DIGITAL SIGNAL PROCESSORS & ARCHITECTURES (R18A0427)

Lecture Notes B. TECH
(IV YEAR – II SEM) (2022-2023)

Prepared by:

Dr. N. Subash, Associate Professor
Mr. M. Sreedhar Reddy, Assistant Professor

Department of Electronics and Communication Engineering

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY
(Autonomous Institution - UGC, Govt. of India)
Recognized under 2(f) and 12 (B) of UGC ACT 1956 (Affiliated to JNTUH, Hyderabad,
Approved by AICTE-Accredited by NBA & NAAC–'A'Grade-ISO9001:2015Certified)
Maisammaguda, Dhulapally (PostVia.Kompally), Secunderabad–500100,TelanganaState,India

MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY

IV Year B.Tech. ECE-II Sem **L T/P/D C**
3 -/ - /- 3

PROFESSIONAL ELECTIVE - IV
(R18A0427) DIGITAL SIGNAL PROCESSORS AND ARCHITECTURES

COURSE OBJECTIVES

1. To introduce architectural features of programmable DSP Processors of TI and AnalogDevices.
2. To recall digital transform techniques.
3. To give practical examples of DSP Processor architectures for better understanding.
4. To develop the programming knowledge using Instruction set of DSP Processors.
5. To understand interfacing techniques to memory and I/O devices.

UNIT- I:

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.

Computational Accuracy in DSP Implementations:

Number formats for signals and coefficients in DSP systems, Dynamic Range and Precision, Sources of error in DSP implementations, A/D Conversion errors, DSP Computational errors,D/A Conversion Errors, Compensating filter.

UNIT- II:

Architectures for Programmable DSP Devices:

Basic Architectural features, DSP Computational Building Blocks, Bus Architecture and Memory, Data Addressing Capabilities, Address Generation UNIT, Programmability and Program Execution, Speed Issues, Features for External interfacing.

UNIT- III:

Programmable Digital Signal Processors:

Commercial Digital signal-processing Devices, Data Addressing modes of TMS320C54XX DSPs, Data Addressing modes of TMS320C54XX Processors, Memory space of TMS320C54XX Processors, Program Control, TMS320C54XX instructions and Programming, On-Chip Peripherals, Interrupts of TMS320C54XX processors, Pipeline operation of TMS320C54XX Processors.

UNIT – IV:

Analog Devices Family of DSP Devices:

Analog Devices Family of DSP Devices – ALU and MAC block diagram, Shifter Instruction, Base Architecture of ADSP 2100, ADSP-2181 high performance Processor.

Introduction to Blackfin Processor - The Blackfin Processor, Introduction to Micro Signal Architecture, Overview of Hardware Processing Units and Register files, Address ArithmeticUnit, Control Unit,

Bus Architecture and Memory, Basic Peripherals.

UNIT – V:

Interfacing Memory and I/O Peripherals to Programmable DSP Devices:

Memory space organization, External bus interfacing signals, Memory interface, Parallel I/Ointerface, Programmed I/O, Interrupts and I/O, Direct memory access (DMA).

TEXT BOOKS:

1. Digital Signal Processing – Avtar Singh and S. Srinivasan, Thomson Publications, 2004.
2. A Practical Approach to Digital Signal Processing - K Padmanabhan, R. Vijayarajeswaran,Ananthi. S, New Age International, 2006/2009
3. Embedded Signal Processing with the Micro Signal Architecture Publisher: Woon-Seng Gan,Sen M. Kuo, Wiley-IEEE Press, 2007

REFERENCE BOOKS :

1. Digital Signal Processors, Architecture, Programming and Applications – B. Venkataramani and M. Bhaskar, 2002, TMH.
2. Digital Signal Processing – Jonatham Stein, 2005, John Wiley.
3. DSP Processor Fundamentals, Architectures & Features – Lapsley et al. 2000, S. Chand & Co.
4. Digital Signal Processing Applications Using the ADSP-2100 Family by The Applications Engineering Staff of Analog Devices, DSP Division, Edited by Amy Mar, PHI
5. The Scientist and Engineer's Guide to Digital Signal Processing by Steven W. Smith, Ph.D., California Technical Publishing, ISBN 0-9660176-3-3, 1997 6.Embedded Media Processing by David J. Katz and Rick Gentile of Analog Devices, Newnes, ISBN 0750679123, 2005

COURSE OUTCOMES:

Upon completion of the course the student will be able to:

1. To distinguish between the architectural features of general purpose processors and DSPprocessors
2. Understand the architectures of TMS 320C54XX and ADSP2100 DSP devices
3. Able to write assembly language programs using instruction set of TMS320C54XX
4. Can interface various devices to DSP Processors

UNIT-1

Introduction to Digital Signal Processing

1.1 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.

Fig 1.1: A Typical DSP System

1.2 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.

1.3 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).

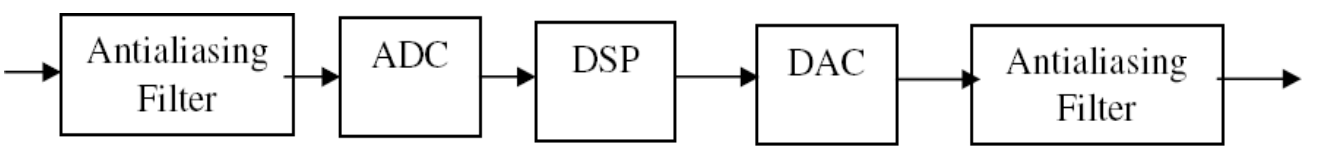


Fig 1.2 The Block Diagram of a DSP System

Signals that occur in a typical DSP are as shown in figure 1.3.

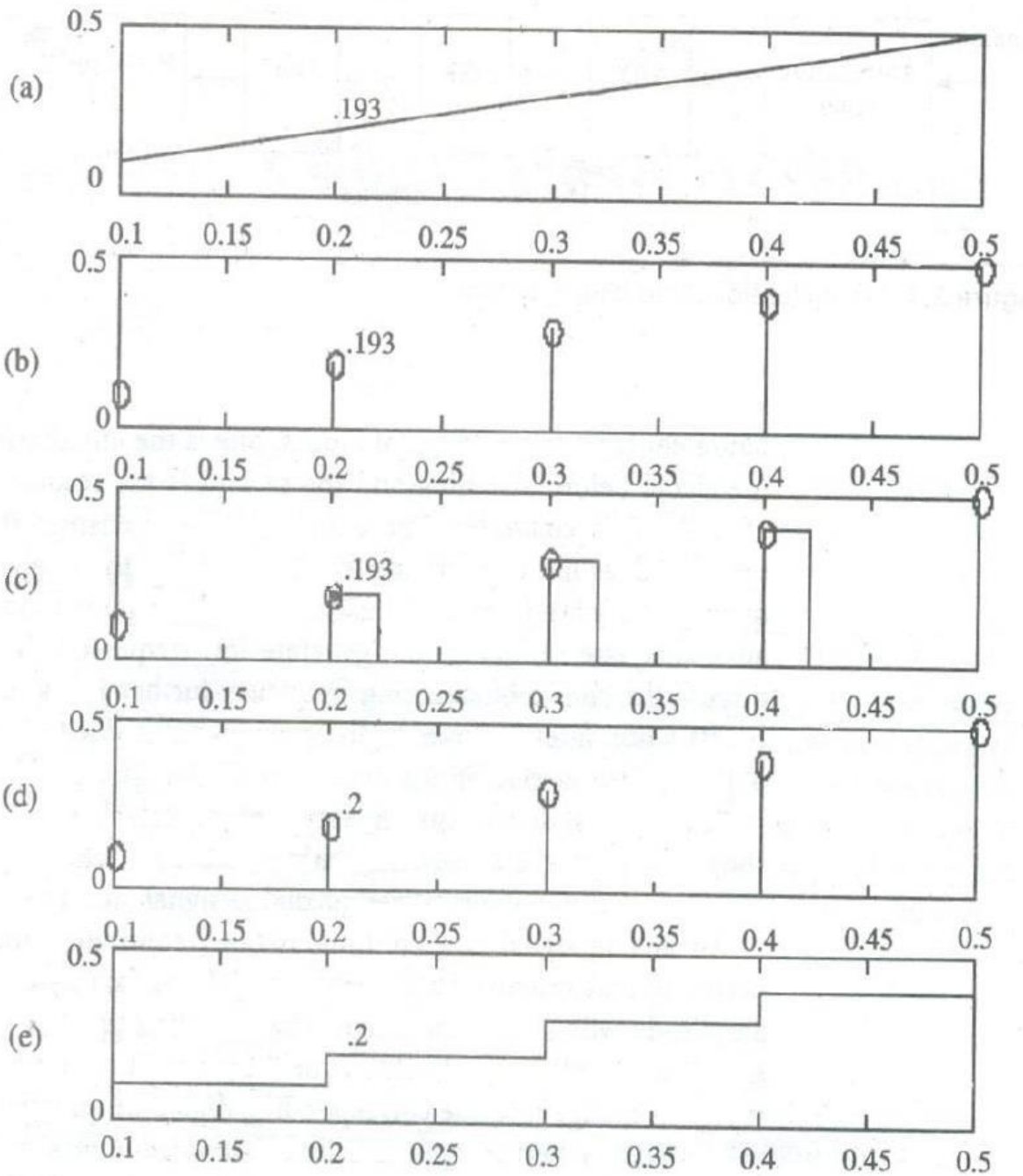


Fig 1.3: (a) Continuous time signal (b) Sampled Signal (c) Sampled Data Signal (d) Quantized Signal (e) DAC Output

1.4 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.

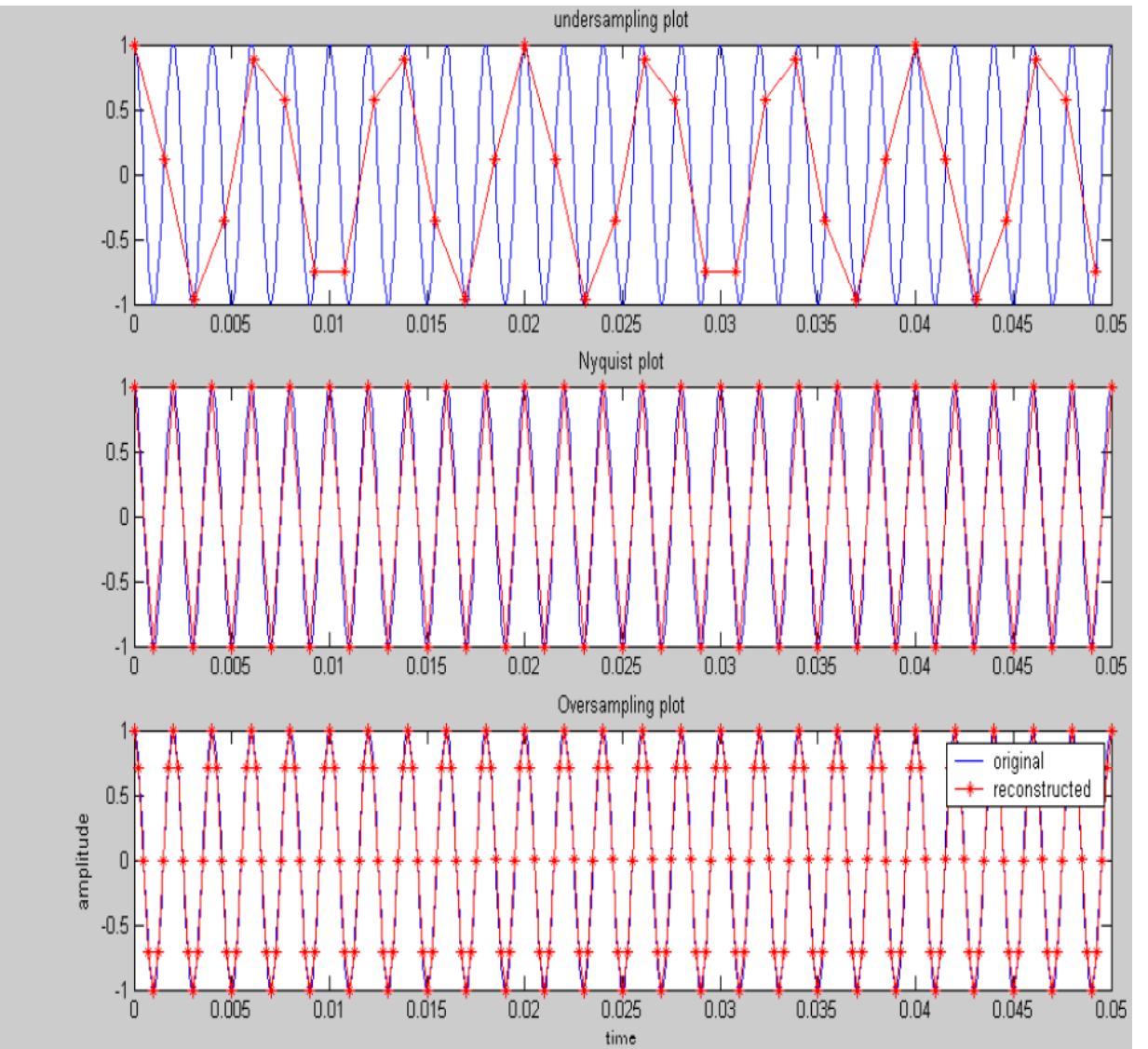


Fig 1.4 Verification of Sampling Theorem

1.5 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 πnθ

The quantity □is called as digital frequency.

θ = 2πfT = 2πf/fs radians

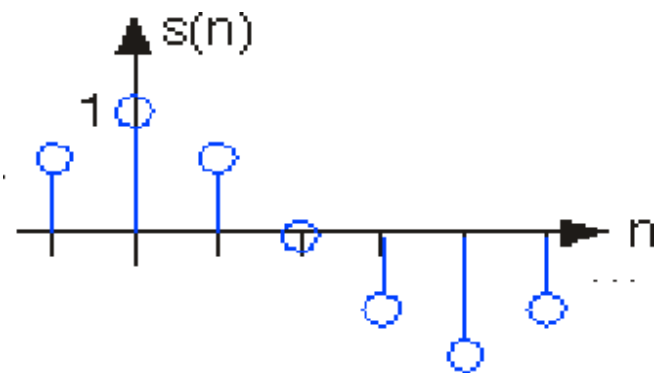


Fig 1.5 A Cosine Waveform

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 $\mathbf{X(e^{j\theta})=\sum x(n)\, e^{-jn\theta}}$

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

1.6 Discrete Fourier Transform and Fast Fourier Transform

1.6.1 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 domainsequence X (K) are called DFT Pair and is given by,

$$DFT(FFT):$$

$$X(k)=\sum_{n=0}^{N-1}x(n)\cdot e^{-j\left(\frac{2\pi}{N}\right)nk}\ (k=0,1,\ldots,N-1)$$

$$IDFT(IFFT):$$

$$x(n)=\frac{1}{N}\sum_{k=0}^{N-1}X(k)\cdot e^{j\left(\frac{2\pi}{N}\right)nk}\ (n=0,1,\ldots,N-1)$$

1.6.2 The Relationship between DFT and Frequency Response:

We have,

$$X(e^{j\theta})=\Sigma x(n)\, e^{-jn\theta}$$

Also

$$X(K)=\Sigma x(n)\, e^{-j2\pi nk/N}$$

$$\therefore X(K)=X(e^{j\theta}) \text{ at } \theta=2\pi k/N$$

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 $e^{-j2\square/N}$. Those algorithms were called as Fast Fourier Transform Algorithms. The following table depicts the complexity involved in the computation using DFT algorithms.

Table 1.1 Complexity in DFT algorithm

Operations	Number of Computations
Complex Multiplications	N^2
Complex Additions	N (N-1)
Real Multiplications	$4N^2$
Real Additions	2N (2N-1)
Trigonometric Functions	$2N^2$

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.

1.7 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.

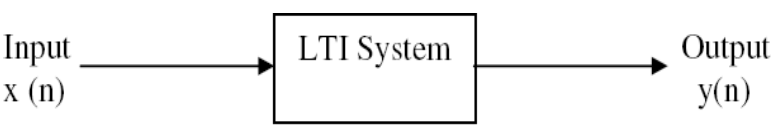


Fig 1.6 An LTI System

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.

1.7.1 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.

1.7.2 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, $\mathbf{X(Z)= \sum x(n)\ z^{-n}}$

1.7.3 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.

1.8 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) =□$\sum a_k y(n-k) + \sum$□$b_k\ x(n-k)$

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

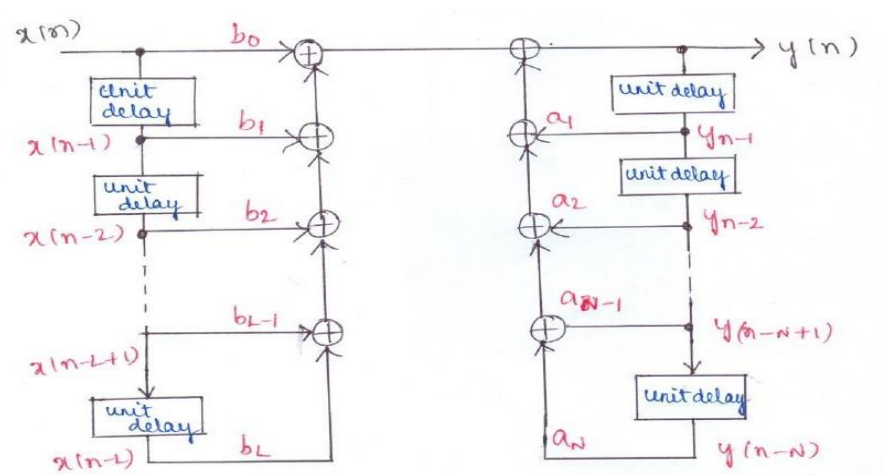


Fig 1.7 Structure of a Digital Filter

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.

1.8.1 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

$$y(n) = \Sigma b_k x(n-k)$$

The frequency response of an FIR filter is given as

$$H(e^{j\theta}) = \Sigma b_k e^{-jk\theta}$$

$$H(Z) = \Sigma b_k Z^{-k}$$

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.

1.8.2 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 suchfilters is given by,

$$H(Z) = (b_0+b_1z^{-1}+b_2z^{-2}+\ldots\ldots\ldots\ldots b_Lz^{-L}) / (1-a_1z^{-1}-a_2z^{-2}-\ldots\ldots\ldots a_Nz^{-N})$$

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.

1.8.3 FIR Filter Design

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

$$H(e^{j\theta}) = \Sigma b_k e^{-jk\theta}$$

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

$$b_k = (1/2\pi) \int H(e^{j\theta}) e^{-jk\theta} d\theta$$

1.8.4 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.

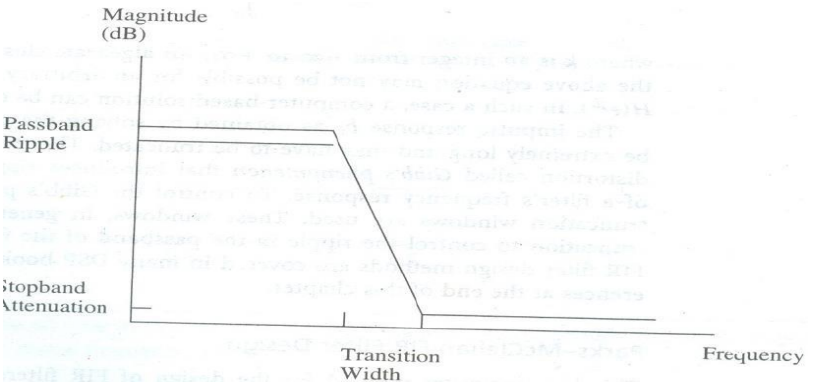


Fig 1.11 Lowpass Filter Specifications

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

1.9 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.

1.9.1 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,

$$y(m)=w(mM)=\sum b_k\, x(mM-k) \quad \text{where } w(n)=\sum b_k\, x(n-k)$$

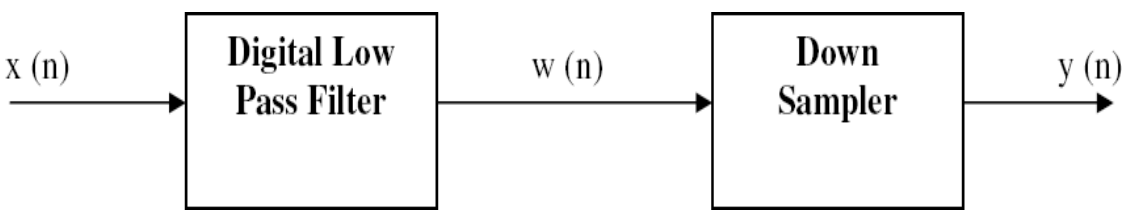


Fig 1.12 Decimation Process

1.9.2 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,

$$y(m)= \Sigma\, b_k\, w(m-k)$$

where $w(n)= x(m/L)$, $m=0,\pm L, \pm 2L\ldots\ldots$

0 Otherwise

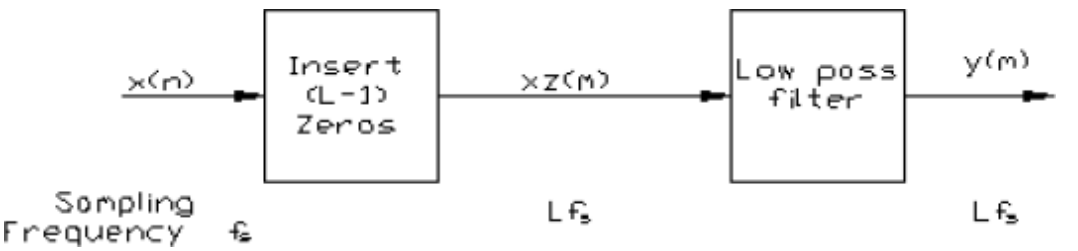
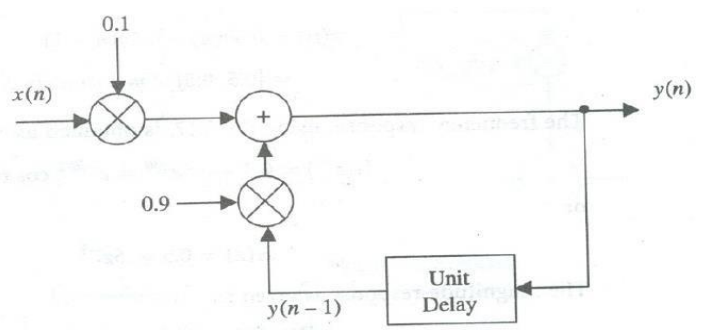


Fig 1.13 Interpolation Process

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.



UNIT-2

Architectures for Programmable Digital Signal Processing Devices

2.1 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,

a. Arithmetic operations such as ADD, SUBTRACT, MULTIPLY etc
b. Logical operations such as AND, OR, NOT, XOR etc
c. Multiply and Accumulate (MAC) operation
d. Signal scaling operation

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

a. On chip registers to store immediate results
b. On chip memories to store signal samples (RAM)
c. On chip memories to store filter coefficients (ROM)

2.2 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.

2.2.1 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:

a. Accuracy
b. Dynamic range
c. 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.

