Presentation on

DIGITAL SIGNAL PROCESSORS AND ARCHITECTURE

(ECE)

B.TECH VIII -Semester (AUTONOMOUS-R16)

Prepared by,

Ms. C. Devisupraja

Assistant Professor

COURSE OUTCOMES

CO 1	Understand the basics of Digital Signal Processing and transforms.
CO 2	Able to distinguish between the architectural features of General purpose processors and DSP processors.
CO 3	Understand the architectures of TMS320C54xx devices.
CO 4	Discuss about various memory and parallel I/O interfaces.
CO 5	Discuss about various memory and parallel I/O interfaces.

UNIT-1

Introduction: Digital signal-processing system, discrete Fourier Transform (DFT) and fast Fourier transform (FFT), differences between DSP and other micro processor architectures; Number formats: Fixed point, floating point and block floating point formats, IEEE-754 floating point, dynamic range and precision, relation between data word size and instruction word size; Sources of error in DSP implementations: A/D conversion errors, DSP computational errors, D/A conversion errors, Q-notation.

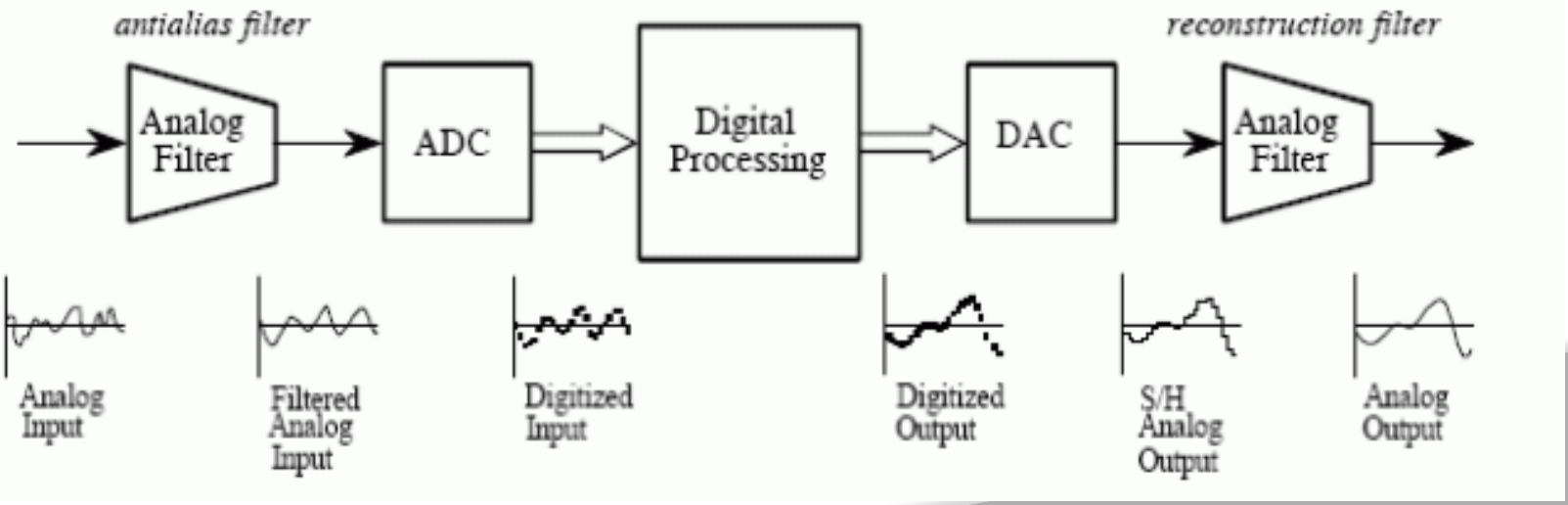
UNIT-1 CLO'S

CLO 1	Understand how digital to analog (D/A) and analog to digital (A/D) converters operate on a signal and be able to model these operations mathematically.
CLO 2	Understand the inter-relationship between DFT and various transforms.
CLO 3	Understand the IEE-754 floating point and source of errors in DSP implementations .
CLO 4	Understand the fast computation of DFT and appreciate the FFT Processing

DIGITAL SIGNAL PROCESSING SYSTEM

•A DSP system uses a computer or a digital processor to process signals.

- Represent signals by a sequence of numbers
  - Sampling or analog-to-digital conversions
- Perform processing on these numbers with a digital processor
  - Digital signal processing
- Reconstruct analog signal from processed numbers



