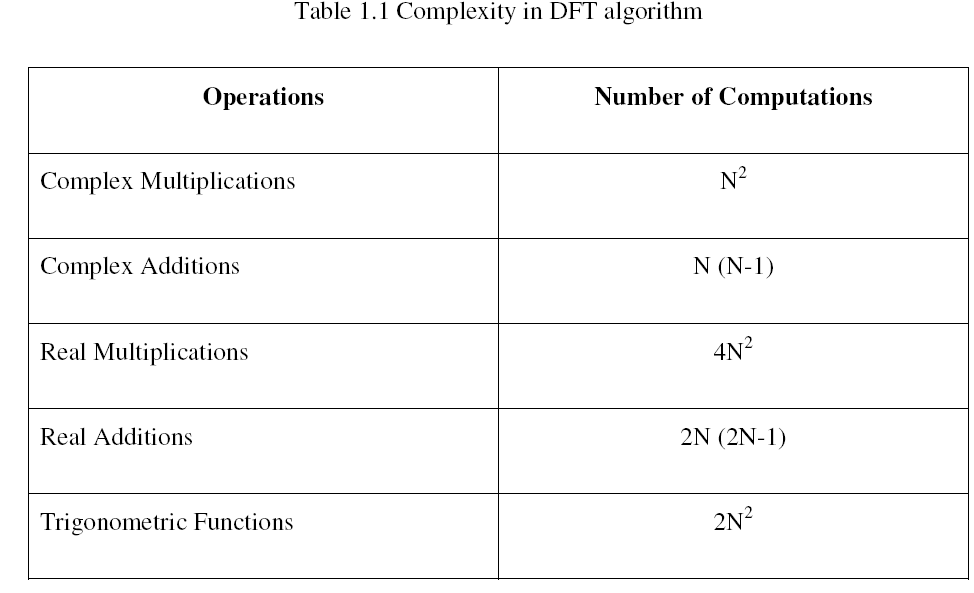
We have,



From the above expression it is clear that we can use DFT to find the Frequency response of a discrete signal. Spacing between the elements of X(k) is given as  f=fs/N=1/NT=1/T0.Where T0 is the signal record length.

It is clear from the expression of  f that, in order to minimize the spacing between the samples N has to be a large value. Although DFT is an efficient technique of obtaining the frequency response of a sequence, it requires more number of complex operations like additions and multiplications.

Thus many improvements over DFT were proposed. One such technique is to use the periodicity property of the twiddle factor e-**j2 /N**. Those algorithms were called as Fast Fourier Transform Algorithms. The following table depicts the complexity involved in the computation using DFT algorithms.



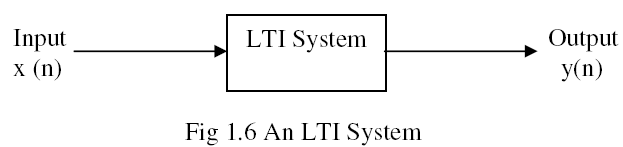
FFT algorithms are classified into two categories via

1. Decimation in Time FFT
2. Decimation in Frequency FFT

In decimation in time FFT the sequence is divided in time domain successively till we reach the sequences of length 2. Whereas in Decimation in Frequency FFT, the sequence X(K) is divided successively. The complexity of computation will get reduced considerably in case of FFT algorithms.

### Linear Time Invariant Systems

A system which satisfies superposition theorem is called as a linear system and a system that has same input output relation at all times is called a Time Invariant System. Systems, which satisfy both the properties, are called LTI systems.



LTI systems are characterized by its impulse response or unit sample response in time domain whereas it is characterized by the system function in frequency domain.

#### Convolution

Convolution is the operation that related the input output of an LTI system, to its unit sample response. The output of the system y (n) for the input x (n) and the impulse response of the system

being h (n) is given as y (n) = x(n) \* h(n) = ∑  x(k) h(n-k), x(n) is the input of the system, h(n) is the impulse response of the system, y(n) is the output of the system.

#### Z Transformation

Z Transformations are used to find the frequency response of the system. The Z Transform for a discrete sequence x (n) is given by, **X(Z)= ∑x(n) z-n**

#### The System Function

An LTI system is characterized by its System function or the transfer function. The system function of a system is the ratio of the Z transformation of its output to that of its input. It is denoted as H (Z) and is given by H (Z) = Y (Z)/ X (Z).

The magnitude and phase of the transfer function H (Z) gives the frequency response of the system. From the transfer function we can also get the poles and zeros of the system by solving its numerator and denominator respectively.

### Digital Filters

Filters are used to remove the unwanted components in the sequence. They are characterized by the impulse response h (n). The general difference equation for an Nth order filter is given by

y (n) =**∑**aky(n-k)+ **∑** bk x(n-k)

A typical digital filter structure is as shown in figure 1.7.

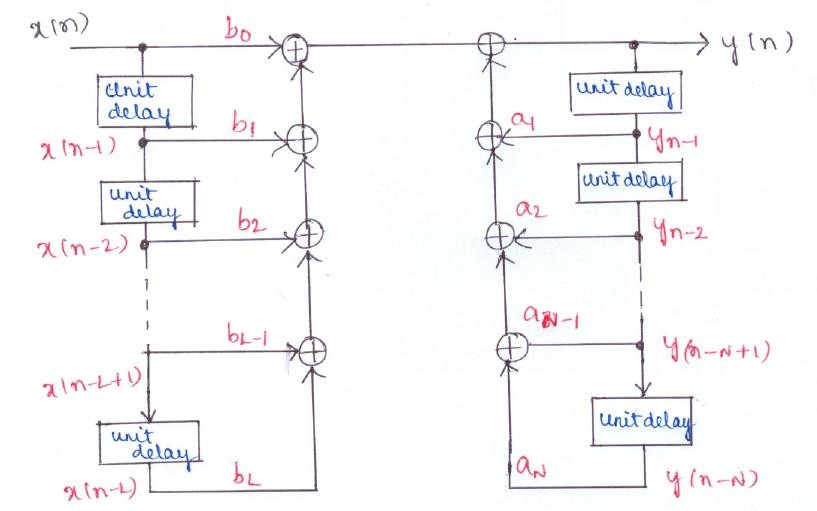


Fig 1.7 Structure of a Digital Filter

Values of the filter coefficients vary with respect to the type of the filter. Design of a digital filter involves determining the filter coefficients. Based on the length of the impulse response, digital filters are classified into two categories via Finite Impulse Response (FIR) Filters and Infinite Impulse Response (IIR) Filters.

#### FIR Filters

FIR filters have impulse responses of finite lengths. In FIR filters the present output depends only on the past and present values of the input sequence but not on the previous output sequences. Thus they are non recursive hence they are inherently stable.FIR filters possess linear phase response. Hence they are very much applicable for the applications requiring linear phase response.

The difference equation of an FIR filter is represented as



The frequency response of an FIR filter is given as



The major drawback of FIR filters is, they require more number of filter coefficients to realize a desired response as compared to IIR filters. Thus the computational time required will also be more.

#### IIR Filters

Unlike FIR filters, IIR filters have infinite number of impulse response samples. They are recursive filters as the output depends not only on the past and present inputs but also on the past outputs. They generally do not have linear phase characteristics. Typical system function of such filters is given by,



Stability of IIR filters depends on the number and the values of the filter coefficients. The major advantage of IIR filters over FIR is that, they require lesser coefficients compared to FIR filters for the same desired response, thus requiring less computation time.

#### FIR Filter Design

Frequency response of an FIR filter is given by the following expression,



Design procedure of an FIR filter involves the determination of the filter coefficients bk.



#### IIR Filter Design

IIR filters can be designed using two methods viz using windows and direct method. In this approach, a digital filter can be designed based on its equivalent analog filter. An analog filter is designed first for the equivalent analog specifications for the given digital specifications. Then using appropriate frequency transformations, a digital filter can be obtained. The filter specifications consist of passband and stopband ripples in dB and Passband and Stopband frequencies in rad/sec.

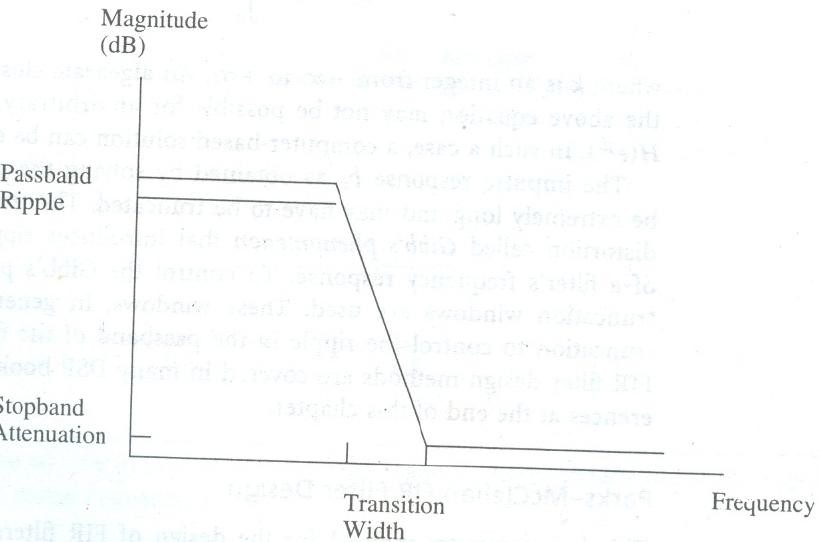


Fig 1.11 Lowpass Filter Specifications

Direct IIR filter design methods are based on least squares fit to a desired frequency response. These methods allow arbitrary frequency response specifications.

### Decimation and Interpolation

Decimation and Interpolation are two techniques used to alter the sampling rate of a sequence. Decimation involves decreasing the sampling rate without violating the sampling theorem whereas interpolation increases the sampling rate of a sequence appropriately by considering its neighboring samples.

* + 1. *Decimation*

Decimation is a process of dropping the samples without violating sampling theorem. The factor by which the signal is decimated is called as decimation factor and it is denoted by M. It is given by,



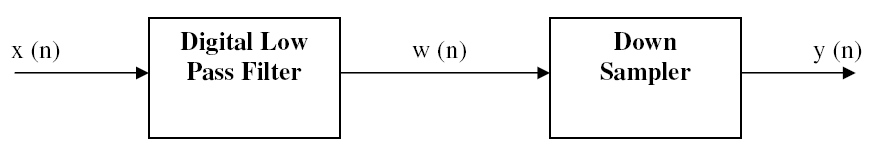
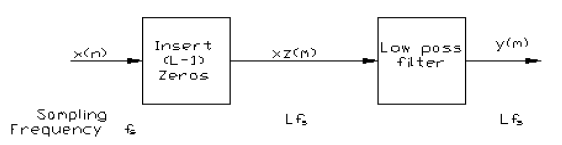


Fig 1.12 Decimation Process

* + 1. *Interpolation*

Interpolation is a process of increasing the sampling rate by inserting new samples in between. The input output relation for the interpolation, where the sampling rate is increased by a factor L, is given as,





### Fig 1.13 Interpolation Process

**Problems:**

* + - 1. **Obtain the transfer function of the IIR filter whose difference equation is given by y (n)= 0.9y (n-1)+0.1x (n)**

y (n)= 0.9y (n-1)+0.1x (n)

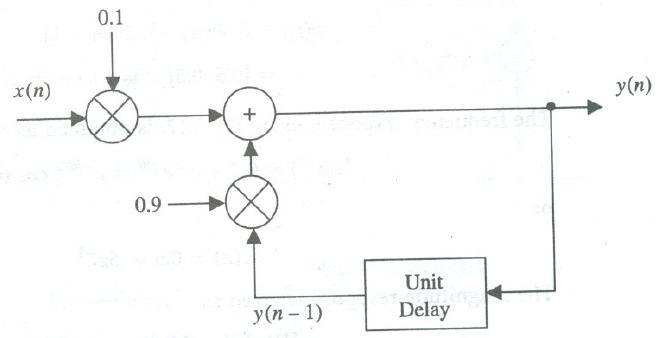
Taking Z transformation both sides Y (Z) = 0.9 Z-1 Y (Z) + 0.1 X (Z)

Y (Z) [1- 0.9 Z-1] = 0.1 X (Z)

The transfer function of the system is given by the expression, H (Z)= Y(Z)/X(Z)

= 0.1/ [ 1- 0.9 Z-1]

Realization of the IIR filter with the above difference equation is as shown in figure.



### Let x(n)= [0 3 6 9 12] be interpolated with L=3. If the filter coefficients of the filters are bk=[1/3 2/3 1 2/3 1/3], obtain the interpolated sequence

After inserting zeros,

w (m) = [0 0 0 3 0 0 6 0 0 9 0 0 12]

bk=[1/3 2/3 1 2/3 1/3] We have,

y(m)=  bk w(m-k) = b-2 w(m+2)+ b-1 w(m+1)+ b0 w(m)+ b1 w(m-1)+ b2 w(m-2)

Substituting the values of m, we get

y(0)= b-2 w(2)+ b-1 w(1)+ b0 w(0)+ b1 w(-1)+ b2 w(-2)= 0

y(1)= b-2 w(3)+ b-1 w(2)+ b0 w(1)+ b1 w(0)+ b2 w(-1)=1

y(2)= b-2 w(4)+ b-1 w(3)+ b0 w(2)+ b1 w(1)+ b2 w(0)=2

Similarly we get the remaining samples as, y (n) = [ 0 1 2 3 4 5 6 7 8 9 10 11 12]

### Basic Architectural Features

A programmable DSP device should provide instructions similar to a conventional microprocessor. The instruction set of a typical DSP device should include the following,

1. Arithmetic operations such as ADD, SUBTRACT, MULTIPLY etc
2. Logical operations such as AND, OR, NOT, XOR etc
3. Multiply and Accumulate (MAC) operation
4. Signal scaling operation

In addition to the above provisions, the architecture should also include,

1. On chip registers to store immediate results
2. On chip memories to store signal samples (RAM)
3. On chip memories to store filter coefficients (ROM)

### DSP Computational Building Blocks

Each computational block of the DSP should be optimized for functionality and speed and in the meanwhile the design should be sufficiently general so that it can be easily integrated with other blocks to implement overall DSP systems.

#### Multipliers

The advent of single chip multipliers paved the way for implementing DSP functions on a VLSI chip. Parallel multipliers replaced the traditional shift and add multipliers now days. Parallel

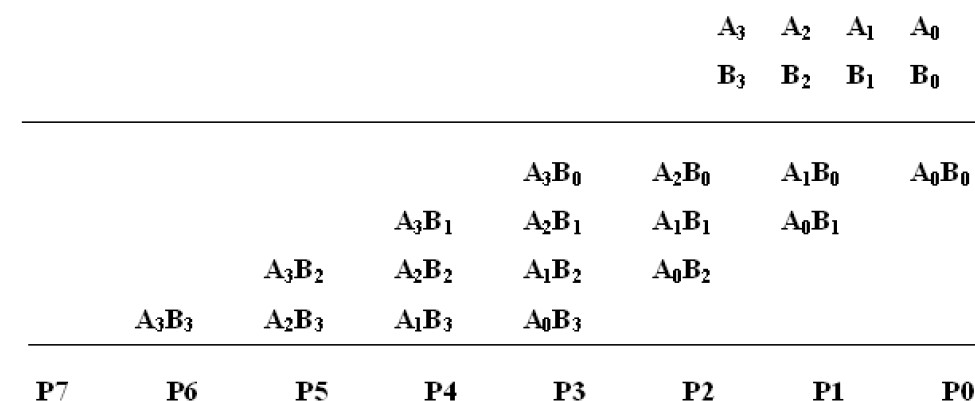
multipliers take a single processor cycle to fetch and execute the instruction and to store the result. They are also called as Array multipliers. The key features to be considered for a multiplier are:

1. Accuracy
2. Dynamic range
3. Speed

The number of bits used to represent the operands decides the accuracy and the dynamic range of the multiplier. Whereas speed is decided by the architecture employed. If the multipliers are implemented using hardware, the speed of execution will be very high but the circuit complexity will also increases considerably. Thus there should be a tradeoff between the speed of execution and the circuit complexity. Hence the choice of the architecture normally depends on the application.

#### Parallel Multipliers

Consider the multiplication of two unsigned numbers A and B. Let A be represented using m bits as (Am-1 Am-2 …….. A1 A0) and B be represented using n bits as (Bn-1 Bn-2 …….. B1 B0). Then the product of these two numbers is given by,



This operation can be implemented paralleling using Braun multiplier whose hardware structure is as shown in the figure 2.1.

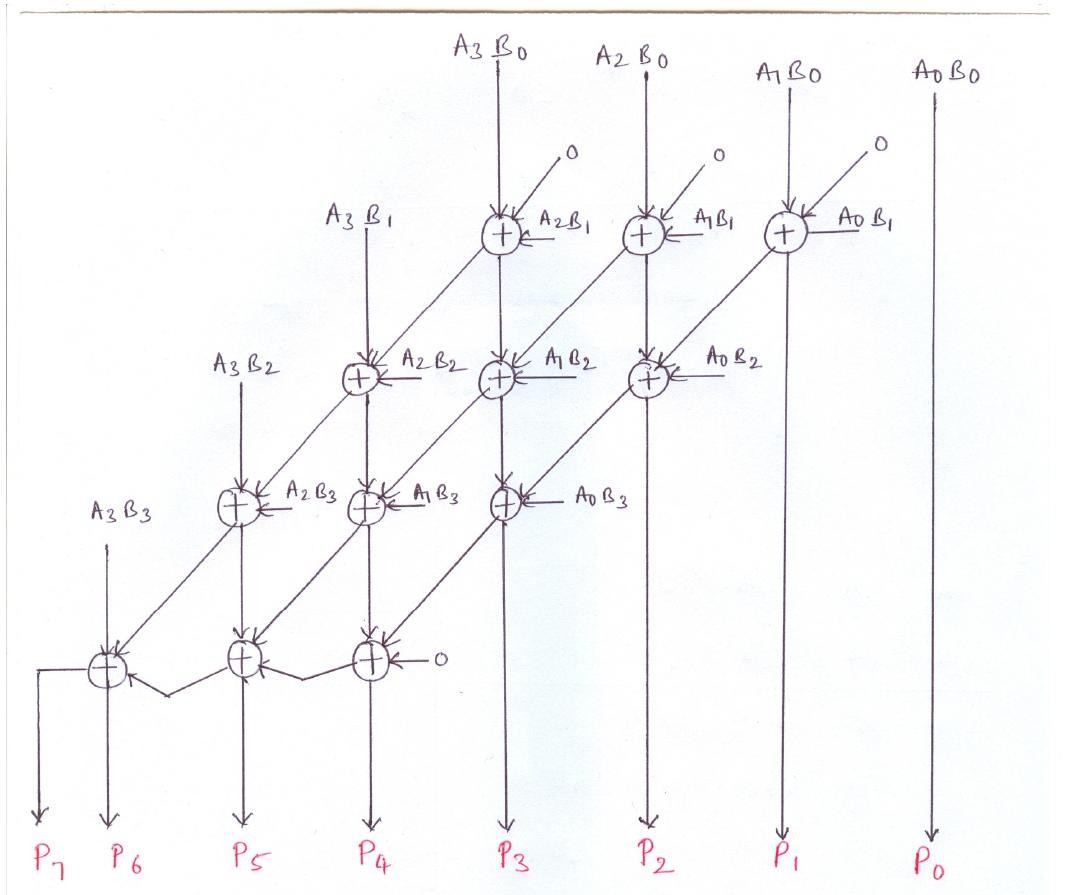
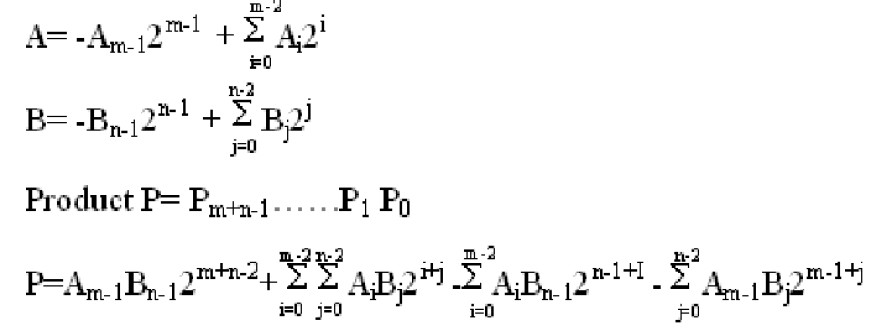


Fig 2.1 Braun Multiplier for a 4X4 Multiplication

#### Multipliers for Signed Numbers

In the Braun multiplier the sign of the numbers are not considered into account. In order to implement a multiplier for signed numbers, additional hardware is required to modify the Braun multiplier. The modified multiplier is called as Baugh-Wooley multiplier.

Consider two signed numbers A and B,



#### Speed

Conventional Shift and Add technique of multiplication requires n cycles to perform the multiplication of two n bit numbers. Whereas in parallel multipliers the time required will be the longest path delay in the combinational circuit used. As DSP applications generally require very high speed, it is desirable to have multipliers operating at the highest possible speed by having parallel implementation.

#### Bus Widths

Consider the multiplication of two n bit numbers X and Y. The product Z can be at most 2n bits long. In order to perform the whole operation in a single execution cycle, we require two buses of width n bits each to fetch the operands X and Y and a bus of width 2n bits to store the result Z to the memory. Although this performs the operation faster, it is not an efficient way of implementation as it is expensive. Many alternatives for the above method have been proposed. One such method is to use the program bus itself to fetch one of the operands after fetching the instruction, thus requiring only one bus to fetch the operands. And the result Z can be stored back to the memory using the same operand bus. But the problem with this is the result Z is 2n bits long whereas the operand bus is just n bits long. We have two alternatives to solve this problem, a. Use the n bits operand bus and save Z at two successive memory locations. Although it stores the exact value of Z in the memory, it takes two cycles to store the result.

b. Discard the lower n bits of the result Z and store only the higher order n bits into the memory. It is not applicable for the applications where accurate result is required. Another alternative can be used for the applications where speed is not a major concern. In which latches are used for inputs and outputs thus requiring a single bus to fetch the operands and to store the result (Fig 2.2).

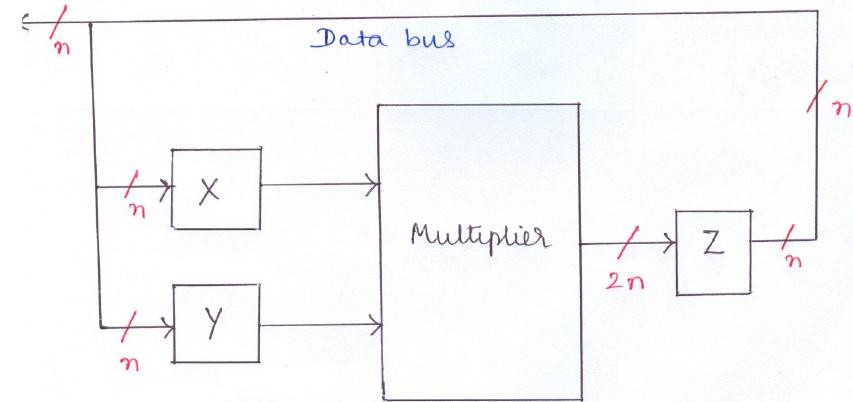


Fig 2.2: A Multiplier with Input and Output Latches

#### Shifters

Shifters are used to either scale down or scale up operands or the results. The following scenarios give the necessity of a shifter

1. While performing the addition of N numbers each of n bits long, the sum can grow up to n+log2 N bits long. If the accumulator is of n bits long, then an overflow error will occur. This can be overcome by using a shifter to scale down the operand by an amount of log2N.
2. Similarly while calculating the product of two n bit numbers, the product can grow up to 2n bits long. Generally the lower n bits get neglected and the sign bit is shifted to save the sign of the product.
3. Finally in case of addition of two floating-point numbers, one of the operands has to be shifted appropriately to make the exponents of two numbers equal.

From the above cases it is clear that, a shifter is required in the architecture of a DSP.

#### Barrel Shifters

In conventional microprocessors, normal shift registers are used for shift operation. As it requires one clock cycle for each shift, it is not desirable for DSP applications, which generally involves more shifts. In other words, for DSP applications as speed is the crucial issue, several shifts are to be accomplished in a single execution cycle. This can be accomplished using a barrel shifter, which connects the input lines representing a word to a group of output lines with the required shifts determined by its control inputs. For an input of length n, log2 n control lines are required. And an dditional control line is required to indicate the direction of the shift.

The block diagram of a typical barrel shifter is as shown in figure 2.3.

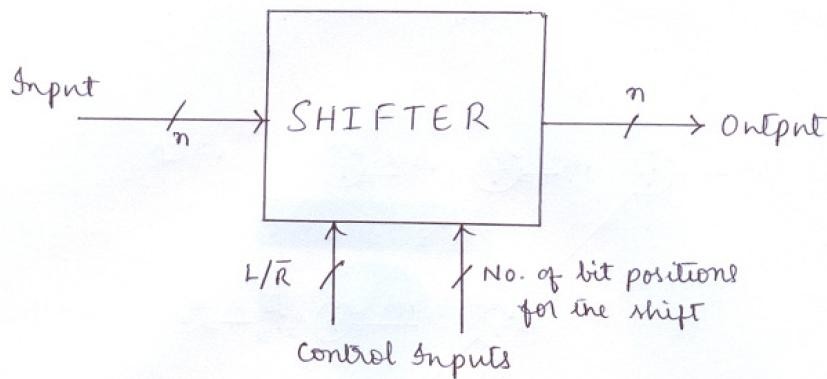


Fig 2.3 A Barrel Shifter

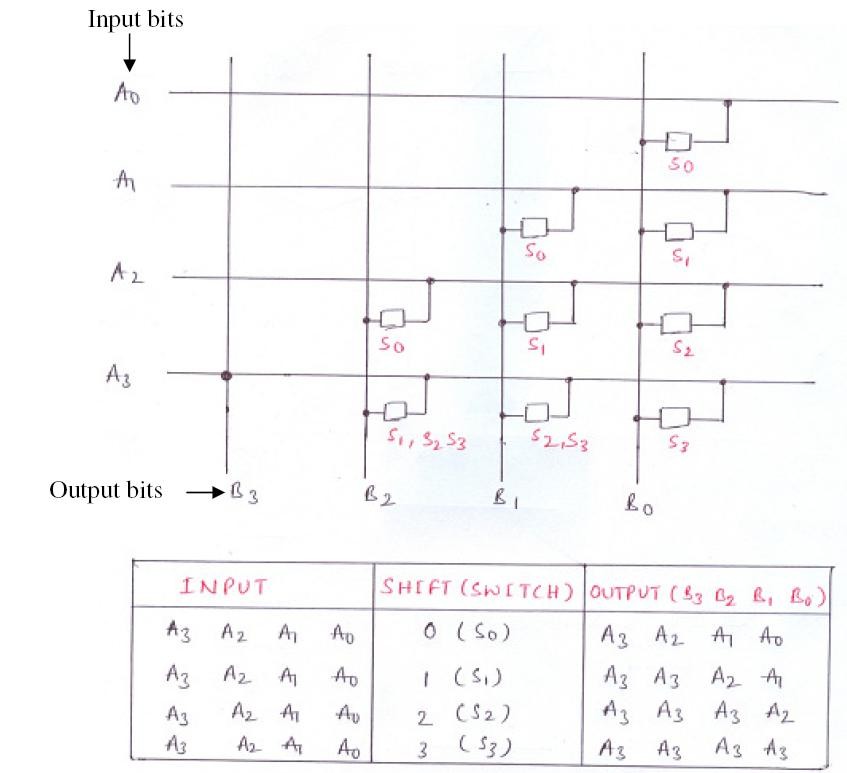


Fig 2.4 Implementation of a 4 bit Shift Right Barrel Shifter

Figure 2.4 depicts the implementation of a 4 bit shift right barrel shifter. Shift to right by 0, 1, 2 or 3 bit positions can be controlled by setting the control inputs appropriately.

### 2.3 Multiply and Accumulate Unit

Most of the DSP applications require the computation of the sum of the products of a series of successive multiplications. In order to implement such functions a special unit called a multiply and Accumulate (MAC) unit is required. A MAC consists of a multiplier and a special register called Accumulator. MACs are used to implement the functions of the type A+BC. A typical MAC unit is as shown in the figure 2.5.