2.2.2 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,

				A_3	A_2	A_1	A_0
				B_3	B_2	B_1	B_0
				A_3B_0	A_2B_0	A_1B_0	A_0B_0
			A_3B_1	A_2B_1	A_1B_1	A_0B_1	
		A_3B_2	A_2B_2	A_1B_2	A_0B_2		
	A_3B_3	A_2B_3	A_1B_3	A_0B_3			
P7	**P6**	**P5**	**P4**	**P3**	**P2**	**P1**	**P0**

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

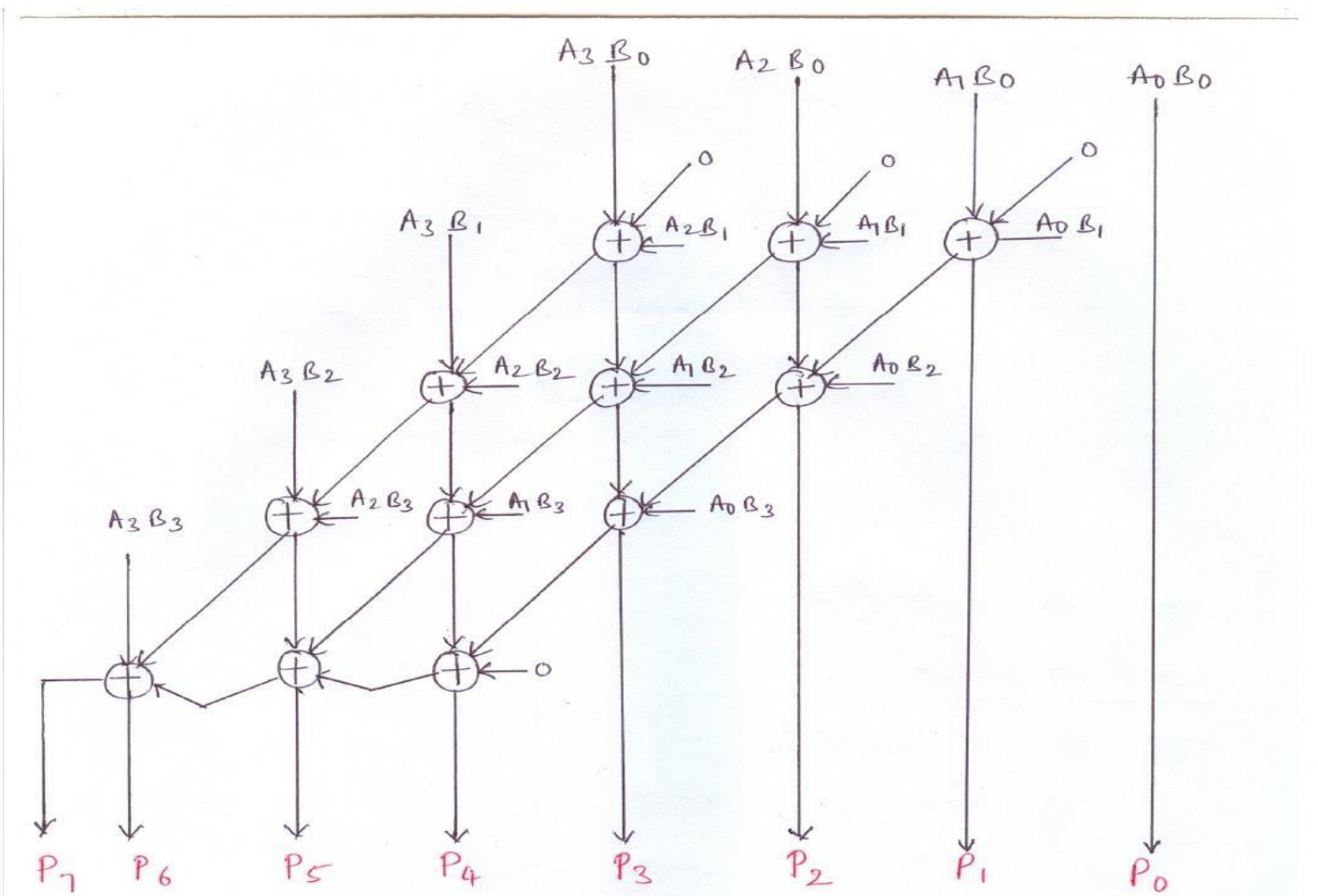


Fig 2.1 Braun Multiplier for a 4X4 Multiplication

2.2.3 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,

$$A = -A_{m-1}2^{m-1} + \sum_{i=0}^{m-2} A_i 2^i$$

$$B = -B_{n-1}2^{n-1} + \sum_{j=0}^{n-2} B_j 2^j$$

$$\text{Product } P = P_{m+n-1} \ldots\ldots P_1 P_0$$

$$P = A_{m-1}B_{n-1}2^{m+n-2} + \sum_{i=0}^{m-2}\sum_{j=0}^{n-2} A_i B_j 2^{i+j} - \sum_{i=0}^{m-2} A_i B_{n-1} 2^{n-1+I} - \sum_{j=0}^{n-2} A_{m-1} B_j 2^{m-1+j}$$

2.2.4 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.

2.2.5 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).

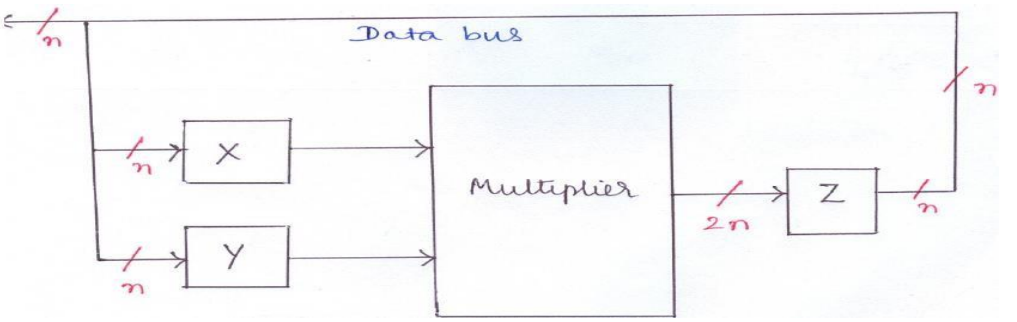


Fig 2.2: A Multiplier with Input and Output Latches

2.2.6 Shifters

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

a. 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.

b. 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.

c. 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.

2.2.7 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 ddditional 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.

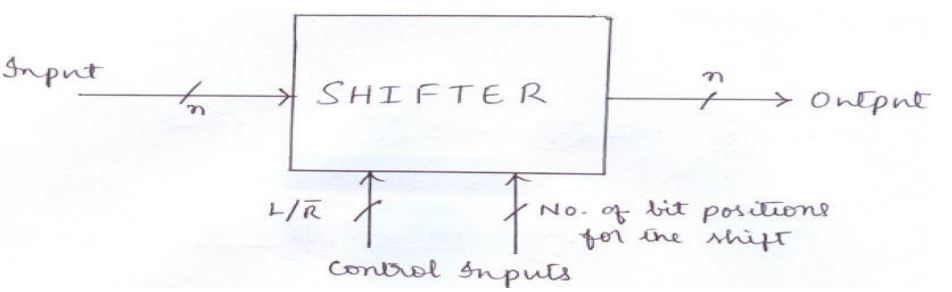
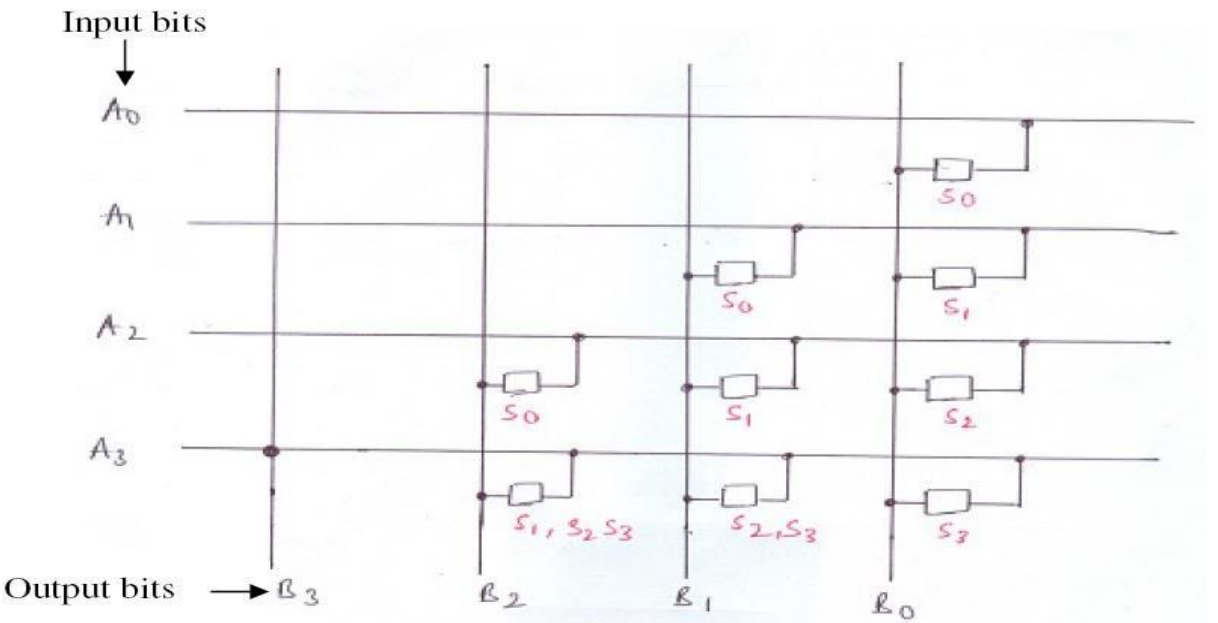


Fig 2.3 A Barrel Shifter



INPUT				SHIFT (SWITCH)	OUTPUT (B_3 B_2 B_1 B_0)			
A_3	A_2	A_1	A_0	0 (S_0)	A_3	A_2	A_1	A_0
A_3	A_2	A_1	A_0	1 (S_1)	A_3	A_3	A_2	A_1
A_3	A_2	A_1	A_0	2 (S_2)	A_3	A_3	A_3	A_2
A_3	A_2	A_1	A_0	3 (S_3)	A_3	A_3	A_3	A_3

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.

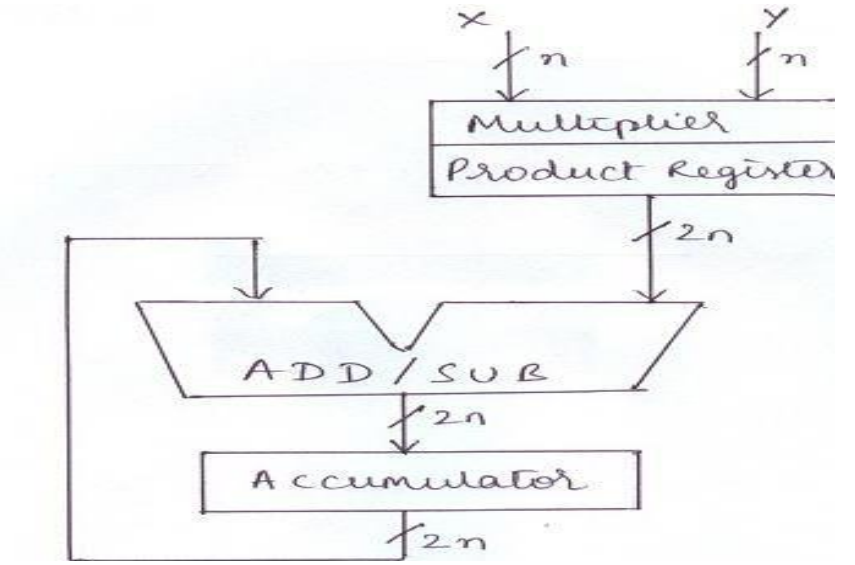


Fig 2.5 A MAC Unit

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

a. Using shifters at the input and the output of the MAC

b. Providing guard bits in the accumulator

c. 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.

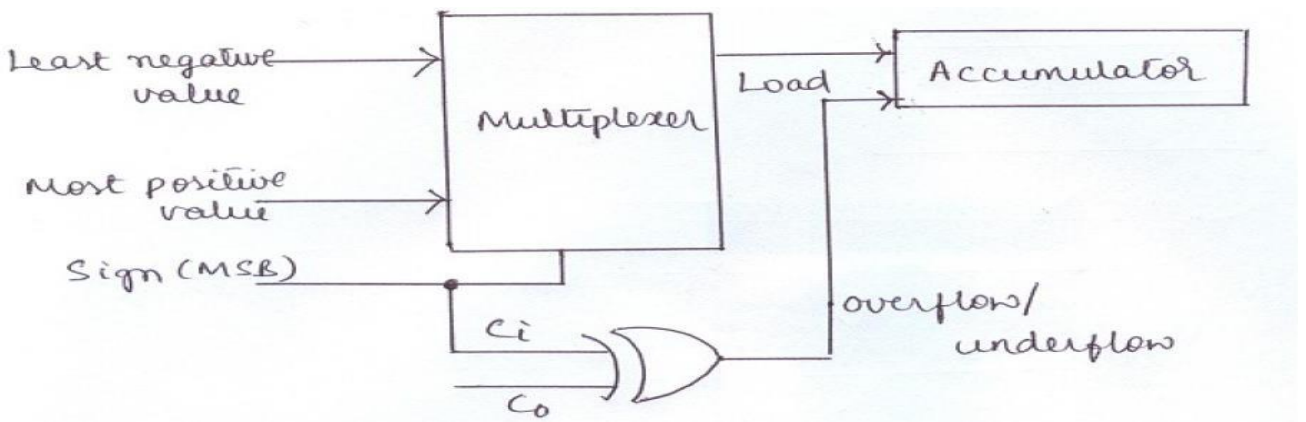


Fig 2.7: Schematic Diagram of the Saturation Logic

2.4 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.

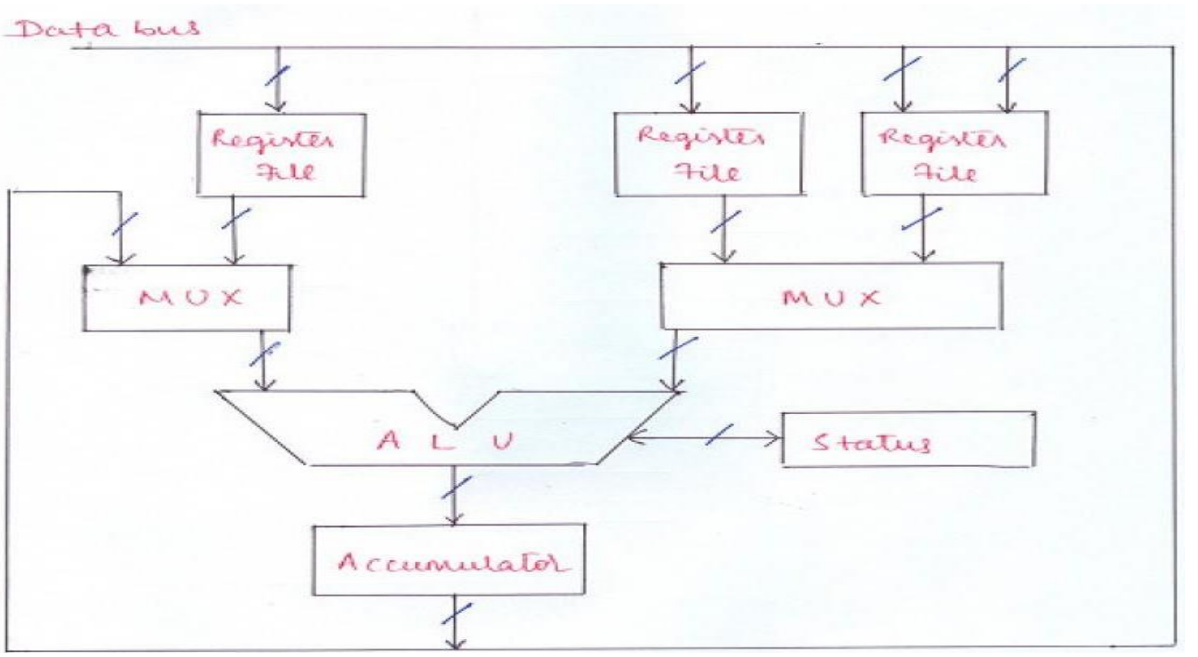


Fig 2.8 Arithmetic Logic Unit of a DSP

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.

2.5 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 thisarchitecture 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.

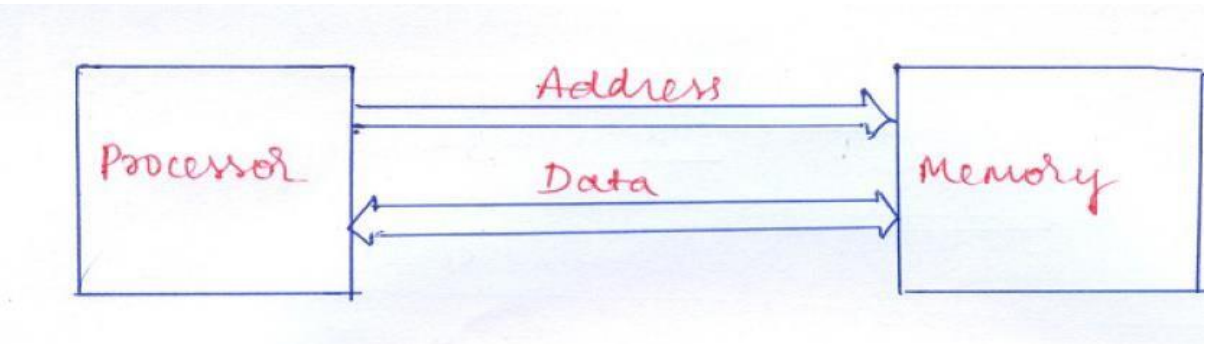


Fig 2.9 Von Neumann Architecture

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.

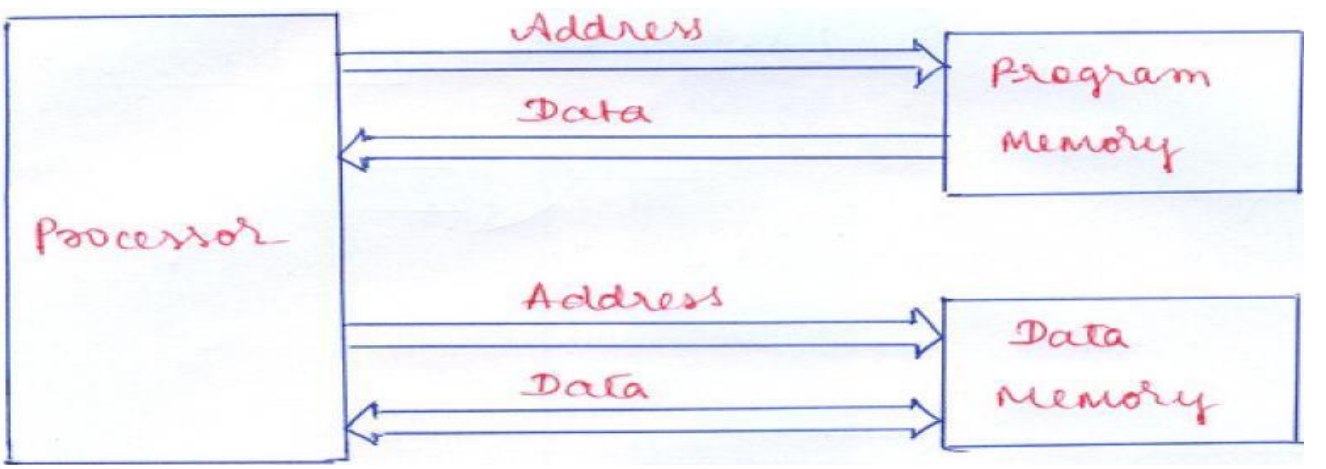


Fig 2.10 Harvard Architecture

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, thusincreasing 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 fetchedtogether (Figure 2.11).

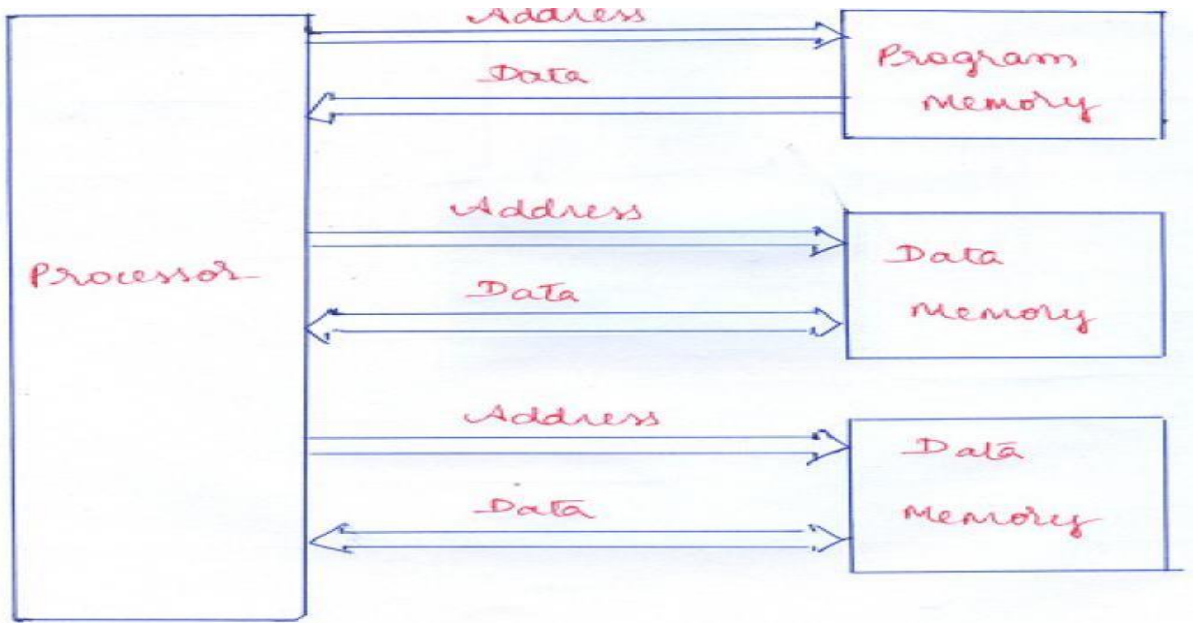


Fig 2.11 Harvard Architecture with Dual Data Memory

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.

2.5.1 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.

2.5.2 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.

a. As many DSP algorithms require instructions to be executed repeatedly, the instruction can bestored in the external memory, once it is fetched can reside in the instruction cache.
b. The access times for memories on-chip should be sufficiently small so that it can be accessed morethan once in every execution cycle.
c. On-chip memories can be configured dynamically so that they can serve different purpose atdifferent times.

2.6 Data Addressing Capabilities

Data accessing capability of a programmable DSP device is configured by means of its addressing modes. The summary of the addressing modes used in DSP is as shown in the table below.

Table 2.1 DSP Addressing Modes

Addressing Mode	Operand	Sample Format	Operation
Immediate	Immediate Value	ADD #imm	#imm +A → A
Register	Register Contents	ADD reg	reg +A → A
Direct	Memory Address Register	ADD mem	mem+A → A
Indirect	Memory contents with address in the register	ADD *addreg	*addreg +A → A

2.6.1 Immediate Addressing Mode

In this addressing mode, data is included in the instruction itself.