DIGITAL SIGNAL PROCESSING SYSTEM

- Analog input – analog output

  -Digital recording of music

- Analog input – digital output

  - Touch tone phone dialing

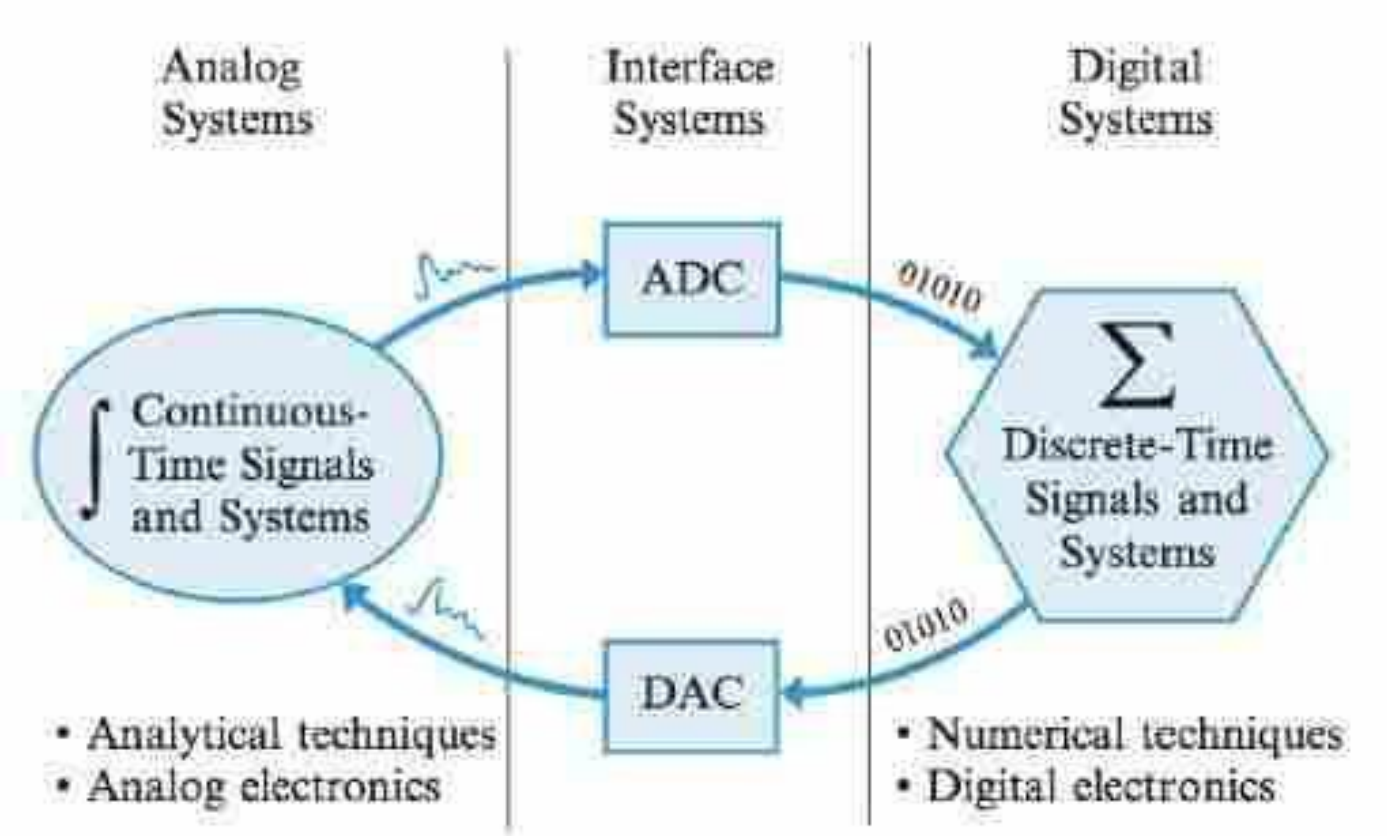
- Digital input – analog output

  -Text to speech

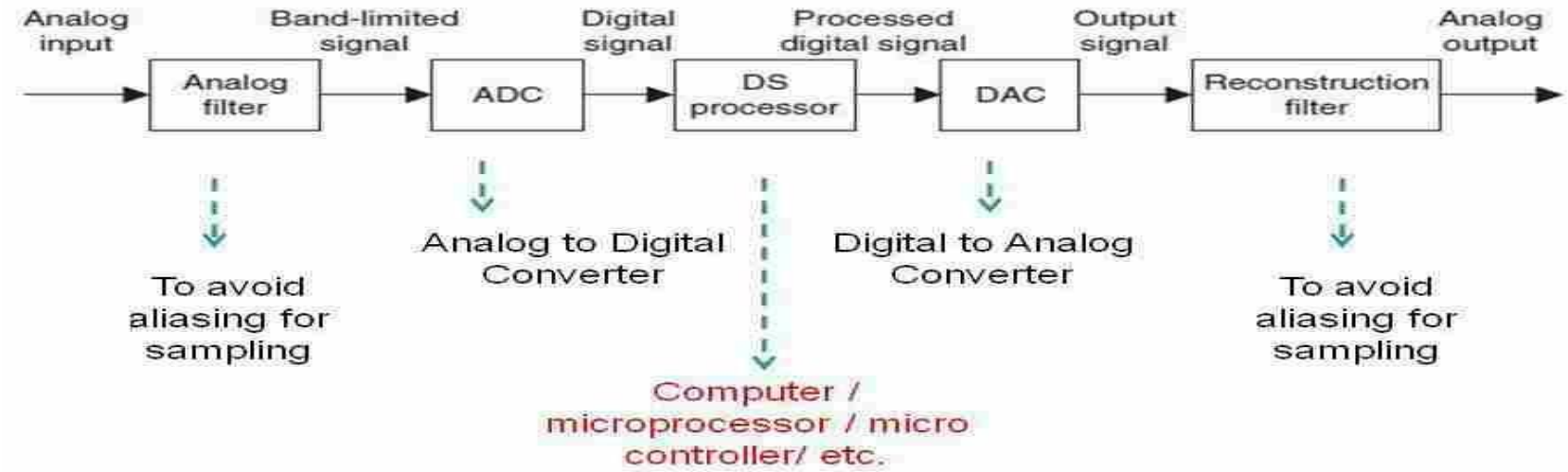
- Digital input – digital output

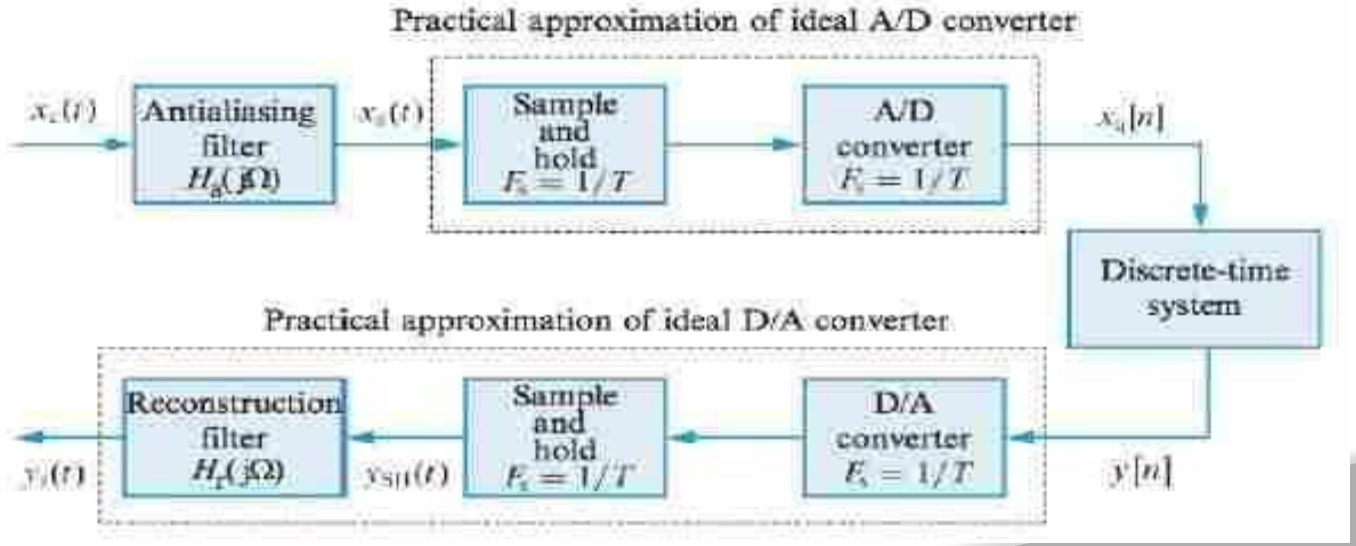
  -Compression of a file on computer

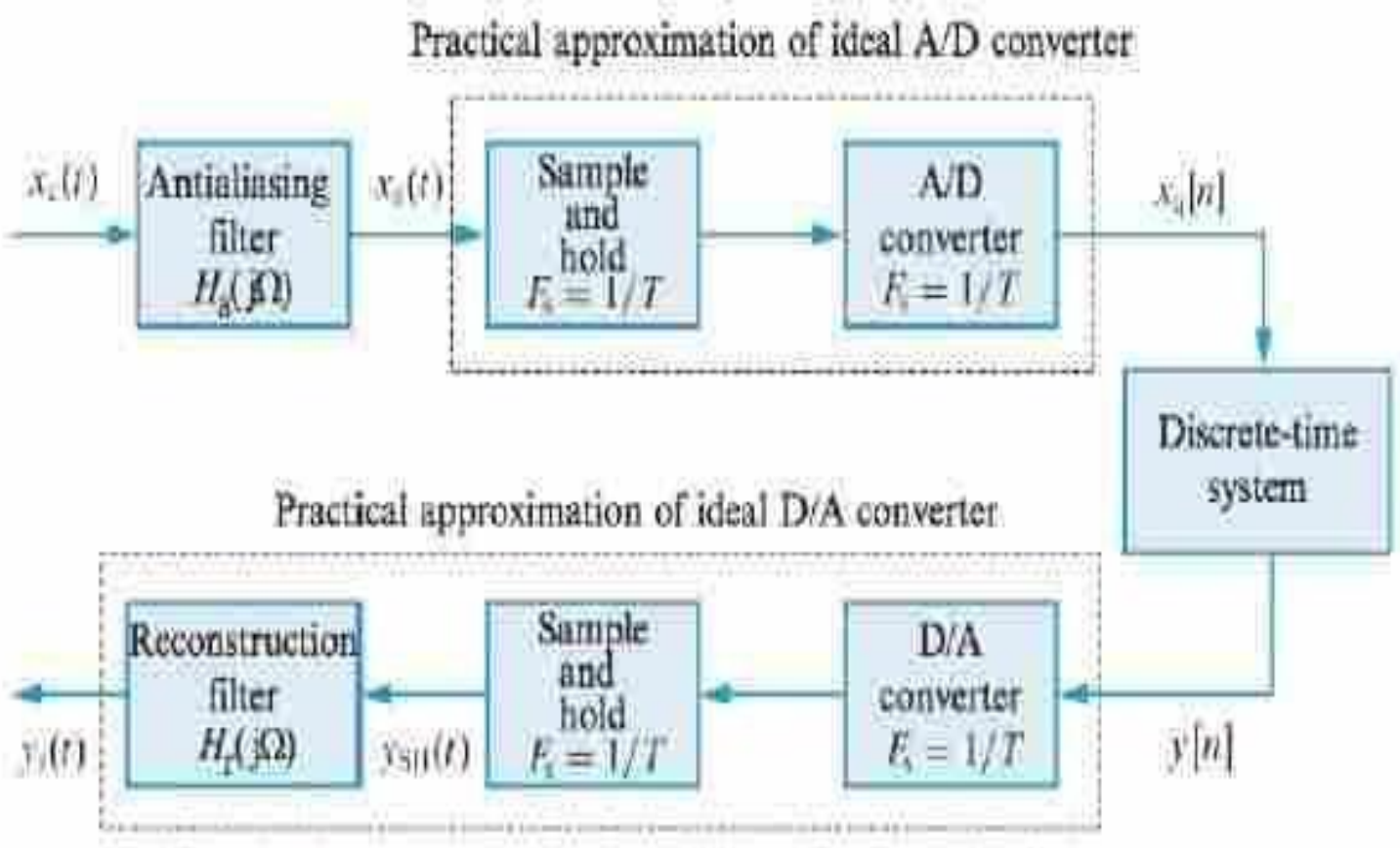
ANALOG, DIGITAL, MIXED SIGNAL PROCESSING