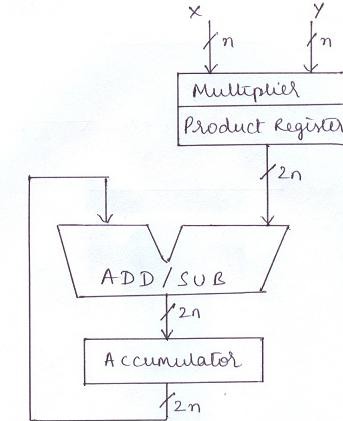


Fig 2.5 A MAC Unit

Although addition and multiplication are two different operations, they can be performed in parallel. By the time the multiplier is computing the product, accumulator can accumulate the product of the previous multiplications. Thus if N products are to be accumulated, N-1 multiplications can overlap with N-1 additions. During the very first multiplication, accumulator will be idle and during the last accumulation, multiplier will be idle. Thus N+1 clock cycles are required to compute the sum of N products.

#### 2.3.1 Overflow and Underflow

While designing a MAC unit, attention has to be paid to the word sizes encountered at the input of the multiplier and the sizes of the add/subtract unit and the accumulator, as there is a possibility of overflow and underflows. Overflow/underflow can be avoided by using any of the following methods viz

1. Using shifters at the input and the output of the MAC
2. Providing guard bits in the accumulator
3. Using saturation logic

### Shifters

Shifters can be provided at the input of the MAC to normalize the data and at the output to de normalize the same.

### Guard bits

As the normalization process does not yield accurate result, it is not desirable for some applications. In such cases we have another alternative by providing additional bits called guard bits in the accumulator so that there will not be any overflow error. Here the add/subtract unit also has to be modified appropriately to manage the additional bits of the accumulator.

### Saturation Logic

Overflow/ underflow will occur if the result goes beyond the most positive number or below the least negative number the accumulator can handle. Thus the overflow/underflow error can be resolved by loading the accumulator with the most positive number which it can handle at the time of overflow and the least negative number that it can handle at the time of underflow. This method is called as saturation logic. A schematic diagram of saturation logic is as shown in figure 2.7. In saturation logic, as soon as an overflow or underflow condition is satisfied the accumulator will be loaded with the most positive or least negative number overriding the result computed by the MAC unit.

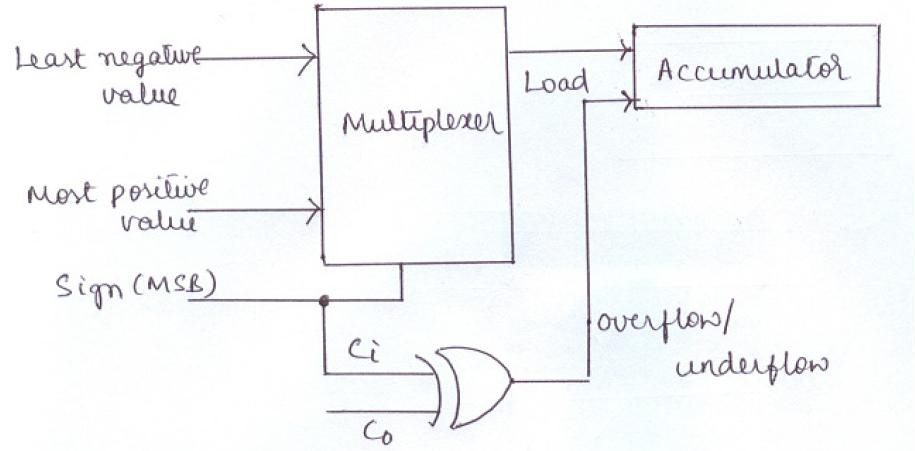


Fig 2.7: Schematic Diagram of the Saturation Logic

### Arithmetic and Logic Unit

A typical DSP device should be capable of handling arithmetic instructions like ADD, SUB, INC, DEC etc and logical operations like AND, OR , NOT, XOR etc. The block diagram of a typical ALU for a DSP is as shown in the figure 2.8.

It consists of status flag register, register file and multiplexers.

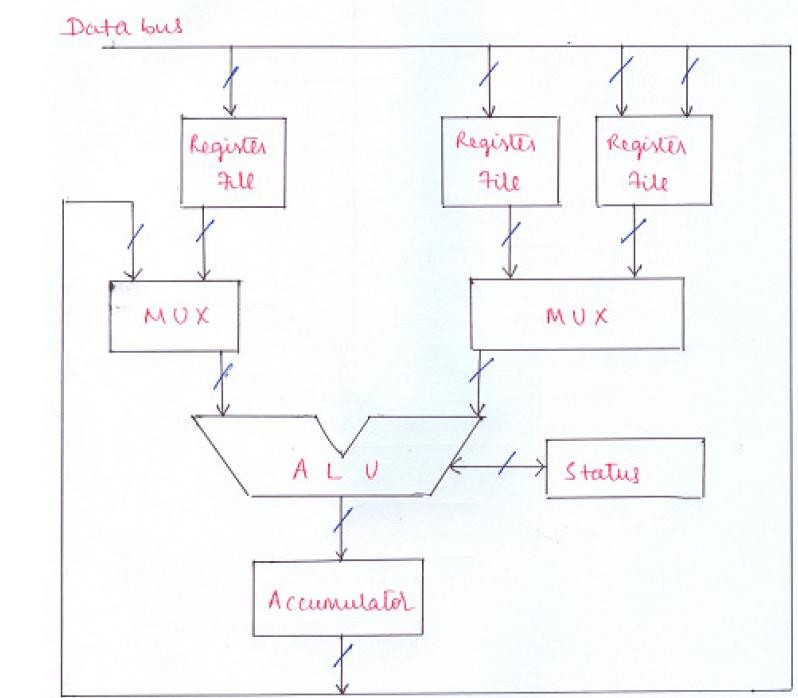


Fig 2.8 Arithmetic Logic Unit of a DSP

### Status Flags

ALU includes circuitry to generate status flags after arithmetic and logic operations. These flags include sign, zero, carry and overflow.

### Overflow Management

Depending on the status of overflow and sign flags, the saturation logic can be used to limit the accumulator content.

### Register File

Instead of moving data in and out of the memory during the operation, for better speed, a large set of general purpose registers are provided to store the intermediate results.

### Bus Architecture and Memory

Conventional microprocessors use Von Neumann architecture for memory management wherein the same memory is used to store both the program and data (Fig 2.9). Although this architecture is simple, it takes more number of processor cycles for the execution of a single instruction as the same bus is used for both data and program.

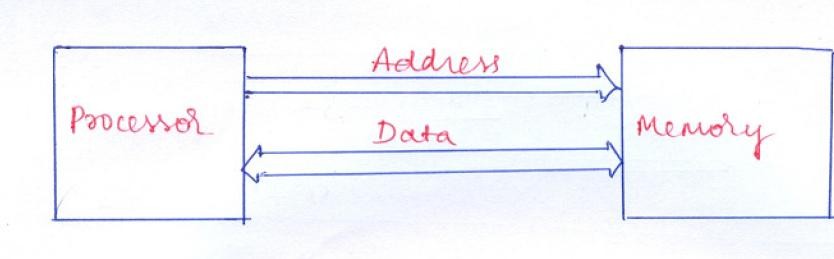


Fig 2.9 Von Neumann Architecture

In order to increase the speed of operation, separate memories were used to store program and data and a separate set of data and address buses have been given to both memories, the architecture called as Harvard Architecture. It is as shown in figure 2.10.

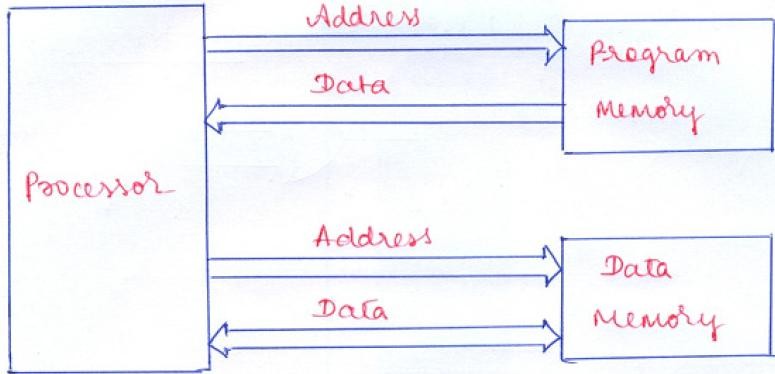


Fig 2.10 Harvard Architecture

Although the usage of separate memories for data and the instruction speeds up the processing, it will not completely solve the problem. As many of the DSP instructions require more than one operand, use of a single data memory leads to the fetch the operands one after the other, thus increasing the delay of processing. This problem can be overcome by using two separate data memories for storing operands separately, thus in a single clock cycle both the operands can be fetched together (Figure 2.11).

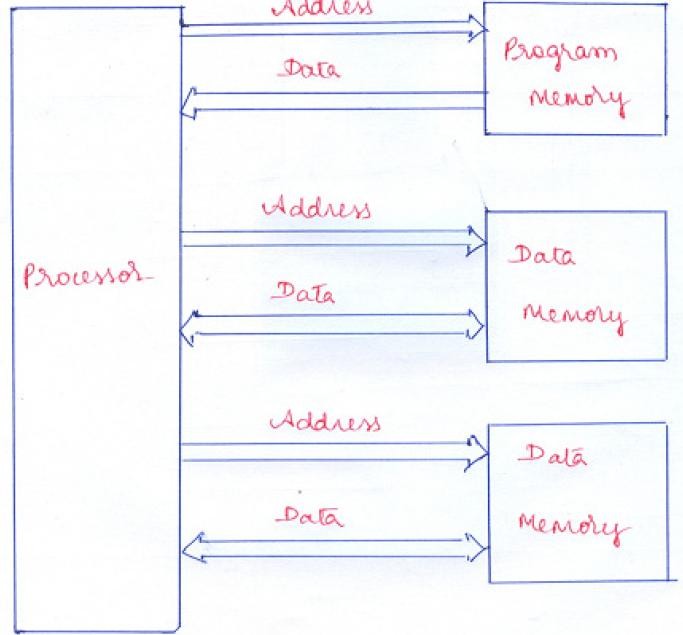


Fig 2.11 Harvard Architecture with Dual Data Memory

Although the above architecture improves the speed of operation, it requires more hardware and interconnections, thus increasing the cost and complexity of the system. Therefore there should be a trade off between the cost and speed while selecting memory architecture for a DSP.

#### On-chip Memories

In order to have a faster execution of the DSP functions, it is desirable to have some memory located on chip. As dedicated buses are used to access the memory, on chip memories are faster. Speed and size are the two key parameters to be considered with respect to the on-chip memories.

### Speed

On-chip memories should match the speeds of the ALU operations in order to maintain the single cycle instruction execution of the DSP.

### Size

In a given area of the DSP chip, it is desirable to implement as many DSP functions as possible. Thus the area occupied by the on-chip memory should be minimum so that there will be a scope for implementing more number of DSP functions on- chip.

#### Organization of On-chip Memories

Ideally whole memory required for the implementation of any DSP algorithm has to reside on- chip so that the whole processing can be completed in a single execution cycle. Although it looks as a better solution, it consumes more space on chip, reducing the scope for implementing any functional block on-chip, which in turn reduces the speed of execution. Hence some other alternatives have to be thought of. The following are some other ways in which the on-chip memory can be organized.

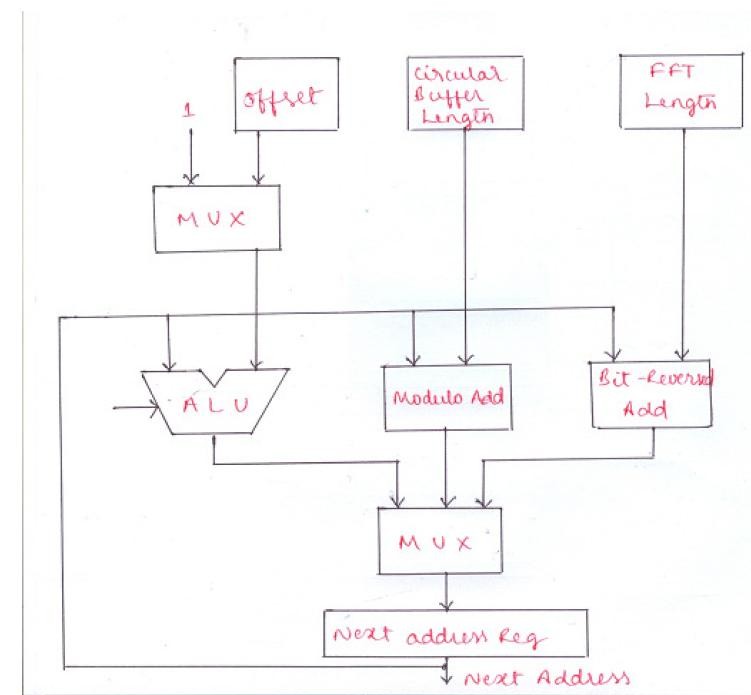
1. As many DSP algorithms require instructions to be executed repeatedly, the instruction can be stored in the external memory, once it is fetched can reside in the instruction cache.
2. The access times for memories on-chip should be sufficiently small so that it can be accessed more than once in every execution cycle.
3. On-chip memories can be configured dynamically so that they can serve different purpose at different times.
   1. **Address Generation Unit**

The main job of the Address Generation Unit is to generate the address of the operands required to carry out the operation. They have to work fast in order to satisfy the timing constraints. As the address generation unit has to perform some mathematical operations in order to calculate the operand address, it is provided with a separate ALU.

Address generation typically involves one of the following operations.

1. Getting value from immediate operand, register or a memory location
2. Incrementing/ decrementing the current address
3. Adding/subtracting the offset from the current address
4. Adding/subtracting the offset from the current address and generating new address according to circular addressing mode
5. Generating new address using bit reversed addressing mode

The block diagram of a typical address generation unit is as shown in figure 2.13.



### Fig 2.13 Address generation unit

* 1. **Programmability and program Execution**

A programmable DSP device should provide the programming capability involving branching, looping and subroutines. The implementation of repeat capability should be hardware based so that it can be programmed with minimal or zero overhead. A dedicated register can be used as a counter. In a normal subroutine call, return address has to be stored in a stack thus requiring memory access for storing and retrieving the return address, which in turn reduces the speed of operation. Hence a LIFO memory can be directly interfaced with the program counter.

#### Program Control

Like microprocessors, DSP also requires a control unit to provide necessary control and timing signals for the proper execution of the instructions. In microprocessors, the controlling is micro coded based where each instruction is divided into microinstructions stored in micro memory. As this mechanism is slower, it is not applicable for DSP applications. Hence in DSP the controlling is hardwired base where the Control unit is designed as a single, comprehensive, hardware unit. Although it is more complex it is faster.

#### Program Sequencer

It is a part of the control unit used to generate instruction addresses in sequence needed to access instructions. It calculates the address of the next instruction to be fetched. The next address can be from one of the following sources.

1. Program Counter
2. Instruction register in case of branching, looping and subroutine calls
3. Interrupt Vector table
4. Stack which holds the return address

The block diagram of a program sequencer is as shown in figure 2.14.

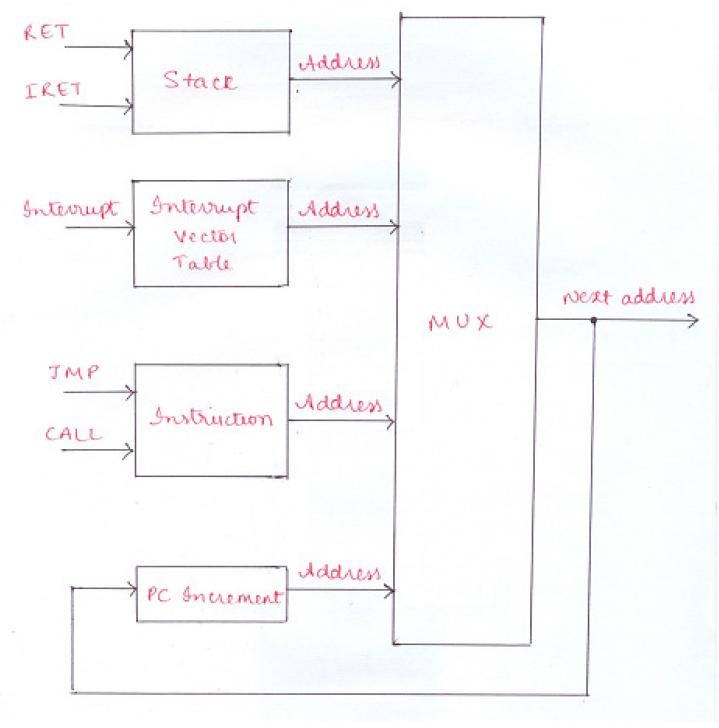


Fig 2.14 Program Sequencer

Program sequencer should have the following circuitry:

1. PC has to be updated after every fetch
2. Counter to hold count in case of looping
3. A logic block to check conditions for conditional jump instructions
4. Condition logic-status flag

## UNIT-2

### Introduction:

Leading manufacturers of integrated circuits such as Texas Instruments (TI), Analog devices & Motorola manufacture the digital signal processor (DSP) chips. These manufacturers have developed a range of DSP chips with varied complexity.

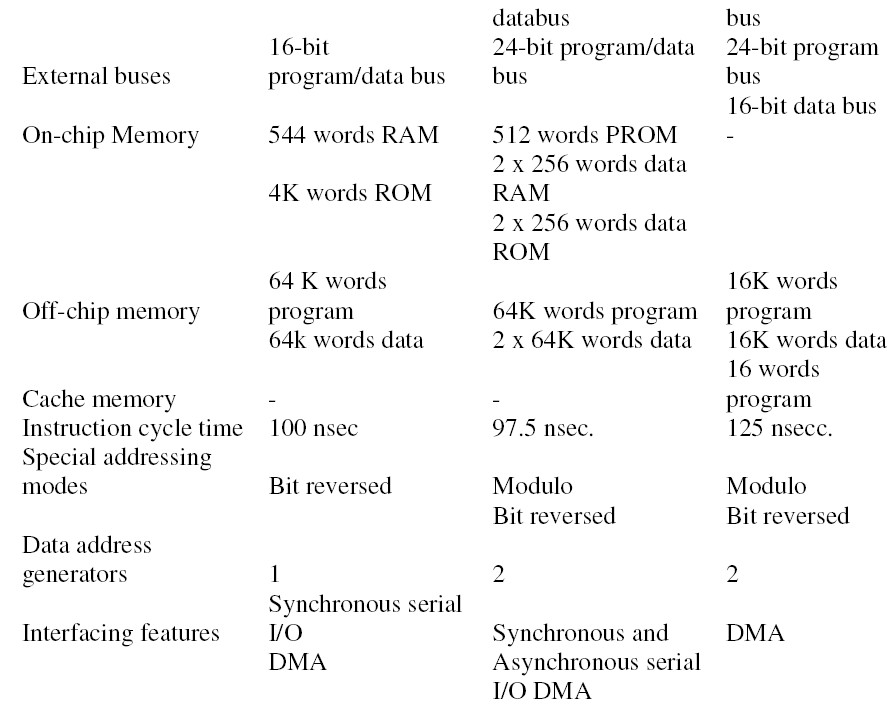
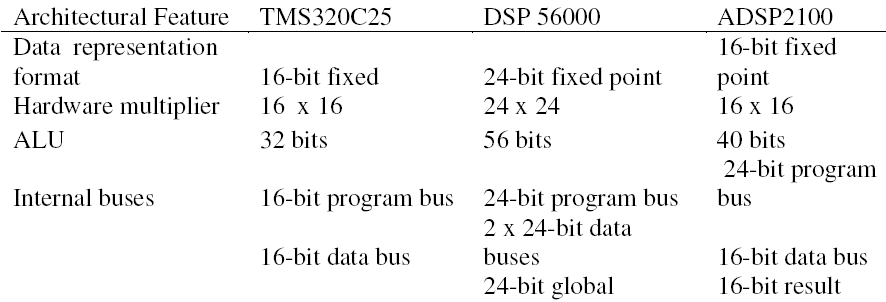
The TMS320 family consists of two types of single chips DSPs: 16-bit fixed point &32-bit floating- point. These DSPs possess the operational flexibility of high-speed controllers and the numerical capability of array processors

### Commercial Digital Signal-Processing Devices:

There are several families of commercial DSP devices. Right from the early eighties, when these devices began to appear in the market, they have been used in numerous applications, such as communication, control, computers, Instrumentation, and consumer electronics. The architectural features and the processing power of these devices have been constantly upgraded based on the advances in technology and the application needs. However, their basic versions, most of them have Harvard architecture, a single-cycle hardware multiplier, an address generation unit with dedicated address registers, special addressing modes, on-chip peripherals interfaces. Of the various families of programmable DSP devices that are commercially available, the three most popular ones are those

from Texas Instruments, Motorola, and Analog Devices. Texas Instruments was one of the first to come out with a commercial programmable DSP with the introduction of its TMS32010 in 1982.

### Summary of the Architectural Features of three fixed-Points DSPs

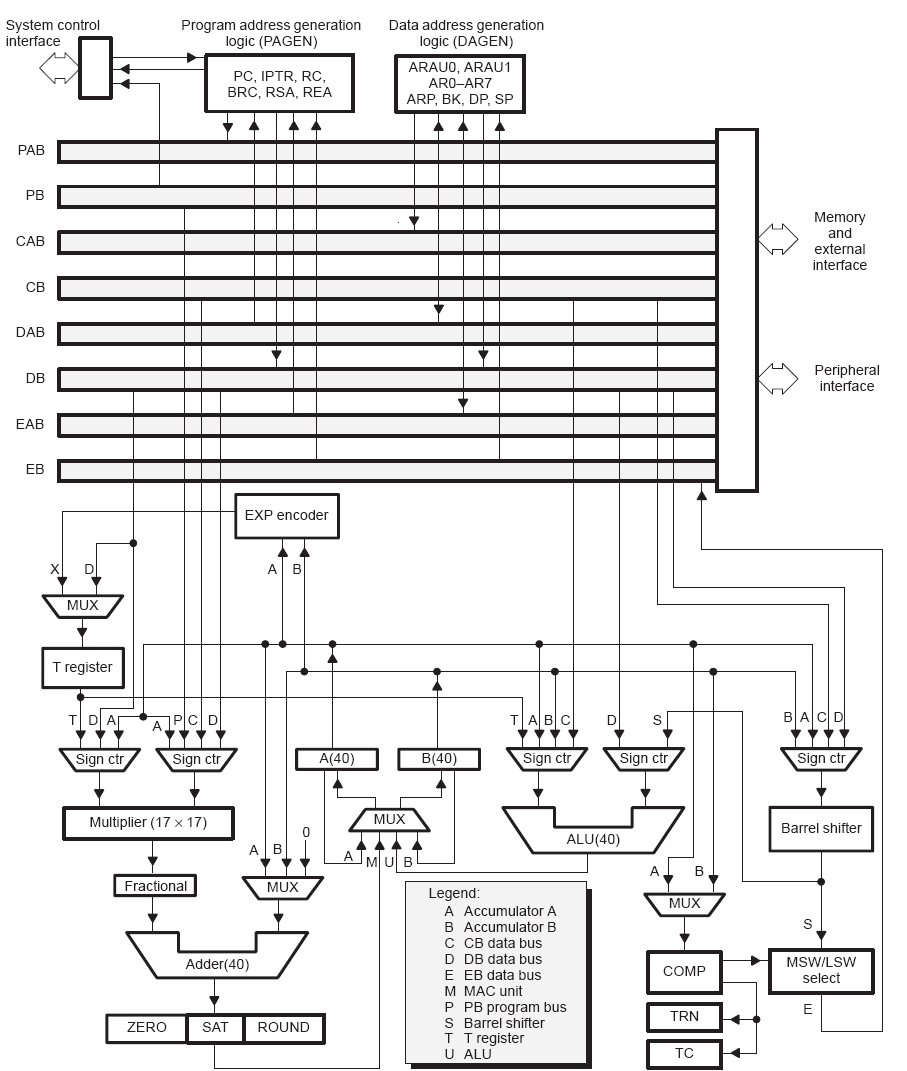


* 1. **The architecture of TMS320C54xx digital signal processors:**

TMS320C54xx processors retain in the basic Harvard architecture of their predecessor, TMS320C25, but have several additional features, which improve their performance over it. Figure 3.1 shows a functional block diagram of TMS320C54xx processors. They have one program and three data memory spaces with separate buses, which provide simultaneous accesses to program instruction and two data operands and enables writing of result at the same time. Part of the memory is implemented on-chip and consists of combinations of ROM, dual-access RAM, and single-access RAM. Transfers between the memory spaces are also possible.

The central processing unit (CPU) of TMS320C54xx processors consists of a 40- bit arithmetic logic unit (ALU), two 40-bit accumulators, a barrel shifter, a 17x17 multiplier, a 40-bit adder, data address generation logic (DAGEN) with its own arithmetic unit, and program address generation logic (PAGEN). These major functional units are supported by a number of registers and logic in the architecture. A powerful instruction set with a hardware-supported, single-instruction repeat and block repeat operations, block memory move instructions, instructions that pack two or three simultaneous reads, and arithmetic instructions with parallel store and load make these devices very efficient for running high-speed DSP algorithms.

Several peripherals, such as a clock generator, a hardware timer, a wait state generator, parallel I/O ports, and serial I/O ports, are also provided on-chip. These peripherals make it convenient to interface the signal processors to the outside world. In these following sections, we examine in detail the various architectural features of the TMS320C54xx family of processors.



**Figure 3.1**.Functional architecture for TMS320C54xx processors.

### Bus Structure:

The performance of a processor gets enhanced with the provision of multiple buses to provide simultaneous access to various parts of memory or peripherals. The 54xx architecture is built around four pairs of 16-bit buses with each pair consisting of an address bus and a data bus. As shown in Figure 3.1, these are The program bus pair (**PAB, PB**); which carries the instruction code from the program memory. Three data bus pairs (**CAB, CB; DAB, DB**; and **EAB, EB**); which interconnected the various units within the CPU. In Addition the pair CAB, CB and DAB, DB are used to read from the data memory, while The pair **EAB, EB**; carries the data to be written to the memory. The ‘54xx can generate up to two data-memory addresses per cycle using the two auxiliary register arithmetic unit (ARAU0 and ARAU1) in the DAGEN block. This enables accessing two operands simultaneously.

### Central Processing Unit (CPU):

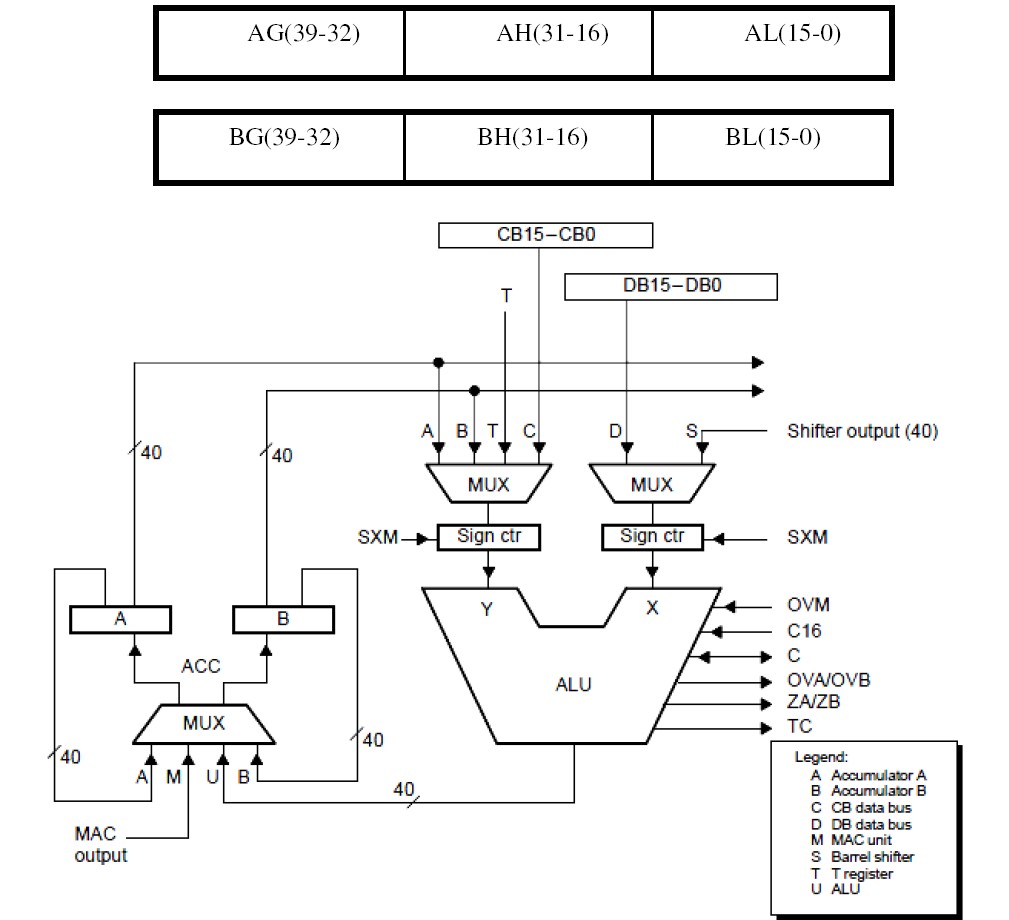
The ‘54xx CPU is common to all the ‘54xx devices. The ’54xx CPU contains a 40-bit arithmetic logic unit (**ALU**); two 40-bit accumulators (**A** and **B**); a barrel shifter; a

17 x 17-bit multiplier; a 40-bit adder; a compare, select and store unit (**CSSU**); an exponent encoder(**EXP**); a data address generation unit (**DAGEN**); and a program address generation unit (**PAGEN**).