2.6.2 Register Addressing Mode

In this mode, one of the registers will be holding the data and the register has to be specified in the instruction.

2.6.3 Direct Addressing Mode

In this addressing mode, instruction holds the memory location of the operand.

2.6.4 Indirect Addressing Mode

In this addressing mode, the operand is accessed using a pointer. A pointer is generally a register, which holds the address of the location where the operands resides. Indirect addressing mode can be extended to inculcate automatic increment or decrement capabilities, which has lead to the following addressing modes.

Table 2.2 Indirect Addressing Modes

Addressing Mode	Sample Format	Operation
Post Increment	ADD *addreg+	A ⟶ A + *addreg addreg ⟶ addreg+1
Post Decrement	ADD *addreg-	A ⟶ A + *addreg addreg ⟶ addreg-1
Pre Increment	ADD +*addreg	addreg ⟶ addreg+1 A ⟶ A + *addreg
Pre Decrement	ADD -*addreg	addreg ⟶ addreg-1 A ⟶ A + *addreg
Post_Add_Offset	ADD *addreg, offsetreg+	A ⟶ A + *addreg addreg ⟶ addreg+offsetreg
Post_Sub_Offset	ADD *addreg, offsetreg-	A ⟶ A + *addreg addreg ⟶ addreg-offsetreg
Pre_Add_Offset	ADD offsetreg+,*addreg	addreg ⟶ addreg+offsetreg A ⟶ A + *addreg
Pre_Sub_Offset	ADD offsetreg-,*addreg	addreg ⟶ addreg-offsetreg A ⟶ A + *addreg

2.7 Special Addressing Modes

For the implementation of some real time applications in DSP, normal addressing modes will not completely serve the purpose. Thus some special addressing modes are required for such applications.

2.7.1 Circular Addressing Mode

While processing the data samples coming continuously in a sequential manner, circularbuffers are used. In a circular buffer the data samples are stored sequentially from the initial location till the buffer gets filled up. Once the buffer gets filled up, the next data samples will get stored once again from the initial location. This process can go forever as long as the data samples are processed ina rate faster than the incoming data rate.

Circular Addressing mode requires three registers viz

a. Pointer register to hold the current location (PNTR)

b. Start Address Register to hold the starting address of the buffer (SAR)

c. End Address Register to hold the ending address of the buffer (EAR)

There are four special cases in this addressing mode. They are

a. SAR < EAR & updated PNTR > EAR

b. SAR < EAR & updated PNTR < SAR

c. SAR >EAR & updated PNTR > SAR

d. SAR > EAR & updated PNTR < EAR

The buffer length in the first two case will be (EAR-SAR+1) whereas for the next tow cases (SAR-EAR+1)

The pointer updating algorithm

```
; Pointer Updating Algorithm

Updated PNTR ◄—— PNTR ± increment

If SAR < EAR
        And if Updated PNTR > EAR then
                New PNTR ◄—— Updated PNTR – Buffer size
        And if Updated PNTR < SAR then
                New PNTR          Updated PNTR + Buffer size

If SAR > EAR
        And if Updated PNTR > SAR then
                New PNTR ◄—— Updated PNTR – Buffer size
        And if Updated PNTR < EAR then
                New PNTR ◄—— Updated PNTR + Buffer size

Else
        New PNTR ◄—— Updated PNTR
```

Four cases explained earlier are as shown in the figure 2.12.

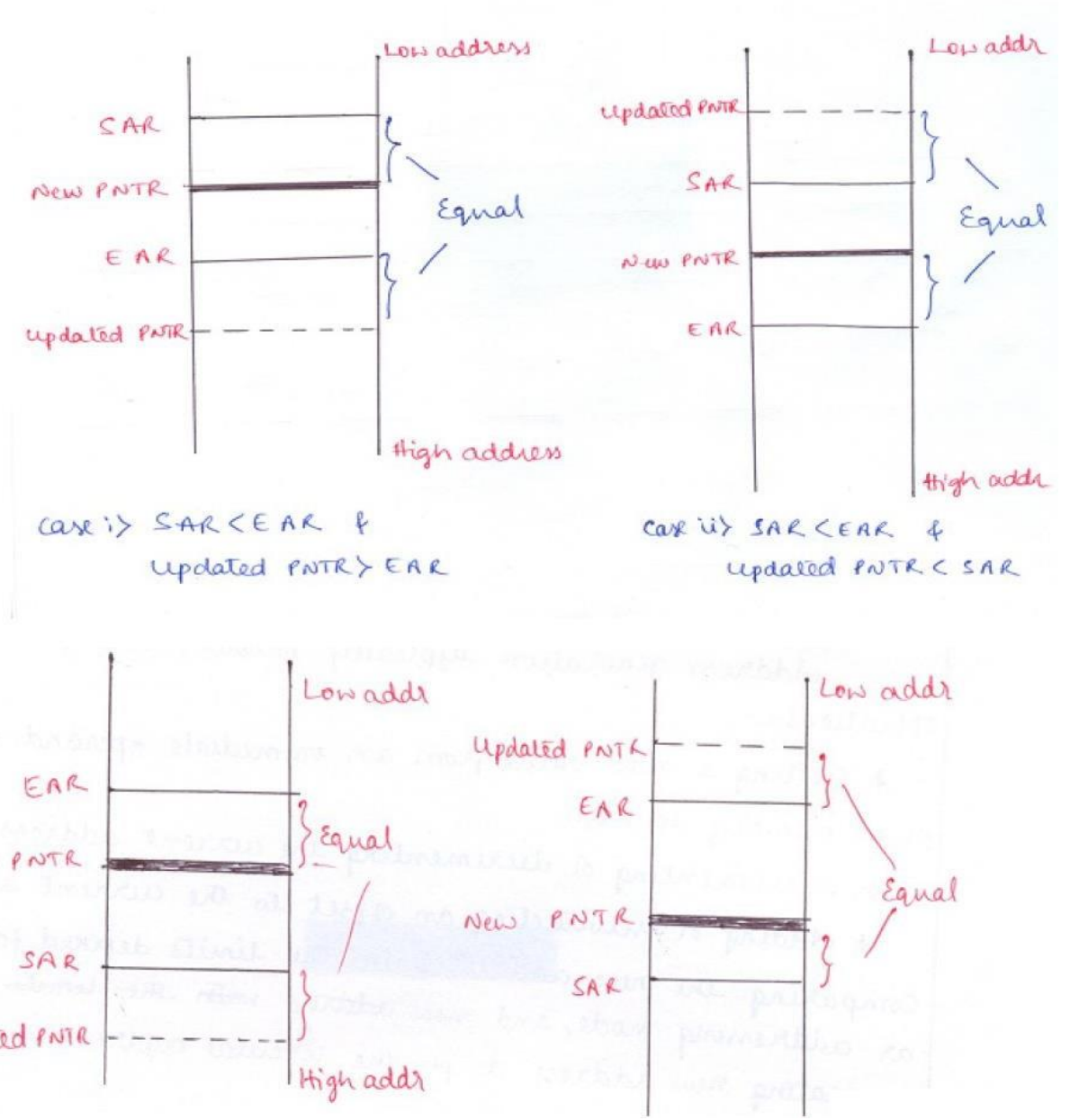


Fig 2.12 Special Cases in Circular Addressing Mode

2.7.2 Bit Reversed Addressing Mode

To implement FFT algorithms we need to access the data in a bit reversed manner. Hence a special addressing mode called bit reversed addressing mode is used to calculate the index of the next data to be fetched. It works as follows. Start with index 0. The present index can be calculated by adding half the FFT length to the previous index in a bit reversed manner, carry being propagated fromMSB to LSB.

Current index= Previous index+ B (1/2(FFT Size))

2.8 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.

a. Getting value from immediate operand, register or a memory location

b. Incrementing/ decrementing the current address

c. Adding/subtracting the offset from the current address

d. Adding/subtracting the offset from the current address and generating new address according tocircular addressing mode

e. Generating new address using bit reversed addressing mode

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

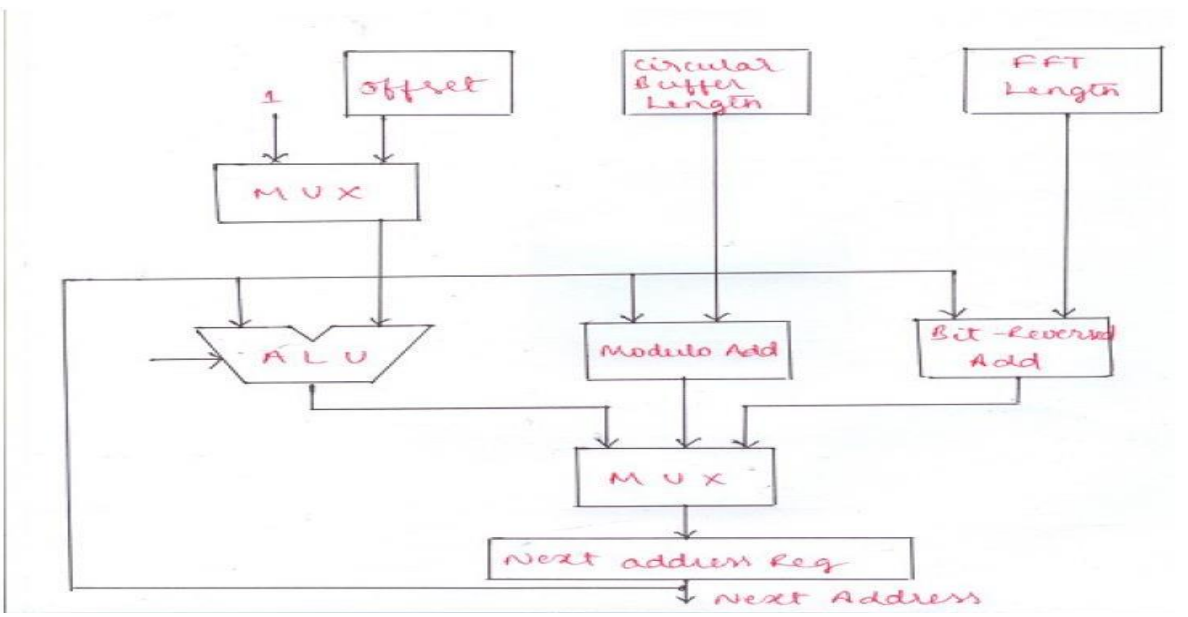


Fig 2.13 Address generation unit

2.9 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.

2.9.1 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.

2.9.2 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.

a. Program Counter

b. Instruction register in case of branching, looping and subroutine calls

c. Interrupt Vector table

d. Stack which holds the return address

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

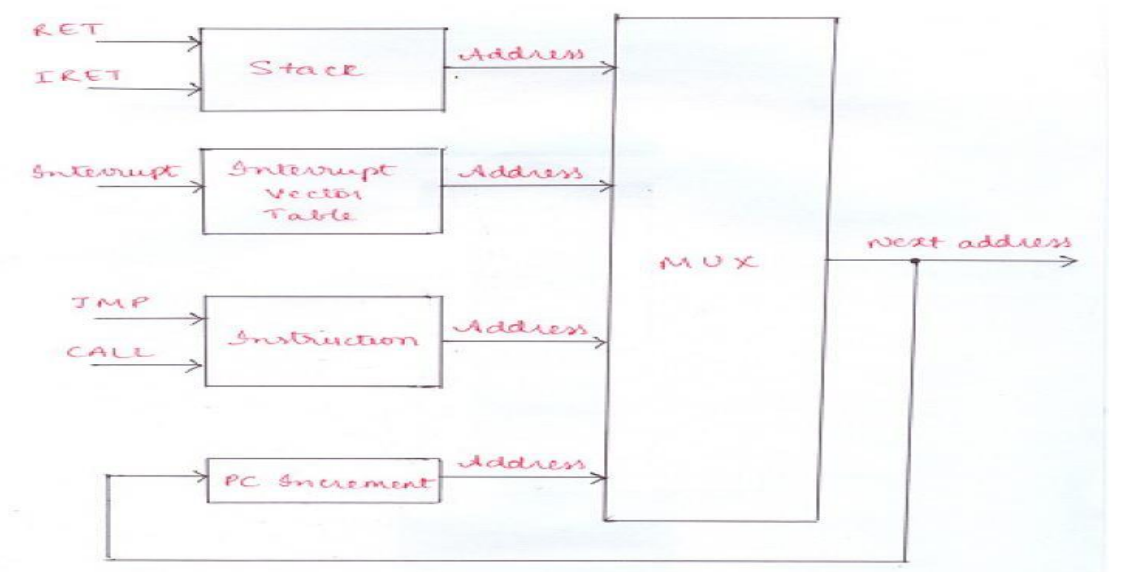


Fig 2.14 Program Sequencer

Program sequencer should have the following circuitry:

a. PC has to be updated after every fetch

b. Counter to hold count in case of looping

c. A logic block to check conditions for conditional jump instructions

d. Condition logic-status flag

Problems:

1). Investigate the basic features that should be provided in the DSP architecture to be used toimplement the following N^{th} order FIR filter.

Solution:-

$y(n)= \sum h(i)\, x(n-i)$ n=0,1,2…
In order to implement the above operation in a DSP, the architecture requires the following features

i. A RAM to store the signal samples x (n)
ii. A ROM to store the filter coefficients h (n)
iii. An MAC unit to perform Multiply and Accumulate operation
iv. An accumulator to store the result immediately
v. A signal pointer to point the signal sample in the memory
vi. A coefficient pointer to point the filter coefficient in the memory
vii. A counter to keep track of the count
viii. A shifter to shift the input samples appropriately

2). It is required to find the sum of 64, 16 bit numbers. How many bits should theaccumulator have so that the sum can be computed without the occurrence of overflow error or loss of accuracy?

The sum of 64, 16 bit numbers can grow up to (16+ log2 64)=22 bits long. Hence the accumulator should be 22 bits long in order to avoid overflow error from occurring.

1. In the previous problem, it is decided to have an accumulator with only 16 bitsbut shift the numbers before the addition to prevent overflow, by how many bits should each number be shifted?

As the length of the accumulator is fixed, the operands have to be shifted by an amount of log2 64 = 6 bits prior to addition operation, in order to avoid the condition of overflow.

2. If all the numbers in the previous problem are fixed point integers, what is theactual sum of the numbers?

The actual sum can be obtained by shifting the result by 6 bits towards left side after the sum being computed. Therefore
Actual Sum= Accumulator content X 2^{6}

3. If a sum of 256 products is to be computed using a pipelined MAC unit, and if the MAC execution time of the unit is 100nsec, what will be the total time required to complete theoperation?

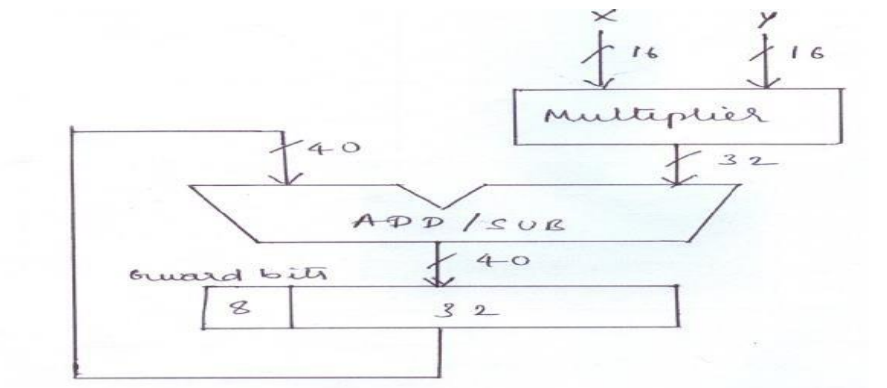
As N=256 in this case, MAC unit requires N+1=257execution cycles. As the single MAC execution time is 100nsec, the total time required will be, (257*100nsec)=25.7usec

4. Consider a MAC unit whose inputs are 16 bit numbers. If 256 products are to be summed up in this MAC, how many guard bits should be provided for the

accumulator to prevent overflow condition from occurring?

As it is required to calculate the sum of 256, 16 bit numbers, the sum can be as long as (16+ log2 256)=24 bits. Hence the accumulator should be capable of handling these 22 bits. Thus the guard bits required will be (24-16)= 8 bits.

The block diagram of the modified MAC after considering the guard or extention bits is as shown in the figure



5. What are the memory addresses of the operands in each of the following cases of indirect addressing modes? In each case, what will be the content of the *addreg* after the memory access? Assume that the initial contents of the *addreg* and the *offsetreg* are 0200h and 0010h,respectively.

a. **ADD **addreg***

b. **ADD *+*addreg***

c. ***ADD*** **offsetreg+,*addreg**

d. **ADD **addreg,offsetreg-***

Instruction	Addressing Mode	Operand Address	addreg Content after Access
ADD **addreg-*	Post Decrement	0200h	0200-01=01FFh
ADD +**addreg*	Pre Increment	0200+01=0201h	0201h
ADD *offsetreg+,*addreg*	Pre_Add_Offset	0200+0010=0210h	0210h
ADD **addreg,offsetreg-*	Post_Sub_Offset	0200h	0200-0010=01F0h

6. A DSP has a circular buffer with the start and the end addresses as 0200h and 020Fh respectively. What would be the new values of the address pointer of the buffer if, in the courseof address computation, it gets updated to

0212h

b. 01FCh

Buffer Length= (EAR-SAR+1) = 020F-0200+1=10h

a. New Address Pointer= Updated Pointer-buffer length = 0212-10=0202h
b. New Address Pointer= Updated Pointer+ buffer length = 01FC+10=020Ch

7. Repeat the previous problem for SAR= 0210h and EAR=0201h

Buffer Length= (SAR-EAR+1)= 0210-0201+1=10h
c. New Address Pointer= Updated Pointer- buffer length = 0212-10=0202h
d. New Address Pointer= Updated Pointer+ buffer length = 01FC+10=020Ch

9. Compute the indices for an 8-point FFT using Bit reversed Addressing Mode

Start with index 0. Therefore the first index would be (000)

Next index can be calculated by adding half the FFT length, in this case it is (100) to the previous index. i.e. Present Index= (000)+B (100)= (100)
Similarly the next index can be calculated as
Present Index= (100)+B (100)= (010)
The process continues till all the indices are calculated. The following table summarizes the calculation.

Index in Binary	BCD value	Bit reversed index	BCD value
000	0	000	0
001	1	100	4
010	2	010	2
011	3	110	6
100	4	001	1
101	5	101	5
110	6	011	3
111	7	111	7

UNIT-3

Programmable Digital Signal Processors

3.1 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

3.2 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

Architectural Feature	TMS320C25	DSP 56000	ADSP2100
Data representation format	16-bit fixed	24-bit fixed point	16-bit fixed point
Hardware multiplier	16 x 16	24 x 24	16 x 16
ALU	32 bits	56 bits	40 bits
Internal buses	16-bit program bus 16-bit data bus	24-bit program bus 2 x 24-bit data buses 24-bit global	24-bit program bus 16-bit data bus 16-bit result

		databus	bus
External buses	16-bit program/data bus	24-bit program/data bus	24-bit program bus 16-bit data bus
On-chip Memory	544 words RAM 4K words ROM	512 words PROM 2 x 256 words data RAM 2 x 256 words data ROM	-
Off-chip memory	64 K words program 64k words data	64K words program 2 x 64K words data	16K words program 16K words data
Cache memory	-	-	16 words program
Instruction cycle time	100 nsec	97.5 nsec.	125 nsecc.
Special addressing modes	Bit reversed	Modulo Bit reversed	Modulo Bit reversed
Data address generators	1	2	2
Interfacing features	Synchronous serial I/O DMA	Synchronous and Asynchronous serial I/O DMA	DMA

3.3. 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.

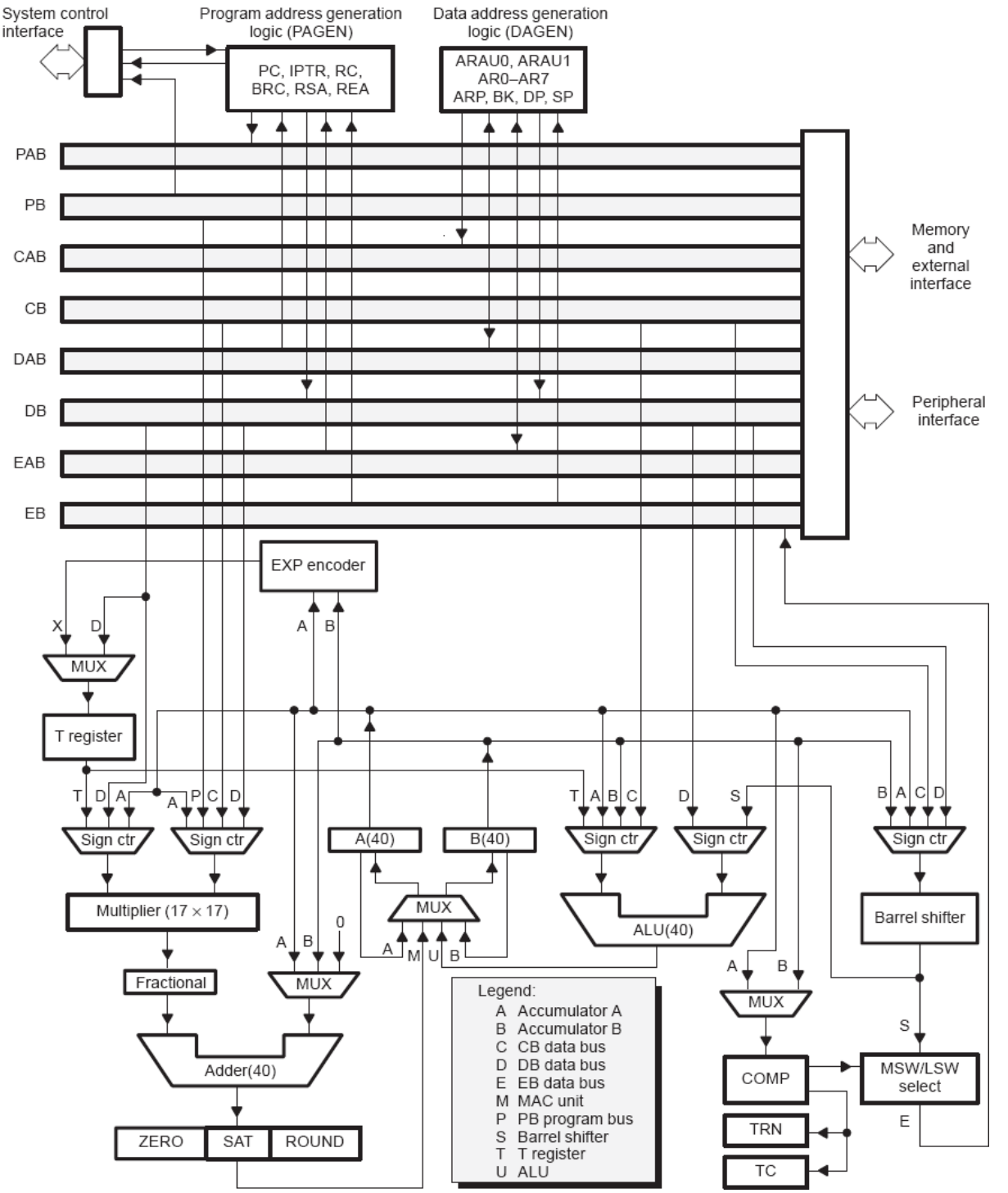


Figure 3.1.Functional architecture for TMS320C54xx processors.