DIGITAL SIGNAL PROCESSING



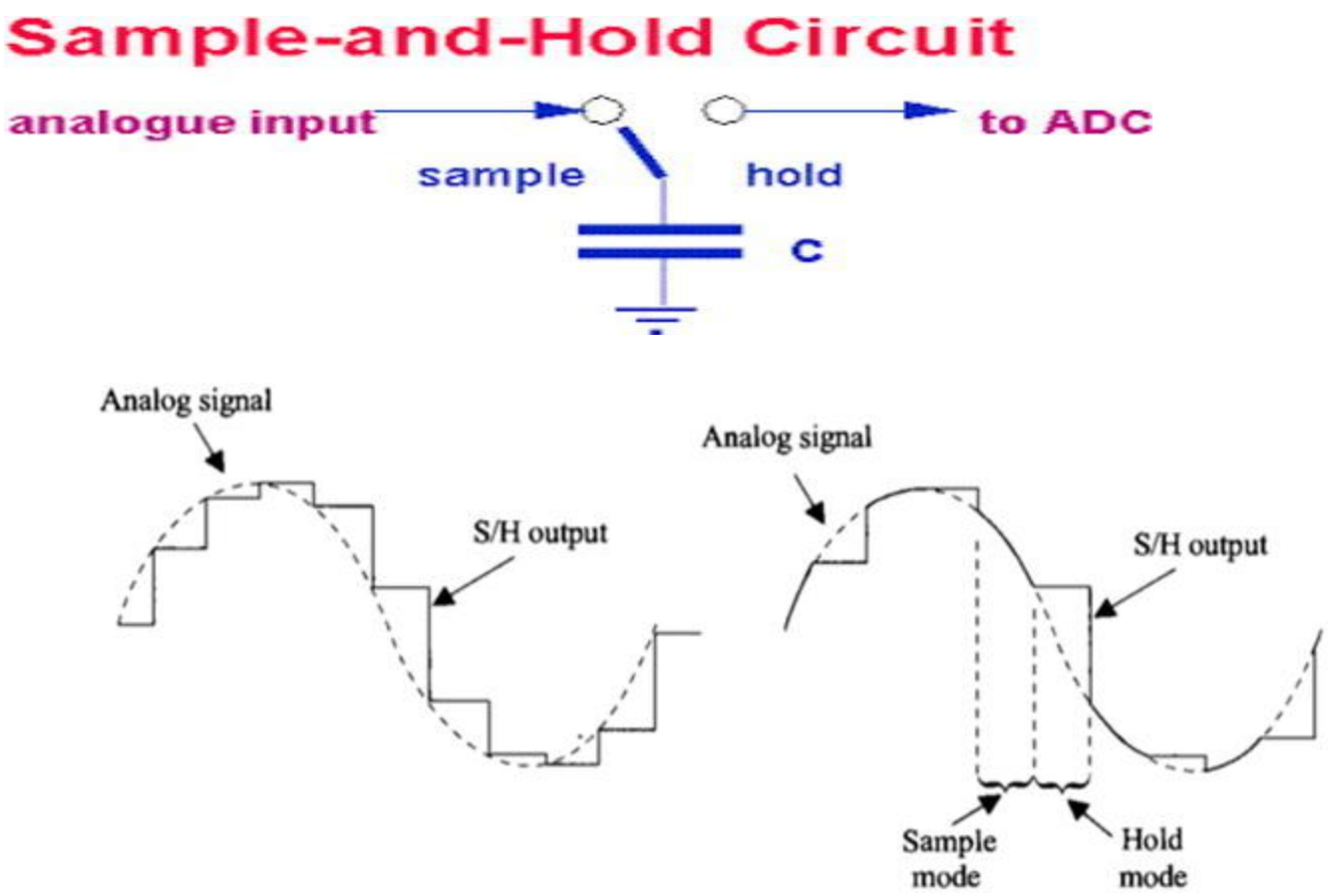


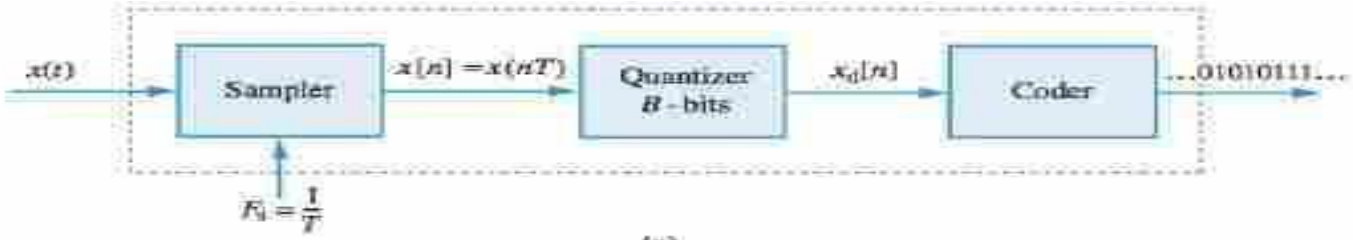

Practical approximation of ideal A/D converter
$x_c(t)$
Antialiasing filter $H_a(j\Omega)$
$x_a(t)$
Sample and hold $F_s = 1/T$
A/D converter $F_s = 1/T$
$x_q[n]$
Discrete-time system
Practical approximation of ideal D/A converter
Reconstruction filter $H_r(j\Omega)$
$y_r(t)$
$y_{sh}(t)$
Sample and hold $F_s = 1/T$
D/A converter $F_s = 1/T$
$y[n]$

SAMPLE AND HOLD CIRCUIT

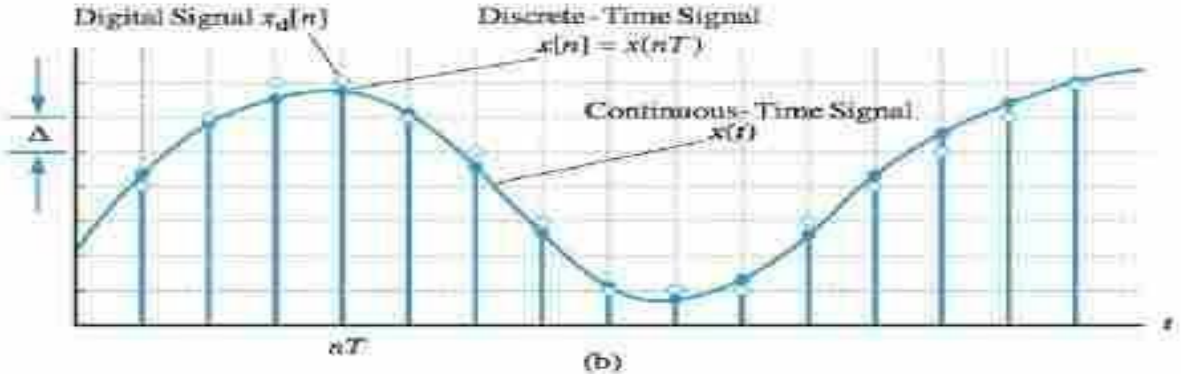
- The main function of low pass ant aliasing filter is to band limit the input signal to the folding frequency without distortion.
- It should be noted that even if the signal is band limited, there is always wide-band additive noise which will be folded back to create aliasing.
- When an analog voltage is connected directly to an ADC, the conversion process can be adversely affected if the voltage is changing during the conversion time.
- The quality of conversion process can be improved by using sample and hold
  - circuit

SAMPLE AND HOLD CIRCUIT





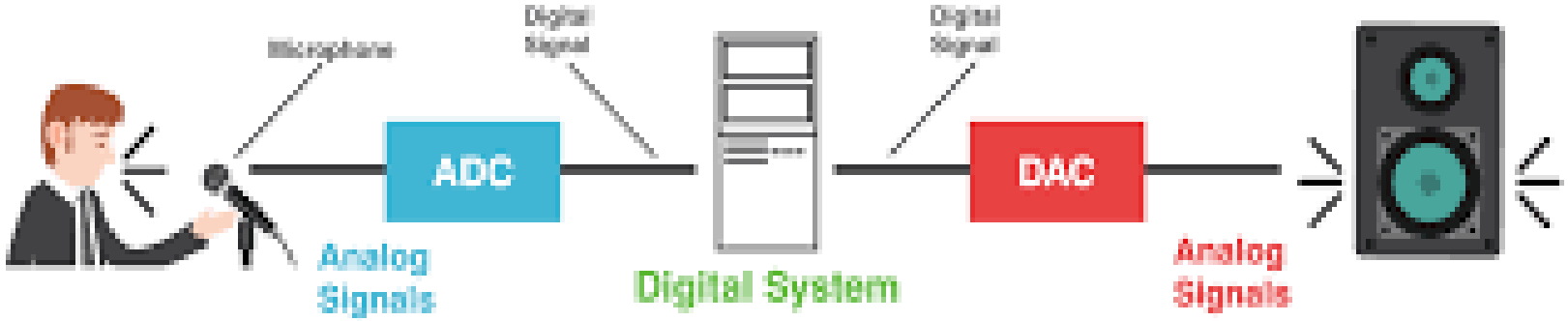
(a)

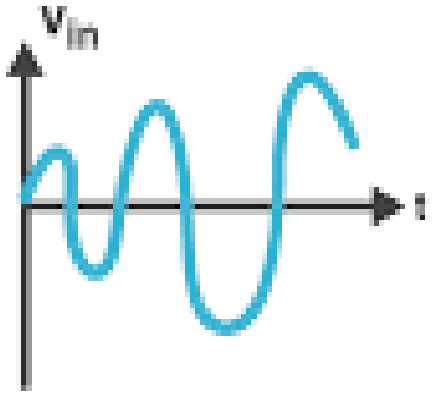


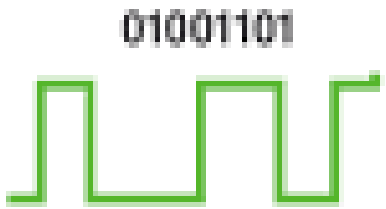
(b)

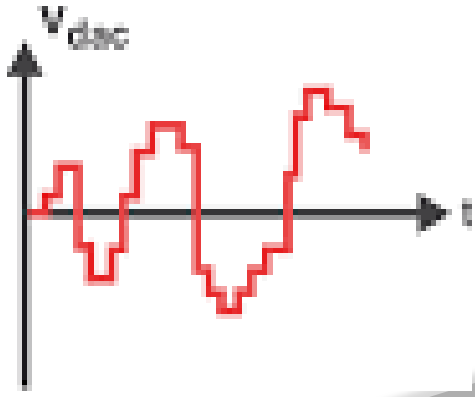
- *Quantization* converts a continuous-amplitude signal $x(t)$ into a discrete-amplitude signal $x_d[n]$.
- In theory, we are dealing with discrete-time signals; in practice, we are dealing with digital signals.

DIGITAL CONVERSION









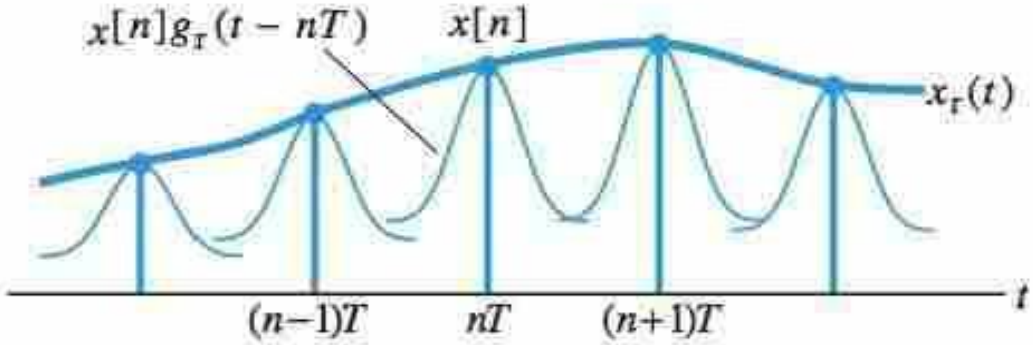
RECONSTRUCTION

- A general formula that describes a broad class of reconstruction processes is given by

$$x_r(t) = \sum_n x[n] g_r(t - nT)$$

where $g_r(t)$ is an interpolating reconstruction function.