The ALU performs 2’s complement arithmetic operations and bit-level Boolean operations on 16, 32, and 40-bit words. It can also function as two separate 16-bit ALUs

and perform two 16-bit operations simultaneously. Figure 3.2 show the functional diagram of the ALU of the TMS320C54xx family of devices.

**Accumulators A and B** store the output from the ALU or the multiplier/adder block and provide a second input to the ALU. Each accumulators is divided into three parts: guards bits (bits 39-32), high- order word (bits-31-16), and low-order word (bits 15- 0), which can be stored and retrieved individually. Each accumulator is memory-mapped and partitioned. It can be configured as the destination registers. The guard bits are used as a head margin for computations.

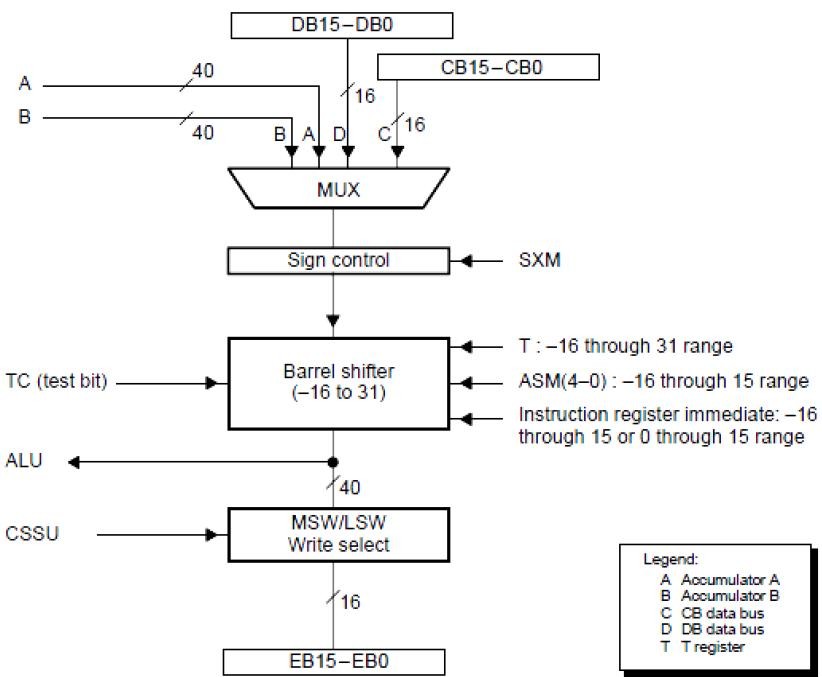


**Figure 3.2**.Functional diagram of the central processing unit of the TMS320C54xx processors.

**Barrel shifter:** provides the capability to scale the data during an operand read or write.

No overhead is required to implement the shift needed for the scaling operations. The’54xx barrel shifter can produce a left shift of 0 to 31 bits or a right shift of 0 to 16 bits on the input data. The shift count field of status registers ST1, or in the temporary

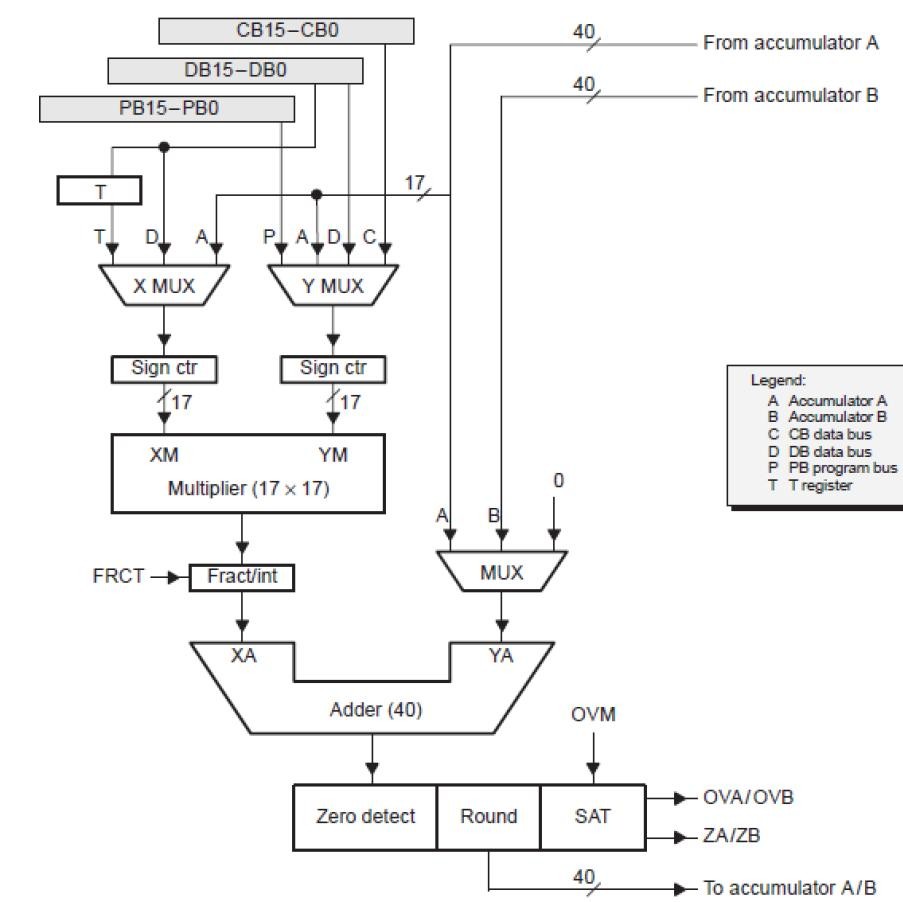
register T. Figure 3.3 shows the functional diagram of the barrel shifter of TMS320C54xx processors. The barrel shifter and the exponent encoder normalize the values in an accumulator in a single cycle. The LSBs of the output are filled with0s, and the MSBs can be either zero filled or sign extended, depending on the state of the sign-extension mode bit in the status register ST1. An additional shift capability enables the processor to perform numerical scaling, bit extraction, extended arithmetic, and overflow prevention operations.



**Figure 3.3**.Functional diagram of the barrel shifter

**Multiplier/adder unit:** The kernel of the DSP device architecture is multiplier/adder unit. The multiplier/adder unit of TMS320C54xx devices performs 17 x 17 2’s complement multiplication with a 40-bit addition effectively in a single instruction cycle.

In addition to the multiplier and adder, the unit consists of control logic for integer and fractional computations and a 16-bit temporary storage register, T. Figure 3.4 show the functional diagram of the multiplier/adder unit of TMS320C54xx processors. The compare, select, and store unit (CSSU) is a hardware unit specifically incorporated to accelerate the add/compare/select operation. This operation is essential to implement the *Viterbi* algorithm used in many signal-processing applications. The exponent encoder unit supports the EXP instructions, which stores in the T register the number of leading redundant bits of the accumulator content. This information is useful while shifting the accumulator content for the purpose of scaling.



**Figure 3.4.** Functional diagram of the multiplier/adder unit of TMS320C54xx processors.

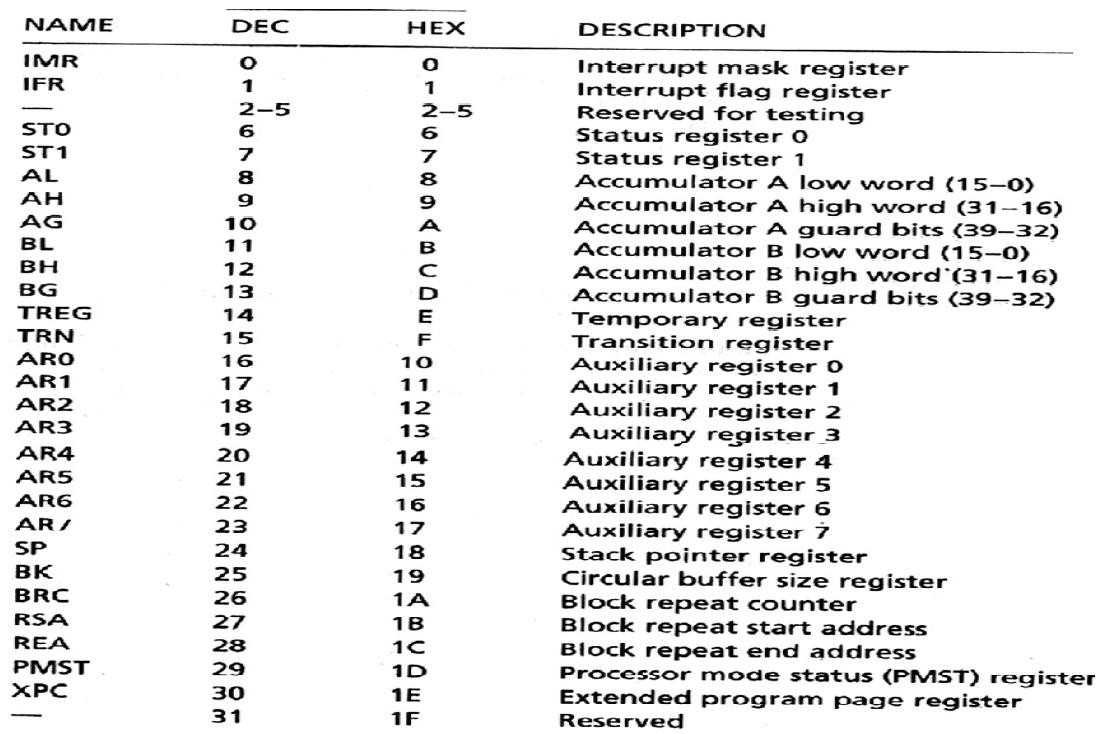
### Internal Memory and Memory-Mapped Registers:

The amount and the types of memory of a processor have direct relevance to the efficiency and performance obtainable in implementations with the processors. The ‘54xx memory is organized into three individually selectable spaces: program, data, and I/O spaces. All ‘54xx devices contain both RAM and ROM. RAM can be either dual-access type (DARAM) or single-access type (SARAM). The on-chip RAM for these processors is organized in pages having 128 word locations on each page.

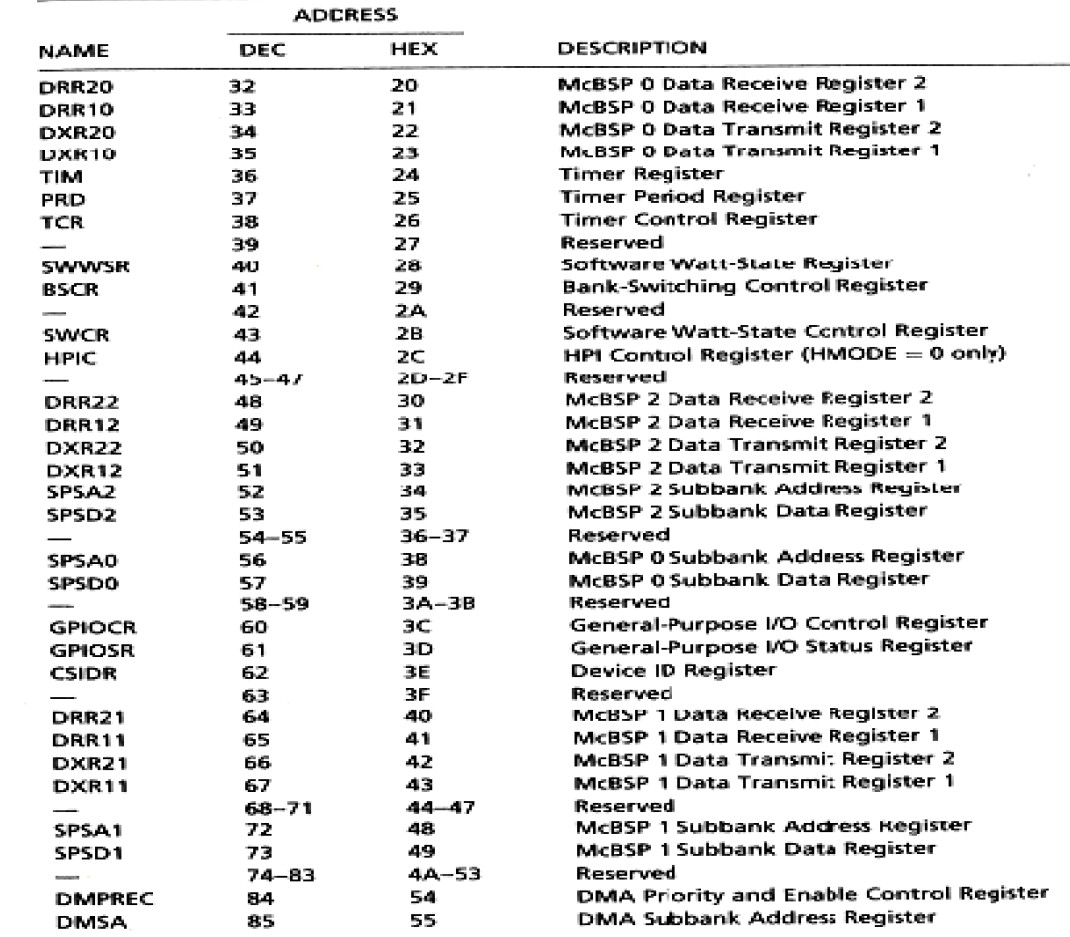
The ‘54xx processors have a number of CPU registers to support operand addressing and computations. The CPU registers and peripherals registers are all located on page 0 of the data

memory. Figure 3.5(a) and (b) shows the internal CPU registers and peripheral registers with their addresses. The processors mode status (PMST) registers

that is used to configure the processor. It is a memory-mapped register located at address 1Dh on page 0 of the RAM. A part of on-chip ROM may contain a boot loader and look-up tables for function such as sine, cosine, *μ- law, and A-* law.



**Figure 3.5(a)** Internal memory-mapped registers of TMS320C54xx processors.



**Figure 3.5(b).**peripheral registers for the TMS320C54xx processors

### Status registers (ST0,ST1):

**ST0:** Contains the status of flags (OVA, OVB, C, TC) produced by arithmetic operations & bit manipulations.

**ST1:** Contain the status of various conditions & modes. Bits of ST0&ST1registers can be set or clear with the SSBX & RSBX instructions.

**PMST:** Contains memory-setup status & control information.

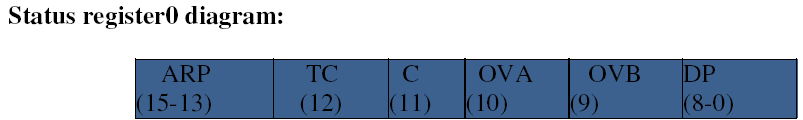


Figure 3.6(a). ST0 diagram

ARP: Auxiliary register pointer. TC: Test/control flag.

C: Carry bit.

OVA: Overflow flag for accumulator A. OVB: Overflow flag for accumulator B. DP: Data-memory page pointer.

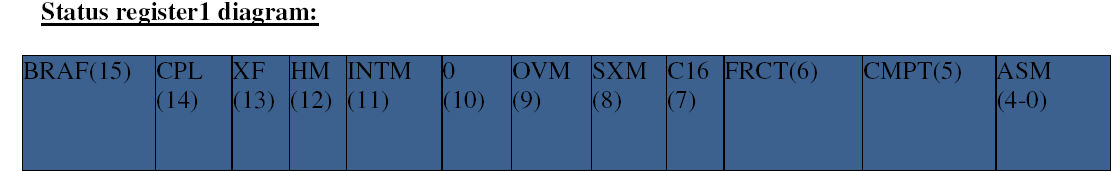


Figure 3.6(b). ST1 diagram

### BRAF: Block repeat active flag

BRAF=0, the block repeat is deactivated. BRAF=1, the block repeat is activated.

### CPL: Compiler mode

CPL=0, the relative direct addressing mode using data page pointer is selected. CPL=1, the relative direct addressing mode using stack pointer is selected.

**HM:** Hold mode, indicates whether the processor continues internal execution or acknowledge for external interface.

### INTM: Interrupt mode, it globally masks or enables all interrupts.

INTM=0\_all unmasked interrupts are enabled. INTM=1\_all masked interrupts are disabled. 0: Always read as 0

### OVM: Overflow mode.

OVM=1\_the destination accumulator is set either the most positive value or the most negative value. OVM=0\_the overflowed result is in destination accumulator.

### SXM: Sign extension mode.

SXM=0 \_Sign extension is suppressed.

SXM=1\_Data is sign extended

**C16: Dual 16 bit/double-Precision arithmetic mode.** C16=0\_ALU operates in double-Precision arithmetic mode. C16=1\_ALU operates in dual 16-bit arithmetic mode.

### FRCT: Fractional mode.

FRCT=1\_the multiplier output is left-shifted by 1bit to compensate an extra sign bit.

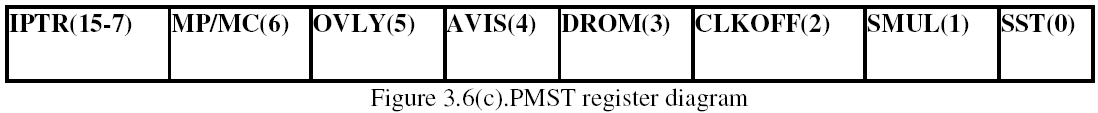
### CMPT: Compatibility mode.

CMPT=0\_ ARP is not updated in the indirect addressing mode. CMPT=1\_ARP is updated in the indirect addressing mode.

### ASM: Accumulator Shift Mode.

5 bit field, & specifies the Shift value within -16 to 15 range.

### Processor Mode Status Register (PMST):



**INTR: Interrupt vector pointer**, point to the 128-word program page where the interrupt vectors reside.

MP/MC: Microprocessor/Microcomputer mode, MP/MC=0, the on chip ROM is enabled.

MP/MC=1, the on chip ROM is enabled.

**OVLY: RAM OVERLAY,** OVLY enables on chip dual access data RAM blocks to be mapped into program space.

**AVIS:** It enables/disables the internal program address to be visible at the address pins.

**DROM: Data ROM**, DROM enables on-chip ROM to be mapped into data space. CLKOFF: CLOCKOUT off.

### SMUL: Saturation on multiplication. SST: Saturation on store.

**3.4 Data Addressing Modes of TMS320C54X Processors**:

Data addressing modes provide various ways to access operands to execute instructions and place results in the memory or the registers. The 54XX devices offer seven basic addressing modes

1. Immediate addressing.
2. Absolute addressing.
3. Accumulator addressing.
4. Direct addressing.
5. Indirect addressing.
6. Memory mapped addressing
7. Stack addressing.
   * 1. Immediate addressing:

The instruction contains the specific value of the operand. The operand can be short (3,5,8 or 9 bit in length) or long (16 bits in length). The instruction syntax for short operands occupies one memory location,

Example: LD #20, DP.

RPT #0FFFFh.

* + 1. Absolute Addressing:

The instruction contains a specified address in the operand.

1. Dmad addressing. MVDK Smem,dmad, MVDM dmad,MMR
2. Pmad addressing. MVDP Smem,pmad, MVPD pmem,Smad
3. PA addressing. PORTR PA, Smem, 4.\*(lk) addressing .
   * 1. Accumulator Addressing:

Accumulator content is used as address to transfer data between Program and Data memory.

Ex: READA \*AR2

* + 1. Direct Addressing:

Base address + 7 bits of value contained in instruction = 16 bit address. A page of 128 locations can be accessed without change in DP or SP.Compiler mode bit (CPL) in ST1 register is used.

If CPL =0 selects DP CPL = 1 selects SP,

It should be remembered that when SP is used instead of DP, the effective address is computed by adding the 7-bit offset to SP.

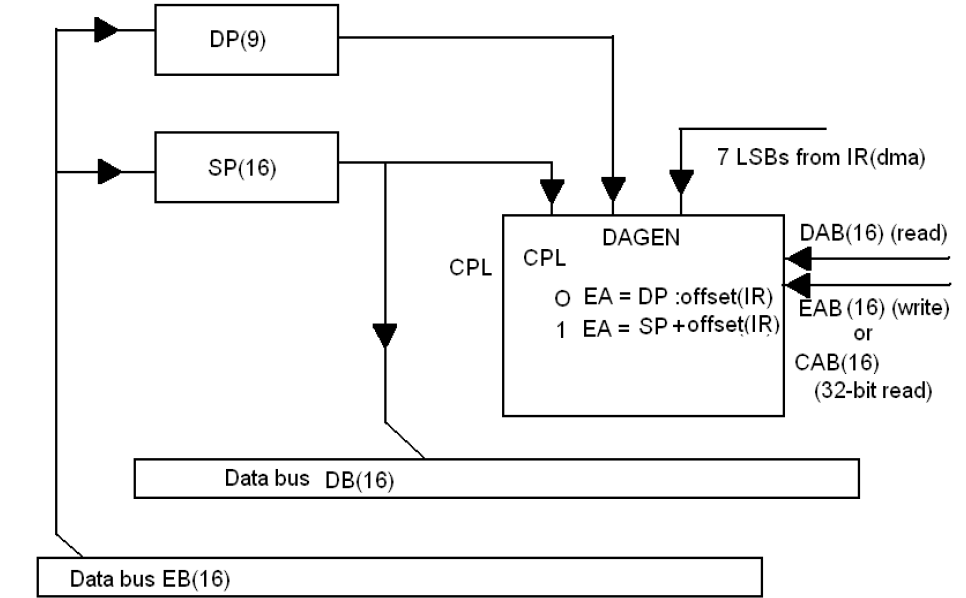


Figure 3.7 Block diagram of the direct addressing mode for TMS320C54xx Processors.

### Indirect Addressing:

* Data space isaccessed by address present in an auxiliary register.

TMS320C54xx have 8, 16 bit auxiliary register (AR0 – AR 7). Two auxiliary register arithmetic units (ARAU0 & ARAU1)

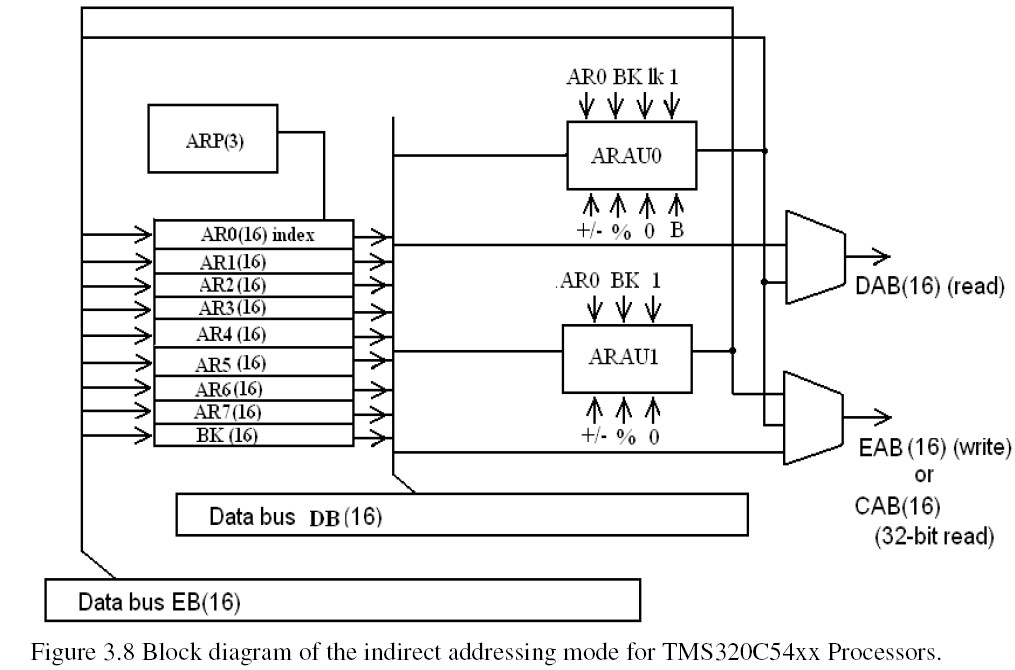
Used to access memory location in fixed step size. AR0 register is used for indexed and bit reverse addressing modes.

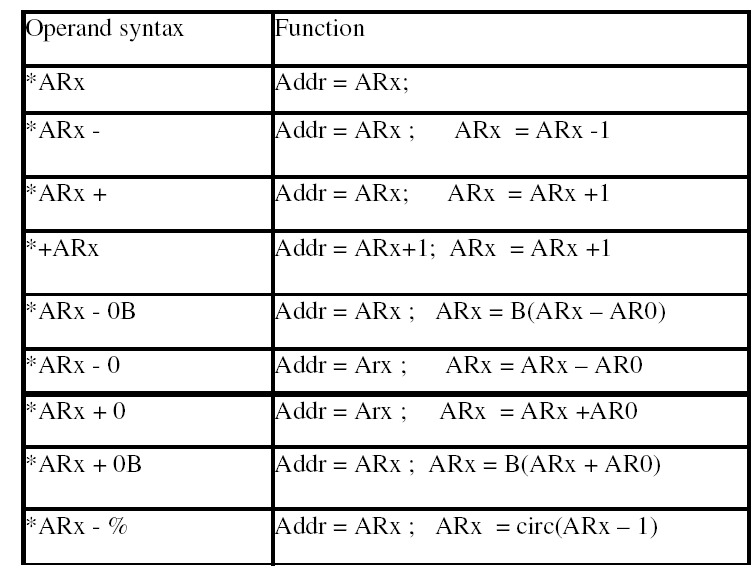
* For single– operand addressing MOD \_ type of indirect addressing ARF \_ AR used for addressing

ARP depends on (CMPT) bit in ST1

CMPT = 0, Standard mode, ARP set to zero

CMPT = 1, Compatibility mode, Particularly AR selected by ARP





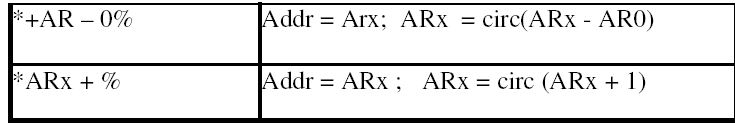
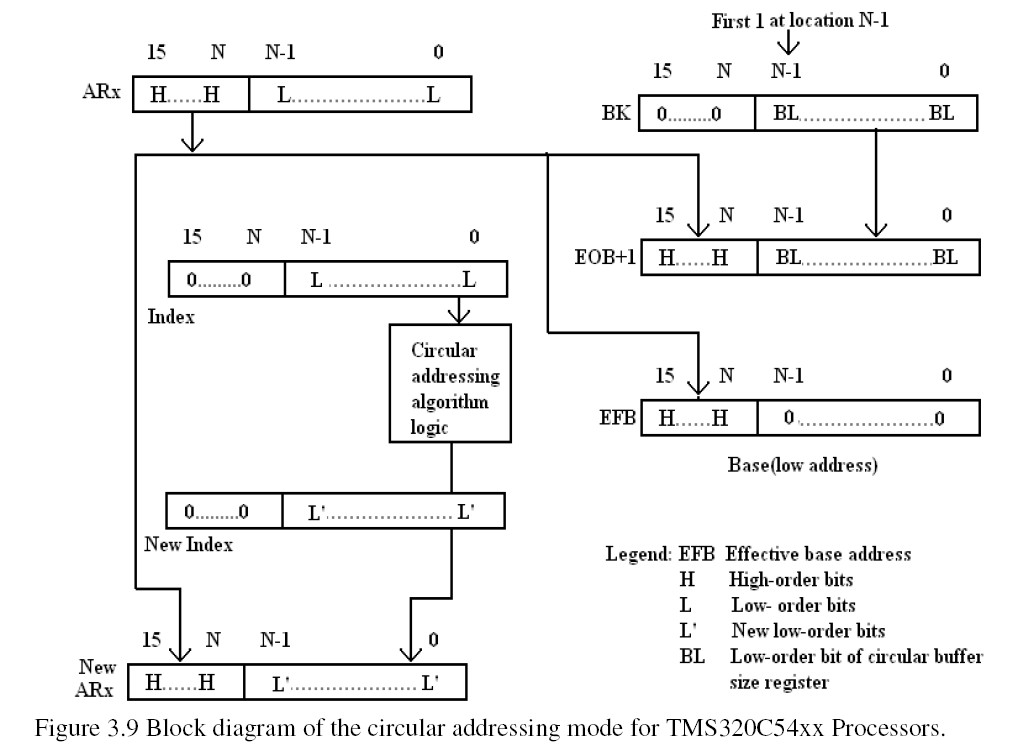


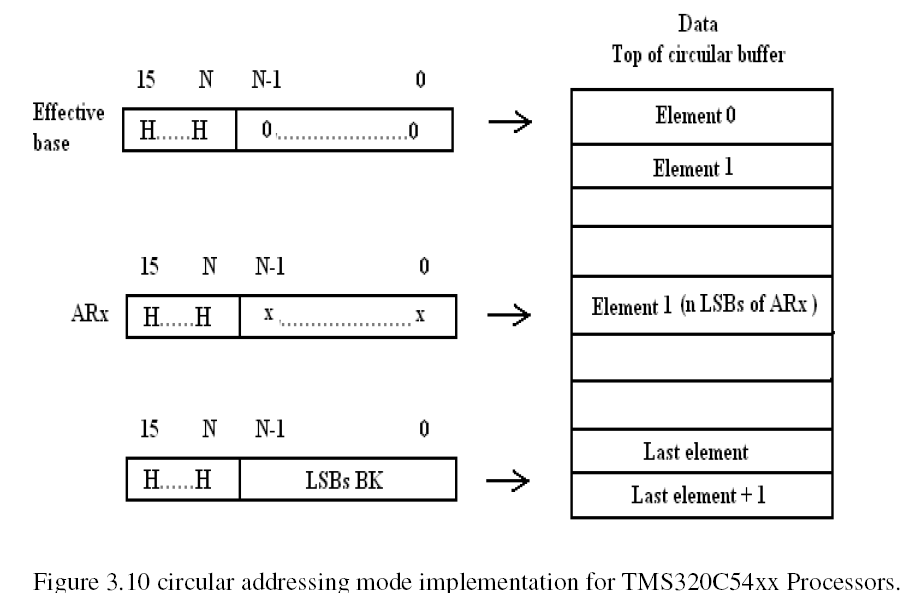
Table 3.2 Indirect addressing options with a single data –memory operand.