3.3.1 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 '54xxcan generate up to two data-memory addresses per cycle using the two auxiliary register arithmeticunit (ARAU0 and ARAU1) in the DAGEN block. This enables accessing two operands simultaneously.

3.3.2 Central Processing Unit (CPU):

The '54xx CPU is common to all the '54xx devices. The '54xx CPU contains a 40-bitarithmetic 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.

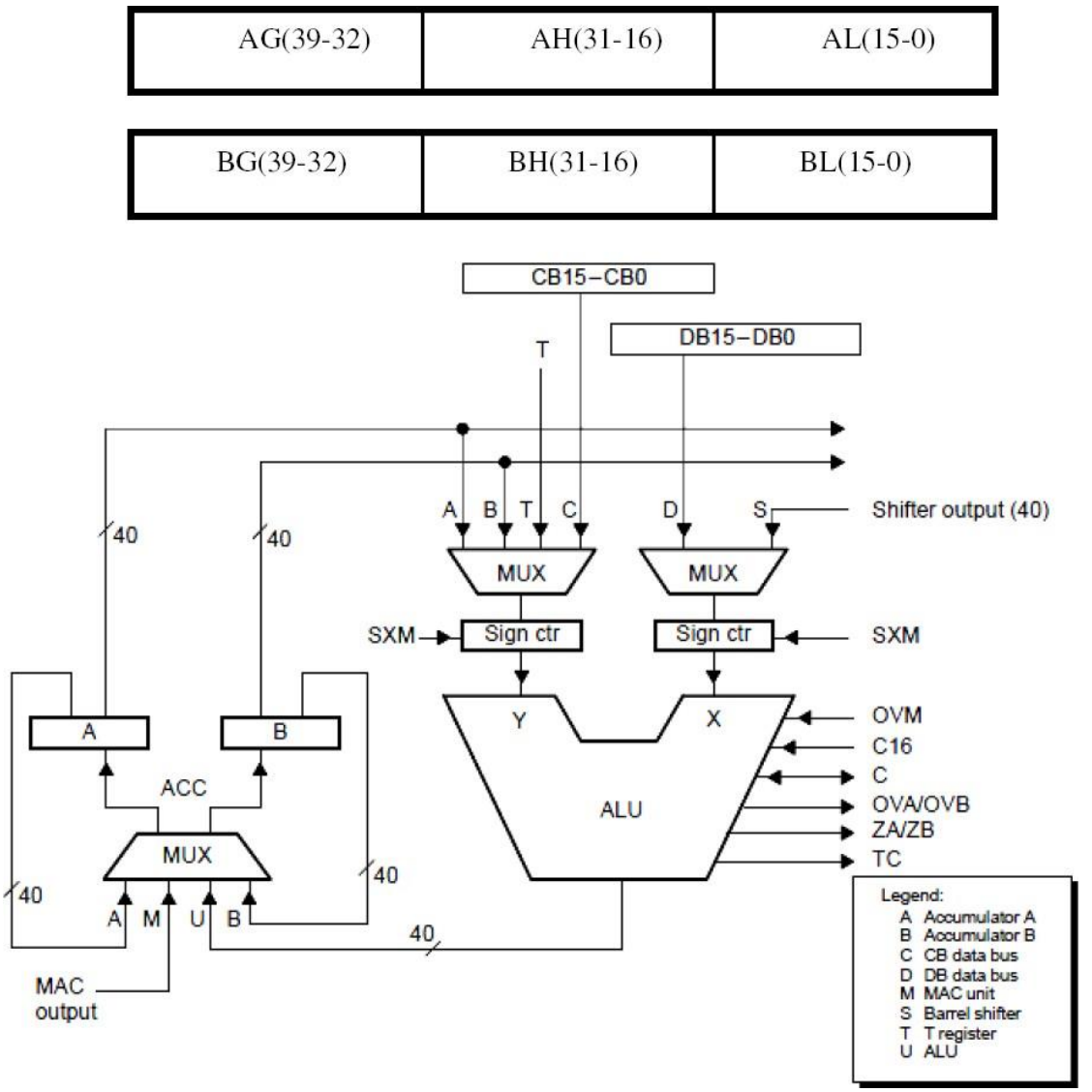


Figure 3.2.Functional diagram of the central processing unit of the TMS320C54xxprocessors.

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.

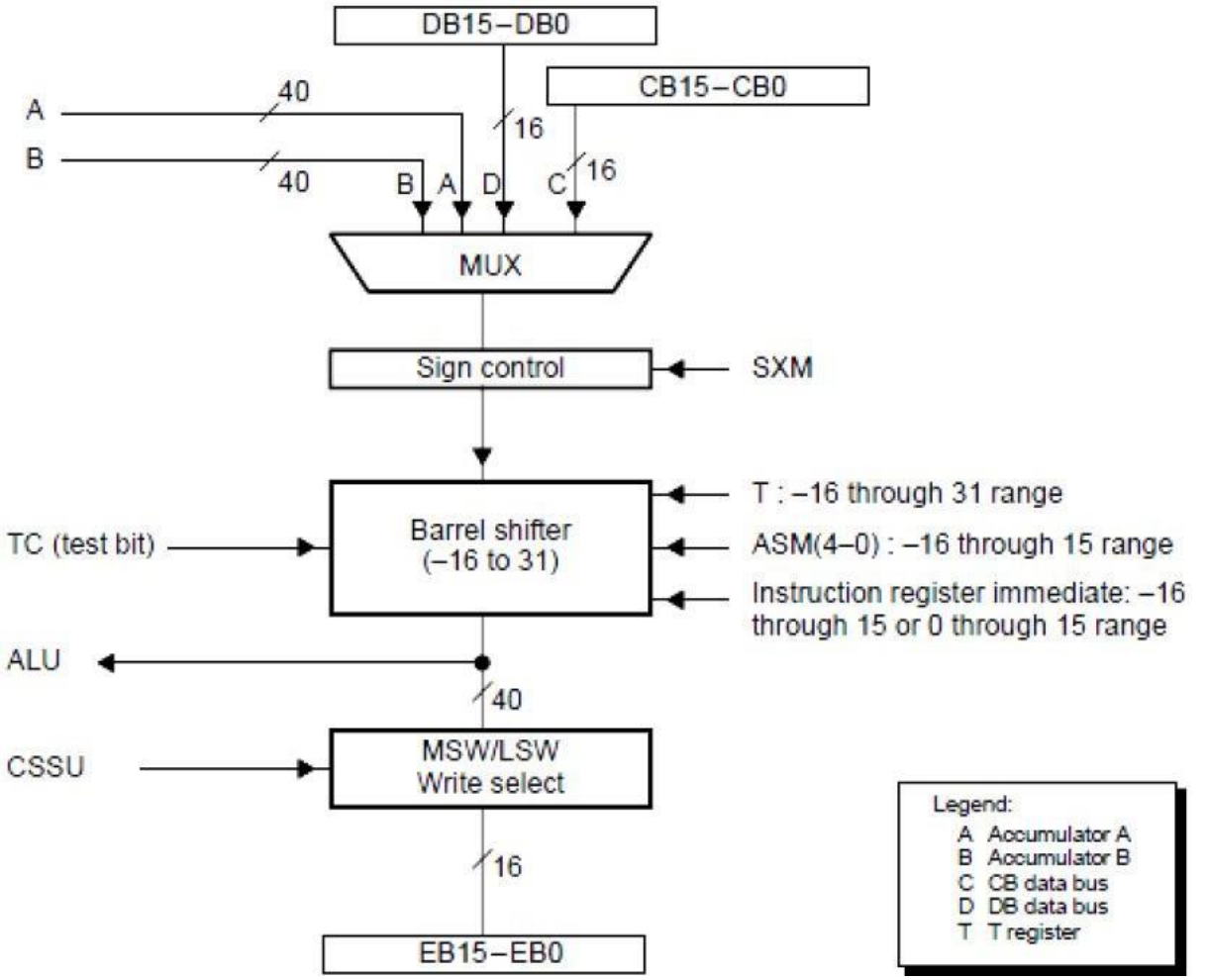


Figure 3.3.Functional diagram of the barrel shifter

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.

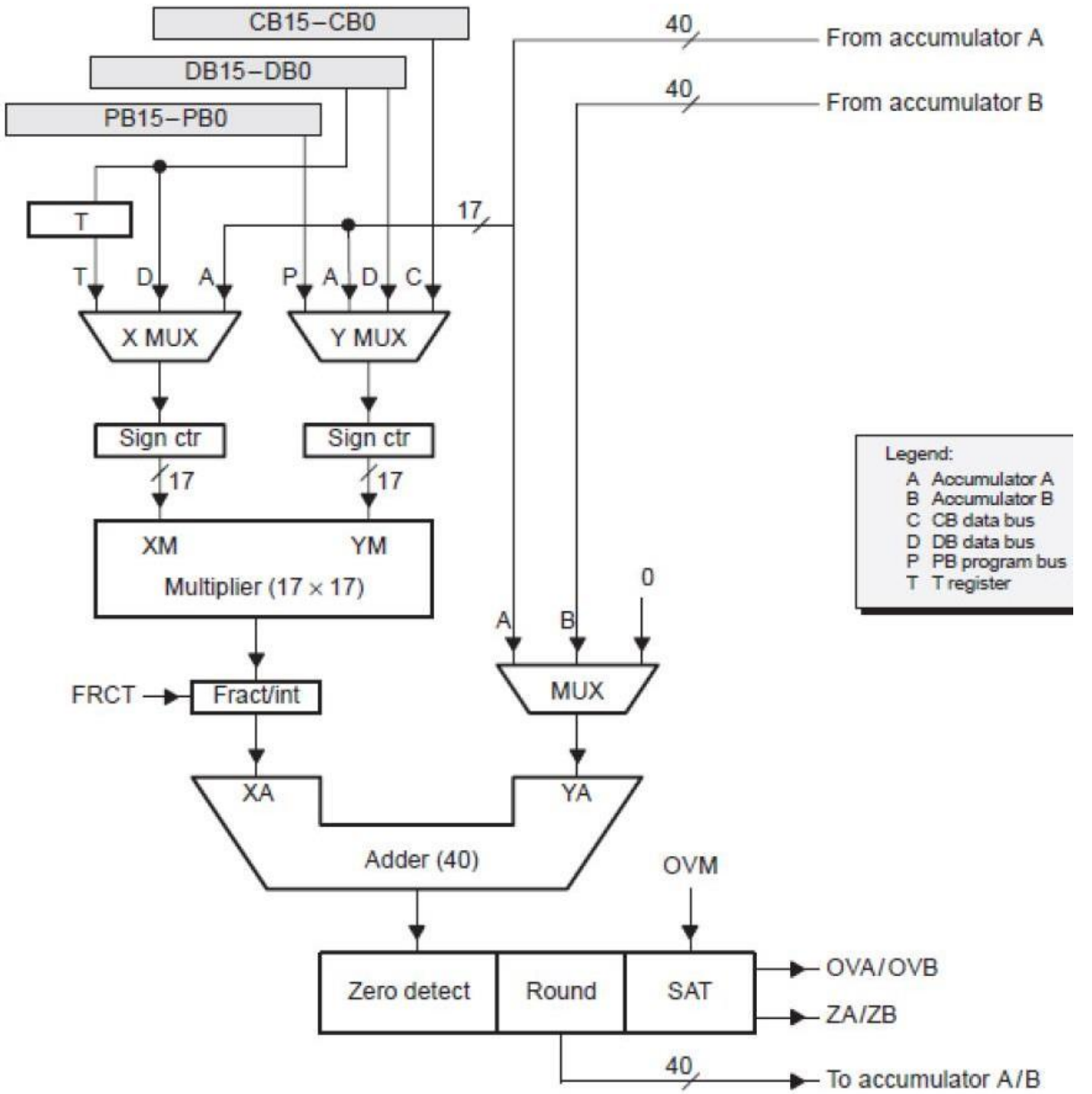


Figure 3.4. Functional diagram of the multiplier/adder unit of TMS320C54xx processors.

3.3.3 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). Theon-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.

NAME	DEC	HEX	DESCRIPTION
IMR	0	0	Interrupt mask register
IFR	1	1	Interrupt flag register
—	2–5	2–5	Reserved for testing
ST0	6	6	Status register 0
ST1	7	7	Status register 1
AL	8	8	Accumulator A low word (15–0)
AH	9	9	Accumulator A high word (31–16)
AG	10	A	Accumulator A guard bits (39–32)
BL	11	B	Accumulator B low word (15–0)
BH	12	C	Accumulator B high word (31–16)
BG	13	D	Accumulator B guard bits (39–32)
TREG	14	E	Temporary register
TRN	15	F	Transition register
AR0	16	10	Auxiliary register 0
AR1	17	11	Auxiliary register 1
AR2	18	12	Auxiliary register 2
AR3	19	13	Auxiliary register 3
AR4	20	14	Auxiliary register 4
AR5	21	15	Auxiliary register 5
AR6	22	16	Auxiliary register 6
AR7	23	17	Auxiliary register 7
SP	24	18	Stack pointer register
BK	25	19	Circular buffer size register
BRC	26	1A	Block repeat counter
RSA	27	1B	Block repeat start address
REA	28	1C	Block repeat end address
PMST	29	1D	Processor mode status (PMST) register
XPC	30	1E	Extended program page register
—	31	1F	Reserved

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

NAME	ADDRESS DEC	ADDRESS HEX	DESCRIPTION
DRR20	32	20	McBSP 0 Data Receive Register 2
DRR10	33	21	McBSP 0 Data Receive Register 1
DXR20	34	22	McBSP 0 Data Transmit Register 2
DXR10	35	23	McBSP 0 Data Transmit Register 1
TIM	36	24	Timer Register
PRD	37	25	Timer Period Register
TCR	38	26	Timer Control Register
—	39	27	Reserved
SWWSR	40	28	Software Watt-State Register
BSCR	41	29	Bank-Switching Control Register
—	42	2A	Reserved
SWCR	43	2B	Software Watt-State Control Register
HPIC	44	2C	HPI Control Register (HMODE = 0 only)
—	45–47	2D–2F	Reserved
DRR22	48	30	McBSP 2 Data Receive Register 2
DRR12	49	31	McBSP 2 Data Receive Register 1
DXR22	50	32	McBSP 2 Data Transmit Register 2
DXR12	51	33	McBSP 2 Data Transmit Register 1
SPSA2	52	34	McBSP 2 Subbank Address Register
SPSD2	53	35	McBSP 2 Subbank Data Register
—	54–55	36–37	Reserved
SPSA0	56	38	McBSP 0 Subbank Address Register
SPSD0	57	39	McBSP 0 Subbank Data Register
—	58–59	3A–3B	Reserved
GPIOCR	60	3C	General-Purpose I/O Control Register
GPIOSR	61	3D	General-Purpose I/O Status Register
CSIDR	62	3E	Device ID Register
—	63	3F	Reserved
DRR21	64	40	McBSP 1 Data Receive Register 2
DRR11	65	41	McBSP 1 Data Receive Register 1
DXR21	66	42	McBSP 1 Data Transmit Register 2
DXR11	67	43	McBSP 1 Data Transmit Register 1
—	68–71	44–47	Reserved
SPSA1	72	48	McBSP 1 Subbank Address Register
SPSD1	73	49	McBSP 1 Subbank Data Register
—	74–83	4A–53	Reserved
DMPREC	84	54	DMA Priority and Enable Control Register
DMSA	85	55	DMA Subbank Address Register

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.

Status register0 diagram:

ARP (15-13)	TC (12)	C (11)	OVA (10)	OVB (9)	DP (8-0)

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.

Status register1 diagram:

BRAF(15)	CPL (14)	XF (13)	HM (12)	INTM (11)	0 (10)	OVM (9)	SXM (8)	C16 (7)	FRCT(6)	CMPT(5)	ASM (4-0)

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):

IPTR(15-7)	MP/MC(6)	OVLY(5)	AVIS(4)	DROM(3)	CLKOFF(2)	SMUL(1)	SST(0)

Figure 3.6(c).PMST register diagram

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.

3.4.1 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.

3.4.2 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 .

3.4.3 Accumulator Addressing:

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

Ex: READA *AR2

3.4.4 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 iscomputed by adding the 7-bit offset to SP

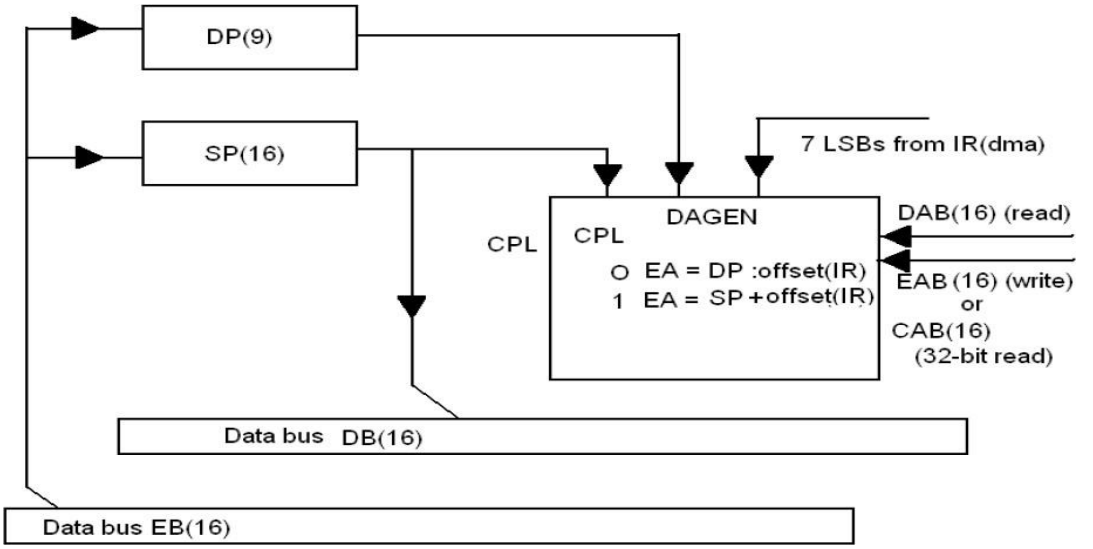


Figure 3.7 Block diagram of the direct addressing mode for TMS320C54xx Processors.

3.4.1 Indirect Addressing:

- Data space is accessed 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

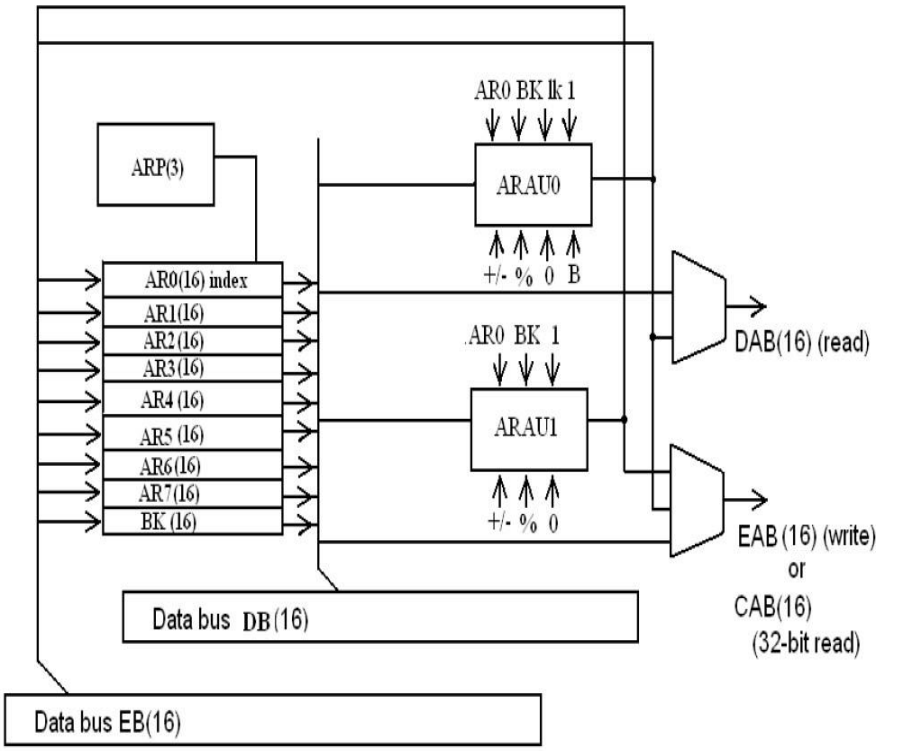


Figure 3.8 Block diagram of the indirect addressing mode for TMS320C54xx Processors.

Operand syntax	Function
*ARx	Addr = ARx;
*ARx -	Addr = ARx ; ARx = ARx -1
*ARx +	Addr = ARx; ARx = ARx +1
*+ARx	Addr = ARx+1; ARx = ARx +1
*ARx - 0B	Addr = ARx ; ARx = B(ARx – AR0)
*ARx - 0	Addr = Arx ; ARx = ARx – AR0
*ARx + 0	Addr = Arx ; ARx = ARx +AR0
*ARx + 0B	Addr = ARx ; ARx = B(ARx + AR0)
*ARx - %	Addr = ARx ; ARx = circ(ARx – 1)
*+AR – 0%	Addr = Arx; ARx = circ(ARx - AR0)
*ARx + %	Addr = ARx ; ARx = circ (ARx + 1)

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 N LSBs 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

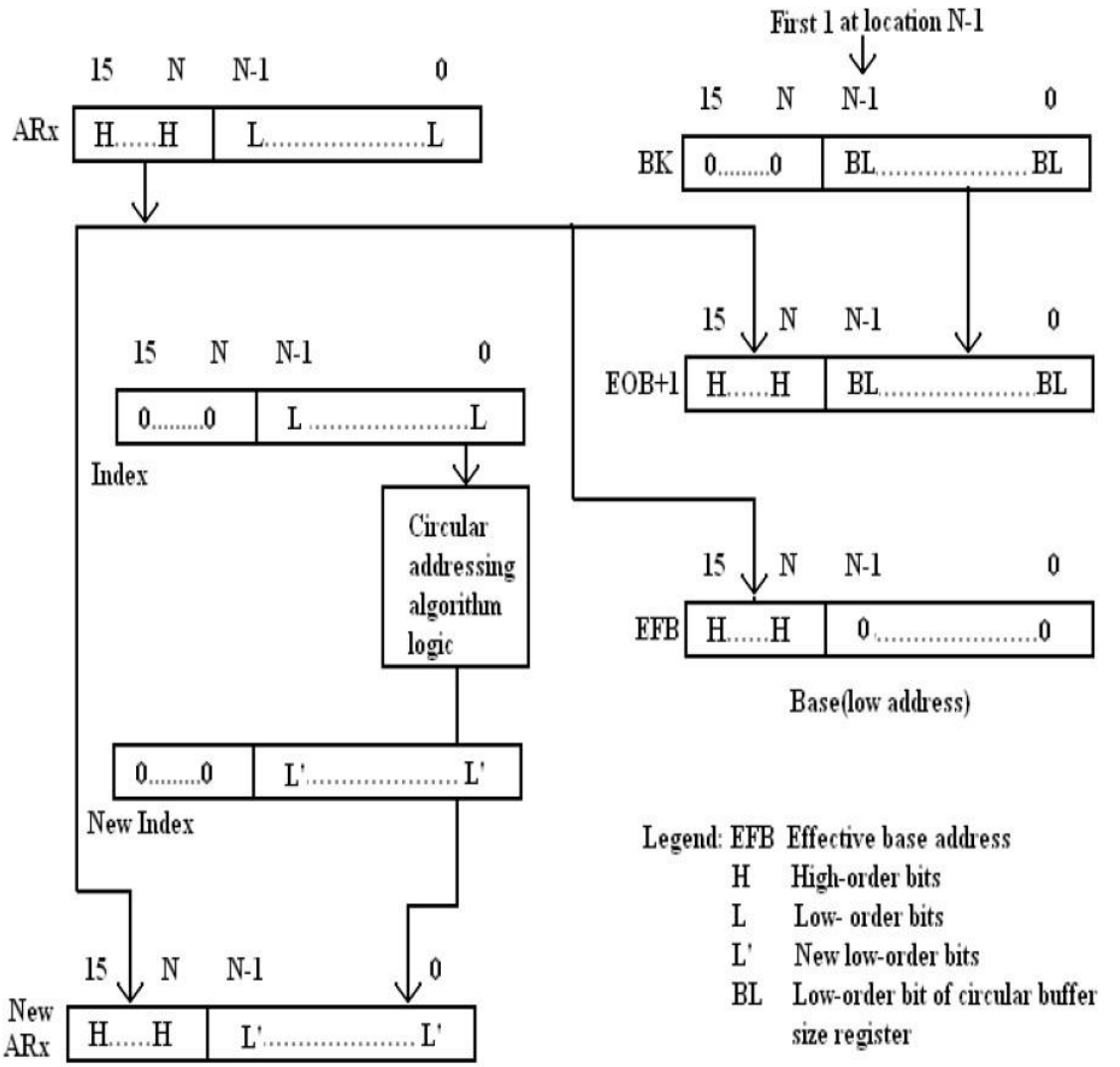


Figure 3.9 Block diagram of the circular addressing mode for TMS320C54xx Processors.