- The process of fitting a continuous function to a set of samples is known as an *interpolation*.



- Thus, if the interpolation function has duration greater than or equal to T, the addition of the overlapping copies "fills the gaps" between samples.

RECONSTRUCTION

- In the Fourier domain, the interpolation formula becomes

$$X_r(j\Omega) = \sum_n x[n] G_r(j\Omega) e^{-j\Omega T} = G_r(j\Omega) \underbrace{\sum_n x[n] e^{-j\Omega T}}_{X(e^{j\Omega T})}$$

- Consequently, we obtain

$$X_r(j\Omega) = G_r(j\Omega)\, X(e^{j\Omega T})$$

- Specifically, if we choose $g_r(t)$ so that

$$G_r(j\Omega) \triangleq G_{BL}(j\Omega) = \begin{cases} T, & |\Omega| \leq \Omega_s/2 \\ 0, & |\Omega| > \Omega_s/2 \end{cases}$$

then $X_r(j\Omega) = X_c(j\Omega)$ and, therefore, $x_r(t) = x_c(t)$.

RECONSTRUCTION

- Evaluating the inverse Fourier transform of $G_{BL}(j\Omega)$, we obtain

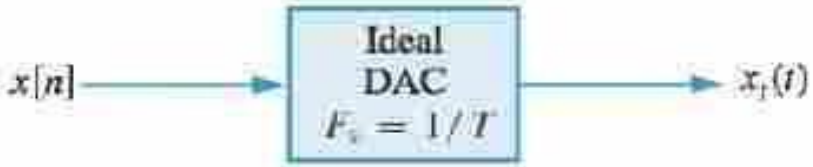
$$g_r(t) \triangleq g_{BL}(t) = \frac{\sin(\pi t/T)}{\pi t/T} = \text{sinc}(t/T)$$

- In this case we obtain:

The ideal interpolation formula

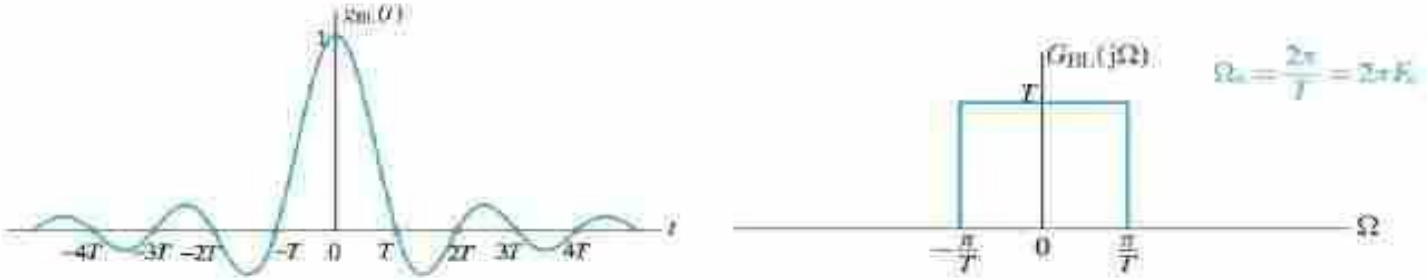
$$x_c(t) = \sum_n x[n] \frac{\sin[\pi(t - nT)/T]}{\pi(t - nT)/T}$$

- The system used to implement the ideal interpolation is known as an *ideal DAC*.

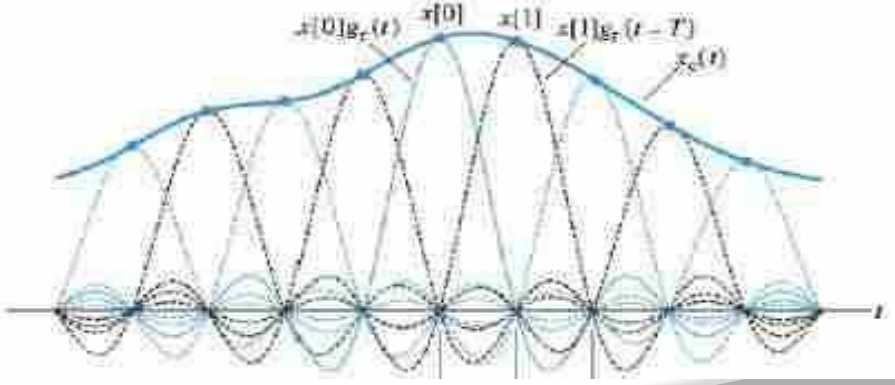


RECONSTRUCTION

- To understand the meaning and implications of the ideal interpolation we look more closely at the sinc function $g_{BL}(t)$.



- We note that $g_{BL}(t) = 0$ at all $t = nT$, except at $t = 0$ where $g_{BL}(t) = 1$. Thus, it is always true that $x_r(nT) = x_c(nT)$ regardless of whether aliasing occurred during sampling.



DISCRETE FOURIER TRANSFORM

• DFT is used for analyzing discrete-time finite duration signals in the frequency domain .

$$DFT(FFT):$$

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

$$IDFT(IFFT):$$

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

DISCRETE FOURIER TRANSFORM

• The Discrete Fourier Transform (DFT) is one of the most important tools in digital signal processing that calculates the spectrum of a finite-duration.

• The representation of a digital signal in terms of its frequency component in a frequency domain is important. The algorithm that transforms the time domain signals to the frequency domain components is known as DFT.

•No of additions to compute DFT is N(N-1)

•No of Multiplications to compute DFT is $N^{2.}$

FAST FOURIER TRANSFORM

- The Fast Fourier Transform (FFT) is an implementation of the DFT which produces almost the same results as the DFT, but it is incredibly more efficient and much faster which often reduces the computation time significantly.
- It is just a computational algorithm used for fast and efficient computation of the DFT. Various fast DFT computation techniques known collectively as the fast Fourier transform, or FFT.

DFT vs FFT

FFT	DFT
FFT is abbreviated as Fast Fourier Transform.	DFT stands for Discrete Fourier Transform.
FFT is a much faster version of the DFT algorithm.	DFT is the discrete version of the Fourier Transform.
Various fast DFT computation techniques are collectively known as the FFT algorithm.	It is the algorithm that transforms the time domain signals to the frequency domain components.
It's an implementation of the DFT.	It establishes a relationship between the time domain and the frequency domain representation
Applications include integer and polynomial multiplication, filtering algorithms, computing isotopic distributions, calculating Fourier series coefficients, etc.	Applications of DFT include solving partial differential applications, detection of targets from radar echoes, correlation analysis, computing polynomial multiplication, spectral analysis, etc.

NUMBER FORMATS

- In DSP, signals are represented as discrete sets of numbers from the input stage along through intermediate processing stages to the output.
- Even DSP structures such as filters require numbers to specify coefficients for operation.
- Two typical formats for these numbers:
  - fixed-point format
  - floating-point format

FIXED-POINT FORMAT

- Simplest scheme
- Number is represented as an integer or fraction using a fixed number of bits.
- An *n*-bit fixed-point signed integer $-2^{n-1} \leq x \leq 2^{n-1} - 1$is represented as:

$$x = -s \cdot 2^{n-1} + b_{n-2} \cdot 2^{n-2} + b_{n-3} \cdot 2^{n-3} + \cdots + b_1 \cdot 2^1 + b_0 \cdot 2$$

where *s* represents the sign of the number (*s* = 0 for positive and *s* = 1 for negative)

FIXED-POINT INTEGER FORMAT

- the simplest scheme of number representation is the format in which the number is represented as an integer or fraction using a fixed no of bits.
- An n-bit fixed point signed integer specifies the value x given as

$$x = -s \cdot 2^{n-1} + b_{n-2} \cdot 2^{n-2} + b_{n-3} \cdot 2^{n-3} + \cdots + b \cdot 2^1 + b \cdot 2^0$$

Eq-1

Where s represents the sign of the numbers: s=0 for positive numbers and s=-1 for negative numbers

- The range of signed integer values that can be represented with this format is -2^{n-1} to $+(2^{n-1}-1)$.

FIXED-POINT FRACTIONAL FORMAT

• Similarly, a fraction can also be represented using a fixed no of bits with an implied binary point after the most significant bit. An bit fixed point signed fraction representation.

$$x = -s \cdot 2^{0} + b_{-1} \cdot 2^{-1} + b_{-2} \cdot 2^{-2} + \cdots + b_{-(n-2)} \cdot 2^{-(n-2)} + b_{-(n-1)} \cdot 2^{-(n-1)}$$

Eq-2

• The range of signed integer values that can be represented with this format is --1 to $+(1-2^{-(n-1)})$.

EXAMPLE 1

What is the range of numbers that can be represented in a fixed-point format using 16 bits if the numbers are treated as

a) Signed Integers

b) Signed Fractions

Sol: a) using 16 bits the range of integers that can be represented is determined by substituting n=16 in eq-1 and is given as -2^{15} to $+ 2^{15} -1$ i.e -32,768 to + 32,767.