Circular Addressing;

* + Used in convolution, correlation and FIR filters.
  + A circular buffer is a sliding window contains most recent data. Circular buffer of size R must start on a N-bit boundary, where 2N > R .
  +  The circular buffer size register (BK): specifies the size of circular buffer.
  + Effective base address (EFB): By zeroing the N LSBs of a user selected AR (ARx).
  +  End of buffer address (EOB) : By repalcing the NLSBs of ARx with the N LSBs of BK. If 0 \_ index + step < BK ; index = index +step;

else if index + step \_ BK ; index = index + step - BK; else if index + step < 0; index + step + BK





### Bit-Reversed Addressing:

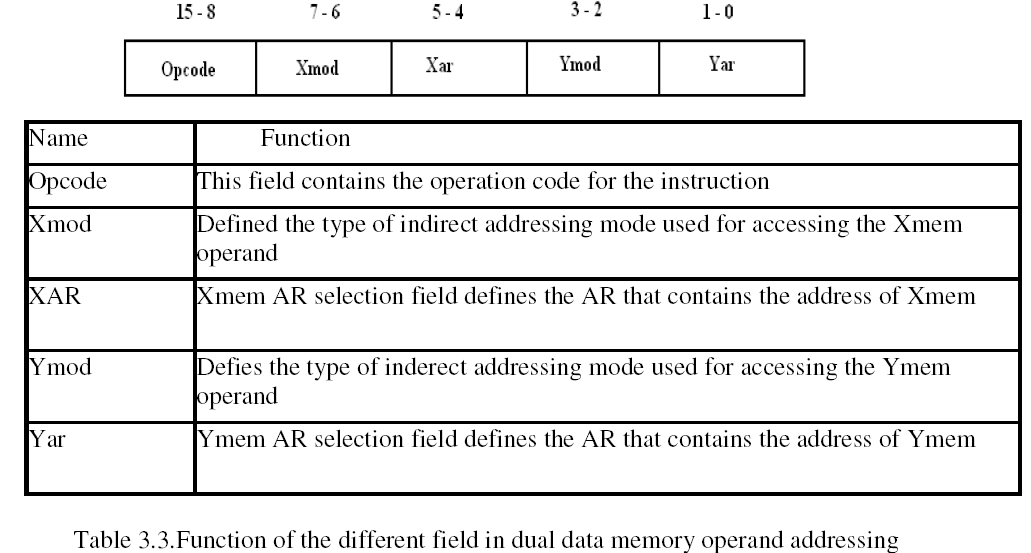
* + - Used for FFT algorithms.
    - AR0 specifies one half of the size of the FFT.
    - The value of AR0 = 2N-1: N = integer FFT size = 2N
    - AR0 + AR (selected register) = bit reverse addressing.
    - The carry bit propagating from left to right.

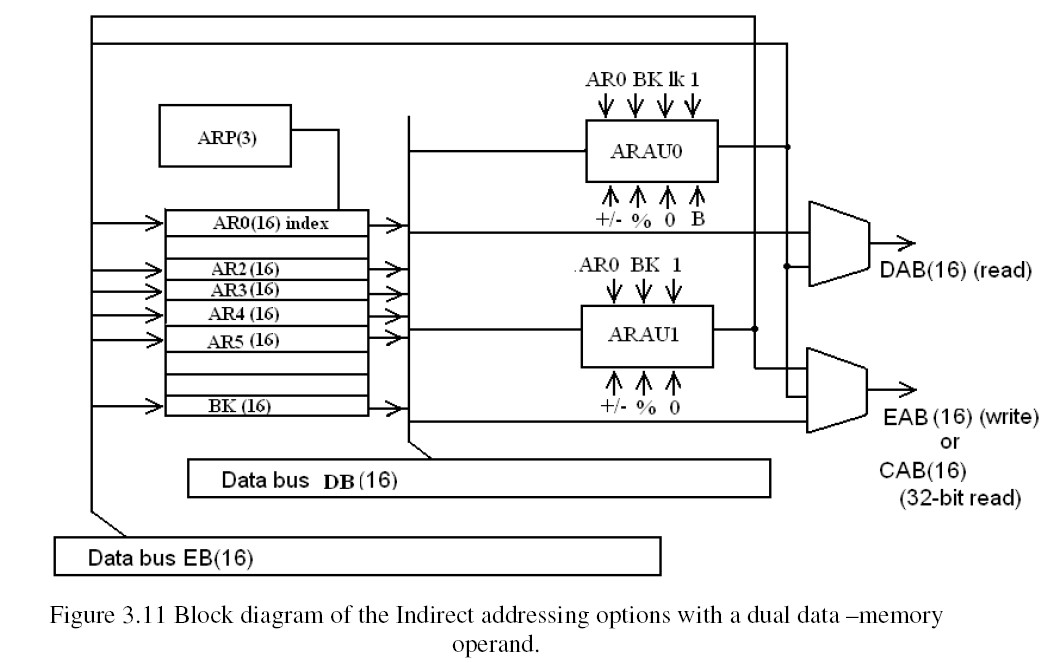
### Dual-Operand Addressing:

Dual data-memory operand addressing is used for instruction that simultaneously perform two reads (32-bit read) or a single read (16-bit read) and a parallel store (16-bit

store) indicated by two vertical bars, II. These instructions access operands using indirect addressing mode.

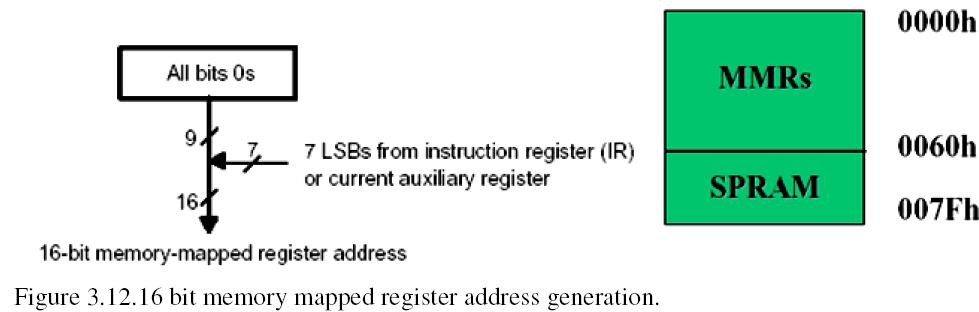
If in an instruction with a parallel store the source operand the destination operand point to the same location, the source is read before writing to the destination. Only 2 bits are available in the instruction code for selecting each auxiliary register in this mode. Thus, just four of the auxiliary registers, AR2-AR5, can be used, The ARAUs together with these registers, provide capability to access two operands in a single cycle. Figure 3.11 shows how an address is generated using dual data- memory operand addressing.





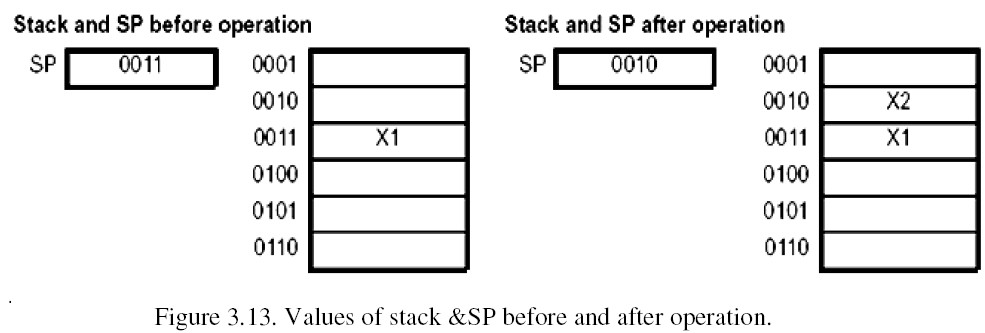
### Memory-Mapped Register Addressing:

* + - * Used to modify the memory-mapped registers without affecting the current data page
      * pointer (DP) or stack-pointer (SP)
        + Overhead for writing to a register is minimal
        + Works for direct and indirect addressing
        + Scratch –pad RAM located on data PAGE0 can be modified
      * STM #x, DIRECT
      * STM #tbl, AR1



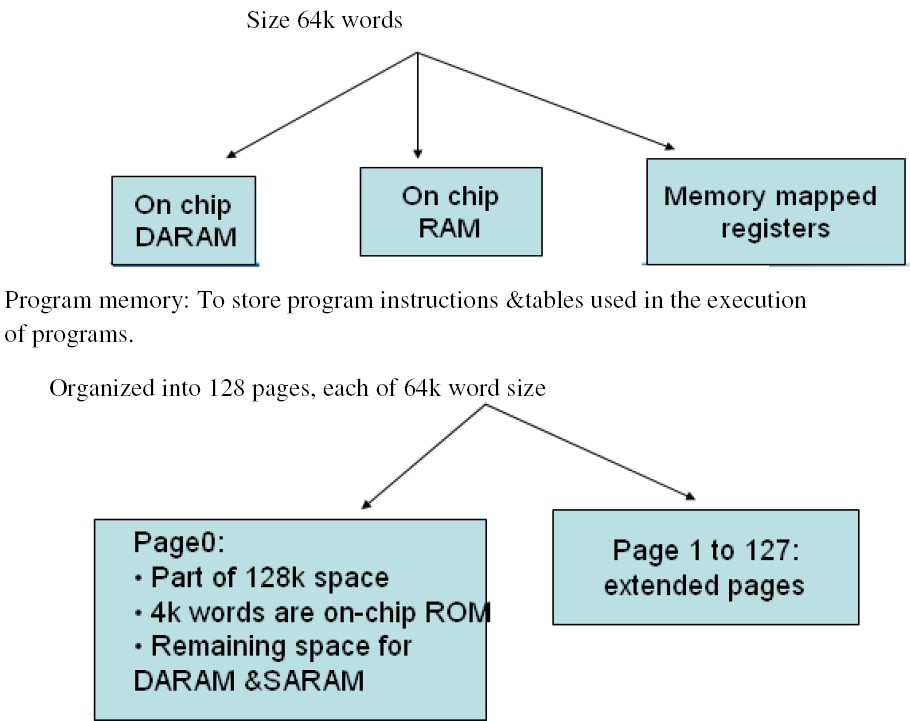
### 3.4.7 Stack Addressing:

* Used to automatically store the program counter during interrupts and subroutines.
* Can be used to store additional items of context or to pass data values.
* Uses a 16-bit memory-mapped register, the stack pointer (SP).
* PSHD X2



### Memory Space of TMS320C54xx Processors

* + - A total of 128k words extendable up to 8192k words.
    - Total memory includes RAM, ROM, EPROM, EEPROM or Memory mapped peripherals.
    -  Data memory: To store data required to run programs & for external memorymapped registers.



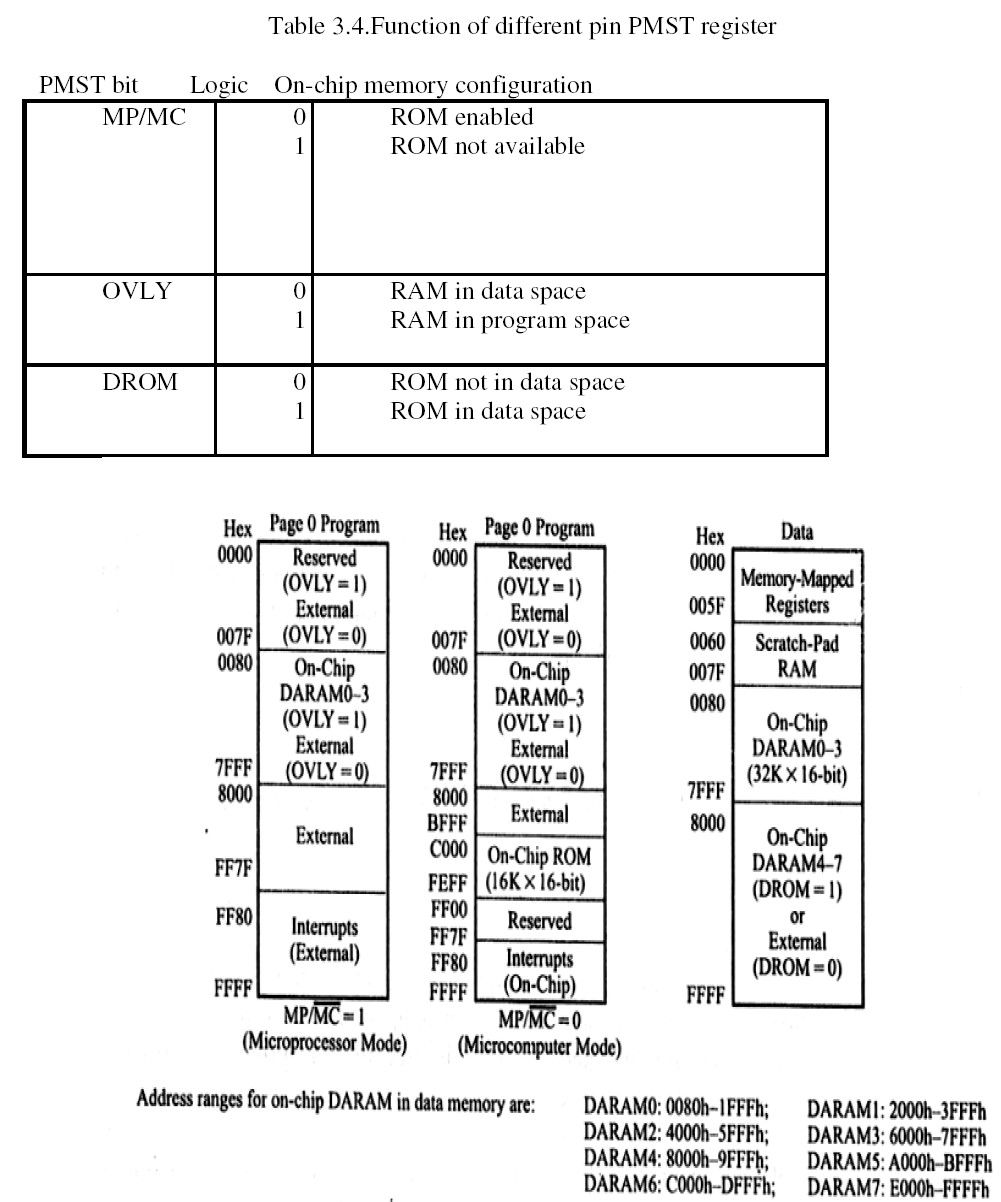


Figure 3.14 Memory map for the TMS320C5416 Processor.

### Program Control

* + - It contains program counter (PC), the program counter related H/W, hard stack, repeat counters &status registers.
    - PC addresses memory in several ways namely:
    - Branch: The PC is loaded with the immediate value following the branch instruction
    - Subroutine call: The PC is loaded with the immediate value following the call instruction
    - Interrupt: The PC is loaded with the address of the appropriate interrupt vector.
    - Instructions such as BACC, CALA, etc ;The PC is loaded with the contents of the accumulator low word
    - End of a block repeat loop: The PC is loaded with the contents of the block repeat program address start register.
    - Return: The PC is loaded from the top of the stack.

### Problems:

1. Assuming the current content of AR3 to be 200h, what will be its contents after each of the following TMS320C54xx addressing modes is used? Assume that the contents of AR0 are 20h.
2. \*AR3+0
3. \*AR3-0
4. \*AR3+
5. \*AR3
6. \*AR3

f. \*+AR3 (40h) g. \*+AR3 (-40h)

### Solution:

1. AR3 ← AR3 + AR0; AR3 = 200h + 20h = 220h
2. AR3← AR3 - AR0;

AR3 = 200h - 20h = 1E0h

1. AR3 ← AR3 + 1; AR3 = 200h + 1 = 201h
2. AR3 ← AR3 - 1; AR3 = 200h - 1 = 1FFh
3. AR3 is not modified.

AR3 = 200h

1. AR3 ← AR3 + 40h; AR3 = 200 + 40h = 240h
2. AR3 ← AR3 - 40h; AR3 = 200 - 40h = 1C0h
3. Assuming the current contents of AR3 to be 200h, what will be its contents after

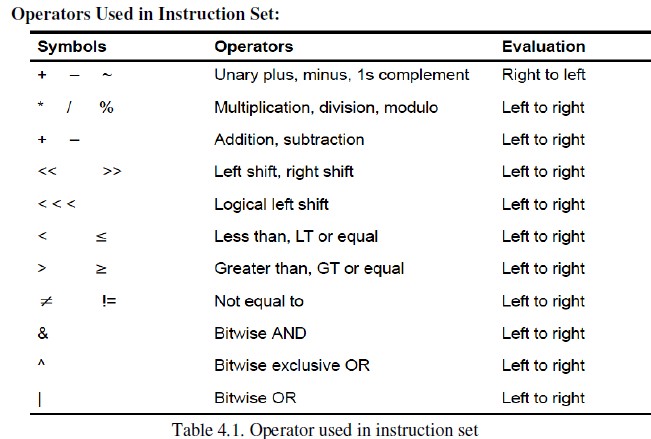
each of the following TMS320C54xx addressing modes is used? Assume that the contents of AR0 are 20h

1. \*AR3 + 0B
2. \*AR3 – 0B

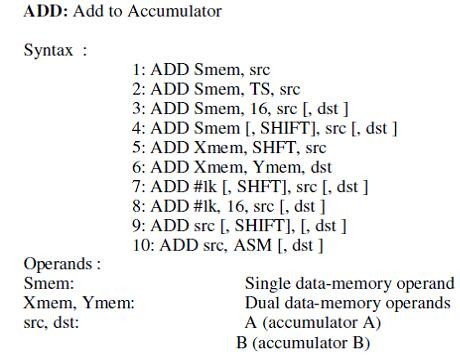
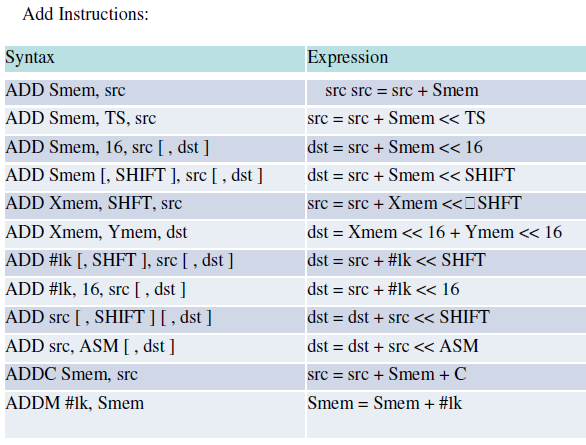
### Solution:

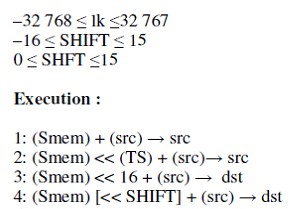
1. AR3 ← AR3 + AR0 with reverse carry propagation; AR3 = 200h + 20h (with reverse carry propagation) = 220h.
2. AR3 ← AR3 - AR0 with reverse carry propagation; AR3 = 200h - 20h (with reverse carry propagation) = 23Fh.

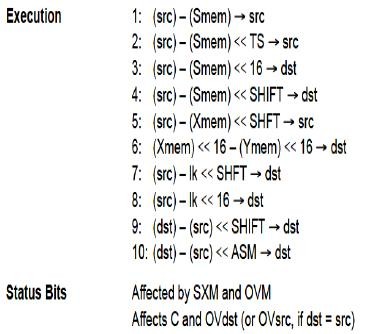
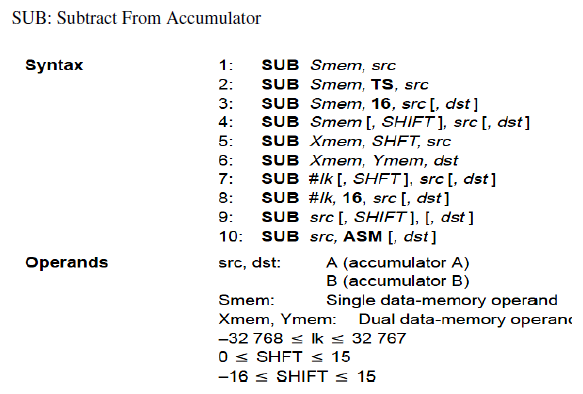
## UNIT-3

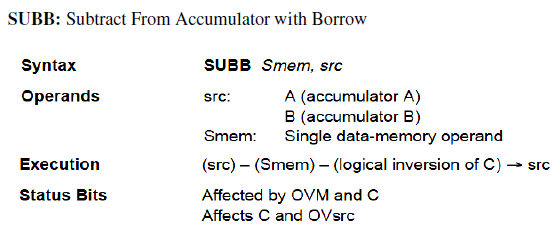


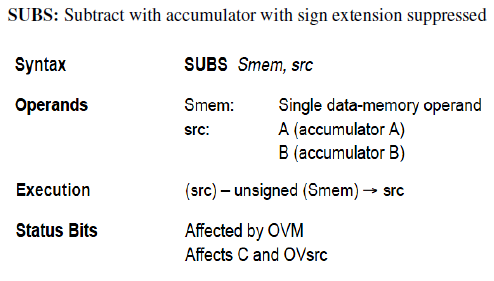
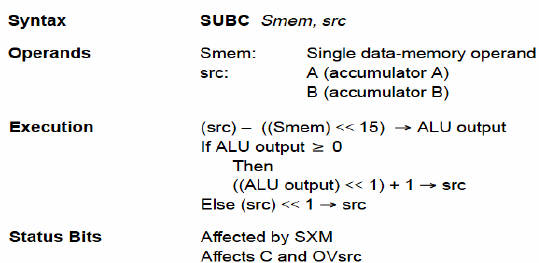
### 4.1.1 Arithmetic Instructions:

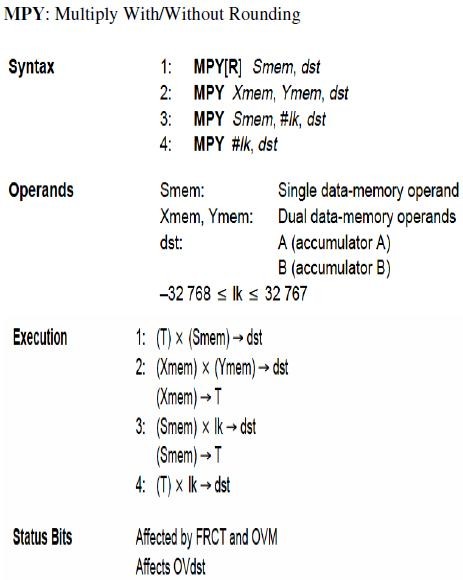


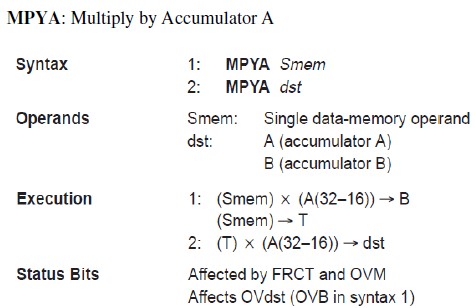


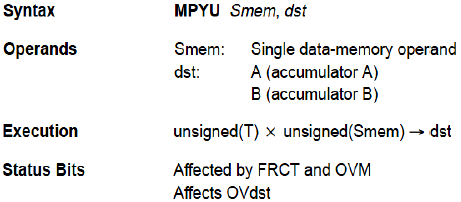


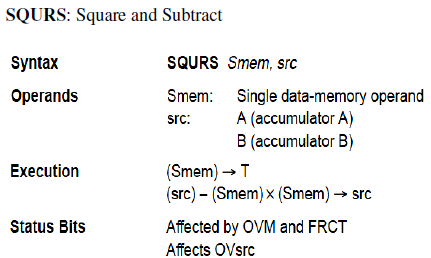
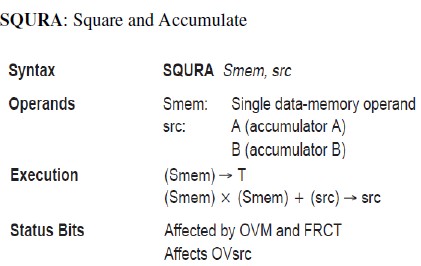
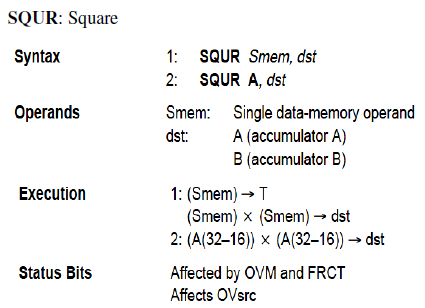