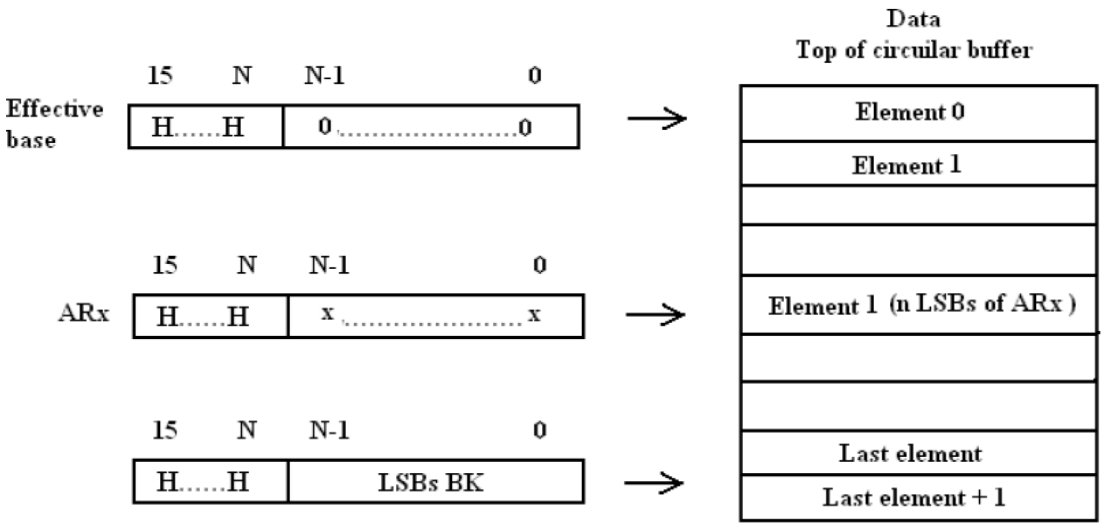


Figure 3.10 circular addressing mode implementation for TMS320C54xx Processors.

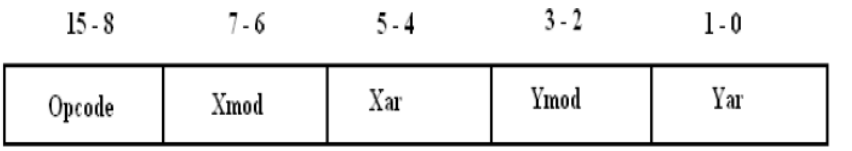
Bit-Reversed Addressing:

- 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.



Name	Function
Opcode	This field contains the operation code for the instruction
Xmod	Defined the type of indirect addressing mode used for accessing the Xmem operand
XAR	Xmem AR selection field defines the AR that contains the address of Xmem
Ymod	Defies the type of inderect addressing mode used for accessing the Ymem operand
Yar	Ymem AR selection field defines the AR that contains the address of Ymem

Table 3.3.Function of the different field in dual data memory operand addressing

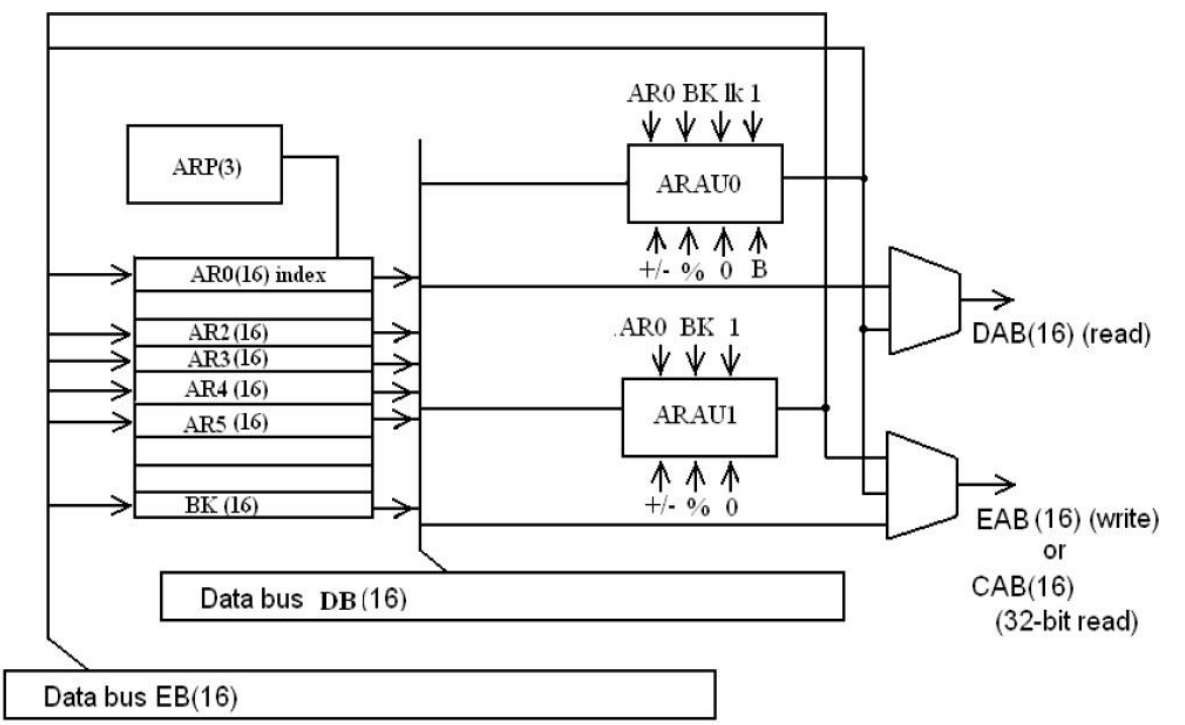


Figure 3.11 Block diagram of the Indirect addressing options with a dual data –memory operand.

3.4.6. 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

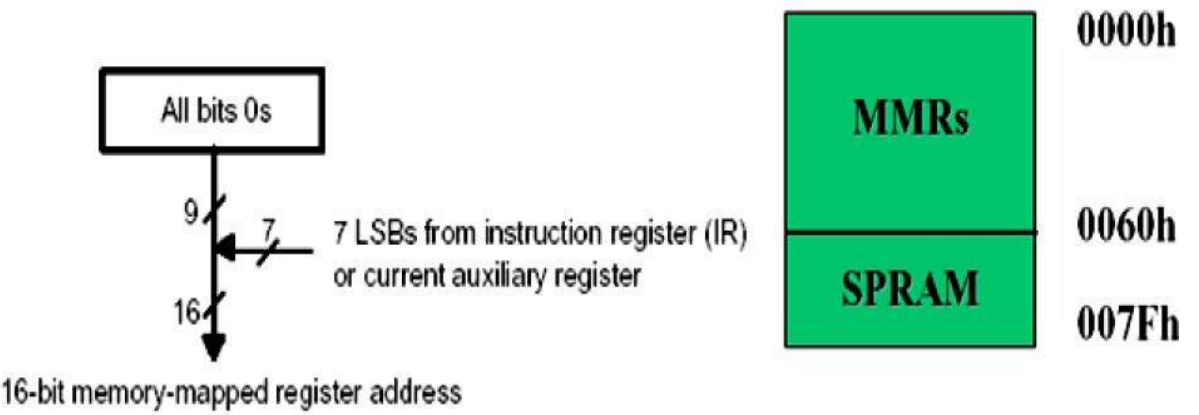


Figure 3.12.16 bit memory mapped register address generation.

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

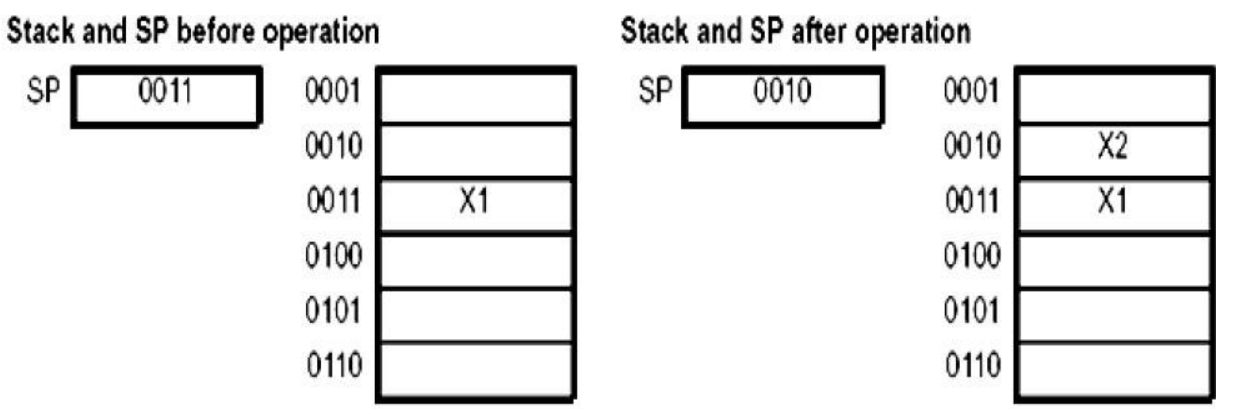
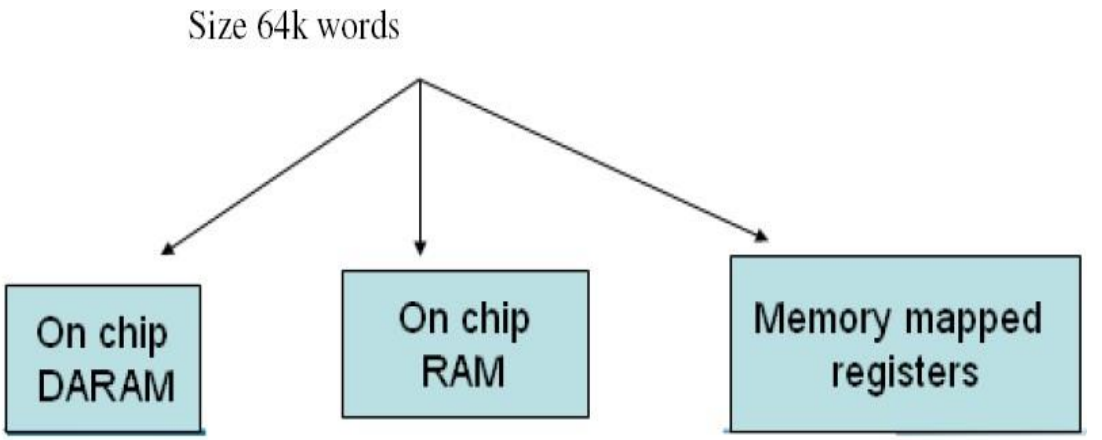


Figure 3.13. Values of stack &SP before and after operation.

3.5. 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 memory mapped registers.



Program memory: To store program instructions &tables used in the execution of programs.

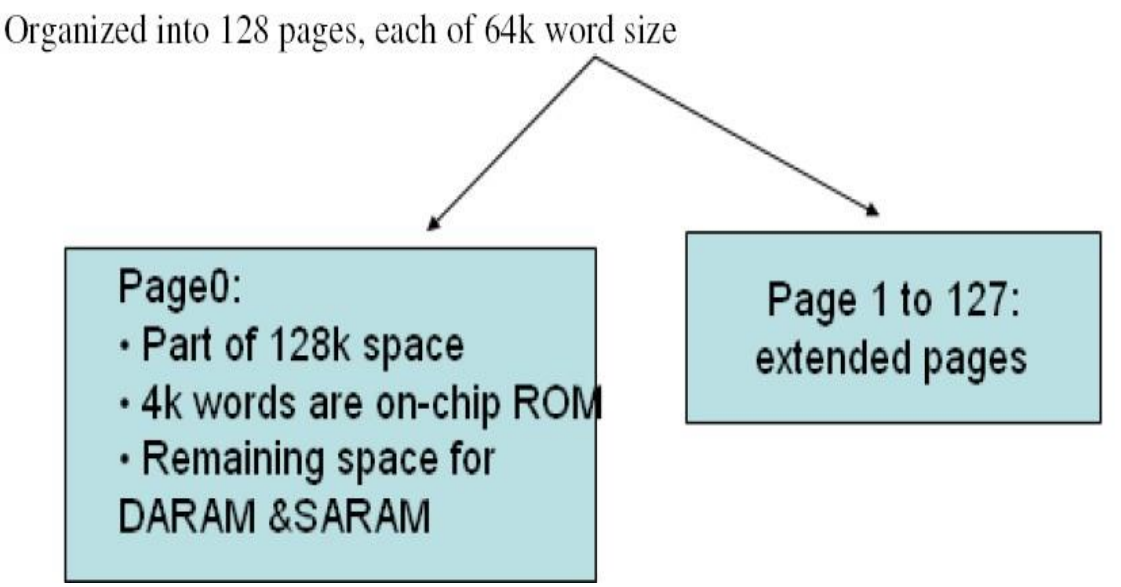
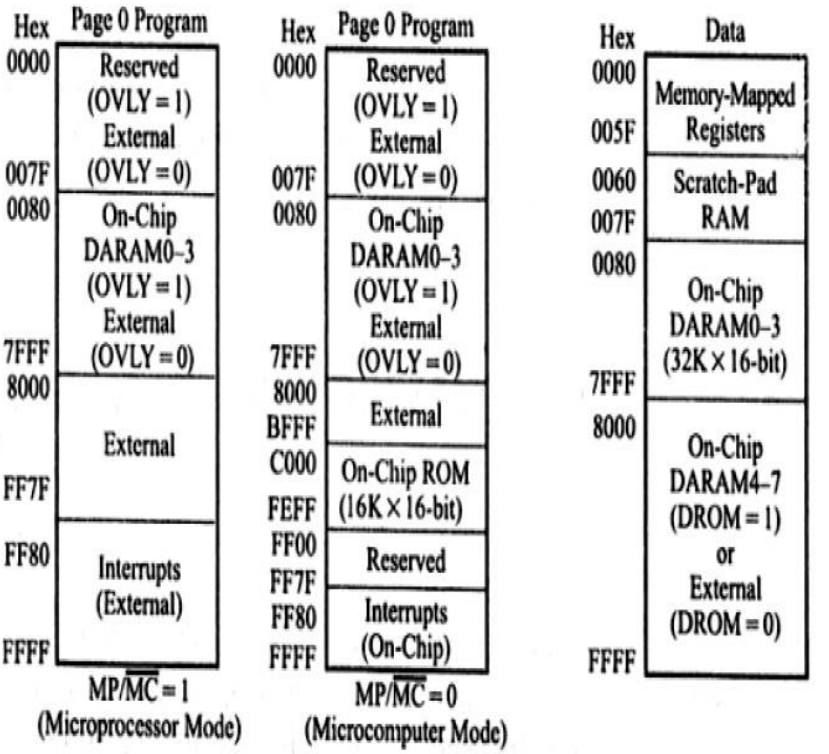


Table 3.4.Function of different pin PMST register

PMST bit	Logic	On-chip memory configuration
MP/MC	0	ROM enabled
	1	ROM not available
OVLY	0	RAM in data space
	1	RAM in program space
DROM	0	ROM not in data space
	1	ROM in data space



Address ranges for on-chip DARAM in data memory are:

DARAM0: 0080h–1FFFh; DARAM1: 2000h–3FFFh
DARAM2: 4000h–5FFFh; DARAM3: 6000h–7FFFh
DARAM4: 8000h–9FFFh; DARAM5: A000h–BFFFh
DARAM6: C000h–DFFFh; DARAM7: E000h–FFFFh

Figure 3.14 Memory map for the TMS320C5416 Processor.

3.6. 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 programaddress 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 aftereach of the following TMS320C54xx addressing modes is used? Assume that the contents of AR0 are 20h.

a. *AR3+0
b. *AR3-0
c. *AR3+
d. *AR3
e. *AR3
f. *+AR3 (40h)
g. *+AR3 (-40h)

Solution:

a. AR3 ← AR3 + AR0; AR3 = 200h + 20h = 220h
b. AR3← AR3 - AR0; AR3 = 200h - 20h = 1E0h
c. AR3 ← AR3 + 1; AR3 = 200h + 1 = 201h
d. AR3 ← AR3 - 1; AR3 = 200h - 1 = 1FFh
e. AR3 is not modified. AR3 = 200h
f. AR3 ← AR3 + 40h; AR3 = 200 + 40h = 240h
g. AR3 ← AR3 - 40h; AR3 = 200 - 40h = 1C0h

2. 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 are20h

a. *AR3 + 0B

b. *AR3 – 0B

Solution:

a. AR3 ← AR3 + AR0 with reverse carry propagation; AR3 = 200h + 20h (with reverse carry propagation) = 220h.

b. AR3 ← AR3 - AR0 with reverse carry propagation; AR3 = 200h - 20h (with reverse carry propagation) = 23Fh.

3.7 On chip peripherals:

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

3.7.1 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

3.7.2 Software programmable wait state generator:

- Extends external bus cycles up to seven machine cycles.

3.7.3 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.

Bit	Name	Function
15-12	Reserved	Reserved; always read as 0.
11	Soft	Used in conjunction with the free bit to determine the state of the timer Soft=0,the timer stops immediately. Soft=1,the timer stops when the counter decrements to 0.
10	Free	Use in conjunction with the soft bit Free=0,the soft bit selects the timer mode free=1,the timer runs free
Bit	Name	Function
9-6	PSC	Timer prescaler counter, specifies the count for the on-chip timer
5	TRB	Timer reload. Reset the on-chip timer.
4	TSS	Timer stop status, stop or starts the on-chip timer.
3-0	TDDR	Timer divide-down ration

Table 4.6. Pin details of software wait state generator

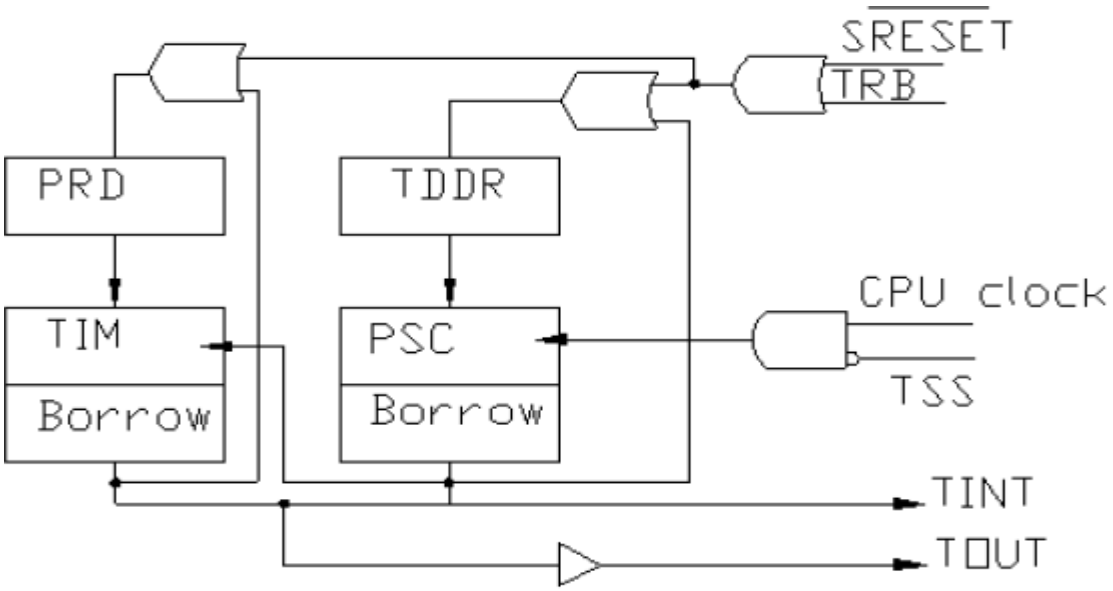


Figure 4.2.Logical block diagram of timer circuit.

3.8 Interrupts of TMS320C54xx Processors:

Many times, when CPU is in the midst of executing a program, a peripheral device may requirea 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 fromwhere 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 servicedfirst 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.

3.9 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 thetype 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 thewrite 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.