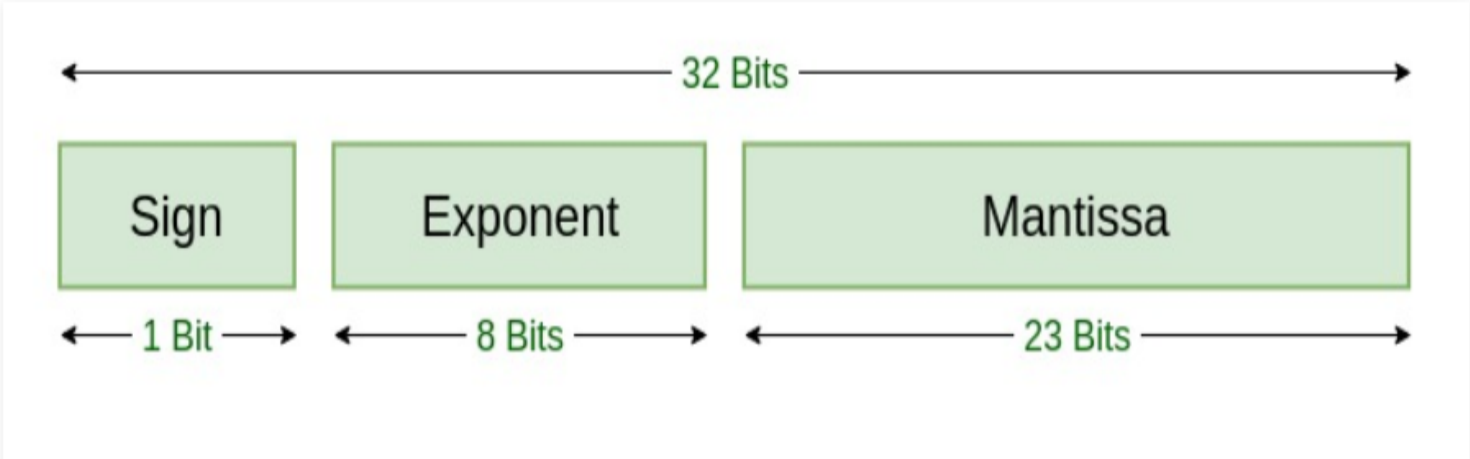
b) the range of fractions, that can be represented is determined by substituting n=16 in eq-2 and is given as -1 to $+ (1-2^{-15})$ i.e -1 to + .999969482.

DOUBLE -PRECISION FIXED -POINT FORMAT

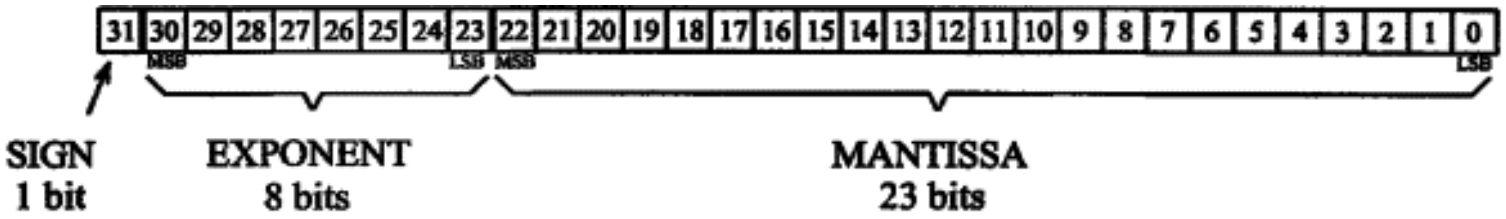
- To increase the range of numbers that can be represented in fixed point format, one obvious approach is to increase its size .
- If the size is doubled , the range of numbers increases substantially, Simply doubling the size and still using the fixed –point format creates is known as double- precision fixed –point format.
- It requires double the storage for the storage for the same data and may need double the number of accesses for the same size of data bus of the DSP device.

IEEE 754 FLOATING –POINT FORMAT

• Floating-point DSPs represent and manipulate rational numbers via a minimum of 32 bits in a manner similar to scientific notation, where a number is represented with a mantissa and an exponent (e.g., A x 2^B, where 'A' is the mantissa and 'B' is the exponent), yielding up to 4,294,967,296 possible bit patterns (2^{32}).



IEEE 754 EXAMPLE



Example 1

0 00000111 11000000000000000000000

\+ 7 0.75

$$+1.75 \times 2^{(7-127)} = +1.316554 \times 10^{-36}$$

Example 2

1 10000001 01100000000000000000000

\- 129 0.375

$$-1.375 \times 2^{(129-127)} = -5.500000$$

DYNAMIC RANGE AND PRECISION

- Dynamic range of a signal is the ratio of the maximum value to the minimum value that the signal can take in the given number representation scheme .
- The Dynamic range of a signal is proportional to the muber of bits used to represent it and increases by 6 dB for every additional bit used for the representation .

$$\text{Dynamic range} = 20\log_{10}(2^n) = 6.02n \text{ dB}$$

- Resolution is defined as the minimum value that can be represented using a number representation format . For instance, if N bits are used to represent a number from 0 to 1 , the smallest value it can take is the resolution and is given as

$$\text{Resolution} = 1/2^N \quad \text{for large N}$$

DYNAMIC RANGE AND PRECISION CONT…

- Resolution of a number representation format is normally expressed as number of bits used in the representation .At times it is also expressed in percentage .
- Precision is an issue related to the speed of DSP implementation. In general, techniques to improve the precision of an implementation reduce its speed.
- Larger word size improves the precision but may pose a problem with the speed of the processor, especially if its bus width is limited.
- For example if the 32-bit product of a 16X16 multiplication has to be preserved without loss of precision , two memory access are required to store and recall this product using a 16-bit bus.

DYNAMIC RANGE AND PRECISION CONT…

- When floating point number representation is used , the exponent determines the dynamic range. Since the exponent in the floating point representation is a power, the dynamic range of a floating point number is very large.
- The resolution or precision of a floating point number is determined by its mantissa.
- since the mantissa uses fewer bits compared to fixed- point representation, the precision of a floating point number representation is smaller than a comparable fixed point representation.

EXAMPLE 1

Calculate the dynamic range and precision of each of the following number representation formats

a) 24-bit , single –precision fixed point format

b) 48-bit , double –precision fixed point format

c) A floating point format with a 16-bit mantissa and an 8-bit exponent .

Sol: a) since each bit gives a dynamic range of 6 dB, the total dynamic range is 24X6=144 dB, Percentage resolution is $(1/2^{24})$ X 100 = 6 X 10^{-6}

b) since each bit gives a dynamic range of 6 dB, the total dynamic range is 48X6=288 dB, Percentage resolution is $(1/2^{48})$ X 100 = 4 X 10^{-13}

c) For floating-point representation , the dynamic range is determined by the no of bits in the exponent. Since there are 8 exponent bits, the dynamic range is (2^8-1)x6= 255X 6= 1530 dB.

EXAMPLE 1 CONT…

- The percentage resolution depends on the number of bits in the mantissa.
- Since there are 16- bits in the mantissa, the resolution is

$$(1/2^{16}) \text{ X } 100 = 1.5 \text{ X } 10^{-3}$$

Format of representation	Number of Bits Used	Dynamic Range	Precision
Fixed-Point	24 bits	144dB	$6X10^{-6}$
Double-Precision	48 bits	288dB	$4X10^{-13}$
Floating -Point	24 bits(16-bit mantissa and 8-bit exponent	1530dB	$1.5X10^{-3}$

SOURCES OF ERRORS IN DSP IMPLEMENTATIONS

- The error in the A/D and D/A in the representation of analog signals by a limited number of bits is called the quantization error.
- The quantization error decreases with the increase in the number of bits used to represent signals in A/D and D/A converters.
- The errors in the DSP calculations are due to limited word length used. These errors depend upon how the algorithm is implemented in a given DSP architecture.
- This error can be reduced by using a larger word length for data and by using rounding , instead of truncation , in calculations.
- Three types of errors
  1. A/D conversion Errors
  2. DSP computational Errors
  3. D/A conversion Errors

UNIT-2

Multiplier and multiplier accumulator, modified bus structures and memory access in PDSPs, multiple access memory, multiport memory, SIMD, VLIW architectures, pipelining, special addressing modes in PDSPs, on-chip peripherals.

UNIT-2 CLO'S

CLO 5	Understand the concept of multiplier and multiplier Accumulator.
CLO 6	Design SMID ,VLIW architectures.
CLO 7	Understand the modified bus structures and memory access in PDSPs.
CLO 8	Understand the special addressing modes in PDSPs