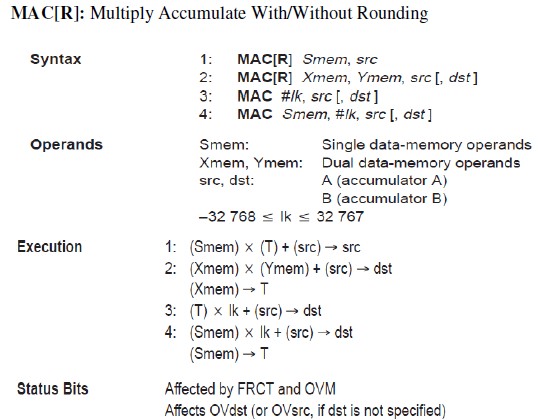


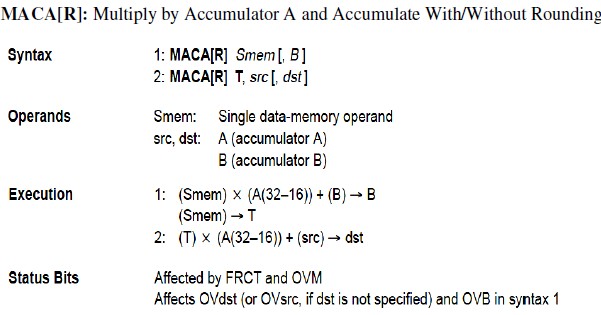


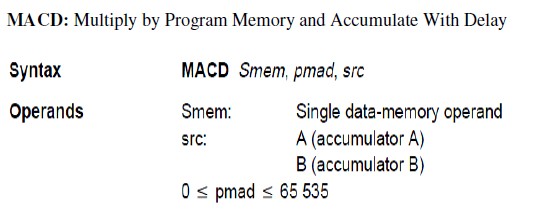


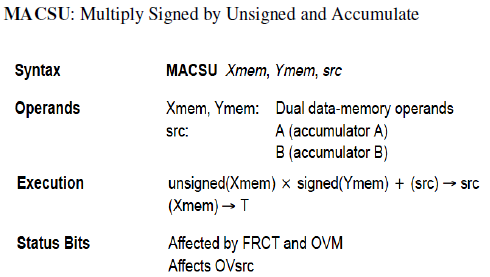
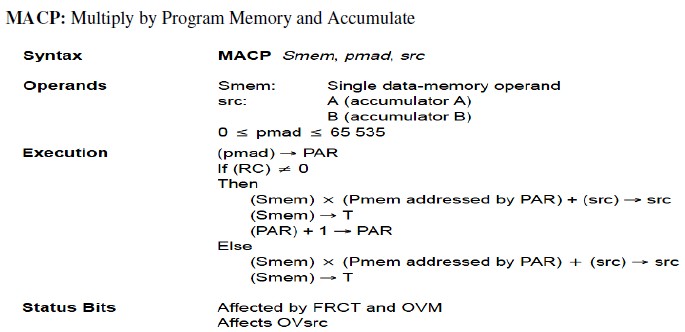


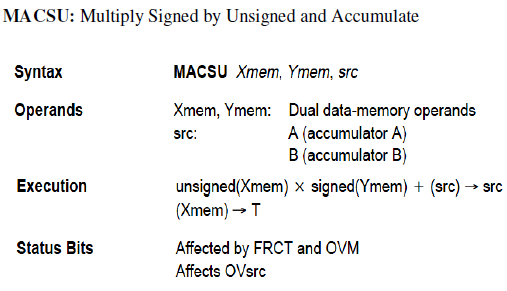


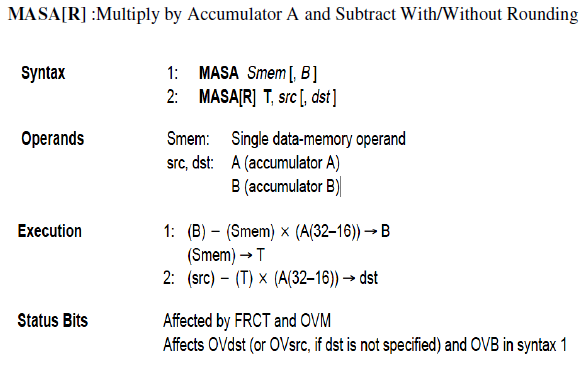
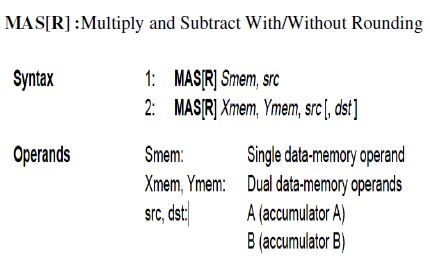


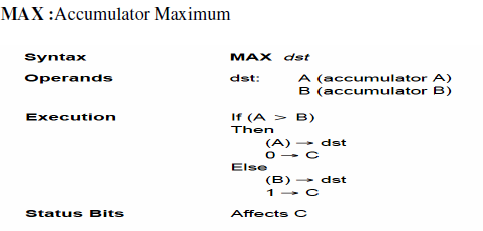


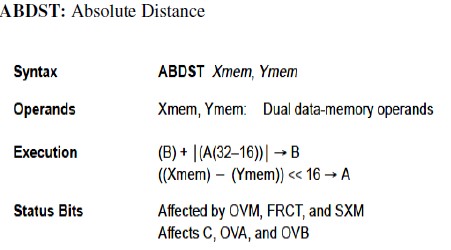
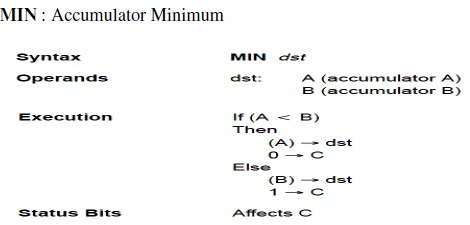


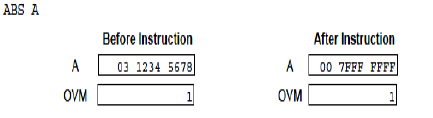


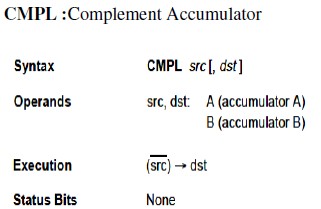


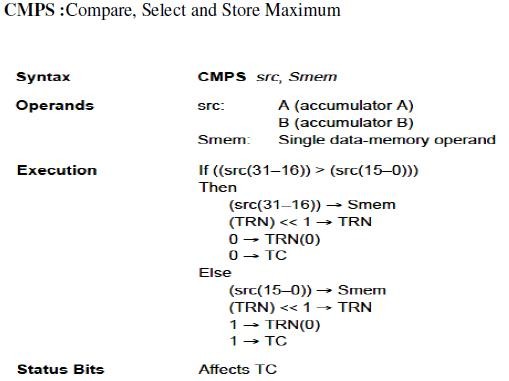
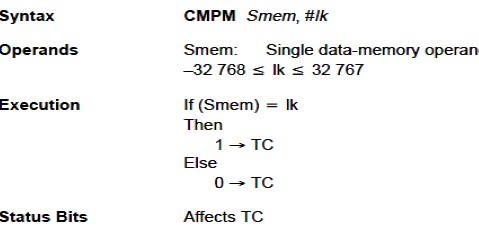


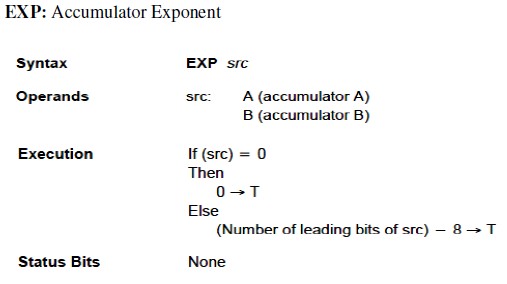


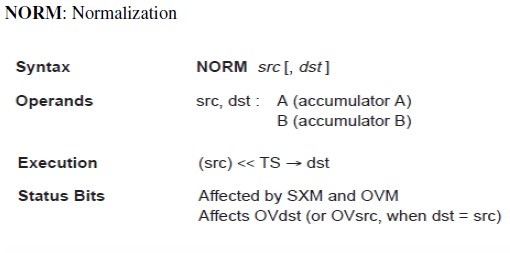
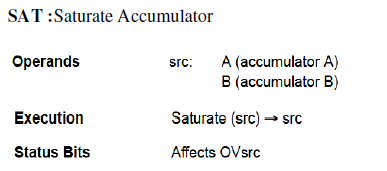


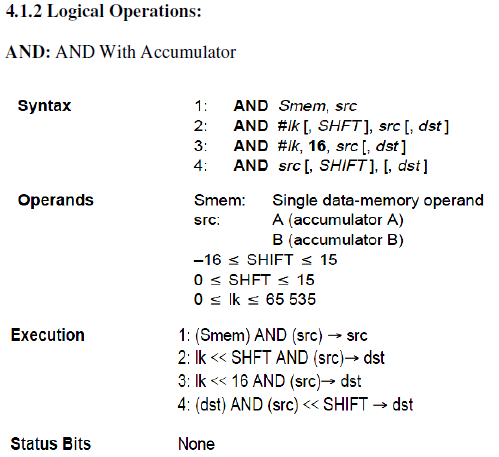




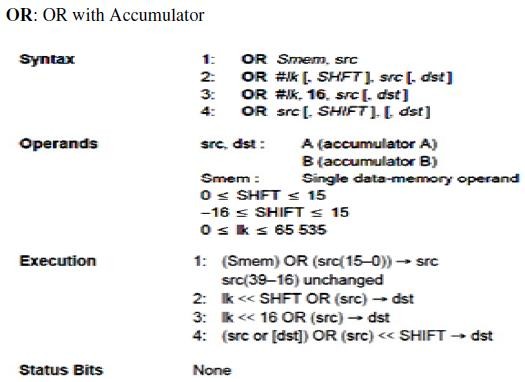


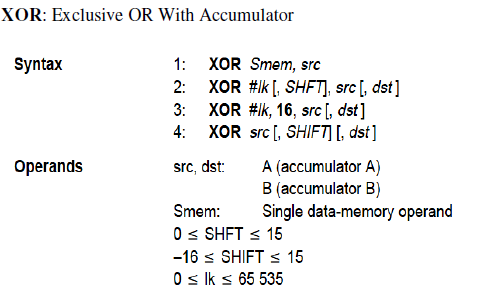
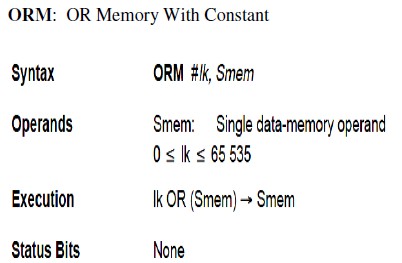


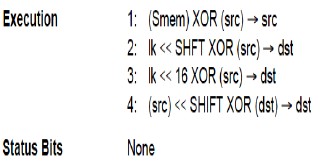


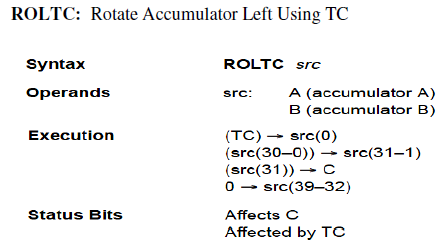
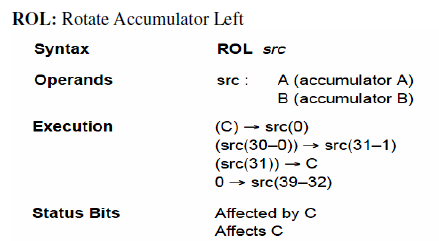
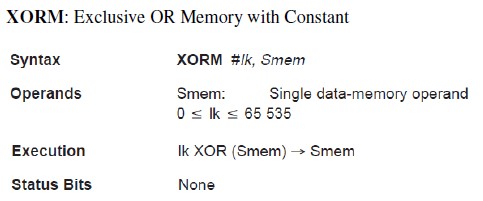


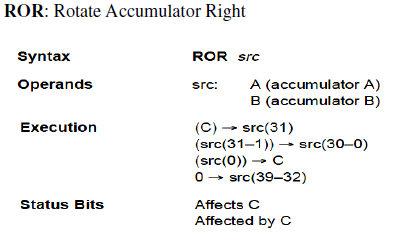


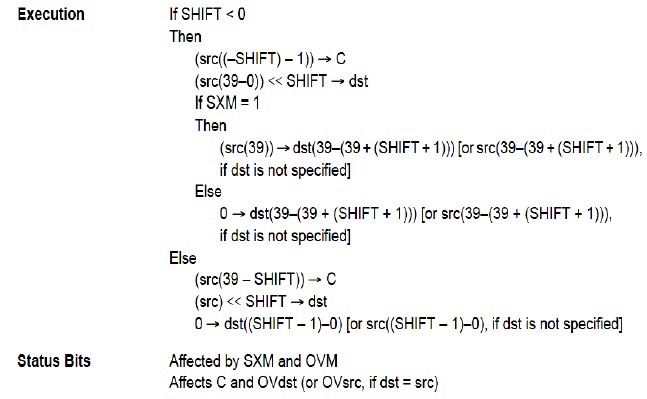
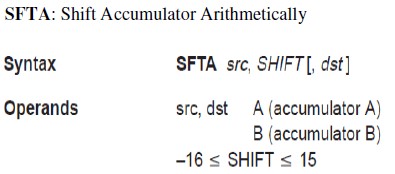


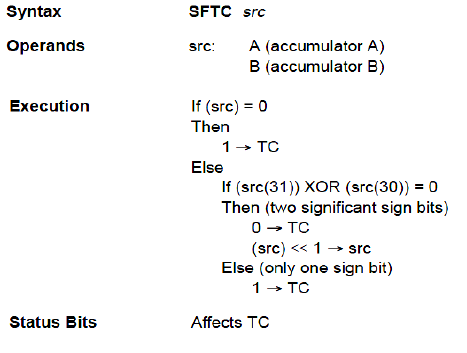


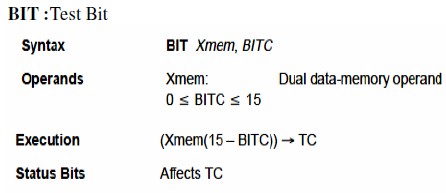
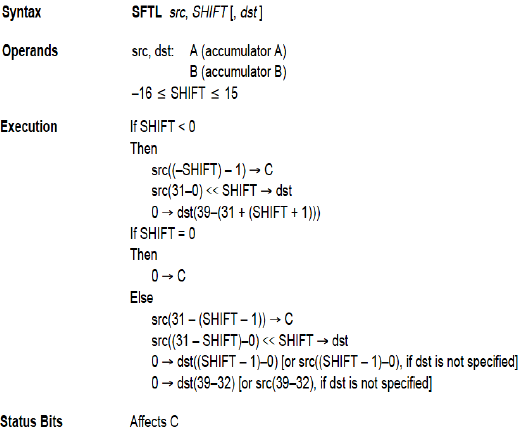












It facilitates interfacing with external devices. The peripherals are:

* + - General purpose I/O pins
    - A software programmable wait state generator.
    - Hardware timer
    - Host port interface (HPI)
    - Clock generator
    - Serial port

### It has two general purpose I/O pins:

* + - * BIO-input pin used to monitor the status of external devices.
      * XF- output pin, software controlled used to signal external devices

### Software programmable wait state generator:

* + - * Extends external bus cycles up to seven machine cycles.

### Hardware Timer

* + - *  An on chip down counter
      *  Used to generate signal to initiate any interrupt or any other process
* Consists of 3 memory mapped registers:
  + The timer register (TIM)
  + Timer period register (PRD)
  + Timer controls register (TCR)
* Pre scaler block (PSC).
* TDDR (Time Divide Down ratio)
* TIN &TOUT

The timer register (TIM) is a 16-bit memory-mapped register that decrements at every pulse from the prescaler block (PSC).

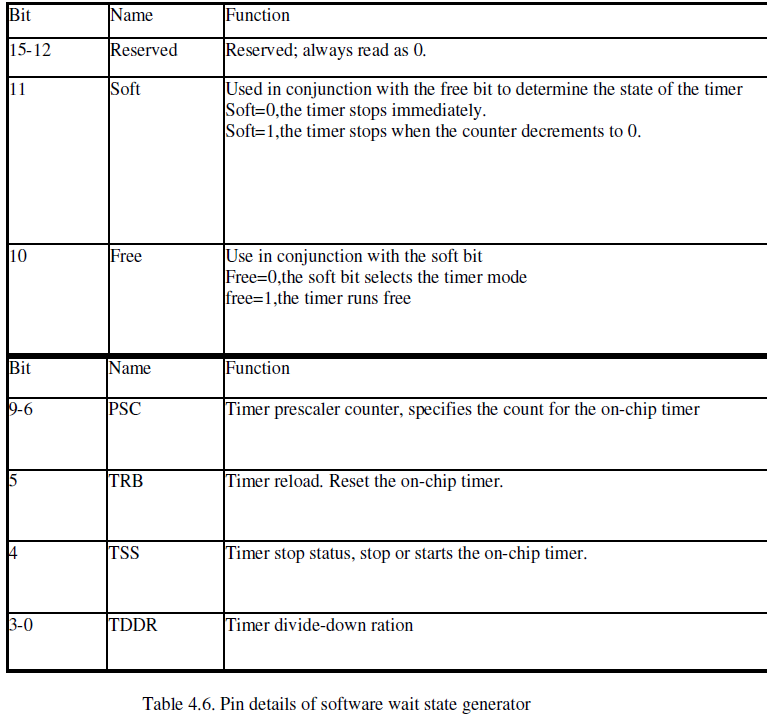
The timer period register (PRD) is a 16-bit memory-mapped register whose contents are loaded onto the TIM whenever the TIM decrements to zero or the device is reset (SRESET).

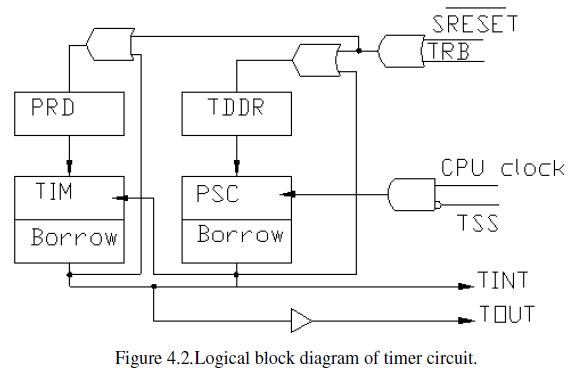
The timer can also be independently reset using the TRB signal. The timer control register (TCR) is a 16-bit memory-mapped register that contains status and control bits. Table shows the functions of the various bits in the TCR.

The prescaler block is also an on-chip counter. Whenever the prescaler bits count down to 0, a clock pulse is given to the TIM register that decrements the TIM register by 1. The TDDR bits contain the divide-down ratio, which is loaded onto the prescaler block after each time the prescaler bits count down to 0.

That is to say that the 4-bit value of TDDR determines the divide-by ratio of the timer clock with respect to the system clock. In other words, the TIM decrements either at the rate of the system clock or at a rate slower than that as decided by the value of the TDDR bits. TOUT and TINT are the output signal generated as the TIM register decrements to 0. TOUT can trigger the start of the conversion signal in an ADC interfaced to the DSP.

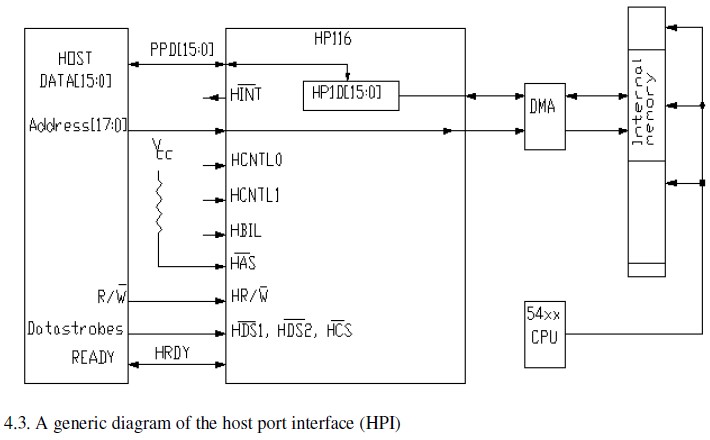
The sampling frequency of the ADC determines how frequently it receives the TOUT signal. TINT is used to generate interrupts, which are required to service a peripheral such as a DRAM controller periodically. The timer can also be stopped, restarted, reset, or disabled by specific status bits.





### Host port interface (HPI):

* Allows to interface to an 8bit or 16bit host devices or a host processor
* Signals in HPI are:
* Host interrupt (HINT)
* HRDY
* HCNTL0 &HCNTL1
* HBIL
* HR/w



Important signals in the HPI are as follows:

* The 16-bit data bus and the 18-bit address bus.
* The host interrupt, Hint, for the DSP to signal the host when it attention is required.
* HRDY, a DSP output indicating that the DSP is ready for transfer.
* HCNTL0 and HCNTL1, control signal that indicate the type of transfer to carry out. The transfer types are data, address, etc.
* HBIL. If this is low it indicates that the current byte is the first byte; if it is high, it indicates that it is second byte.
* HR/W indicates if the host is carrying out a read operation or a write operation

### Clock Generator:

The clock generator on TMS320C54xx devices has two options-an external clock

and the internal clock. In the case of the external clock option, a clock source is directly connected to the device. The internal clock source option, on the other hand, uses an internal clock generator and a phase locked loop (PLL) circuit. The PLL, in turn, can be hardware configured or software programmed. Not all devices of the TMS320C54xx family have all these clock options; they vary from device to device.

### Serial I/O Ports:

Three types of serial ports are available:

* Synchronous ports.
* Buffered ports.
* Time-division multiplexed ports.

The synchronous serial ports are high-speed, full-duplex ports and that provide direct communications with serial devices, such as codec, and analog-to-digital (A/D) converters. A buffered serial port (BSP) is synchronous serial port that is provided with

an auto buffering unit and is clocked at the full clock rate. The head of servicing interrupts. A time- division multiplexed (TDM) serial port is a synchronous serial port that is provided to allow time- division multiplexing of the data. The functioning of each of these on-chip peripherals is controlled by memory-mapped registers assigned to the respective peripheral.

### Interrupts of TMS320C54xx Processors:

Many times, when CPU is in the midst of executing a program, a peripheral device may require a service from the CPU. In such a situation, the main program may be interrupted by a signal generated by the peripheral devices. This results in the processor suspending the main program in order to execute another program, called interrupt service routine, to service the peripheral device. On completion of the interrupt service routine, the processor returns to the main program to continue from where it left.

Interrupt may be generated either by an internal or an external device. It may also be generated by software. Not all interrupts are serviced when they occur. Only those interrupts that are called *nonmaskable* are serviced whenever they occur. Other interrupts, which are called *maskable* interrupts, are serviced only if they are enabled. There is also a priority to determine which interrupt gets serviced first if more than one interrupts occur simultaneously.

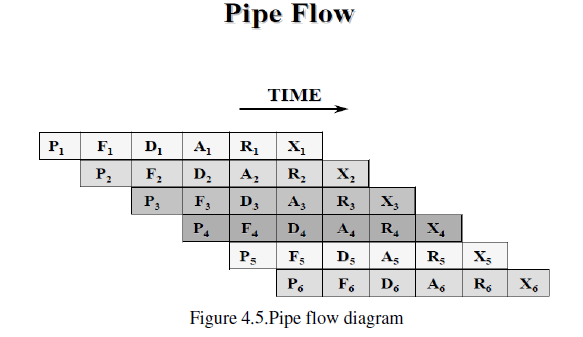
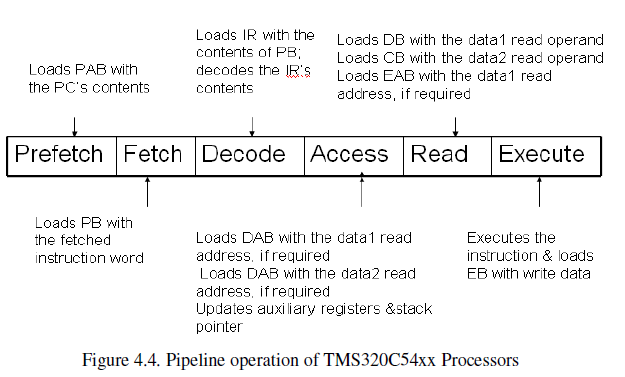
Almost all the devices of TMS320C54xx family have 32 interrupts. However, the

types and the number under each type vary from device to device. Some of these interrupts are reserved for use by the CPU.

### Pipeline operation of TMS320C54xx Processors:

The CPU of ‘54xx devices have a six-level-deep instruction pipeline. The six stages of the pipeline are independent of each other. This allows overlapping execution of instructions. During any given cycle, up to six different instructions can be active, each at a different stage of processing. The six levels of the pipeline structure are program prefetch, program fetch, decode, access, read and execute.

1. During program prefetch, the program address bus, PAB, is loaded with the address of the next instruction to be fetched.
2. In the fetch phase, an instruction word is fetched from the program bus, PB, and loaded into the instruction register, IR. These two phases from the instruction fetch sequence.
3. During the decode stage, the contents of the instruction register, IR are decoded to determine the type of memory access operation and the control signals required for the data-address generation unit and the CPU.
4. The access phase outputs the read operand’s on the data address bus, DAB. If a second operand is required, the other data address bus, CAB, also loaded with an appropriate address. Auxiliary registers in indirect addressing mode and the stack pointer (SP) are also updated.
5. In the read phase the data operand(s), if any, are read from the data buses, DB and CB. This phase completes the two-phase read process and starts the two phase write processes. The data address of the write operand, if any, is loaded into the data write address bus, EAB.
6. The execute phase writes the data using the data write bus, EB, and completes the operand write sequence. The instruction is executed in this phase.



## UNIT-4

### Introduction:

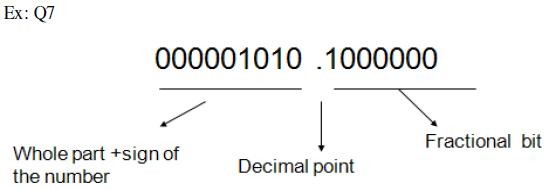
In this unit, we deal with implementations of DSP algorithms & write programs to implement the core algorithms only. However, these programs can be combined with input/output routines to create applications that work with a specific hardware.

* + - Q-notation
    - FIR filters
    - IIR filters
    - Interpolation filters
    - Decimation filters

### The Q-notation:

DSP algorithm implementations deal with signals and coefficients. To use a fixed point DSP device efficiently, one must consider representing filter coefficients and signal samples using fixed- point2’s complement representation. Ex: N=16, Range: -2N-1 to +2N-1 -1(-32768 to 32767).Typically, filter coefficients are fractional numbers.

To represent such numbers, the Q-notation has been developed. The Q-notation specifies the number of fractional bits.

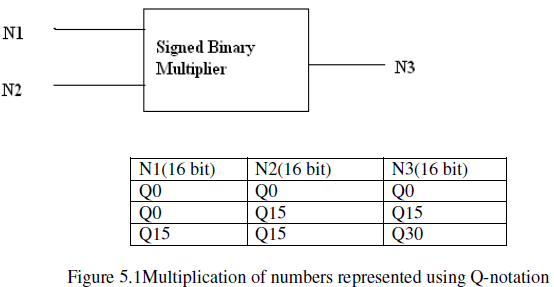


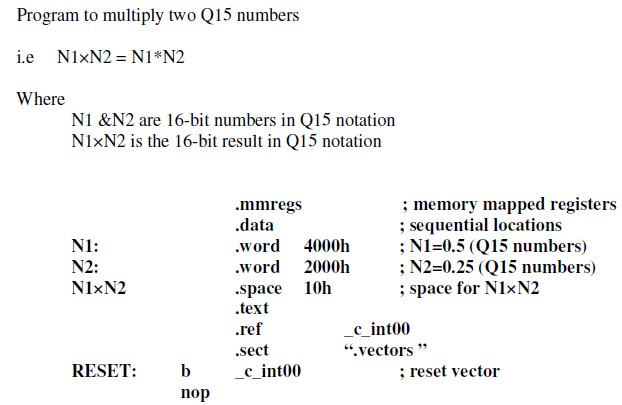
A commonly used notation for DSP implementations is Q15. In the Q15 representation, the least significant 15 bits represent the fractional part of a number. In a processor where 16 bits are used to represent numbers, the Q15 notation uses the MSB to represent the sign of the number and the rest of the bits represent the value of the number.

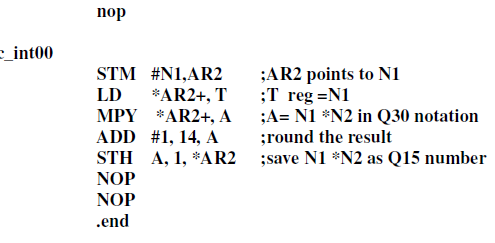
In general, the value of a 16-bit Q15 number N represented as:



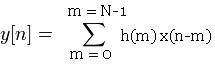
Multiplication of numbers represented using the Q-notation is important for DSP implementations. Figure 5.1(a) shows typical cases encountered in such implementations.







### FIR Filters:

A finite impulse response (FIR) filter of order N can be described by the difference equation.