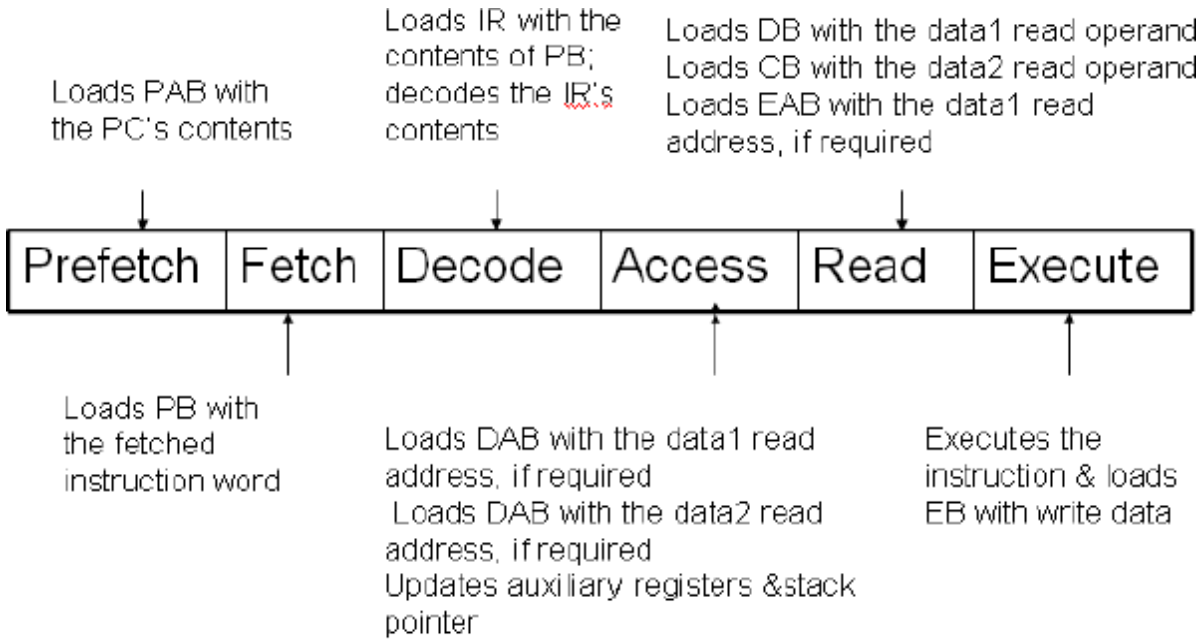


Figure 4.4. Pipeline operation of TMS320C54xx Processors

Pipe Flow

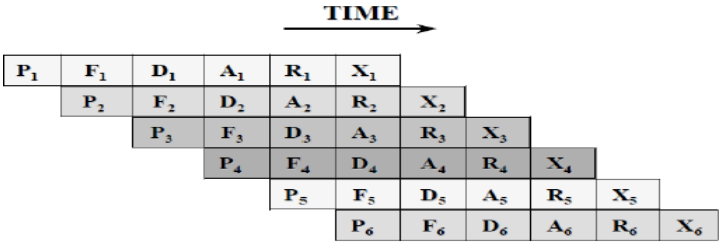
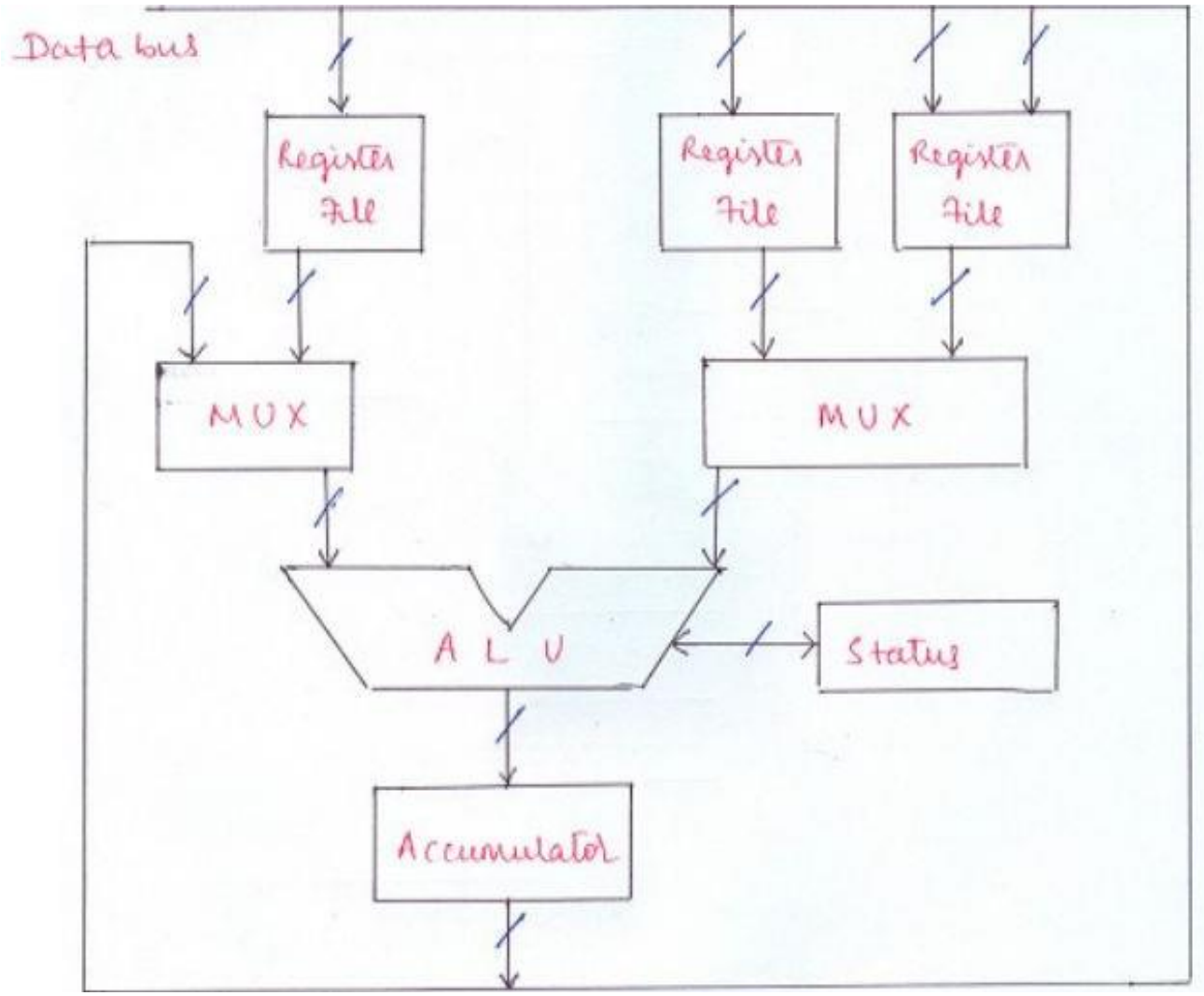


Figure 4.5.Pipe flow diagram

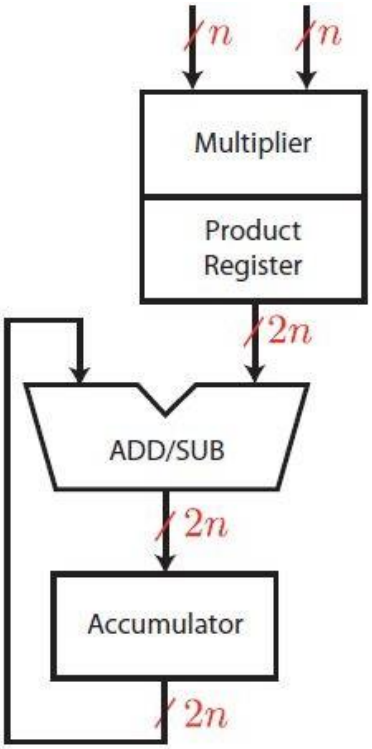
UNIT-IV

Analog Devices Family of DSP Devices:

ALU Block Diagram:



MAC (Multiplier and Accumulate):



- can implement $A + BC$ operations
- clearing the accumulator at the right time (e.g., as an initialization to zero) provides appropriate sum of products

Q: If a sum of 256 products is to be computed using a pipelined MAC unit and if the MAC execution time of the unit is 100 ns, what is the total time required to compute the operation?

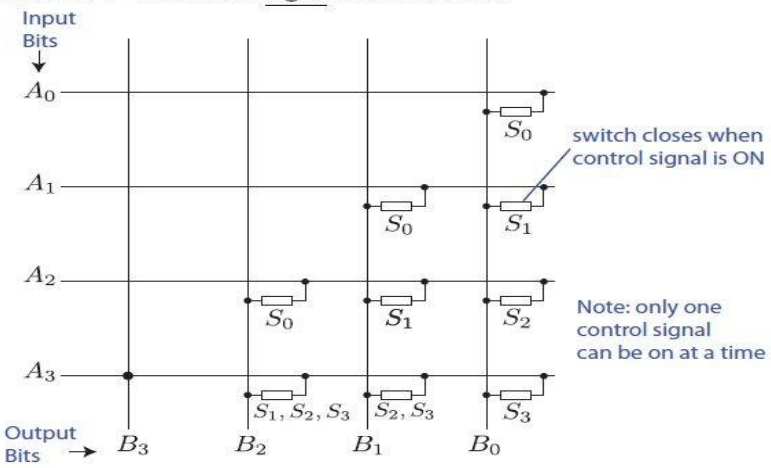
A:

For 256 MAC operations, need 257 execution cycles.

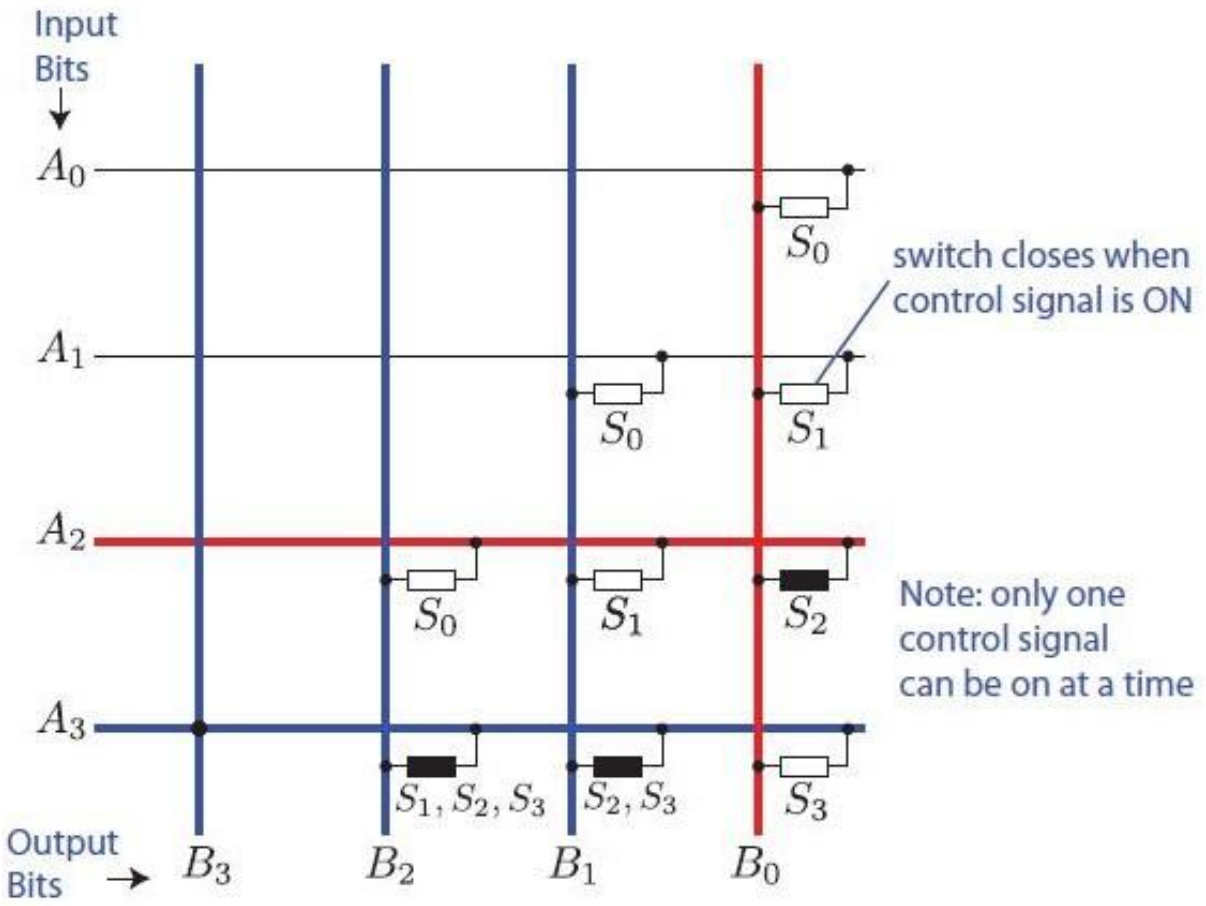
Total time required $= 257 \times 100 \times 10^{-9}$ sec $= \underline{25.7\mu s}$

Shifter Instructions:

Implementation of a 4-bit shift-right barrel shifter:



Input	Shift (Switch)	Output ($B_3B_2B_1B_0$)
$A_3A_2A_1A_0$	0 (S_0)	$A_3A_2A_1A_0$
$A_3A_2A_1A_0$	1 (S_1)	$A_3A_3A_2A_1$
$A_3A_2A_1A_0$	2 (S_2)	$A_3A_3A_3A_2$
$A_3A_2A_1A_0$	3 (S_3)	$A_3A_3A_3A_3$



Base Architecture of ADSP 2100:

The ASDP-2100 is a programmable single-Chip microprocessor optimized for digital signal processing (DSP) and other high-speed numeric processing applications. The ADSP-2100 chip contains an ALU, a multiplier/accumulator, a barrel shifter, two data address generators and a program sequencer; data and program memories are external. The ADSP- 2100 A is a pin-and code-compatible version of the original ADSP-2100 fabricated, in 1.0- µm CMOS. It can operate at a faster clock rate than the ADSP-2100

This section gives a broad overview of the ADSP-2100 internal architecture, using Figure 1.1 to show the architecture of the ADSP-2100processor.

The ADSP-2100 processor contains three full-function and independent

computational units: an arithmetic/logic unit, a multiplier/accumulatorand a barrel shifter. The computational units process 16-bit data directly and provide for multiprecision computation.

Two dedicated data address generators and a complete program sequencer supply addresses. The sequencer supports single-cycle conditional branching and executes program loops with zero overhead. Dual address generators allow the processor to output simultaneous addresses for dual operand fetches. Together the sequencer and data address generators allow computational operations to execute with maximum efficiency. The ADSP-2100 family uses a modified Harvard architecture in which data memory stores data, and program memory stores both instructions and data. Able to store data in both program and data memory, ADSP-2100 processors are capable of fetching two operandson the same instruction cycle.

The internal components are supported by five internal buses.

- Program Memory Address (PMA) bus
- Program Memory Data (PMD) bus
- Data Memory Address (DMA) bus
- Data Memory Data (DMD) bus
- Result (R) bus (which interconnects the computational units)

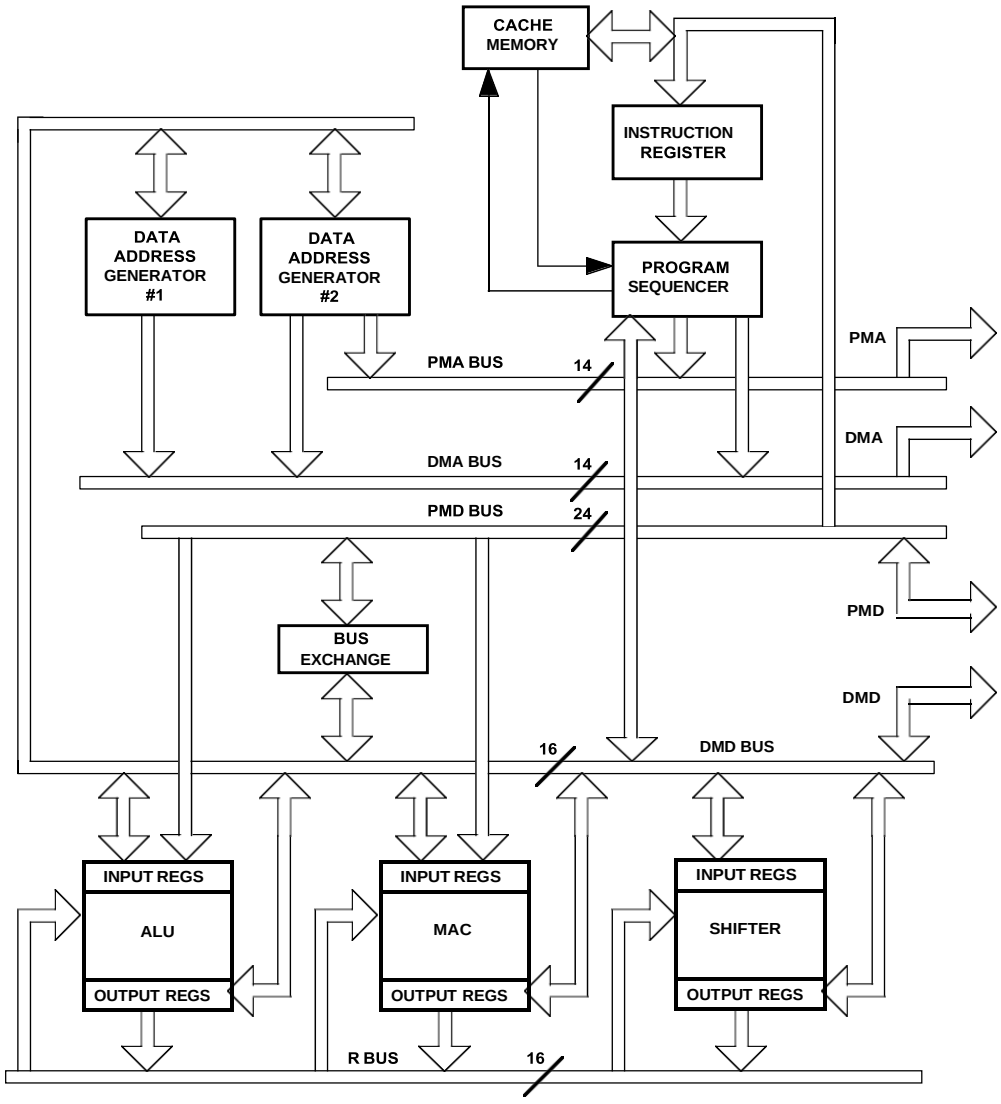


Figure 1.1 ADSP-2100 Internal Architecture

On the ADSP-2100, the four memory buses are extended off-chip for directconnection to external memories. The program memory data (PMD) bus serves primarily to transfer instructions from off-chip memory to the internal instruction register. Instructions are fetched and loaded into the instruction register duringone processor cycle; they execute during the following cycle while the next instruction is being fetched. The instruction register introduces a single level of pipelining in the program flow. Instructions loaded into theinstruction register are also written into the cache memory, to be described below

The next instruction address i s generated by the program sequencer depending on the current instruction and internal processor status. This address is placed on the program memory address (PMA) bus. The program sequencer uses features such as conditional branching, loop counters and zero-overhead looping to minimize program flow overhead.The program memory address (PMA) bus is 14 bits wide, allowing direct access to up to 16K words of instruction code and 16K words of data. Thestate of the PMDA pin distinguishes between code and data access of program memory. The program memory data (PMD) bus, like the processor's instruction words, is 24 bits wide.

.

The data memory address (DMA) bus is 14 bits wide allowing direct access of up to 16K words of data. The data memory data (DMD) bus is 16bits wide. The data memory data (DMD) bus provides a path for the contents of any register in the processor to be transferred to any other register, or to any external data memory location, *in a single cycle*. The datamemory address can come from two sources: an absolute value specified in the instruction code (direct addressing) or the output of a data address generator (indirect addressing). Only indirect addressing is supported for data fetches via the program memory bus.

The program memory data (PMD) bus can also be used to transfer data toand from the computational units through direct paths or via the PMD- DMD bus exchange unit. The PMD-DMD bus exchange unit permits datato be passed from one bus to the other. It contains hardware to overcome the 8-bit width discrepancy between the two buses when necessary.

Each computational unit contains a set of dedicated input and output registers. Computational operations generally take their operands from input registers and load the result into an output register. The registers actas a stopover point for data between the external memory and the computational circuitry, effectively introducing one pipeline level on input and one level on output. The computational units are arranged side by side rather than in cascade. To avoid excessive pipeline delays when a series of different operations are performed, the internal result (R) bus allows any of the output registers to be used directly (without delay) as the input to another computation.

For a wide variety of calculations, it is desirable to fetch two operands at the same time—one from data memory and one from program memory. Fetching data from program memory, however, makes it impossible to fetch the next instruction from program memory on the same cycle; an additional cycle would be required. To avoid this overhead, the ADSP- 2100 incorporates an instruction cache which holds sixteen words. The benefit of the cache architecture is most apparent when executing a program loop that can be totally contained in the cache memory. In this situation, the ADSP-2100 works like a three-bus system with an instruction fetch and two operand fetches taking place at the same time. Many algorithms are readily coded in loops of sixteen instructions or lessbecause of the parallelism and high-level syntax of the ADSP-2100 assembly language.

Here's how the cache functions: Every instruction loaded into the instruction register is also written into cache memory. As additional instructions are fetched, they overwrite the current contents of cache in acircular fashion. When the current instruction does a program memory data access, the cache automatically sources the instruction register if its contents are valid. Operation of the cache is completely transparent to user.

There are two independent data address generators (DAGs). As a pair, they allow the simultaneous fetch of data stored in program and in data memory for executing dual-operand instructions in a single cycle. One data address generator (DAG1) can supply addresses to the data memoryonly; the other (DAG2) can supply addresses to either the data memory orthe program memory. Each DAG can handle linear addressing as well as modulo addressing for circular buffers.

With its multiple bus structure, the ADSP-2100 supports a high degree of operational parallelism. In a single cycle, the ADSP-2100 can fetch an instruction, compute the next instruction address, perform one or two data transfers, update one or two data address pointers and perform a computation. Every instruction executes in a single cycle.

Figure 1.2, on the next page, is a simplified representation of the ADSP-2100 in a system context. The figure shows the two external memories used by the processor. Program memory stores instructions and is also used to store data. Data memory stores only data. The data memory address space may be shared with memory-mapped peripherals, if desired. Both memories may be accessed by external devices, such as a system host, if desired. Figure 1.2 also shows the processor control interface signals, (RESET, HALT and TRAP) the four interrupt request lines, the bus request and bus grant lines (BR and BG) and the clock input(CLKIN) and output (CLKOUT).

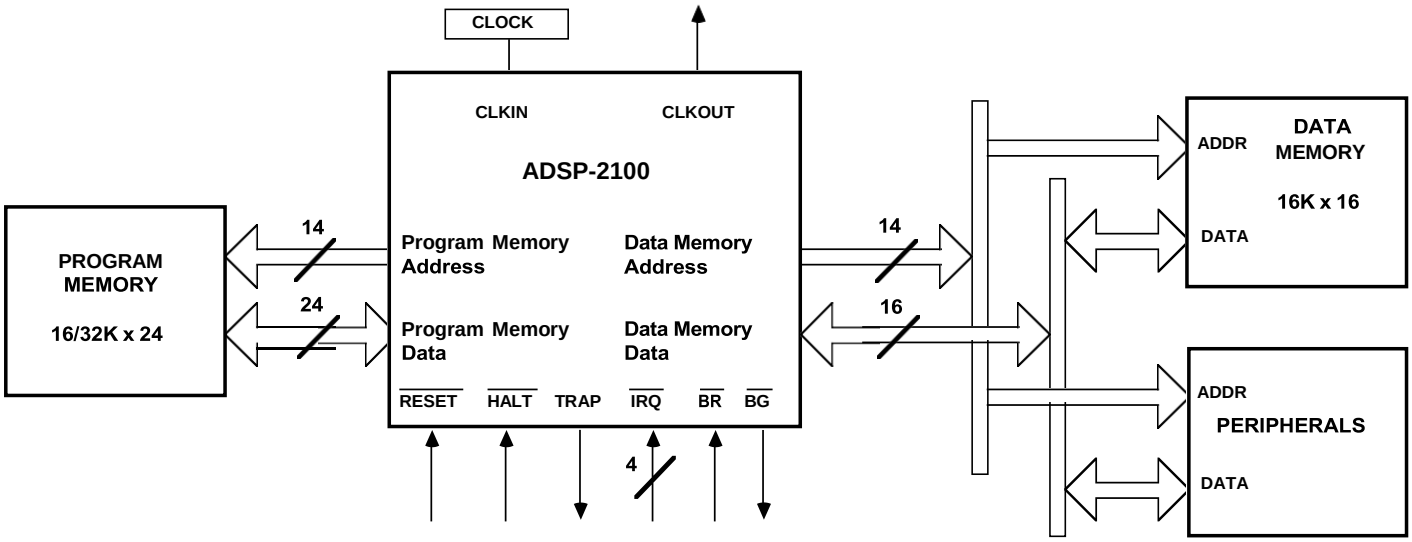


Figure 1.2 ADSP-2100 System

ADSP-2181 high performance Processor:

The ADSP-2181 is a single-chip microcomputer optimized fordigital signal processing (DSP) and other high speed numeric processing applications.