Multiplier and Multiplier Accumulator

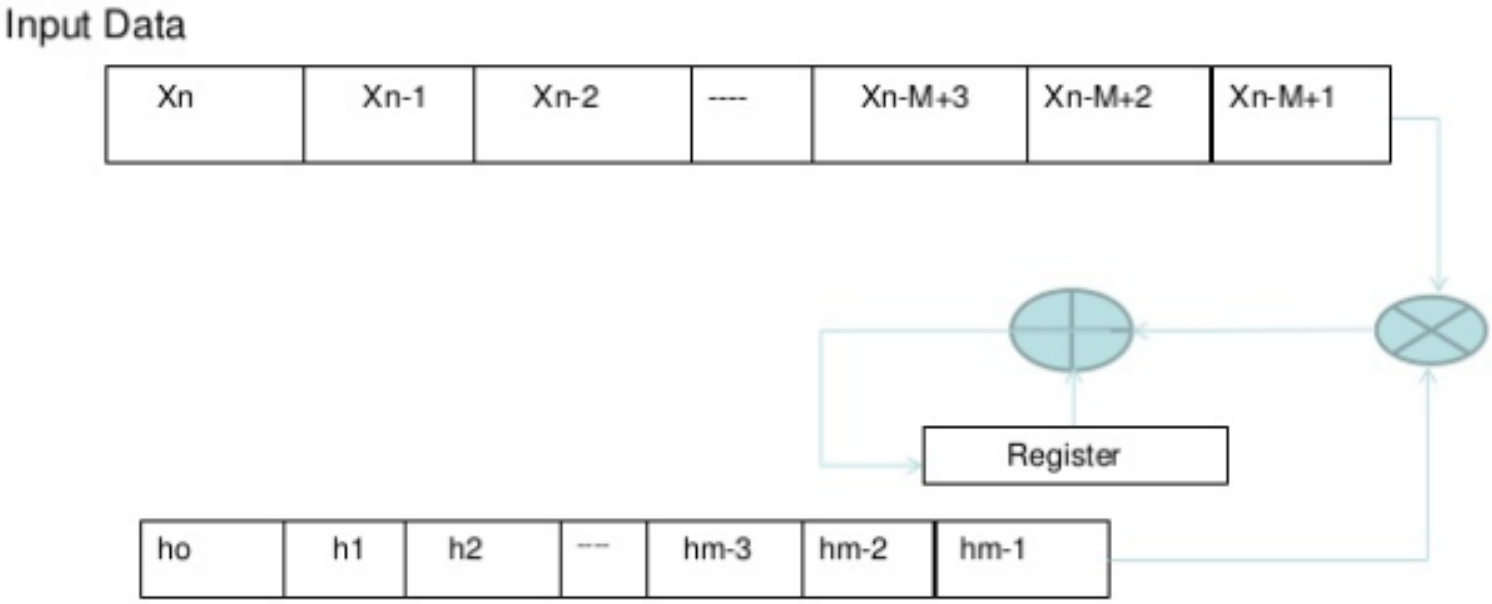
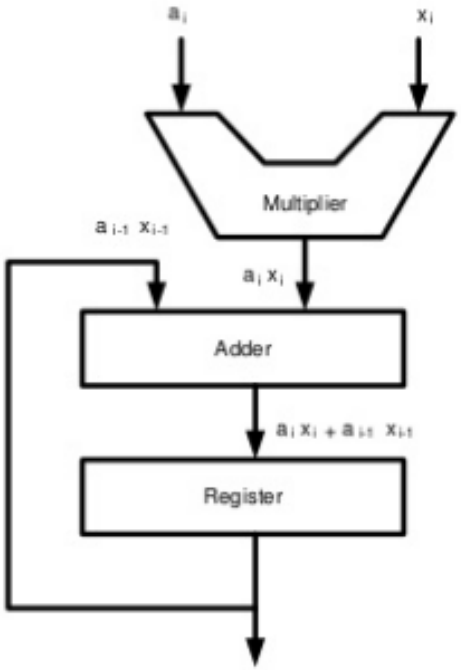


Fig. Implementation of Convolver with single multiplier/Adder

MAC UNIT Cont..

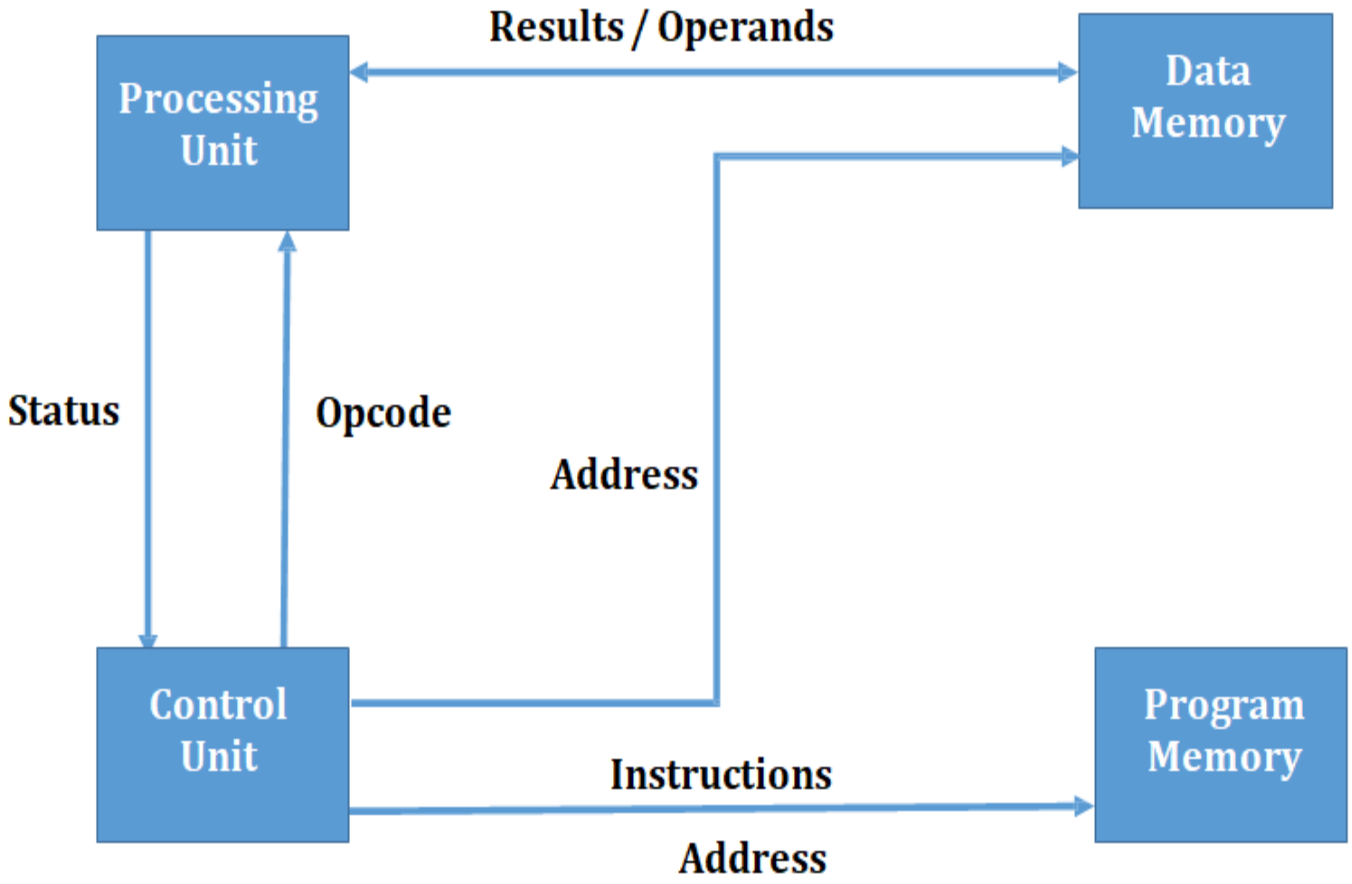
Single-Cycle MAC unit



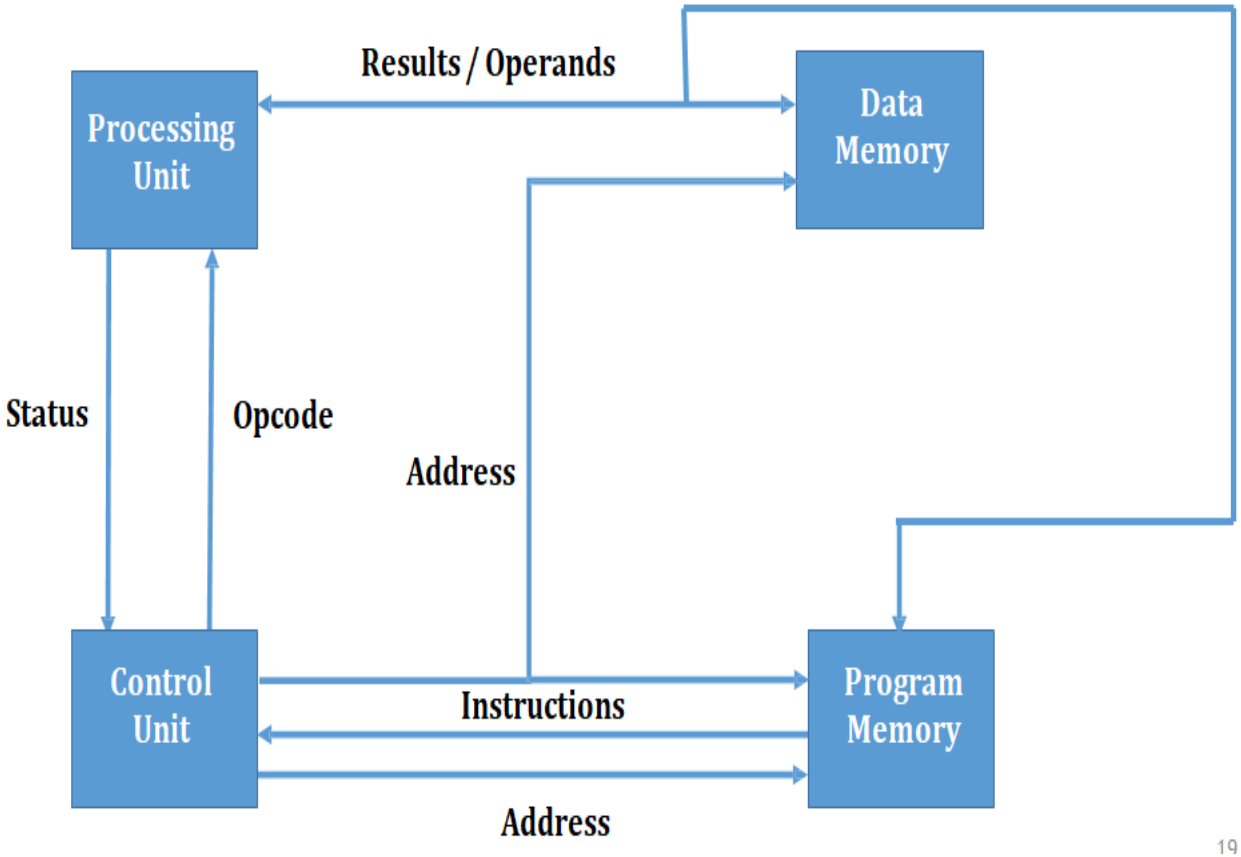
$$\sum_{i=0}^{n} (a_i x_i)$$

Can compute a sum of *n*-products in *n* cycles

MAC in Von Neumann Architecture

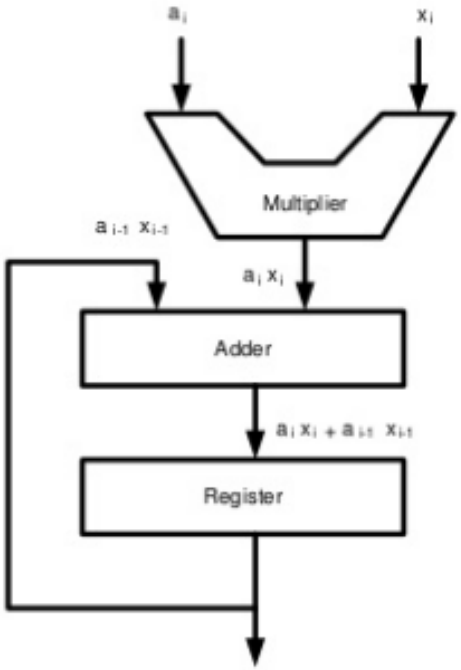


MAC UNIT Cont...