The expanded form is y(n)=h(N-1)x(n-(N-1))+h(N-2)x(n-(N-2))+ ...h(1)x(n-1)+h(0)x(n)

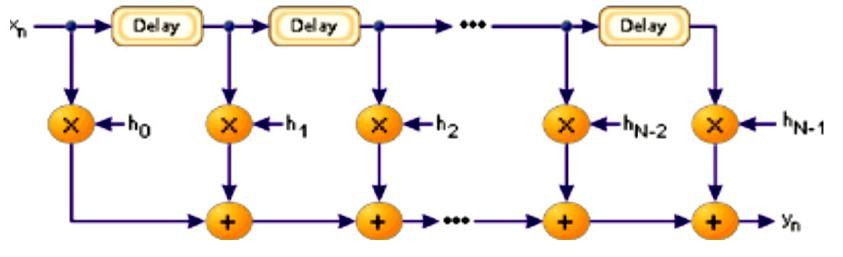
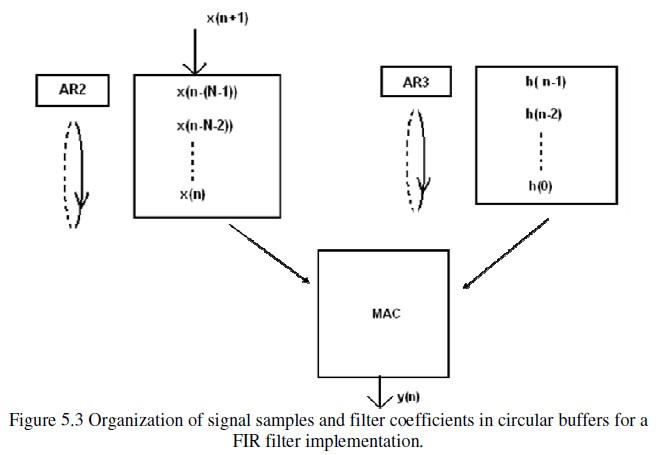


Figure 5.2 A FIR filter implementation block diagram

The implementation requires signal delay for each sample to compute the next output,

y(n+1), is given as y(n+1)=h(N-1)x(n-(N-2))+h(N-2)x(n-(N-3))+ ...h(1)x(n)+h(0)x(n+1) Figure 5.3 shows the memory organization for the implementation of the filter. The filter Coefficients and the signal samples are stored in two circular buffers each of a size equal to the filter. AR2 is used to point to the samples and AR3 to the coefficients. In order to start with the last product, the pointer register AR2 must be initialized to access the signal sample x(2-(N-1)), and the pointer register AR3 to access the filter coefficient h(N-1). As each product is computed and added to the previous result, the pointers advance circularly. At the end of the computation, the signal sample pointer is at the oldest sample, which is replaced with the newest sample to proceed with the next output computation.



### Program to implement an FIR filter:

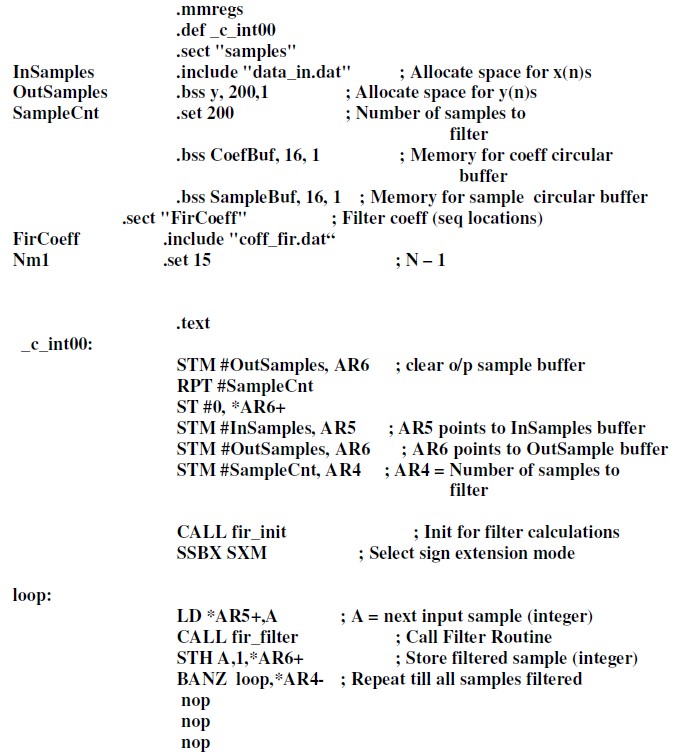
It implements the following equation;

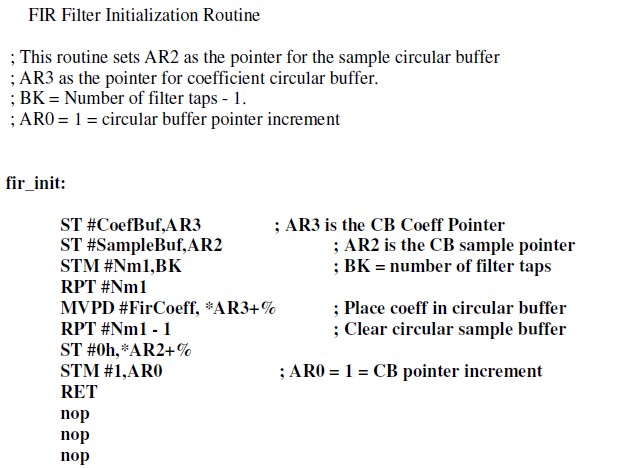
y(n)=h(N-1)x(n-(N-1))+h(N-2)x(n-(N-2))+ ...h(1)x(n-1)+h(0)x(n)

Where N = Number of filter coefficients = 16.

h(N-1), h(N-2),...h(0) etc are filter coefficients (q15numbers) . The coefficients are available in file: coeff\_fir.dat.

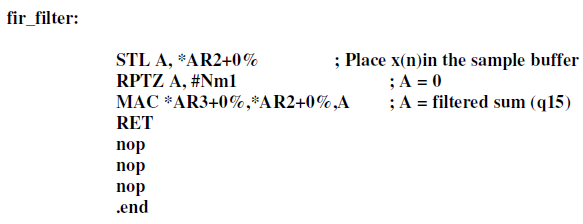
x(n-(N-1)),x(n-(N-2),...x(n) are signal samples(integers). The input x(n) is received from the data file: data\_in.dat. The computed output y(n) is placed in a data buffer.





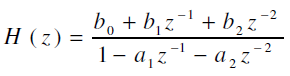
### FIR Filter Routine

; Enter with A=the current sample x(n)-an integer, AR2 pointing to the location for the current sample x(n),andAR3pointingtotheq15coefficienth(N-1). Exit with A = y(n) as q15 number.



### IIR Filters:

An infinite impulse response (IIR) filter is represented by a transfer function, which is a ratio of two polynomials in z. To implement such a filter, the difference equation representing the transfer function can be derived and implemented using multiply and add operations. To show such an implementation, we consider a second order transfer function given by



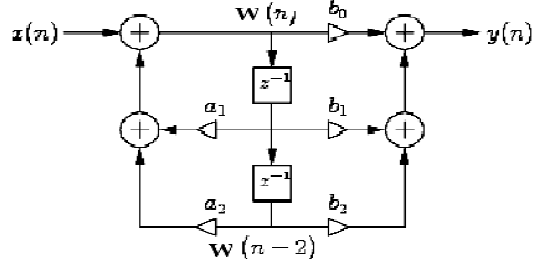
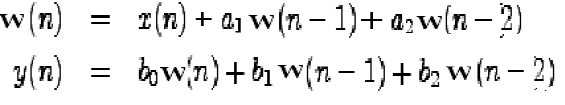


Figure5.4 Block diagram of second order IIR filter



### Program for IIR filter:

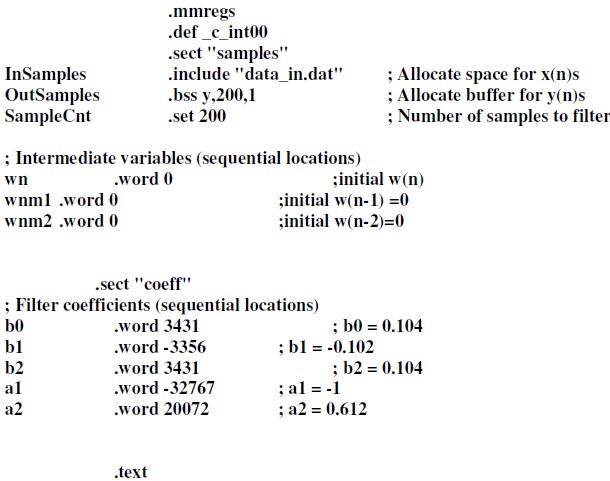
The transfer function is

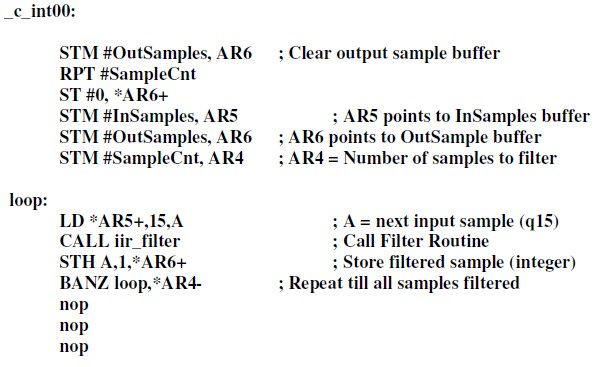


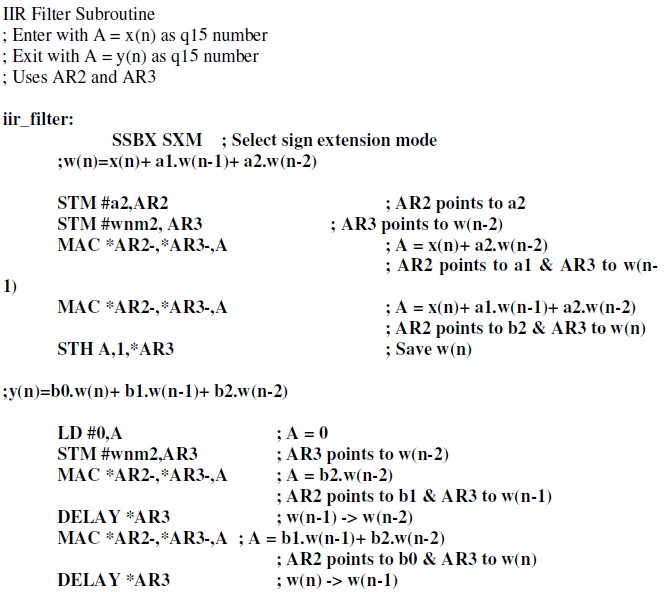
Which is equivalent to the equations: w(n) = x(n) + a1.w(n-1) + a2.w(n-2)

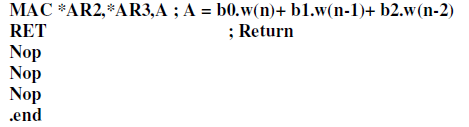
y(n) = b0.w(n) + b1.w(n-1) + b2.w(n-2)

Where w(n), w(n-1), and w(n-2) are the intermediate variables used in computations (integers).a1, a2, b0, b1, and b2 are the filter coefficients (q15 numbers). x(n) is the input sample (integer). Input samples are placed in the buffer, In Samples, from a data file, data\_in.dat y(n) is the computed output (integer). The output samples are placed in a buffer, Out Samples.









### Interpolation Filters:

An *interpolation filter* is used to increase the sampling rate. The interpolation process involves inserting samples between the incoming samples to create additional samples to increase the sampling rate for the output. One way to implement an interpolation filter is to first insert zeros between samples of the original sample sequence. The zero-inserted sequence is then passed through an appropriate lowpass digital FIR filter to generate the interpolated sequence. The interpolation process is depicted in Figure 5.5

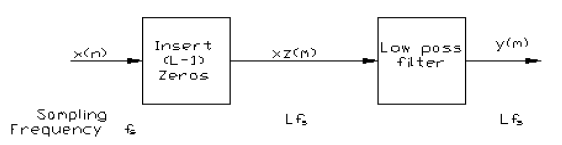
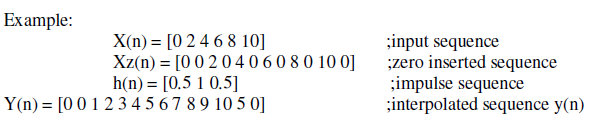


Figure 5.5 :The interpolation process



The kind of interpolation carried out in the examples is called *linear interpolation* because the convolving sequence h(n) is derived based on linear interpolation of samples. Further, in this case, the h(n) selected is just a second-order filter and therefore uses just two adjacent samples to interpolate a sample. A higher-order filter can be used to base interpolation on more input samples. To implement an ideal interpolation. Figure 5.6 shows how an interpolating filter using a 15-tap FIR filter and an interpolation factor of 5 can be implemented. In this example, each incoming samples is followed by four zeros to increase the number of samples by a factor of 5.

The interpolated samples are computed using a program similar to the one used for a FIR filter implementation. One drawback of using the implementation strategy depicted in Figure 5.7 is that there are many multiplies in which one of the multiplying elements is zero. Such multiplies need not be included in computation if the computation is rearranged to take advantage of this fact. One such scheme, based on generating what are called *poly-phase sub-filters*, is available for reducing the computation. For a case where the number of filter coefficients N is a multiple of the interpolating factor L, the scheme implements the interpolation filter using the equation.

Figure 5.7 shows a scheme that uses poly-phase sub-filters to implement the interpolating filter using the 15-tap FIR filter and an interpolation factor of 5. In this implementation, the 15 filter taps are arranged as shown and divided into five 3-tap sub filters. The input samples x(n), x(n-1) and x(n-2) are used five times to generate the five output samples. This implementation requires 15 multiplies as opposed to 75 in the direct implementation of Figure 5.7.

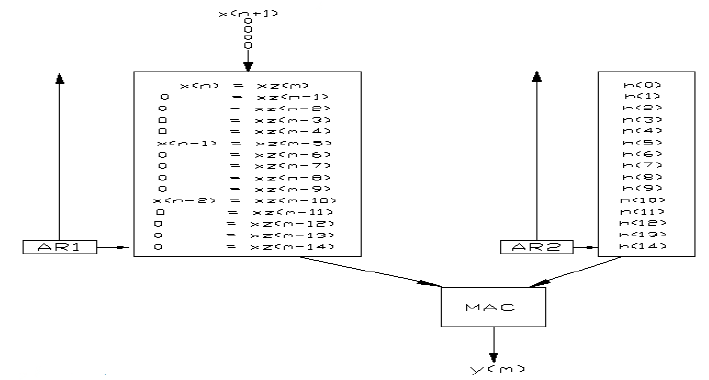


Figure 5.6 interpolating filter using a 15-tap FIR filter and an interpolation factor of 5

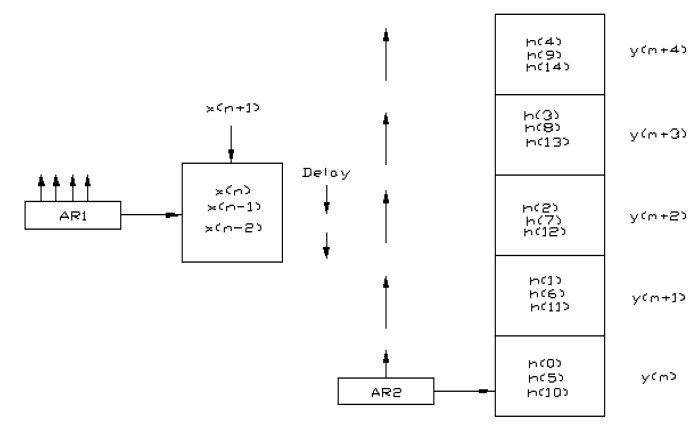
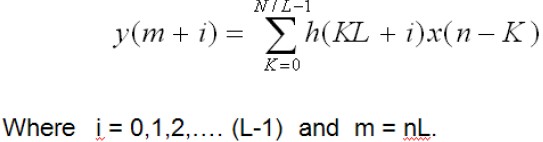


Figure5.7: A scheme that uses poly-phase sub-filters to implement the interpolating filter

Using the 15-tap FIR filter and an interpolation factor of 5



### Decimation Filters:

A decimation filter is used to decrease the sampling rate. The decrease in sampling rate can be achieved by simply dropping samples. For instance, if every other

sample of a sampled sequence is dropped, the sampling the rate of the resulting sequence will be half that of the original sequence. The problem with dropping samples is that the new sequence may violate the sampling theorem, which requires that the sampling frequency must be greater than two times the highest frequency contents of the signal.

To circumvent the problem of violating the sampling theorem, the signal to be decimated is first filtered using a low pass filter. The cutoff frequency of the filter is chosen so that it is less than half the final sampling frequency. The filtered signal can be

decimated by dropping samples. In fact, the samples that are to be dropped need not be computed at all. Thus, the implementation of a decimator is just a FIR filter implementation in which some of the outputs are not calculated.

Figure 5.8 shows a block diagram of a decimation filter. Digital decimation can be implemented as depicted in Figure 5.9 for an example of a decimation filter with decimation factor of

1. It uses a low pass FIR filter with 5 taps. The computation is similar to that of a FIR filter. However, after computing each output sample, the signal array is delayed by three sample intervals by bringing the next three samples into the circular buffer to replace the three oldest samples.

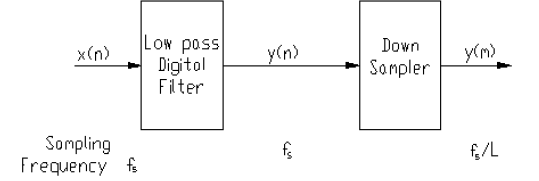
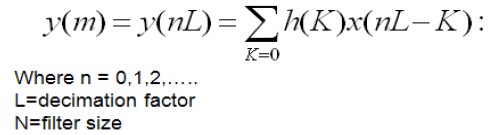


Figure 5.8: The decimation process



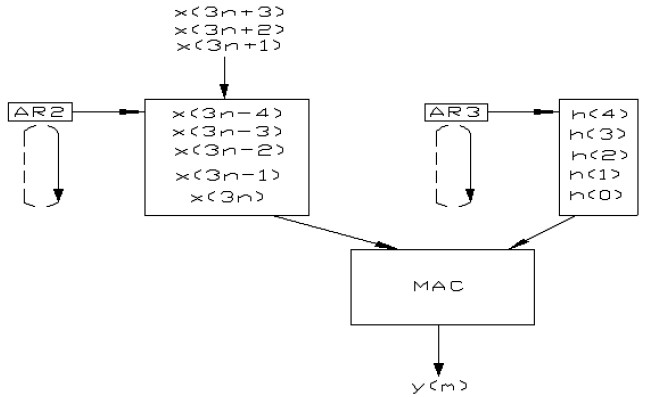


Figure 5.9: Implementation of decimation filter

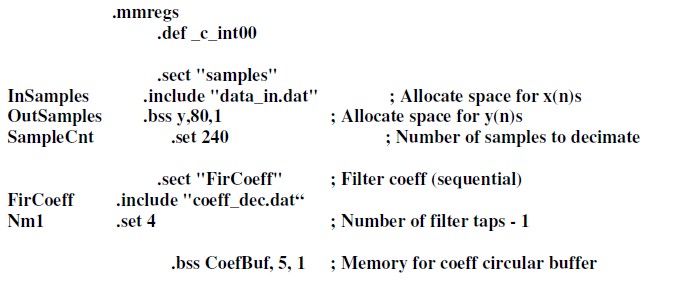
### Implementation of decimation filter

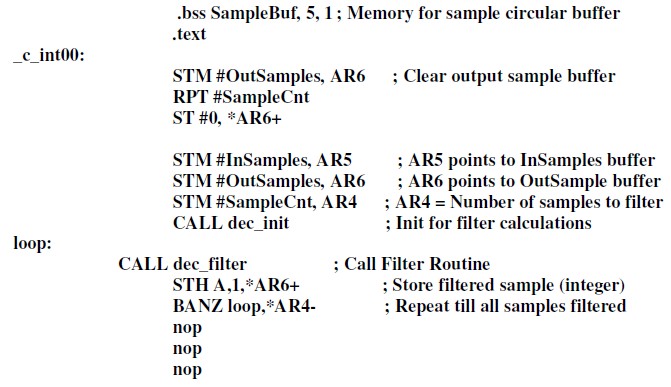
It implements the following equation:

y(m) = h(4)x(3n-4) + h(3)x(3n-3) + h(2)x(3n-2) + h(1)x(3n-1) + h(0)x(3n) followed by the equation

y(m+1) = h(4)x(3n-1) + h(3)x(3n) + h(2)x(3n+1) + h(1)x(3n+2) + h(0)x(3n+3)

and so on for a decimation factor of 3 and a filter length of 5.

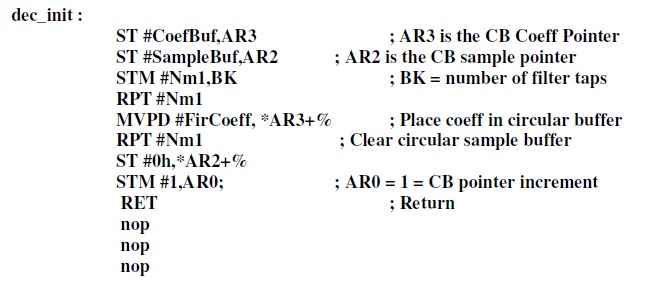




### Decimation Filter Initialization Routine

This routine sets AR2 as the pointer for the sample circular buffer, and AR3 as the pointer for coefficient circular buffer.

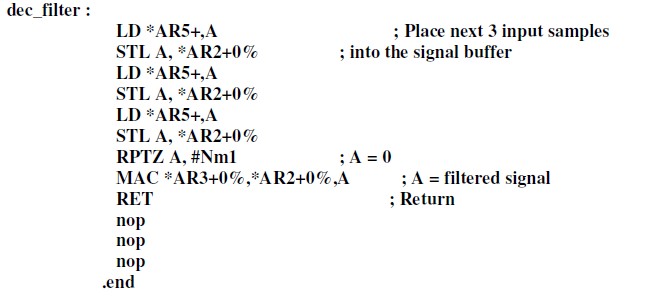
BK = Number of filter taps. ; AR0 = 1 = circular buffer pointer increment.



### FIR Filter Routine

Enter with A = x(n), AR2 pointing to the circular sample buffer, and AR3 to the circular coeff buffer. AR0 = 1.

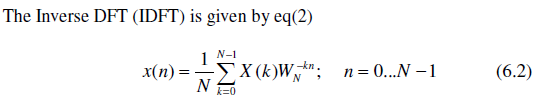
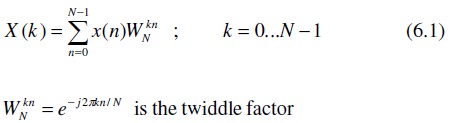
Exit with A = y (n) as q15 number.



## Unit-5

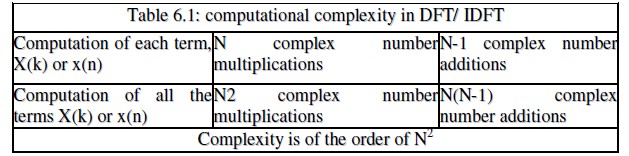
**IMPLEMENTATION OF FFT ALGORITHMS:** Introduction, An FFT Algorithm for DFT

* 1. **Introduction:** The N point Discrete Fourier Transform (DFT) of x(n) is a discrete signal of length N is given by eq(6.1)

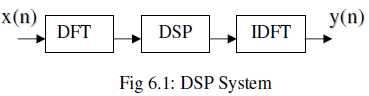


By referring to eq (6.1) and eq (6.2), the difference between DFT & IDFT are seen to be

the sign of the argument for the exponent and multiplication factor, 1/N. The computational complexity in computing DFT / I DFT is thus same (except for the additional multiplication factor in IDFT). The computational complexity in computing each X(k) and all the x(k) is shown in table 6.1.



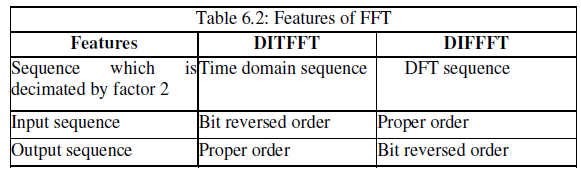
In a typical Signal Processing System, shown in fig 6.1 signal is processed using DSP in the DFT domain. After processing, IDFT is taken to get the signal in its original domain. Though certain amount of time is required for forward and inverse transform, it is because of the advantages of transformed domain manipulation, the signal processing is carried out in DFT domain. The transformed domain manipulations are sometimes simpler. They are also more useful and powerful than time domain manipulation. For example, convolution in time domain requires one of the signals to be folded, shifted and multiplied by another signal, cumulatively. Instead, when the signals to be convolved are transformed to DFT domain, the two DFT are multiplied and inverse transform is taken. Thus, it simplifies the process of convolution.



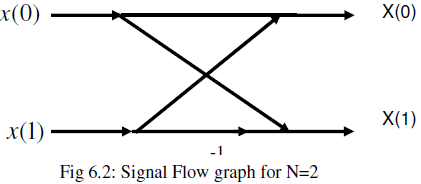
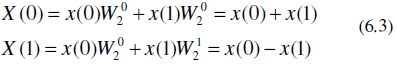
* 1. **An FFT Algorithm for DFT Computation:** As DFT / IDFT are part of signal processing system, there is a need for fast computation of DFT / IDFT. There are algorithms available for fast computation of DFT/ IDFT. There are referred to as Fast Fourier Transform (FFT) algorithms. There are two FFT algorithms: Decimation-In-Time

FFT (DITFFT) and Decimation-In-Frequency FFT (DIFFFT). The computational complexity of both the algorithms are of the order of log2(N). From the hardware / software implementation viewpoint the algorithms have similar structure throughout the

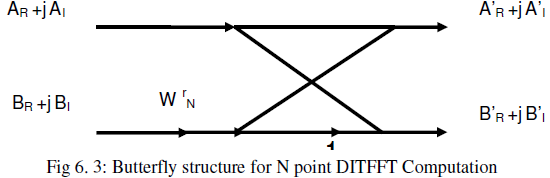
computation. In-place computation is possible reducing the requirement of large memory locations. The features of FFT are tabulated in the table 6.2.

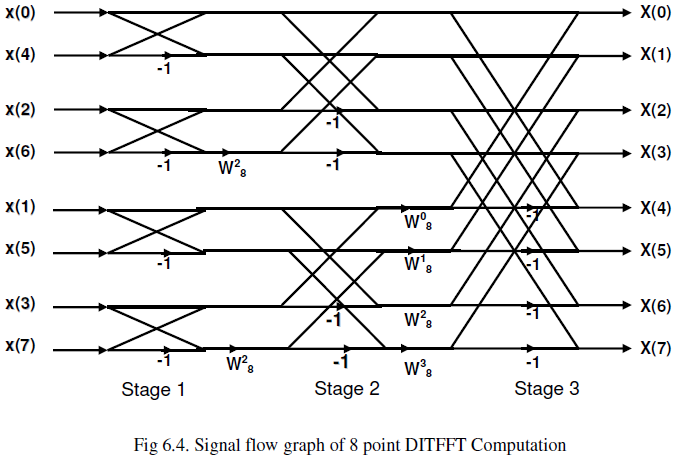


Consider an example of computation of 2 point DFT. The signal flow graph of 2 point DITFFT Computation is shown in fig 6.2. The input / output relations is as in eq (6.3) which are arrived at from eq(6.1).



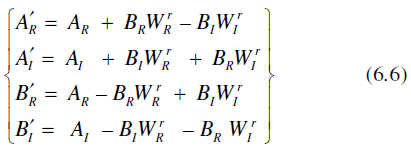
Similarly, the Butterfly structure in general for DITFFT algorithm is shown in fig. 6.3. The signal flow graph for N=8 point DITFFT is shown in fig. 4. The relation between input and output of any Butterfly structure is shown in eq (6.4) and eq(6.5).







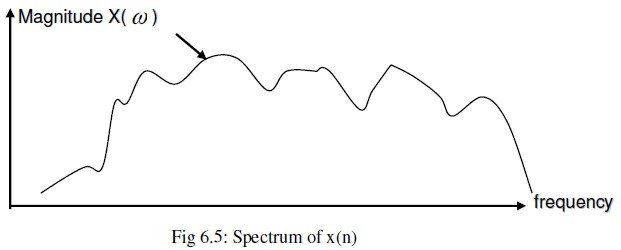
Separating the real and imaginary parts, the four equations to be realized in implementation of DITFFT Butterfly structure are as in eq(6.6).