The ADSP-2181 combines the ADSP-2100 family base archi- tecture (three computational units, data address generators and a program sequencer) with two serial ports, a 16-bit internal DMA port, a byte DMA port, a programmable timer, Flag I/O,extensive interrupt capabilities, and on-chip program and data memory.

The ADSP-2181 integrates 80K bytes of on-chip memory con- figured as 16K words (24-bit) of program RAM, and 16K words(16-bit) of data RAM. Power-down circuitry is also provided to meet the low power needs of battery operated portable equip- ment. The ADSP-2181 is available in 128-lead TQFP and 128- lead PQFP packages.

In addition, the ADSP-2181 supports new instructions, which include bit manipulations—bit set, bit clear, bit toggle, bit test—new ALU constants, new multiplication instruction (x squared), biased rounding, result free ALU operations, I/O memory trans- fers and global interrupt masking for increased flexibility.

Fabricated in a high speed, double metal, low power, CMOS process, the ADSP-2181 operates with a 25 ns instruction cycletime. Every instruction can execute in a single processor cycle.

The ADSP-2181's flexible architecture and comprehensive instruction set allow the processor to perform multiple opera-tions in parallel. In one processor cycle the ADSP-2181 can:

- Generate the next program address
- Fetch the next instruction
- Perform one or two data moves

- Update one or two data address pointers
- Perform a computational operation

This takes place while the processor continues to:

- Receive and transmit data through the two serial ports
- Receive and/or transmit data through the internal DMA port
- Receive and/or transmit data through the byte DMA port
- Decrement timer

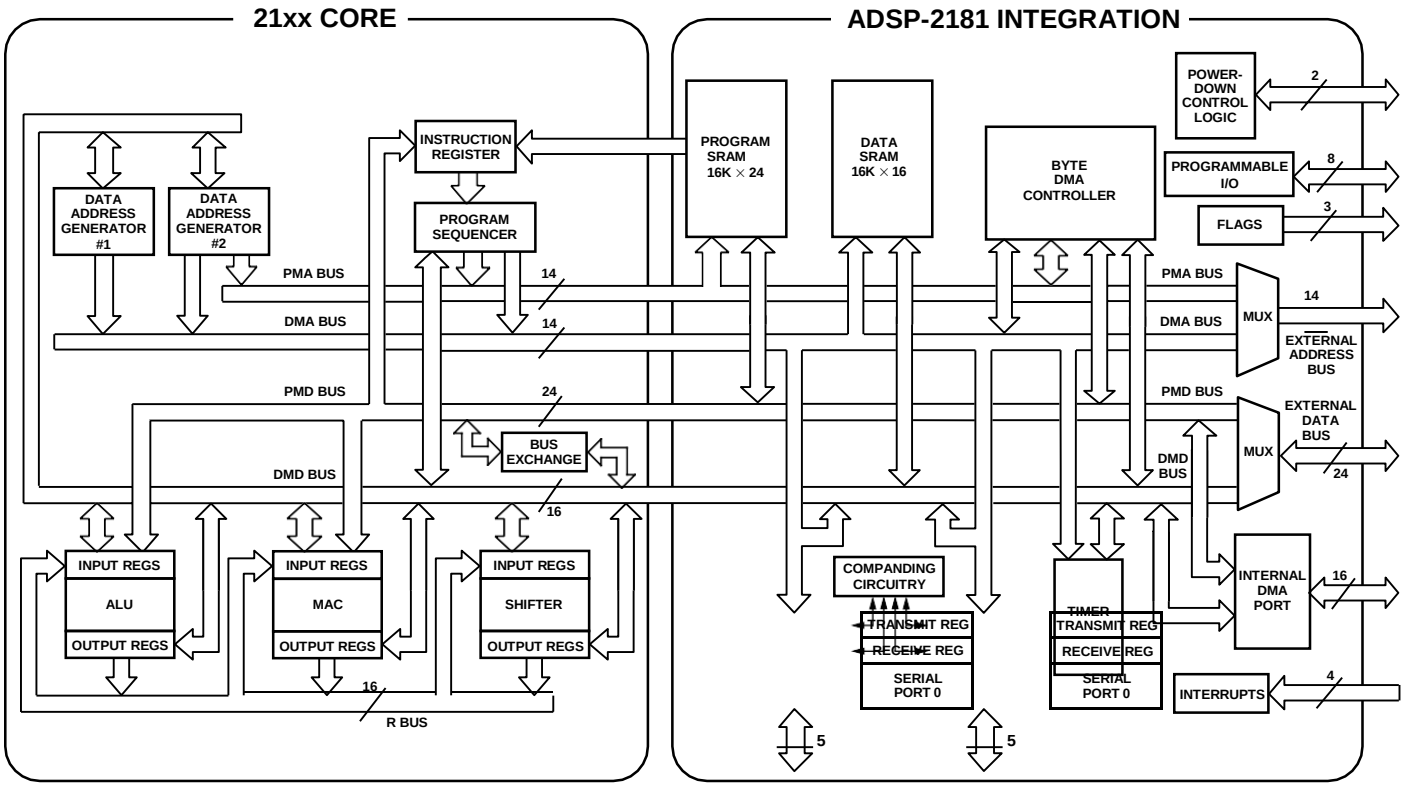


Figure 1. ADSP-2181 Block Diagram

The ADSP-2181 instruction set provides flexible data moves and multifunction (one or two data moves with a computation)instructions. Every instruction can be executed in a single processor cycle. The ADSP-2181 assembly language uses an algebraic syntax for ease of coding and readability. A comprehensive set of development tools supports program development.

Figure 1 is an overall block diagram of the ADSP-2181. The processor contains three independent computational units: the ALU, the multiplier/accumulator (MAC) and the shifter. The computational units process 16-bit data directly and have provi-sions to support multiprecision computations. The ALU per- forms a standard set of arithmetic and logic operations; division primitives are also supported. The MAC performs single-cycle multiply, multiply/add and multiply/subtract operations with 40 bits of accumulation. The shifter performs logical and arithmetic shifts, normalization, denormalization and derive exponent operations. The shifter can be used to efficiently implement numeric format control including multiword and block floating- point representations.

The internal result (R) bus connects the computational units sothat the output of any unit may be the input of any unit on the next cycle.

A powerful program sequencer and two dedicated data address generators ensure efficient delivery of operands to these computational units. The sequencer supports conditional jumps, subroutine calls and returns in a single cycle. With internal loop counters and loop stacks, the ADSP-2181 executes looped code with zero over- head; no explicit jump instructions are required to maintain loops.

Two data address generators (DAGs) provide addresses for simultaneous dual operand fetches (from data memory and program memory). Each DAG maintains and updates four address pointers. Whenever the pointer is used to access data (indirect addressing), it is post-modified by the value of one of four possible modify registers. A length value may be associatedwith each pointer to implement automatic modulo addressing for circular buffers.

Efficient data transfer is achieved with the use of five internalbuses:

- Program Memory Address (PMA) Bus
- Program Memory Data (PMD) Bus
- Data Memory Address (DMA) Bus
- Data Memory Data (DMD) Bus
- Result (R) Bus

The two address buses (PMA and DMA) share a single external address bus, allowing memory to be expanded off-chip, and the two data buses (PMD and DMD) share a single external data
bus. Byte memory space and I/O memory space also share the external buses.

Program memory can store both instructions and data, permit- ting the ADSP-2181 to fetch two operands in a single cycle, one from program memory and one from data memory. The ADSP-2181 can fetch an operand from program memory and the next instruction in the same cycle. In addition to the address and data bus for external memory connection, the ADSP-2181 has a 16-bit Internal DMA port (IDMA port) for connection to external systems. The IDMA port is made up of 16 data/address pins and five control pins. The IDMA port provides transparent, direct access to the DSPs on-chip program and data RAM. An interface to low cost byte-wide memory is provided by the Byte DMA port (BDMA port). The BDMA port is bidirectionaland can directly address up to four megabytes of external RAMor ROM for off-chip storage of program overlays or data tables. The byte memory and I/O memory space interface supports slow memories and I/O memory-mapped peripherals with program- mable wait state generation. External devices can gain control of external buses with bus request/grant signals (*BĂ*, *BŠĬ* and *BŠ*).One execution mode (Go Mode) allows the ADSP-2181 to con- tinue running from on-chip memory. Normal execution mode requires the processor to halt while buses are granted. The ADSP-2181 can respond to 13 possible interrupts, eleven of which are accessible at any given time. There can be up to sixexternal interrupts (one edge-sensitive, two level-sensitive and three configurable) and seven internal interrupts generated bythe timer, the serial ports (SPORTs), the Byte DMA port and the power-down circuitry. There is also a master *ĂESET* signal. The two serial ports provide a complete synchronous serial inter- face with optional companding in hardware and a wide variety of framed or frameless data transmit and receive modes of operation. Each port can generate an internal

programmable serial clock or accept an external serial clock.

The ADSP-2181 provides up to 13 general-purpose flag pins. The data input and output pins on SPORT 1 can be alternatively configured as an input flag and an output flag. In addition, there are eight flags that are programmable as inputs or outputs and three flags that are always outputs.

A programmable interval timer generates periodic interrupts. A 16-bit count register (T COUN T) is decremented every *n* pro- cessor cycles, where *n* is a scaling value stored in an 8-bit regis- ter (TSCALE). When the value of the count register reaches zero, an interrupt is generated and the count register is reloaded from a 16-bit period register (TPERIOD).

Serial Ports

The ADSP-2181 incorporates two complete synchronous serial ports (SPORT 0 and SPORT 1) for serial communications and multiprocessor communication.

Here is a brief list of the capabilities of the ADSP-2181 SPORTs. Refer to the *ADSP-2100 Family User's Manual, Third Edition* for further details.

- SPORTs are bidirectional and have a separate, double-buffered transmit and receive section.
- SPORTs can use an external serial clock or generate their own serial clock internally.
- SPORTs have independent framing for the receive and trans- mit sections. Sections run in a frameless mode or with frame synchronization signals internally or externally generated.
  Frame sync signals are active high or inverted, with either of two pulsewidths and timings.
- SPORTs support serial data word lengths from 3 to 16 bits and provide optional A-law and □-law companding accordingto CCITT recommendation G.711.
- SPORT receive and transmit sections can generate uniqueinterrupts on completing a data word transfer.
- SPORTs can receive and transmit an entire circular buffer ofdata with only one overhead cycle per data word. An interruptis generated after a data buffer transfer.
- SPORT 0 has a multichannel interface to selectively receiveand transmit a 24- or 32-word, time-division multiplexed, serial bitstream.
- SPORT 1 can be configured to have two external interrupts (*IĂQ0* and *IĂQ1*) and the Flag In and Flag Out signals. Theinternally generated serial clock may still be used in this configuration.

UNIT - V

Interfacing Memory & Parallel I/O Peripheralsto DSP Devices

5.1 Introduction: A typical DSP system has DSP with external memory, input devices and output devices. Since the manufacturers of memory and I/O devices are not same as that of manufacturers of DSP and also since there are variety of memory and I/O devices available, the signals generated by DSP may not suit memory and I/O devices to be connected to DSP. Thus, there is a need for interfacing devices the purpose of it being to us e DSP signals to generate the appropriate signals for setting up communication with the memory. DSP with interface is shown in fig. 5.1.

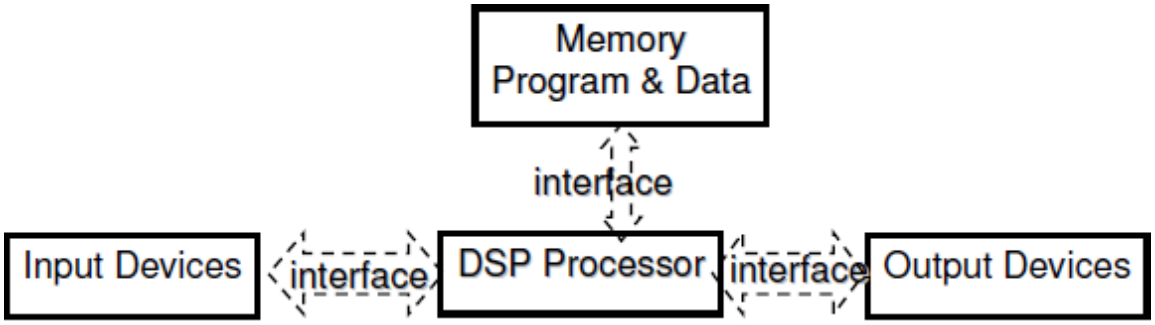


Figure 1DSP SYSTEM WITH INTERFACING

5.2 Memory Space Organization:

Memory Space in TMS320C54xx has 192K words of 16 bits each.Memory is divided into Program Memory, Data Memory and I/O Space, each are of 64K words. Theactual memory and type of memory depends on particular DSP device of the family. If the memory available on a DSP is not sufficient for an application, it can be interfaced to an external memory asdepicted in fig. 7.2. The On- Chip Memory are faster than External Memory. There are no interfacingrequirements. Because they are on-chip, power consumption is less and size is small. It exhibits better performance by DSP because of better data flow within pipeline. The purpose of such memory is tohold Program / Code / Instructions, to hold constant data such as filter coefficients / filter order, also tohold trigonometric tables / kernels of transforms employed in an algorithm. Not only constants arestored in such memory, they are also used to hold variable data and intermediate results so that the processor need not refer to external memory for the purpose.

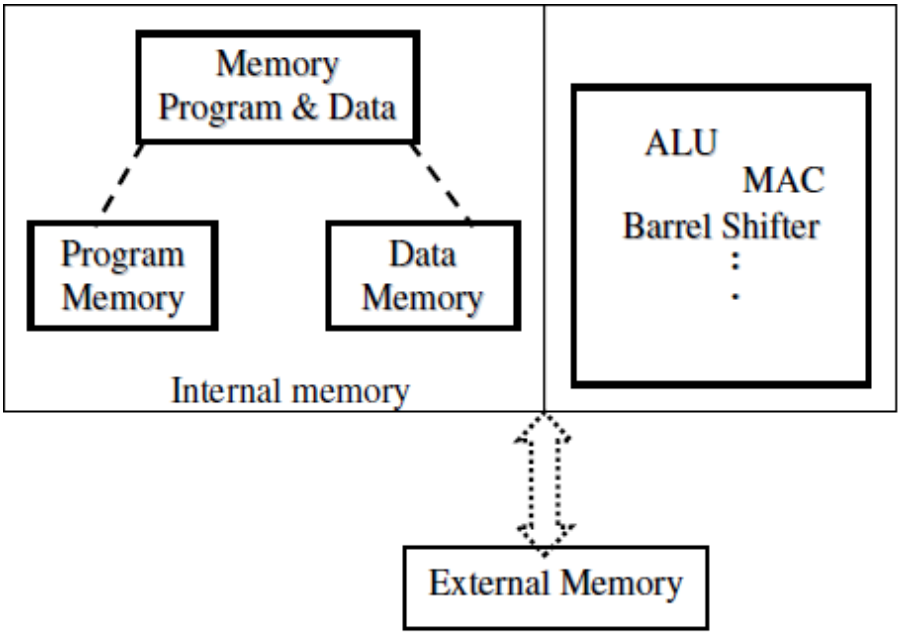


Fig. 7.2: Internal memory and interfacing of external memory

External memory is off-chip. They are slower memory. External Interfacing is required to establish the communication between the memory and the DSP. They can be with large memoryspace. The purpose is being to store variable data and as scratch pad memory. Program memory can be ROM, Dual Access RAM (DARAM), Single Access RAM (SARAM), or a combination of all these. The program memory can be extended externally to 8192K words. That is, 128 pages of 64K words each. The arrangement of memory and DSP in the case of Single Access RAM (SARAM) and Dual Access RAM (DARAM) is shown in fig. 7.3. One set of address bus and data bus is available in the case of SARAM and two sets of address bus and data bus is available in the case of DARAM. TheDSP can thus access two memory locations simultaneously.

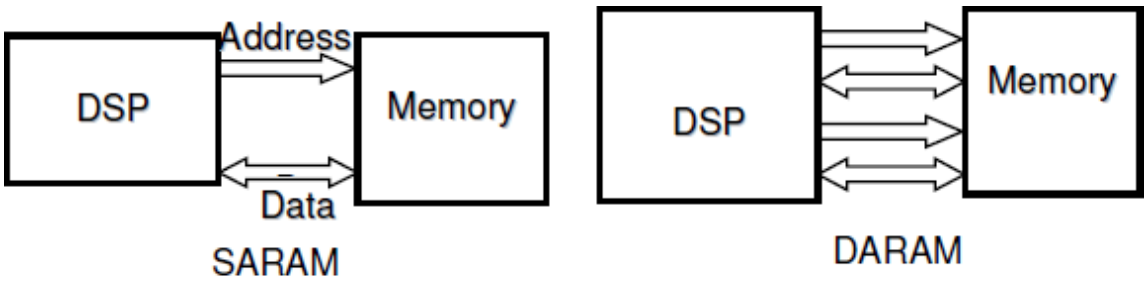


Fig. 7.3: SARAM & DARAM

There are 3 bits available in memory mapped register, PMST for the purpose of on-chip memory mapping. They are microprocessor / microcomputer mode. If this bit is 0, the on-chip ROM isenabled and addressable and if this bit is 1 the on-chip ROM not available. The bit can be manipulated by software / set to the value on this pin at system reset. Second bit is OVLY. It implies RAM Overlay. It enables on-chip DARAM data memory blocks to be mapped into program space. If this bit is 0, on-chip RAM is addressable in data space but not in Program Space and if it is 1, on-chip RAM is mapped into Program & Data Space. The third bit is DROM. It enables on-chip DARAM 4-7 to be mapped into data space. If this bit is 0, on-chipDARAM 4-7 is not mapped into data space and if this bit is 1, on-

chip DARAM 4-7 is mapped into Data Space. On-chip data memory is partitioned into several regions as shown in table 7.1. Data memory can be onchip / off-chip.

Table 7.1: Data memory 64 K

0000-005F 96 locations	Memory Mapped Registers
0060-007F 32 locations	Scratch pad RAM
0080-7FFF	On-chip DARAM 0-3 32Kx16bit
8000-FFFF 32K locations	On-chip DARAM 4-7 for Data

The on-chip memory of TMS320C54xx can be both program & data memory. It enhances speed of program execution by using parallelism. That is, multiple data access capability is provided for concurrent memory operations. The number of operations in single memory access is 3 reads & one write. The external memory to DSP can be interfaced with 16 -23 bit Address Bus, 16 bit Data Bus. Interfacing Signals are generated by the DSP to refer to external memory. The signals required by the memory are typically chip Select, Output Enable and Write Enable. For example, TMS320C5416 has 16K ROM, 64K DARAM and 64K SARAM.

Extended external Program Memory is interfaced with 23 address lines i.e., 8192K locations. The external memory thus interfaced is divided into 128 pages, with 64K words per page.

5.3 External Bus Interfacing Signals:

In DSP there are 16 external bus interfacing signals. The signal is characterized as single bit i.e., single line or multiple bits i.e., Multiple lines / bus. It can be synchronous / asynchronous with clock. The signal can be active low / active high. It can be output / input Signal. The signal carrying line / lines Can be unidirectional / bidirectional Signal. The characteristics of the signal depend on the purpose it serves. The signals available in TMS320C54xx are listed in table 7.2 (a) & table 7.2 (b).

In external bus interfacing signals, address bus and data bus are multi-lines bus. Address bus is unidirectional and carries address of the location refereed. Data bus is bidirectional and carries data to or from DSP. When data lines are not in use, they are tri-stated. Data Space Select, Program Space Select, I/O Space Select are meant for data space, program space or I/O space selection. These interfacing signals are all active low. They are active during the entire operation of data memory / program memory / I/O space reference. Read/Write Signal determines if the DSP is reading the external device or writing.

Table 7.2 (a) External Bus Interfacing Signals		
1	A0-A19	20 bit Address Bus
2	D0-D15	16 bit Data Bus
3	$\overline{DS}$	Data Space Select
4	$\overline{PS}$	Program Space Select
5	$\overline{IS}$	I/O Space Select
6	$R/\overline{W}$	Read/Write Signal
7	$\overline{MSTRB}$	Memory Strobe
8	$\overline{IOTRB}$	I/O Strobe

Read/Write Signal is low when DSP is writing and high when DSP is reading. Strobe Interfacing Signals, Memory Strobe and I/O Strobe both are active low. They remain low during the entire read & write operations of memory and I/O operations respectively. External Bus Interfacing Signals from 1-8 are all are unidirectional except Data Bus which is bidirectional. Address Lines are outgoing signals and all other control signals are also outgoing signals.

Table 7.2 (b) External Bus Interfacing Signals		
9	READY	Data Ready Signal
10	$\overline{HOLD}$	Hold Request
11	$\overline{HLDA}$	Hold Acknowledge
12	$\overline{MSC}$	Micro State Complete
13	$\overline{IRQ}$	Interrupt Request
14	$\overline{IACK}$	Interrupt Acknowledge
15	XF	External Flag Output
16	$\overline{BIO}$	Branch Control Input

Data Ready signal is used when a slow device is to be interfaced. Hold Request and Hold Acknowledge are used in conjunction with DMA controller. There are two Interrupt related signals: Interrupt Request and Interrupt Acknowledge. Both are active low. Interrupt Request typically for data exchange. For example, between ADC / another Processor. TMS320C5416 has 14 hardware interrupts for the purpose of User interrupt, Mc-BSP, DMA and timer. The External Flag is active high,asynchronous and outgoing control signal. It initiates an action or informs about the completion of a transaction to the peripheral device. Branch Control Input is a active low, asynchronous, incoming control signal. A low on this signal makes the DSP to respond or attend to the peripheral device. It informs about the completion of a transaction to the DSP.

5.4 The Memory Interface:

The memory is organized as several locations of certain number of bits. The number of locations decides the address bus width and memory capacity. The number of bits per locations decides the data bus width and hence word length. Each location has unique address. The demand of 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 thecases, more number of memory ICs are required.