MAC UNIT Cont..

Single-Cycle MAC unit



$$\sum_{i=0}^{n} (a_i x_i)$$

Can compute a sum of *n*-products in *n* cycles

MAC UNIT

Multiplier and Multiplier Accumulator

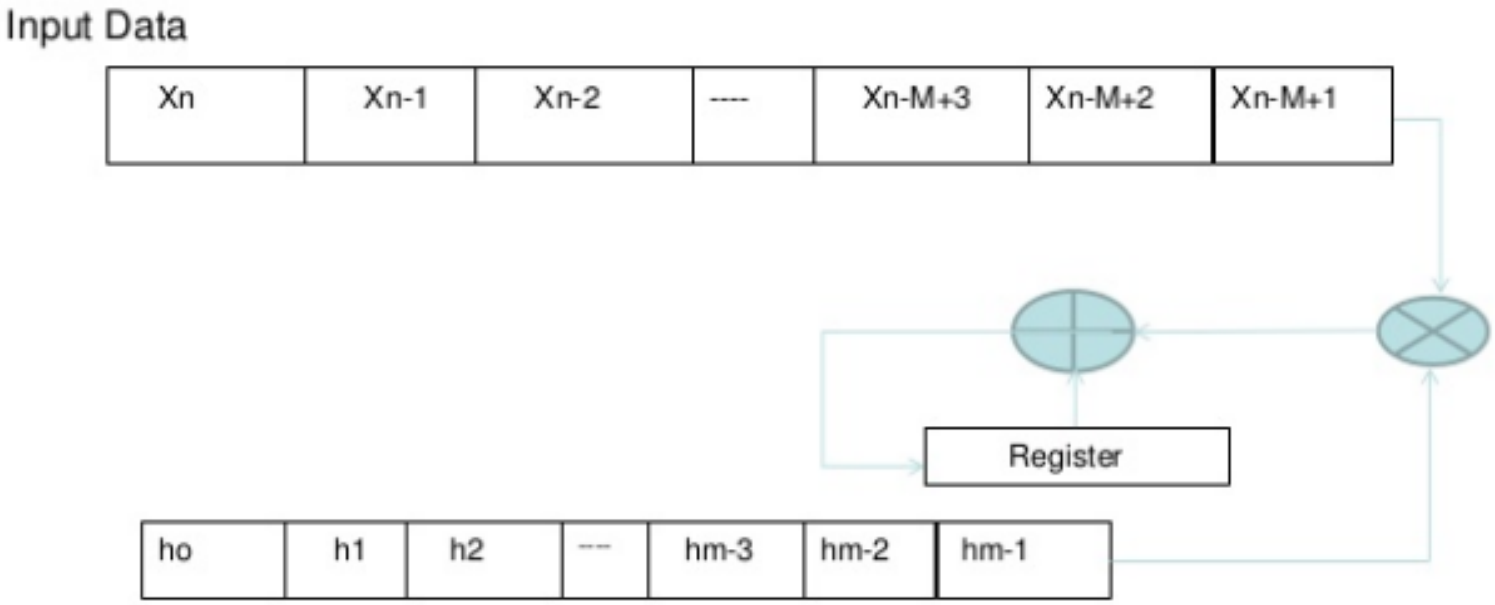


Fig. Implementation of Convolver with single multiplier/Adder

MAC IN VON NEUMANN ARCHITECTURE

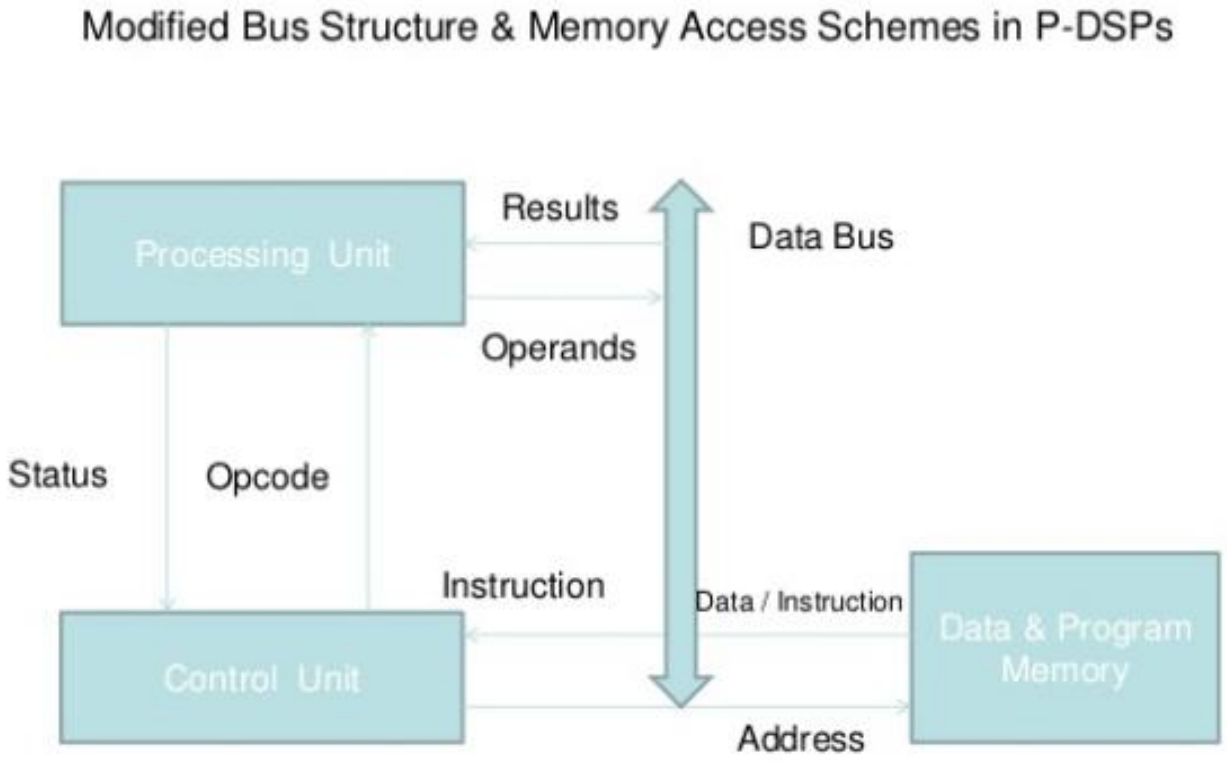


Fig: Van Neumann Architecture

MAC UNIT CONT...

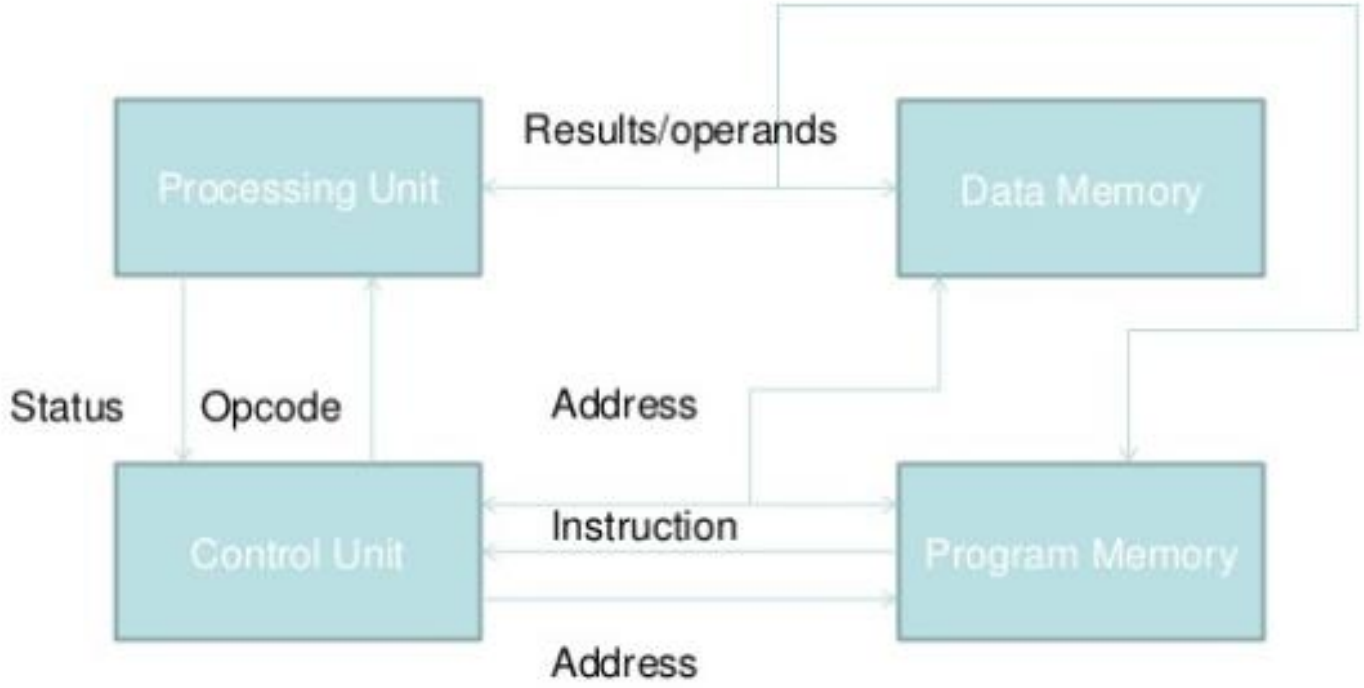


Fig: Modified harvard architecture

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.
b. This can be overcome by using a shifter to scale down the operand by an amount of log2N.
c. Similarly while calculating the product of two n bit numbers, the product can grow up to 2n bits long.
d. Generally the lower n bits get neglected and the sign bit is shifted to save the sign of the product.