Observe that with N=2^M, the number of stages in signal flow graph=M, number of multiplications = (N/2)log2(N) and number of additions = (N/2)log2(N). Number of Butterfly Structures per stage = N/2. They are identical and hence in-place computation is possible. Also reusability of hardware designed for implementing Butterfly structure is

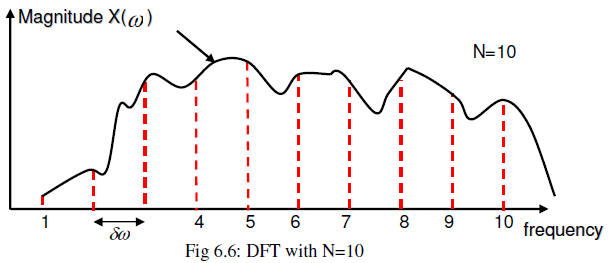
possible. However in case FFT is to be computed for a input sequence of length other than 2^M the sequence is extended to N=2^M by appending additional zeros. The process will not alter the

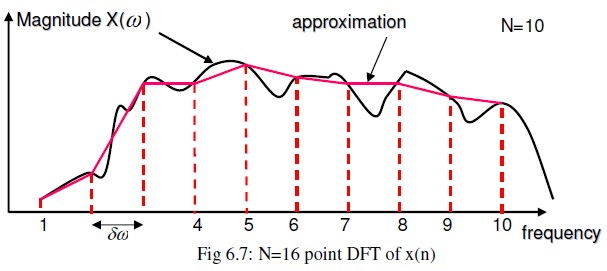
information content of the signal. It improves frequency resolution. To make the point clear, consider a sequence whose spectrum is shown in fig. 6.5.



The spectrum is sampled to get DFT with only N=10. The same is shown in fig 6.

The variations in the spectrum are not traced or caught by the DFT with N=10. For example, dip in the spectrum near sample no. 2, between sample no.7 & 8 are not represented in DFT. By increasing N=16, the DFT plot is shown in fig. 6.7. As depicted in fig 6.7, the approximation to the spectrum with N=16 is better than with N=10. Thus, increasing N to a suitable value as required by an algorithm improves frequency resolution.

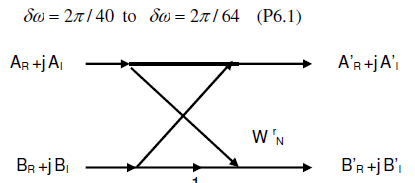




**Problem P6.1:** What minimum size FFT must be used to compute a DFT of 40 points? What must be done to samples before the chosen FFT is applied? What is the frequency resolution achieved?

Solution:

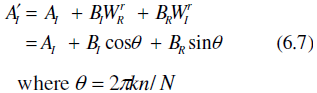
Minimum size FFT for a 40 point sequence is 64 point FFT. Sequence is extended to 64 by appending additional 24 zeros. The process improves frequency resolution from



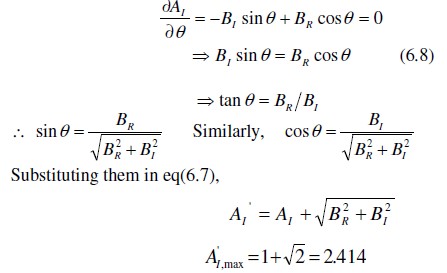
* 1. **Overflow and Scaling:** In any processing system, number of bits per data in signal

processing is fixed and it is limited by the DSP processor used. Limited number of bits leads to overflow and it results in erroneous answer. InQ15 notation, the range of numbers that can be represented is -1 to 1. If the value of a number exceeds these limits, there will be underflow / overflow. Data is scaled down to avoid overflow.

However, it is an additional multiplication operation. Scaling operation is simplified by selecting scaling factor of 2^-n. And scaling can be achieved by right shifting data by n bits. Scaling factor is defined as the reciprocal of maximum possible number in the operation. Multiply all the numbers at the beginning of the operation by scaling factor so that the maximum number to be processed is not more than 1. In the case of DITFFT computation, consider for example,

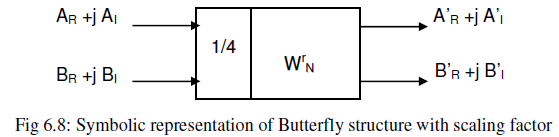


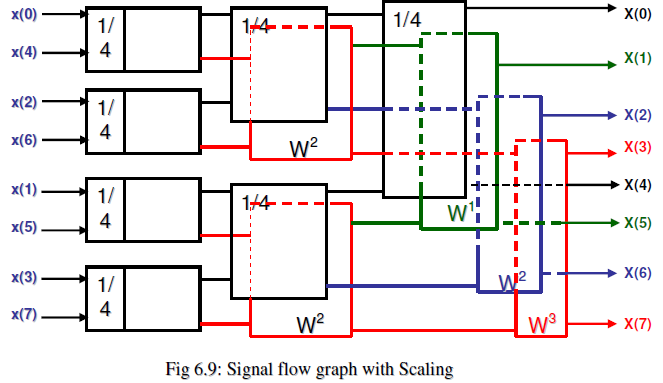
To find the maximum possible value for LHS term, Differentiate and equate to zero



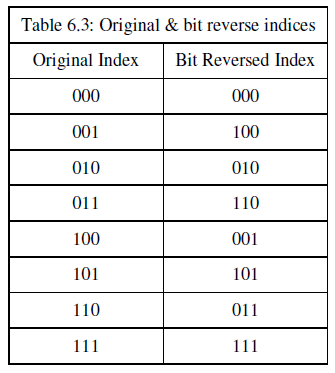
Thus scaling factor is 1/2.414=0.414. A scaling factor of 0.4 is taken so that it can be implemented by shifting the data by 2 positions to the right. The symbolic representation

of Butterfly Structure is shown in fig. 6.8. The complete signal flow graph with scaling factor is shown in fig. 6.9.





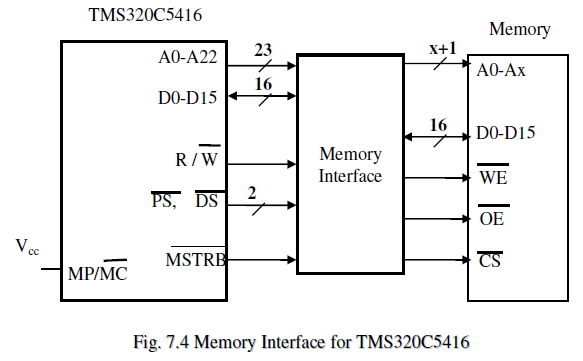
* 1. **Bit-Reversed Index Generation:** As noted in table 6.2, DITFFT algorithm requires input in bit reversed order. The input sequence can be arranged in bit reverse order by reverse carry add operation. Add half of DFT size (=N/2) to the present bit reversed ndex to get next bit reverse index. And employ reverse carry propagation while adding bits from left to right. The original index and bit reverse index for N=8 is listed in table 6.3



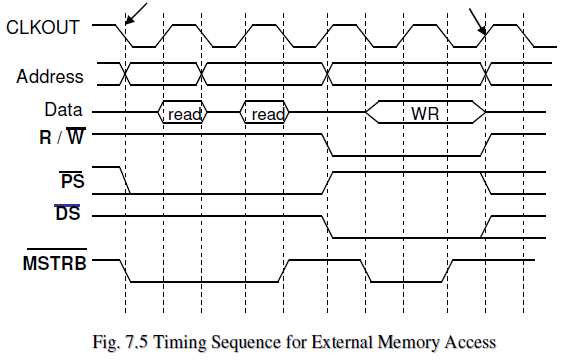
Consider an example of computing bit reverse index. The present bit reversed index be

* 1. an application may be such that memory capacity required is more than that available in a memory IC. That means there are insufficient words in memory IC. Or the word length required may be more than that is available in a memory IC. Thus, there may be insufficient word length. In both the cases, more number of memory ICs are required.

Typical signals in a memory device are address bus to carry address of referred memory location. Data bus carries data to or from referred memory location. Chip Select Signal selects one or more memory ICs among many memory ICs in the system. Write Enable enables writing of data available on data bus to a memory location. Output Enable signal enables the availability of data from a memory location onto the data bus. The address bus is unidirectional, carries address into the memory IC. Data bus is bidirectional. Chip Select, Write Enable and Output Enable control signals are active high or low and they carry signals into the memory ICs. The task of the memory interface is to use DSP signals and generate the appropriate signals for setting up communication with the memory. The logical spacing of interface is shown in fig. 7.4.

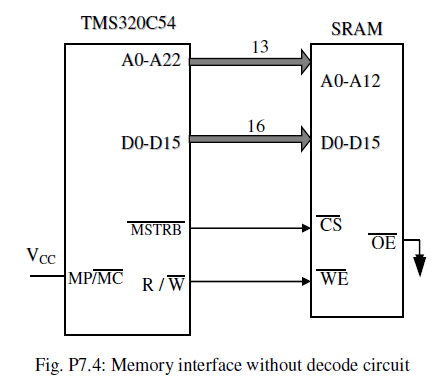


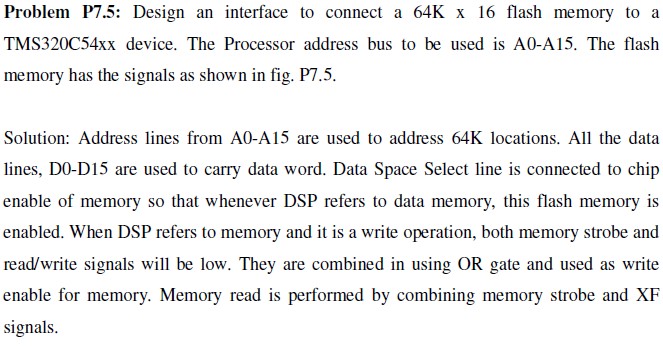
The timing sequence of memory access is shown in fig. 7.5. There are two read operations, both referring to program memory. Read Signal is high and Program Memory Select is low. There is one Write operation referring to external data memory. Data Memory Select is low and Write Signal low. Read and write are to memory device and hence memory strobe is low. Internal program memory reads take one clock cycle and External data memory access require two clock cycles.

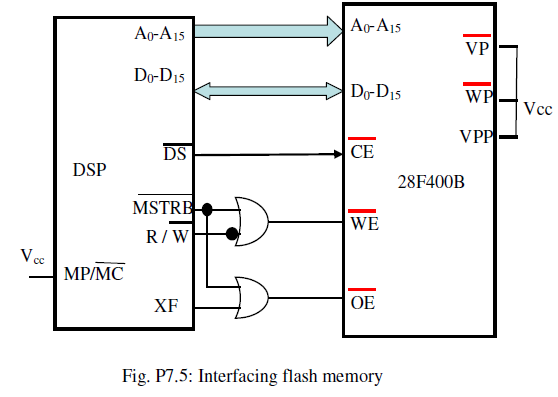


Effects of ‘No decode’ interface are

* Fast memory Access
* ENTIRE Address space is used by the Device that is connected
* Memory responds to 0000-1FFFh and also to all combinations of address bits A13- A19 (In the example quoted)
* Program space select & data space select lines are not used
* SRAM is thus indistinguishable as a program or data Memory







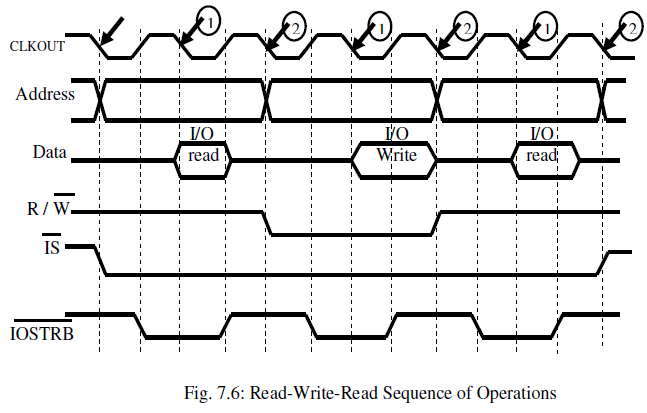
* 1. **Parallel I/O Interface:** I/O devices are interfaced to DSP using unconditional I/O mode, programmed I/O mode or interrupt I/O mode. Unconditional I/O does not require any handshaking signals. DSP assumes the readiness of the I/O and transfers the data with its own speed. Programmed I/O requires handshaking signals. DSP waits for the readiness of the I/O readiness signal which is one of the handshaking signals. After the

completion of transaction DSP conveys the same to the I/O through another handshaking signal. Interrupt I/O also requires handshaking signals. DSP is interrupted by the I/O indicating the readiness

of the I/O. DSP acknowledges the interrupt, attends to the interrupt. Thus, DSP need not wait for the I/O to respond. It can engage itself in execution as long as there is no interrupt.

* 1. **: Programmed I /O interface:** The timing diagram in the case of programmed I/O is shown in fig.

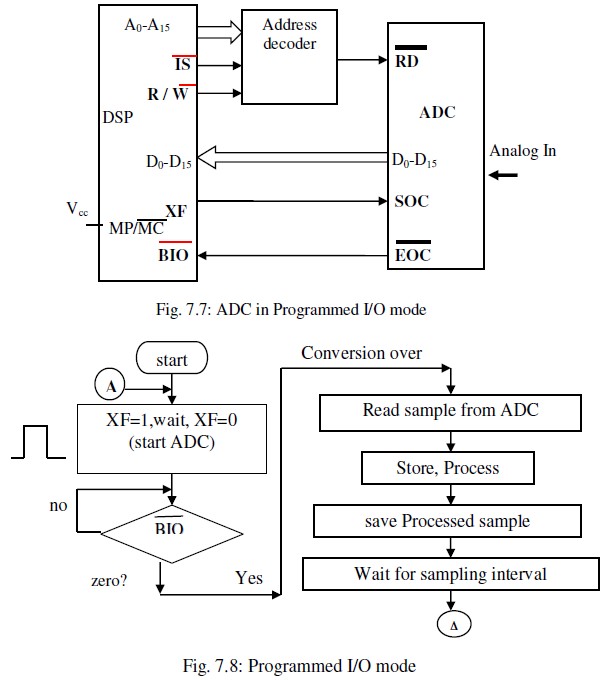
7.6. I/O strobe and I/O space select are issued by the DSP. Two clock cycles each are required for I/O read and I/O write operations.



An example of interfacing ADC to DSP in programmed I/O mode is shown in fig. 7.7. ADC has a start of conversion (SOC) signal which initiates the conversion. In programmed I/O mode, external flag signal is issued by DSP to start the conversion. ADC issues end of conversion (EOC) after completion of conversion. DSP receives Branch input control by ADC when ADC completes the conversion. The DSP issues address of the ADC, I/O strobe and read / write signal as high to read the data. An address decoder does the translation of this information into active low read signal to ADC. The data is supplied on data bus by ADC and DSP reads the same. After reading,

DSP issues start of conversion once again after the elapse of sample interval. Note that

there are no address lines for ADC. The decoded address selects the ADC. During conversion, DSP waits checking branch input control signal status for zero. The flow chart of the activities in programmed I/O is shown in fig. 7.8.



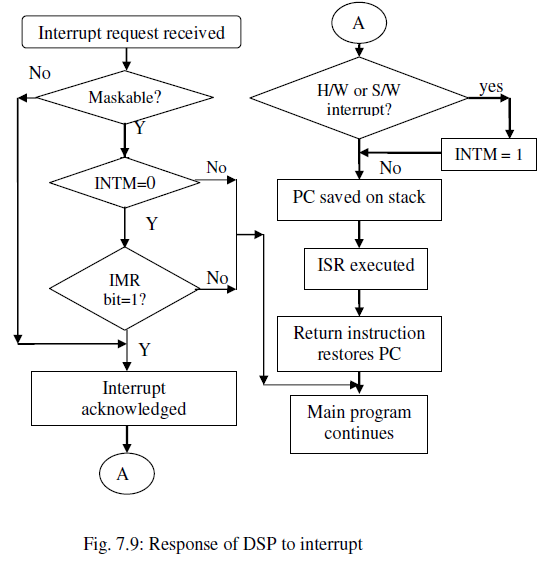
* 1. **Interrupt I/O:** This mode of interfacing I/O devices also requires handshaking signals. DSP is interrupted by the I/O whenever it is ready. DSP Acknowledges the interrupt, after testing certain conditions, attends to the interrupt. DSP need not wait for the I/O to respond. It can engage itself in execution. There are a variety of interrupts. One of the classifications is maskable and nonmaskable. If maskable, DSP can ignore when that interrupt is masked. Another classification is vectored and non- vectored. If vectored, Interrupt Service subroutine (ISR) is in specific location. In Software Interrupt, instruction is written in the program.

In Hardware interrupt, a hardware pin, on the DSP IC will receive an interrupt by the external device. Hardware interrupt is also referred to as external interrupt and software interrupt is referred to as internal interrupt. Internal interrupt may also be due to execution of certain instruction can causing interrupt. In TMS320C54xx there are total of 30 interrupts. Reset, Non-maskable, Timer Interrupt, HPI, one each, 14 Software Interrupts, 4 External user Interrupts, 6 Mc-BSP related Interrupts and 2 DMA related Interrupts. Host Port Interface (HPI) is a 8 bit parallel port. It is possible to interface to a Host Processor using HPI. Information exchange is through on-chip memory of DSP

which is also accessible Host processor.

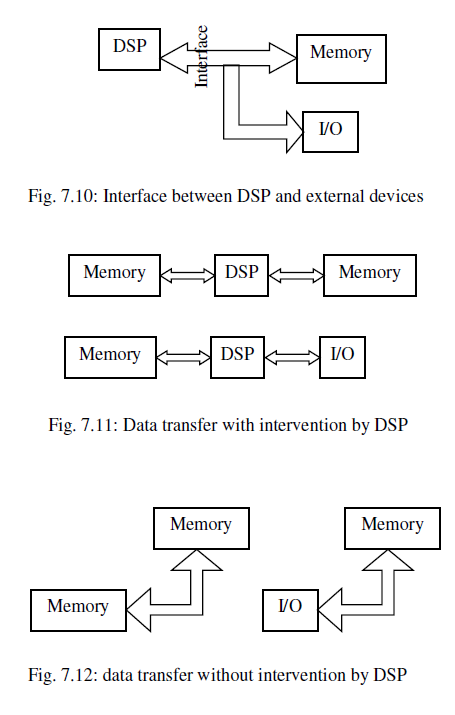
Registers used in managing interrupts are Interrupt flag Register (IFR) and Interrupt Mask Register (IMR). IFR maintains pending external & internal interrupts. One in any bit position implies pending interrupt. Once an interrupt is received, the orresponding bit is set. IMR is used to mask or unmask an interrupt. One implies that the corresponding interrupt is unmasked. Both these registers are Memory Mapped Registers. One flag, Global enable bit (INTM), in ST1 register is used to enable or disable all interrupts globally. If INTM is zero, all unmasked interrupts are enabled. If it is one, all maskable interrupts are disabled.

When an interrupt is received by the DSP, it checks if the interrupt is maskable. If the interrupt is non-maskable, DSP issues the interrupt acknowledgement and thus serves the interrupt. If the interrupt is hardware interrupt, global enable bit is set so that no other interrupts are entertained by the DSP. If the interrupt is maskable, status of the INTM is checked. If INTM is 1, DSP does not respond to the interrupt and it continues with program execution. If the INTM is 0, bit in IMR register corresponding to the interrupt is checked. If that bit is 0, implying that the interrupt is masked, DSP does not respond to the interrupt and continues with its program execution. If the interrupt is unmasked, then DSP issues interrupt acknowledgement. Before branching to the interrupt service routine, DSP saves the PC onto the stack. The same will be reloaded after attending the interrupt so as to return to the program that has been interrupted. The response of DSP to an Interrupt is shown in flow chart in fig. 7.9.



* 1. **: Direct Memory Access (DMA) operation:** In any application, there is data transfer

between DSP and memory and also DSP and I/O device, as shown in fig. 7.10. However, there may be need for transfer of large amount of data between two memory regions or between memory and I/O. DSP can be involved in such transfer, as shown in fig. 7.11. Since amount of data is large, it will engage DSP in data transfer task for a long time. DSP thus will not get utilized for the purpose it is meant for, i.e., data manipulation. The intervention of DSP has to be avoided for two reasons: to utilize DSP for useful signal processing task and to increase the speed of transfer by direct data transfer between memory or memory and I/O. The direct data transfer is referred to as direct memory access (DMA). The arrangement expected is shown in fig. 7.12. DMA controller helps in data transfer instead of DSP.



In DMA, data transfer can be between memory and peripherals which are either internal

or external devices. DMA controller manages DMA operation. Thus DSP is relieved of the task of data transfer. Because of direct transfer, speed of transfer is high. In TMS320C54xx, there are up to 6 independent programmable DMA channels. Each channel is between certain source & destination. One channel at a time can be used for

data transfer and not all six simultaneously. These channels can be prioritized. The speed of transfer measured in terms of number of clock cycles for one DMA transfer depends on several factors such as source and destination location, external interface conditions, number of active DMA channels, wait states and bank switching time. The time for data transfer between two internal memory is 4 cycles for each word.

Requirements of maintaining a channel are source & Destination address for a channel, separately for each channel. Data transfer is in the form of block, with each block having frames of 16

/ 32 bits. Block size, frame size, data are programmable. Along with these, mode of transfer and assignment of priorities to different channels are also to be maintained for the purpose of data transfer.

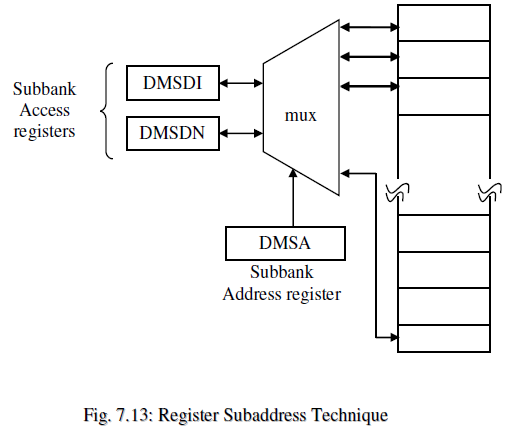
There are five, channel context registers for each DMA channel. They are Source

Address Register (DMSRC), Destination Address Register (DMDST), Element Count Register (DMCTR), Sync select & Frame Count register (DMSFC), Transfer Mode Control Register (DMMCR). There are four reload registers. The context register DMSRC & DMDST are source & destination address holders. DMCTR is for holding number of data elements in a frame. DMSFC is to convey sync event to use to trigger DMA transfer, word size for transfer and for holding frame count. DMMCR Controls transfer mode by specifying source and destination spaces as program memory, data memory or I/O space. Source address reload & Destination address reload are useful in

reloading source address and destination address. Similarly, count reload and frame count reload are used in reloading count and frame count. Additional registers for DMA that are common to all channels are Source Program page address, DMSRCP, Destination Program page address, DMDSTP, Element index address register, Frame index address register.

Number of memory mapped registers for DMA are 6x(5+4) and some common registers

for all channels, amounting to total of 62 registers required. However, only 3 (+1 for priority related) are available. They are DMA Priority & Enable Control Register (DMPREC), DMA sub bank Address Register (DMSA), DMA sub bank Data Register with auto increment (DMSDI) and DMA sub bank Data Register (DMSDN). To access each of the DMA Registers Register sub addressing Technique is employed. The schematic of the arrangement is shown in fig. 7.13. A set of DMA registers of all channels (62) are made available in set of memory locations called sub bank. This voids the need for 62 memory mapped registers. Contents of either DMSDI or DMSDN indicate the code (1’s & 0’s) to be written for a DMA register and contents of DMSA refers to the unique sub address of DMA register to be accessed. Mux routes either DMSDI or DMSDN to the sub bank. The memory location to be written



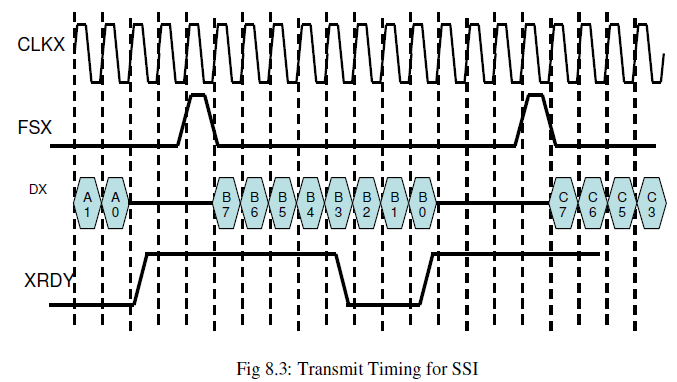
DMSDI is used when an automatic increment of the sub address is required after each access. Thus it can be used to configure the entire set of registers. DMSDN is used when single DMA register access is required. The following examples bring out clearly the method of accessing the DMA registers and transfer of data in DMA mode.

## Recommended Questions:

* + 1. Explain an interface between an A/D converter and the TMS320C54XX processor in the programmed I/O mode.
    2. Describe DMA with respect to TMS320C54XX processors.
    3. Drew the timing diagram for memory interface for read-read-write sequence of operation. Explain the purpose of each signal involved.
    4. Explain the memory interface block diagram for the TMS 320 C54xx processor**.**
    5. Draw the I/O interface timing diagram for read – write read sequence of operation.
    6. What are interrupts? How interrupts are handled by C54xx DSP Processors.
    7. Explain the memory interface block diagram for the TMS 320 C54xx processor.
    8. Draw the I/O interface timing diagram for read – write read sequence of operation.
    9. What are interrupts? How interrupts are handled by C54xx DSP Processors.
    10. Design a data memory system with address range 000800h – 000fffh for a c5416 processor using 2kx8 SRAM memory chips.
    11. Design a data memory system with address range 000800h – 000fffh for a c5416 processor using 2kx8 SRAM memory chips. (**MAY-JUNE 10, 6m)**
    12. Explain an interface between an A/D converter and the TMS320C54XX processor in the programmed I/O mode. . (**JUNE 12, 10m)**
    13. Describe DMA with respect to TMS320C54XX processors. **(June/July 11, 10m)**
    14. Drew the timing diagram for memory interface for read-read-write sequence of operation. Explain the purpose of each signal involved.**(June/July 11, 10m)**
    15. Explain the memory interface block diagram for the TMS 320 C54xx processor**.(Dec 2010)**
    16. Draw the I/O interface timing diagram for read – write read sequence of operation **(Dec 2010)**
    17. What are interrupts? How interrupts are handled by C54xx DSP Processors. **(Dec 2010,12)**
    18. What are interrupts? What are the classes of interrupts available in the TMS320C54xx processor. **(JUNE/July 11, 8m)**

As shown, the receiving or transmit activity is initiated at the rising edge of clock, CLKR

/ CLKX. Reception / Transfer starts after FSR / FSX remains high for one clock cycle. RRDY / XRDY is initially high, goes LOW to HIGH after the completion of data transfer. Each transfer of bit requires one clock cycle. Thus, time required to transfer / receive data word depends on the number of bits in the data word. An example of data word of 8 bits is shown in the fig. 8.2 and fig. 8.3.



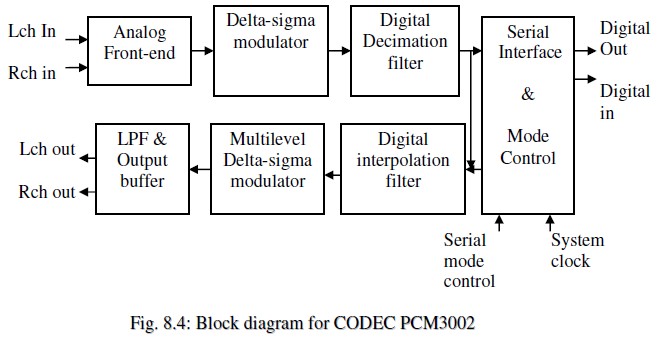
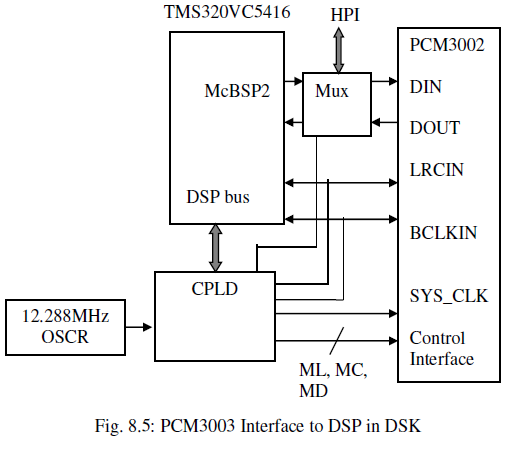


Fig. 8.4 shows the block diagram of PCM3002 CODEC. Analog front end samples signal at 64X over sampling rate. It eliminates need for sample-and-hold circuit and simplifies need for anti aliasing filter. ADC is based on Delta-sigma modulator to convert analog signal to digital form. Decimation filter reduces the sampling rate and thus processing does not need high speed devices. DAC is Delta-sigma modulator, converts digital signal to analog signal. Interpolation increases the sampling rate back to original value. LPF smoothens the analog reconstructed signal by removing high frequency components. The Serial Interface monitors serial data transfer. It accepts built-in ADC output and converts to serial data and transmits the same on DOUT. It also accepts serial data on DIN & gives the

same to DAC. The serial interface works in synchronization with BCLKIN & LRCIN. The Mode Control initializes the serial data transfer. It sets all the desired modes, the number of bits and the mode Control Signals, MD, MC and ML. MD carries Mode Word. MC is the mode Clock Signal. MD to be loaded is sent with reference to this clock. ML is the mode Load Signal. It defines start and end of latching bits into CODEC device.