Typical signals in a memory device are address bus to carry address of referred memory location. Databus 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

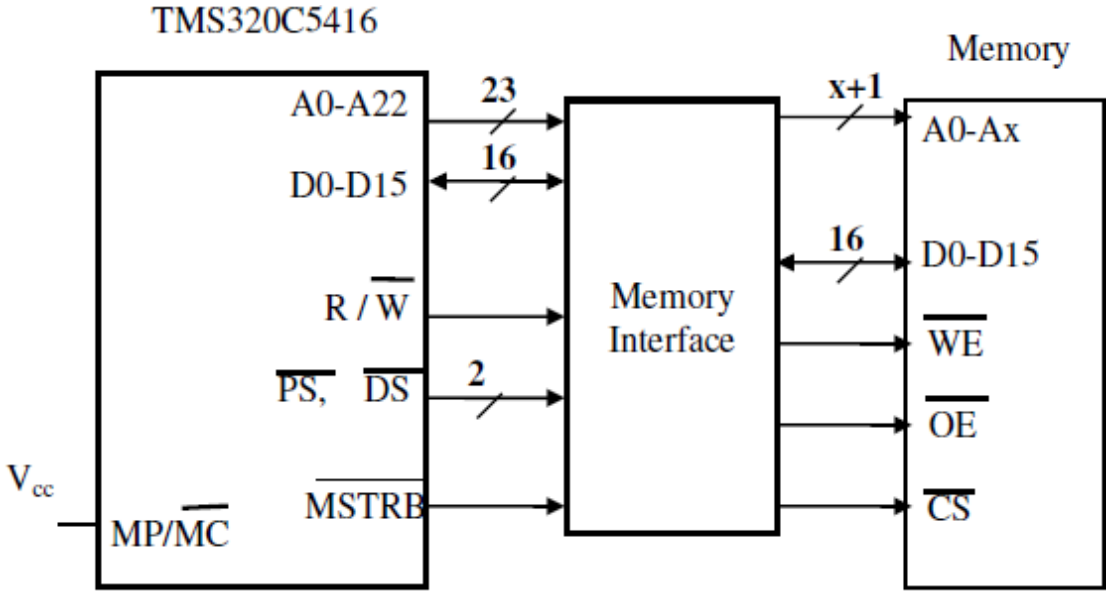


Fig. 7.4 Memory Interface for TMS320C5416

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.

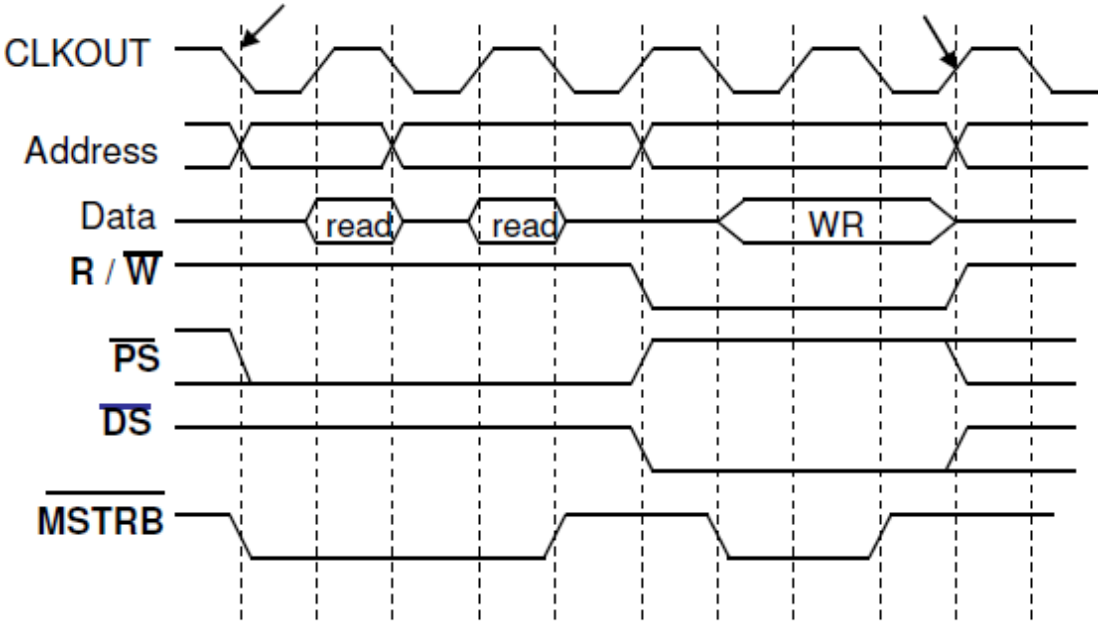


Fig. 7.5 Timing Sequence for External Memory Access

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

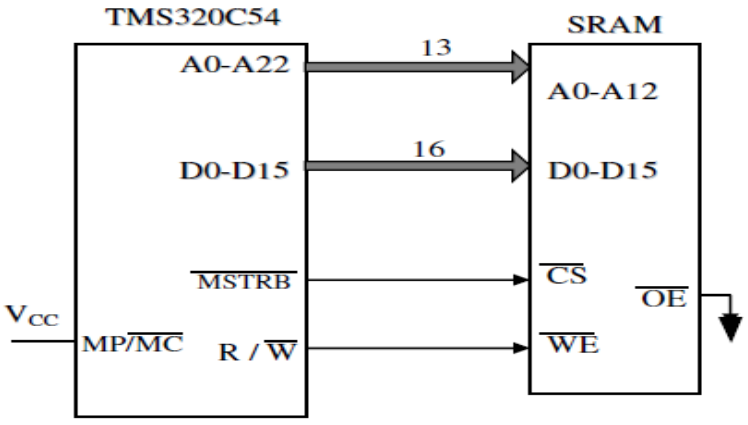


Fig. P7.4: Memory interface without decode circuit

• SRAM is thus indistinguishable as a program or data Memory

Problem P7.5: Design an interface to connect a 64K x 16 flash memory to a TMS320C54xx device. The Processor address bus to be used is A0-A15. The flash memory has the signals as shown in fig. P7.5.

Solution: Address lines from A0-A15 are used to address 64K locations. All the data lines, D0-D15 are used to carry data word. Data Space Select line is connected to chip enable of memory so that whenever DSP refers to data memory, this flash memory is enabled. When DSP refers to memory and it is a write operation, both memory strobe and read/write signals will be low. They are combined in using OR gate and used as write enable for memory. Memory read is performed by combining memory strobe and XF signals.

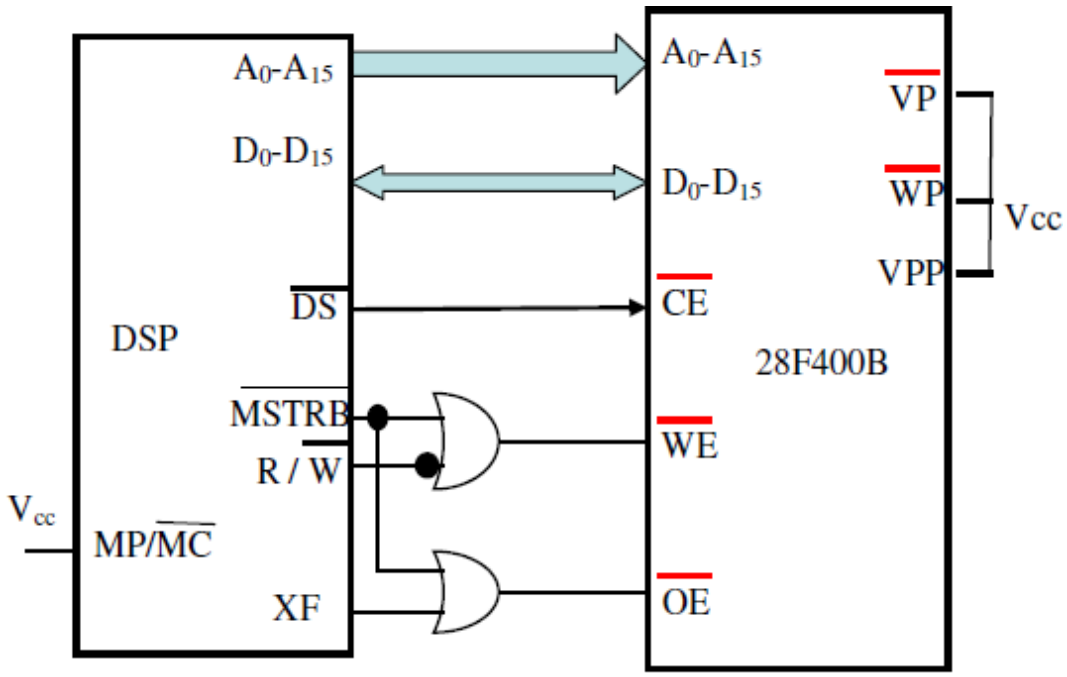


Fig. P7.5: Interfacing flash memory

5.5 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.

5.6 : 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/Oread and I/O write operations.

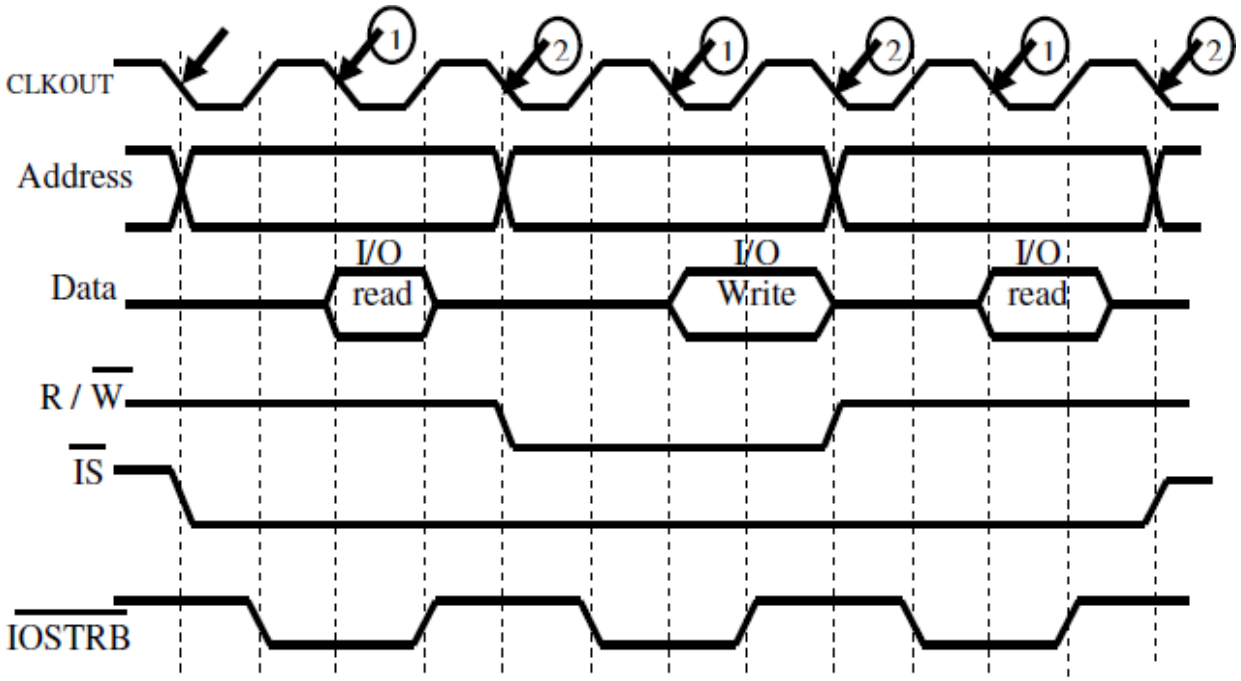


Fig. 7.6: Read-Write-Read Sequence of Operations

An example of interfacing ADC to DSP in programmed I/O mode is shown in fig. 7.7. ADC has a startof 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.

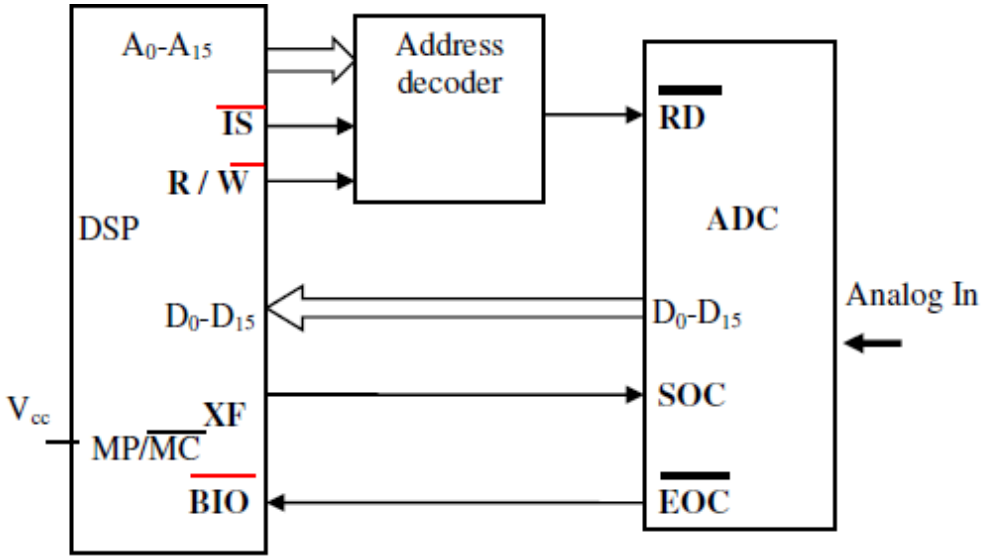


Fig. 7.7: ADC in Programmed I/O mode

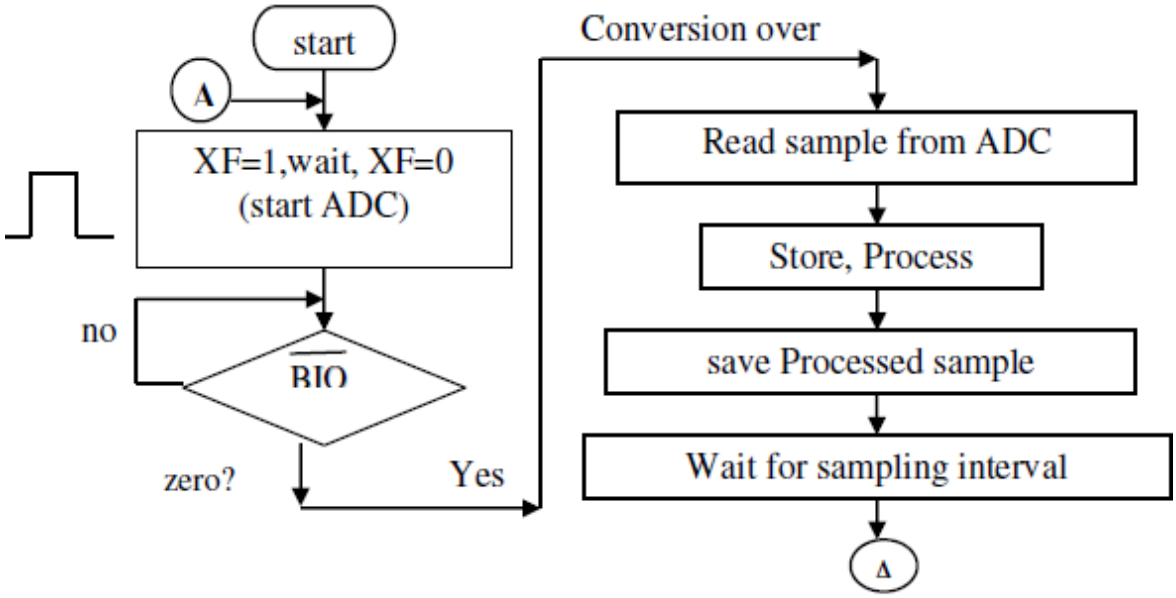


Fig. 7.8: Programmed I/O mode

5.7 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 externaldevice. Hardware interrupt is also referred to as external interrupt and software interrupt is referred toas 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 2DMA 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 Corresponding bit is set. IMR is used to mask or unmask an interrupt. One implies that the corresponding interrupt is unmasked. Both these registersare 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.

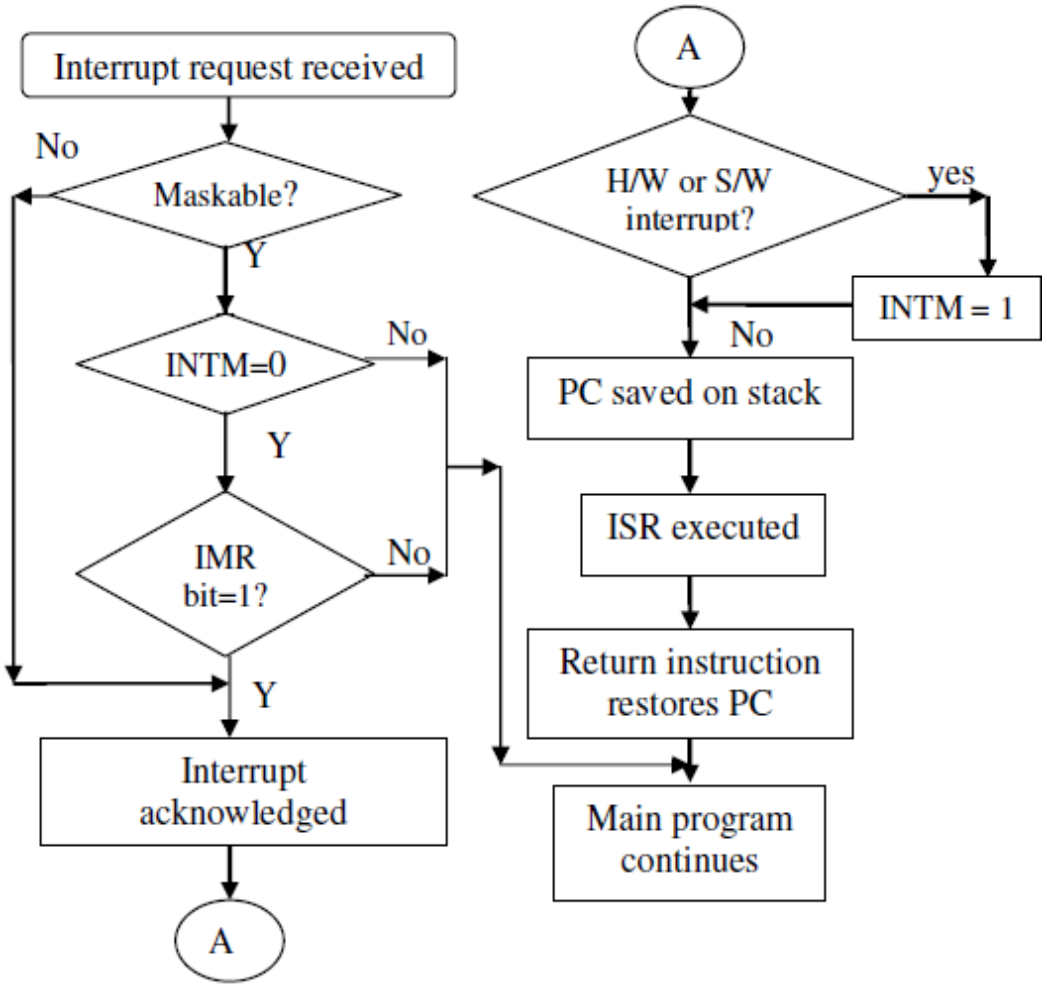


Fig. 7.9: Response of DSP to interrupt

5.8 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 beneed 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: toutilize 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.

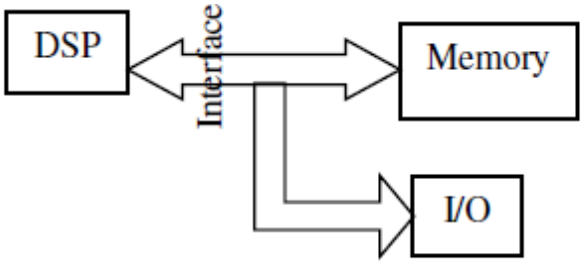


Fig. 7.10: Interface between DSP and external devices

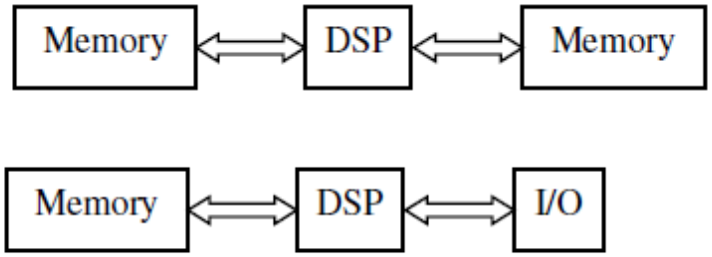


Fig. 7.11: Data transfer with intervention by DSP

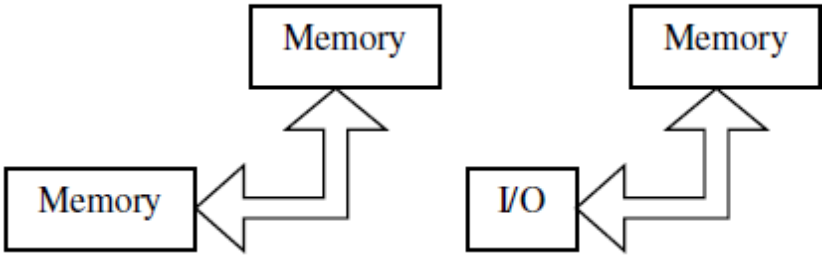


Fig. 7.12: data transfer without intervention by DSP

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 ofdata 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 foreach 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 DMAregister to be accessed. Mux routes either DMSDI or DMSDN to the sub bank. The memory locationto be written

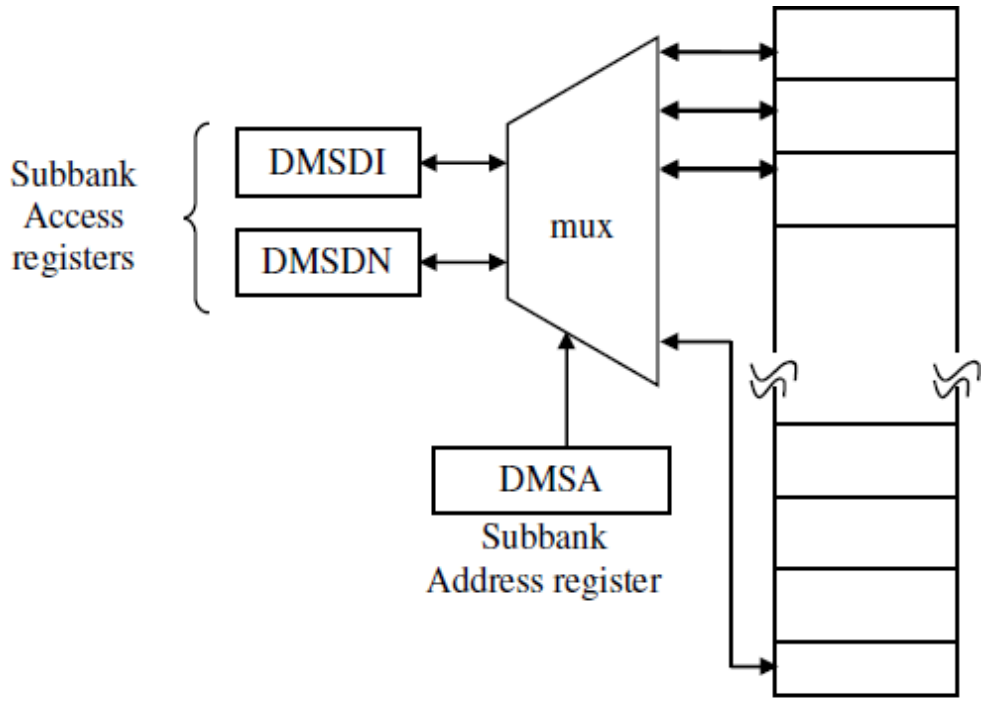


Fig. 7.13: Register Subaddress Technique

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.