Figure 8.5 shows interfacing of PCM3002 to DSP in DSK. DSP is connected to PCM3002 through McBSP2. The same port can be connected to HPI. Mux selects one among these two based on CPLD signal. CPLD in Interface also provides system clock for DSP and for CODEC, Mode control signals for CODEC. CPLD generates BCLKIN and LRCIN signals required for serial interface.



PCM3002 CODEC handles data size of 16 / 20 bits. It has 64x over-sampling, delta-sigma ADC & DAC. It has two channels, called left and right. The CODEC is programmable for digital de-emphasis, digital attenuation, soft mute, digital loop back, power-down mode. System clock, SYSCLK of CODEC can be 256fs, 384fs or 512fs. Internal clock is always 256fs for converters, digital filters. DIN, DOUT are the single line data lines to carry the data into the CODEC and from CODEC. Another signal BCLKIN is data bit clock, the default value of which is CODEC SYSCLK / 4. LRCIN is frame sync signal for Left and Right Channels. The frequency of this signal is same as the sampling

frequency. The default divide factor can be 2, 4, 6 and 8. Thus, sampling rate is minimum of 6 KHz and maximum of 48 KHz.

**Problem P8.1:** A PCM3002 is programmed for the 12 KHz sampling rate. Determine the divisor N that should be written to the CPLD of the DSK and the various clock frequencies for the set up.

**Solution:** CPLD input Clock=12.288MHz (known) Sampling rate fs=CODEC\_SYSCLK / 256 =12KHz (given) CPLD output clock, CODEC\_SYSCLK =12.288 x 106 / N Thus, CODEC\_SYSCLK =256 x 12 KHz

& N=12.288 x 106/(256 x 12 x 103)

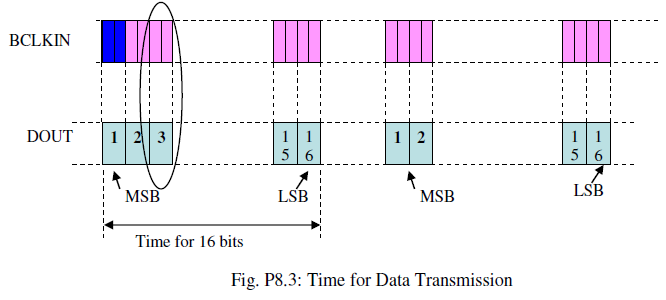
= 4

**Problem P8.3:** Frame Sync is generated by dividing the 8.192MHz clock by 256 for the

serial communication. Determine the sampling rate and the time a 16 bit sample takes when transmitted on the data line.

**Solution:** LRCIN, Frame Sync = 8.192x106/256 =32 KHz Sampling rate fs= frequency of LRCIN=32 KHz

BCLKIN, Bit clock rate=CODEC\_SYSCLK / 4=8.192x106/4=2.048MHz



LRCIN, Frame Sync = 8.192x10^6/256 =32 KHz Sampling rate fs= frequency of LRCIN=32 KHz

BCLKIN, Bit clock rate=CODEC\_SYSCLK / 4=8.192x10^6/4=2.048MHz Bit clock period= 1/2.048x10^6 =0.488x10^-6s

Time for transmitting 16 bits =0.488x10^-6x16 =7.8125x10^-6s (refer fig. P8.3)

The CODEC PCM3002 supports four data formats as listed in table 8.1. The four data formats depend on the number of bits in the data word, if the data is right justified or left justified with respect to LRCIN and if it is I2S (**I**ntegrated **I**nter-chip **S**ound) format.

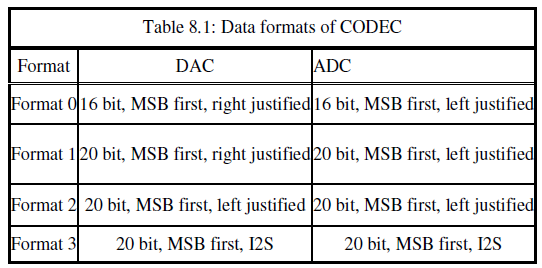
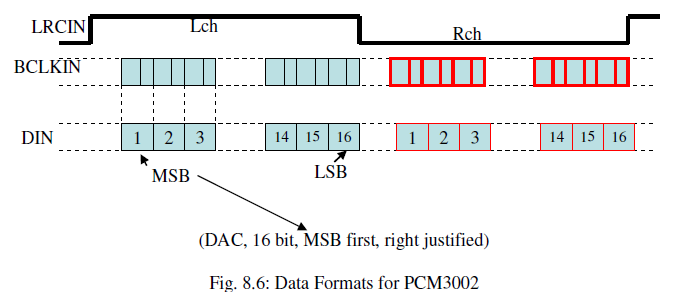
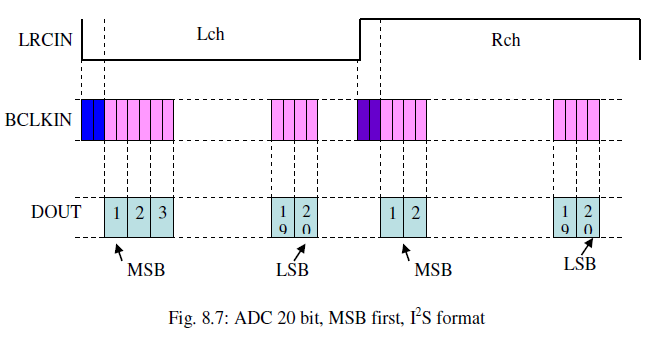


Figure 8.6 and fig. 8.7 depicts the data transaction for CODEC PCM3002. As shown in fig. 8.6, DIN (/ DOUT) carries the data. BCLKIN is the reference for transfer. When LRCIN is high, left channel inputs (/ outputs) the data and when LRCIN is low, right channel inputs (/ outputs) the data. The data bits at the end (/ beginning) of the LRCIN thus Right (/ left) justified.



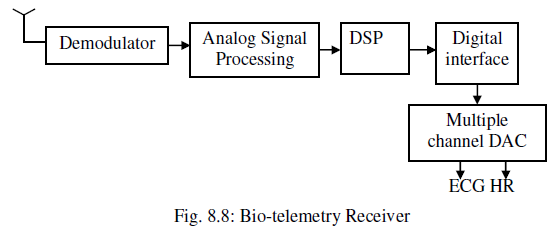
Another data format handled by PCM3002 is I2S (Integrated Inter-chip Sound). It is used for transferring PCM between CD transport & DAC in CD player. LRCIN is low for left channel and high for right channel in this mode of transfer. During the first BCKIN, there is no transmission by ADC. During 2nd BCKIN onwards, there is transmission with MSB first and LSB last. Left channel data is handled first followed by right channel data.



* 1. **DSP Based Bio-telemetry Receiver:** Biotelemetry involves transfer of physiological

information from one remote place to another for the purpose of obtaining experts opinion. The receiver uses radio Frequency links. The schematic diagram of biotelemetry receiver is shown in fig.

8.8. The biological signals may be single dimensional signals such as ECG and EEG or two dimensional signals such as an image, i.e., X-ray. Signal can even be multi dimensional signal i.e., 3D picture. The signals at source are encoded, modulated and transmitted. The signals at destination are decoded, demodulated and analyzed.

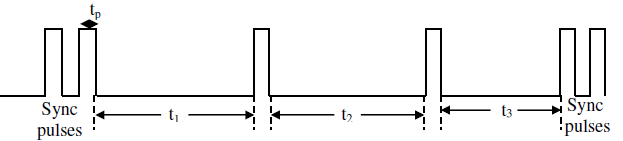


An example of processing ECG signal is considered. The scheme involves modulation of ECG signal by employing Pulse Position Modulation (PPM). At the receiving end, it is

demodulated. This is followed by determination of Heart beat Rate (HR). PPM Signal either encodes single or multiple signals. The principle of modulation being that the position of pulse decides the sample value.

The PPM signal with two ECG signals encoded is shown in fig. 8.9. The transmission requires a sync signal which has 2 pulses of equal interval to mark beginning of a cycle.

The sync pulses are followed by certain time gap based on the amplitude of the sample of 1st signal to be transmitted. At the end of this time interval there is another pulse. This is again followed by time gap based on the amplitude of the sample of the 2nd signal to be transmitted. After encoding all the samples, there is a compensation time gap followed by sync pulses to mark the beginning of next set of samples. Third signal may be encoded in either of the intervals of 1st or 2nd signal. With two signals encoded and the pulse width as tp, the total time duration is 5tp.



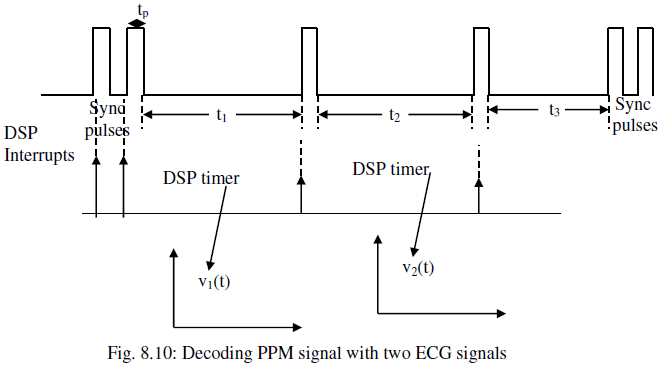
each pullse iinttervall

tt1:: pullse iinttervall correspondiing tto samplle vallue of 1stt siignall tt2:: pullse iinttervall correspondiing tto samplle vallue of 2nd siignall tt3:: compensattiion ttiime iinttervall

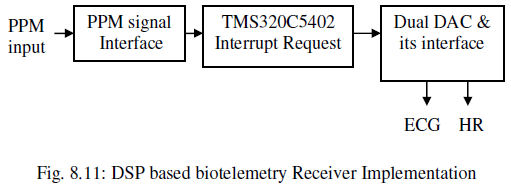
Fig. 8.9: A PPM signal with two ECG signals

Since the time gap between the pulses represent the sample value, at the receiving end the time gap has to be measured and the value so obtained has to be translated to sample value. The scheme for decoding is shown in fig. 8.10. DSP Internal Timer employed. The pulses in PPM generate interrupt signals for DSP. The interrupt start / terminate the timer.

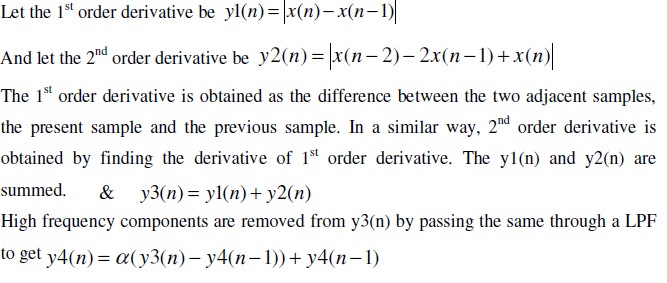
The count in the timer is equivalent to the sample value that has been encoded. Thus, ADC is avoided while decoding the PPM signal.



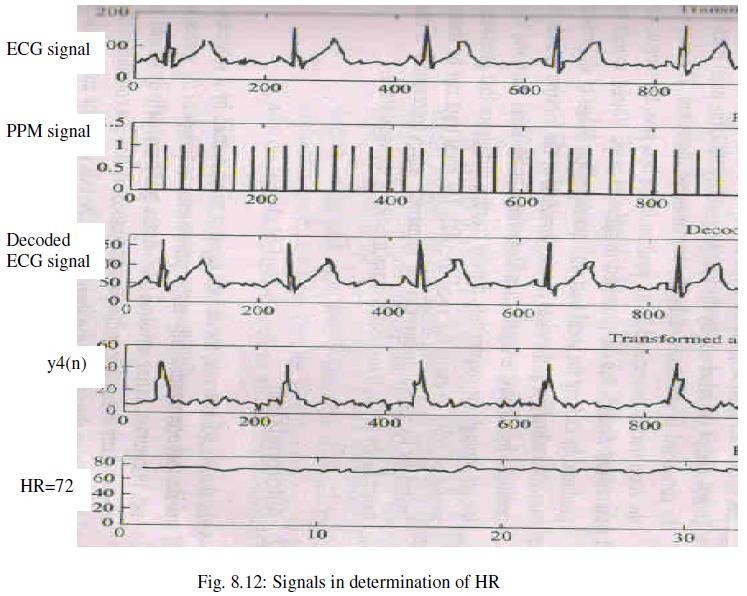
A DSP based PPM signal decoding is shown in fig. 8.11. PPM signal interface generates the interrupt for DSP. DSP entertains the interrupt and starts a timer. When it receives another interrupt, it stops the timer and the count is treated as the digital equivalent of the sample value. The process repeats. Dual DAC converts two signals encoded into analog signals. And heart rate is determined referring to the ECG obtained by decoding



Heart Rate (HR) is a measure of time interval between QRS complexes in ECG signal. QRS complex in ECG is an important segment representing the heart beat. There is periodicity in its appearance indicating the heart rate. The algorithm is based on 1st and 2nd order absolute derivatives of the ECG signal. Since absolute value of derivative is taken, the filter will be a nonlinear filtering.



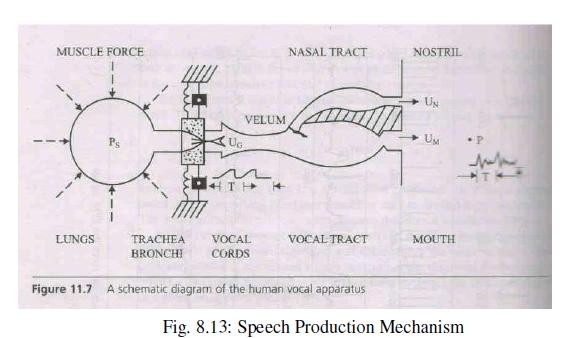
Mean of half of peak amplitudes is determined, which is threshold for detection of QRS complex. QRS interval is then the time interval between two such peaks. Time Interval between two peaks is determined using internal timer of DSP. Heart Rate, heart beat perminute is computed using the relation HR=Sampling rate x 60 / QRS interval. The signals at various stages are shown in fig. 8.12.



* 1. **A Speech Processing System:** The purpose of speech processing is for analysis, transmission or reception as in the case of radio / TV / phone, denoising, compression and so on. There are various applications of speech processing which include identification and verification of speaker, speech synthesis, voice to text conversion and

vice versa and so on. A speech processing system has a vocoder, a voice coding / decoding circuit. Schematic of speech production is shown in fig. 8.13. The vocal tract has vocal cord at one end and mouth at the other end. The shape of the vocal tract depends on position of lips, jaws, tongue and the velum. It decides the sound that is produced. There is another tract, nasal tract. Movement of velum connects or disconnects nasal tract. The overall voice that sounds depends on both, the vocal tract and nasal tract.

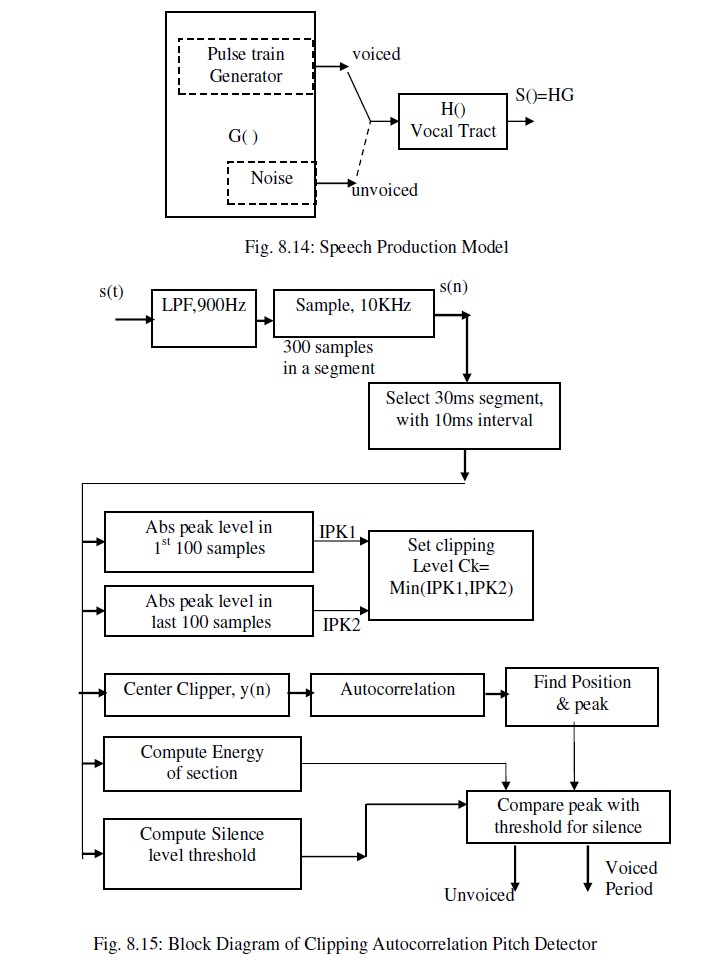
Two types of speech are voiced sound and unvoiced sound. Vocal tract is excited with quasi periodic pulses of air pressure caused by vibration of vocal cords resulting in voiced sound. Unvoiced sound is produced by forcing air through the constriction, formed somewhere in the vocal tract and creating turbulence that produces source of noise to excite the vocal tract.



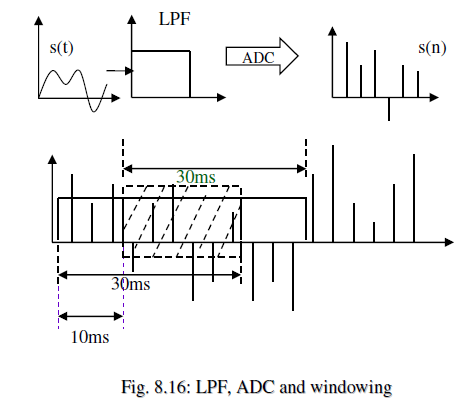
By the understanding of speech production mechanism, a speech production model representing the same is shown in fig. 8.14. Pulse train generator generates periodic pulse train. Thus it represents the voiced speech signal. Noise generator represents unvoiced speech. Vocal tract system is supplied either with periodic pulse train or noise. The final output is the synthesized speech signal.

Sequence of peaks occurs periodically in voiced speech and it is the fundamental frequency of speech. The fundamental frequency of speech differs from person to person and hence sound of speech differs from person to person. Speech is a non stationary signal. However, it can be considered to be relatively stationary in the intervals of 20ms. Fundamental frequency of speech can be determined by

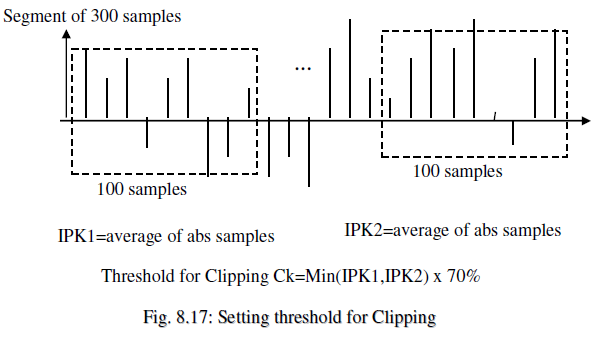
autocorrelation method. In other words, it is a method of determination of pitch period. Periodicity in autocorrelation is because of the fundamental frequency of speech. A three level clipping scheme is discussed here to measure the fundamental frequency of speech. The block diagram for the same is shown in fig. 8.15.



The speech signal s(t) is filtered to retain frequencies up to 900Hz and sampled using ADC to get s(n). The sampled signal is processed by dividing it into set of samples of 30ms duration with 20ms overlap of the windows. The same is shown in fig. 8.16.

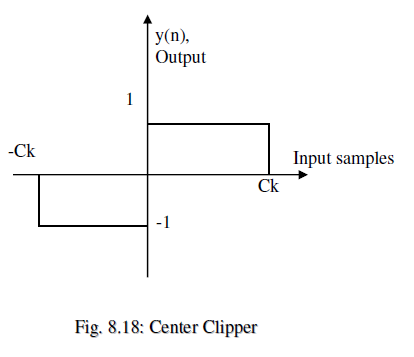


A threshold is set for three level clipping by computing minimum of average of absolute values of 1st 100 samples and last 100 samples. The scheme is shown in fig. 8.17.



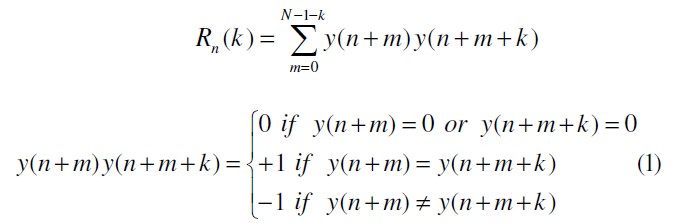
The transfer characteristics of three level clipping circuit is shown in fig. 8.18. If the sample value is greater than +CL, the output y(n) of the clipper is set to 1. If the sample value is more negative than -

CL, the output y(n) of the clipper is set to -1. If the sample value is between –CL and +CL, the output y(n) of the clipper is set to 0.

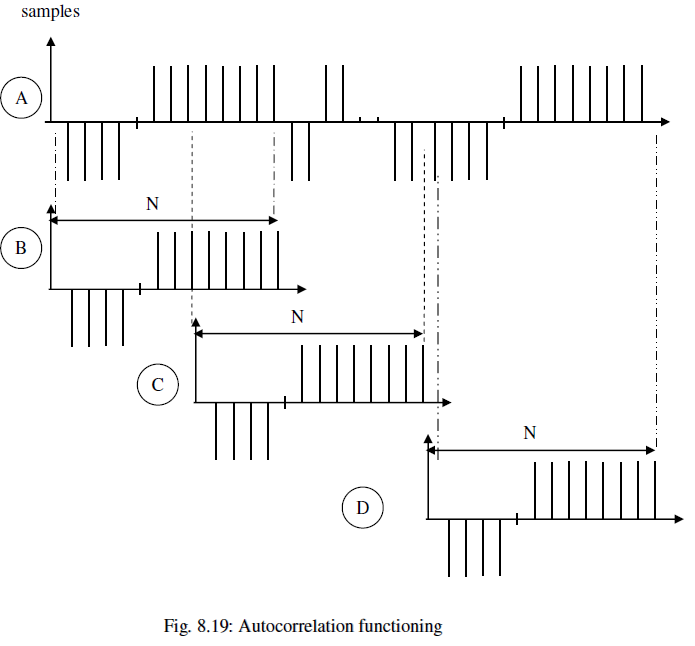


The autocorrelation of y(n) is computed which will be 0,1 or -1 as defined by eq (1). The largest peak in autocorrelation is found and the peak value is compared to a fixed threshold. If the peak value is below threshold, the segment of s(n) is classified as unvoiced segment. If the peak value is above threshold, the segment of s(n) is classified

as voiced segment. The functioning of autocorrelation is shown in fig. 8.19.



As shown in fig. 8.19, A is a sample sequence y(n). B is a window of samples of length N and it is compared with the N samples of y(n). There is maximum match. As the window is moved further, say to a position C the match reduces. When window is moved further say to a position D, again there is maximum match. Thus, sequence y(n) is periodic. The period of repetition can be measured by locating the peaks and finding the time gap between them.



* 1. **An Image Processing System:** In comparison with the ECG or speech signal considered so far, image has entirely different requirements. It is a two dimensional signal. It can be a color or gray image. A color image requires 3 matrices to be maintained for three primary colors-red, green and blue. A gray image requires only one

matrix, maintaining the gray information of each pixel (picture cell). Image is a signal with large amount of data. Of the many processing, enhancement, restoration, etc., image compression is one important processing because of the large amount of data in image.

To reduce the storage requirement and also to reduce the time and band width required to transmit the image, it has to be compressed. Data compression of the order of factor 50 is sometimes preferred. JPEG, a standard for image compression employs lossy compression technique. It is based on discrete cosine transform (DCT). Transform domain compression separates the image signal into low frequency components and high frequency components. Low frequency components are retained

because they represent major variations. High frequency components are ignored because they represent minute variations and our eye is not sensitive to minute variations.

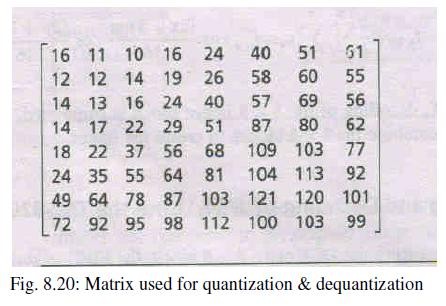
Image is divided into blocks of 8 x 8. DCT is applied to each block. Low frequency coefficients are of higher value and hence they are retained. The amount of high frequency components to be retained is decided by the desirable quality of reconstructed image. Forward DCT is given by eq (2).



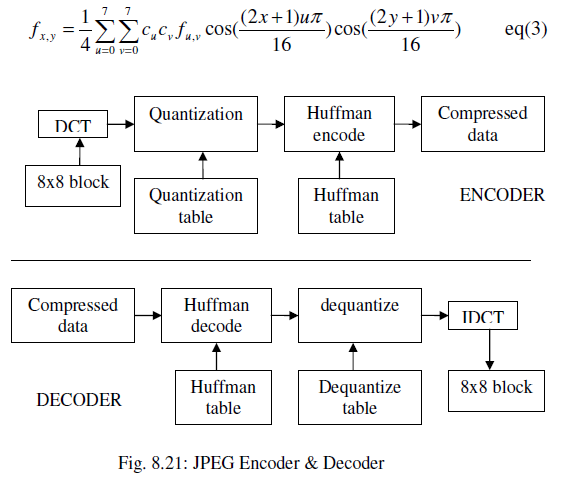
Since the coefficients values may vary with a large range, they are quantized. As already noted low frequency coefficients are significant and high frequency coefficients are insignificant, they are allotted varying number of bits. Significant coefficients are quantized precisely, with more bits and insignificant coefficients are quantized coarsely,

with fewer bits. To achieve this, a quantization table as shown in fig. 8.20 is employed. The contents of Quantization Table indicate the step size for quantization. An entry as smaller value implies smaller step size, leading to more bits for the coefficients and vice

versa.



The quantized coefficients are coded using Huffman coding. It is a variable length coding Huffman Encoding. Shorter codes are allotted for frequently occurring long sequence of 1’s & 0’s. Decoding requires Huffman table and dequantization table. Inverse DCT is taken employing eq(3). The data blocks so obtained are combined to form complete image. The schematic of encoding and decoding is shown in fig. 8.21.



## Recommended Questions:

* + 1. With the help of a block diagram, explain the image compression and reconstruction using JPEG encoder and decoder.
    2. Write a pseudo algorithm heart rate(HR), using the digital signal processor**.**
    3. Explain briefly the building blocks of a PCM3002 CODEC device. What do you understand by a DSP based biotelemetry receiver?
    4. With the help of block diagram explain JPEG algorithm.
    5. Explain with the neat diagram the operation of pitch detector**.**
    6. Explain with a neat diagram, the synchronous serial interface between the C54xx and a CODEC device. Explain the operation of pulse position modulation (PPM) to encode two biomedical signals.
    7. Explain with a neat block diagram the operation, the operation of the pitch detector.
    8. Explain PCM3002 CODEC, with the help of neat block diagram.
    9. Explain DSP based biotelemetry receiver system, with the help of a block schematic diagram.
    10. Explain the memory interface block diagram for the TMS 320 C54xx processor**.(Dec 2010)**
    11. Draw the I/O interface timing diagram for read – write read sequence of operation **(Dec 2010)**
    12. What are interrupts? How interrupts are handled by C54xx DSP Processors. **(Dec 2010,12)**
    13. What are interrupts? What are the classes of interrupts available in the TMS320C54xx processor. **(JUNE/July 11, 8m)**