SHIFTERS CONT..

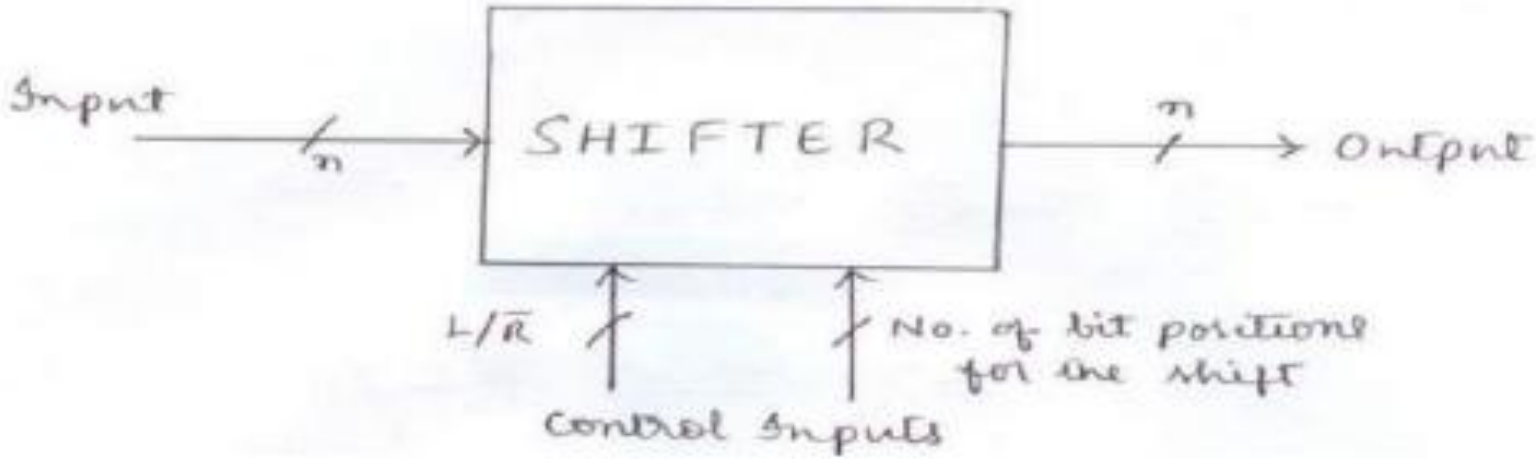
. Finally in case of addition of two floating-point numbers, one of the operands has to be shifted appropriately to make the exponents of two numbers equal.

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

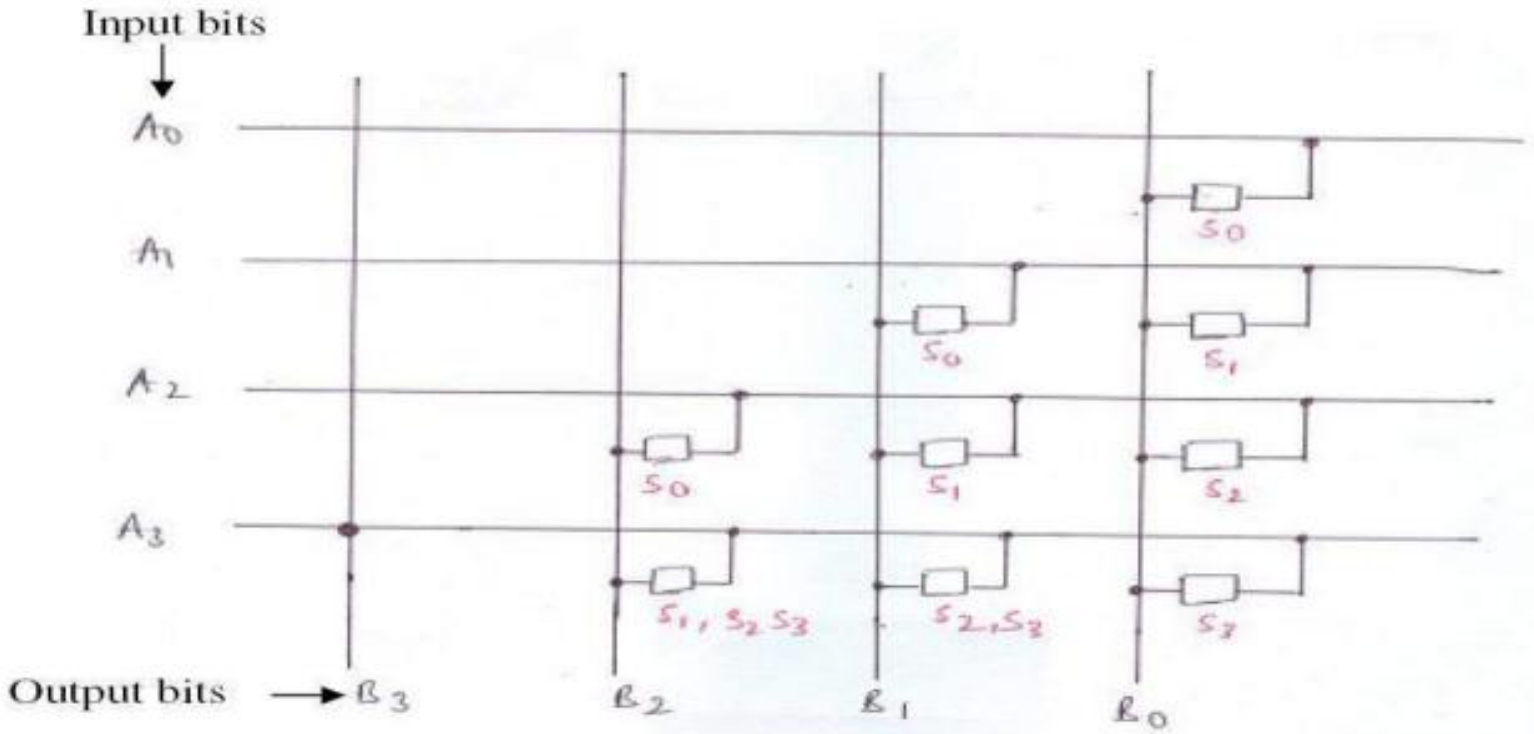
BARREL SHITERS

- 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 Additional control line is required to indicate the direction of the shift

BARREL SHITERS Cont..



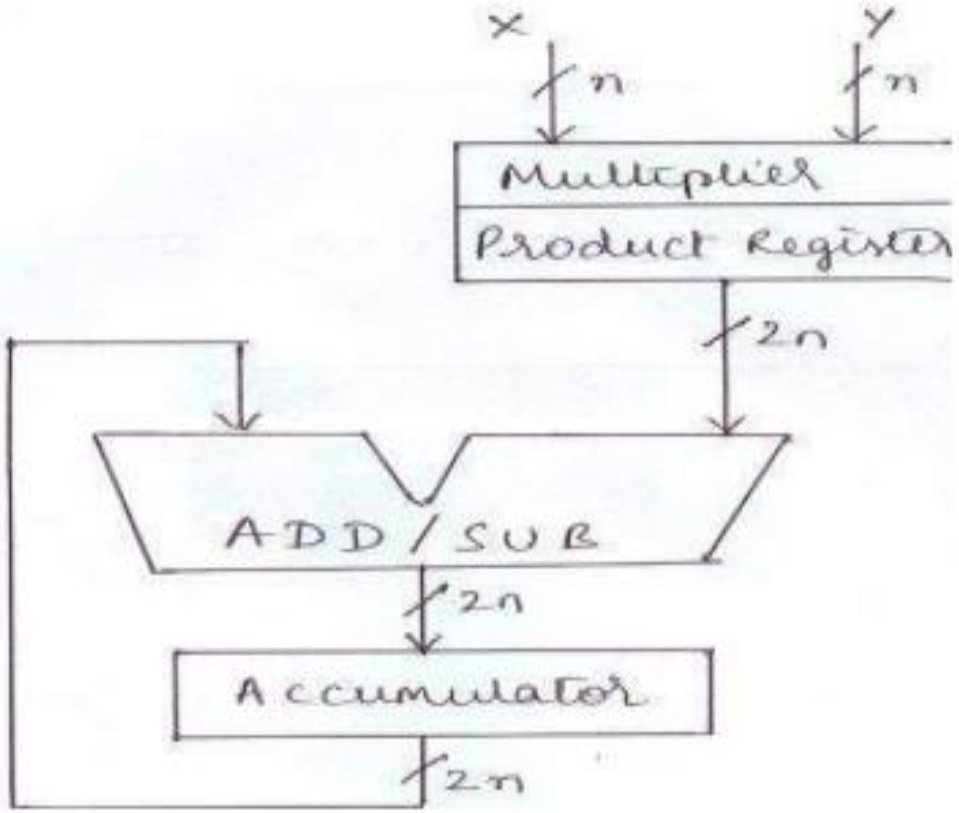
BARREL SHITERS CONT..



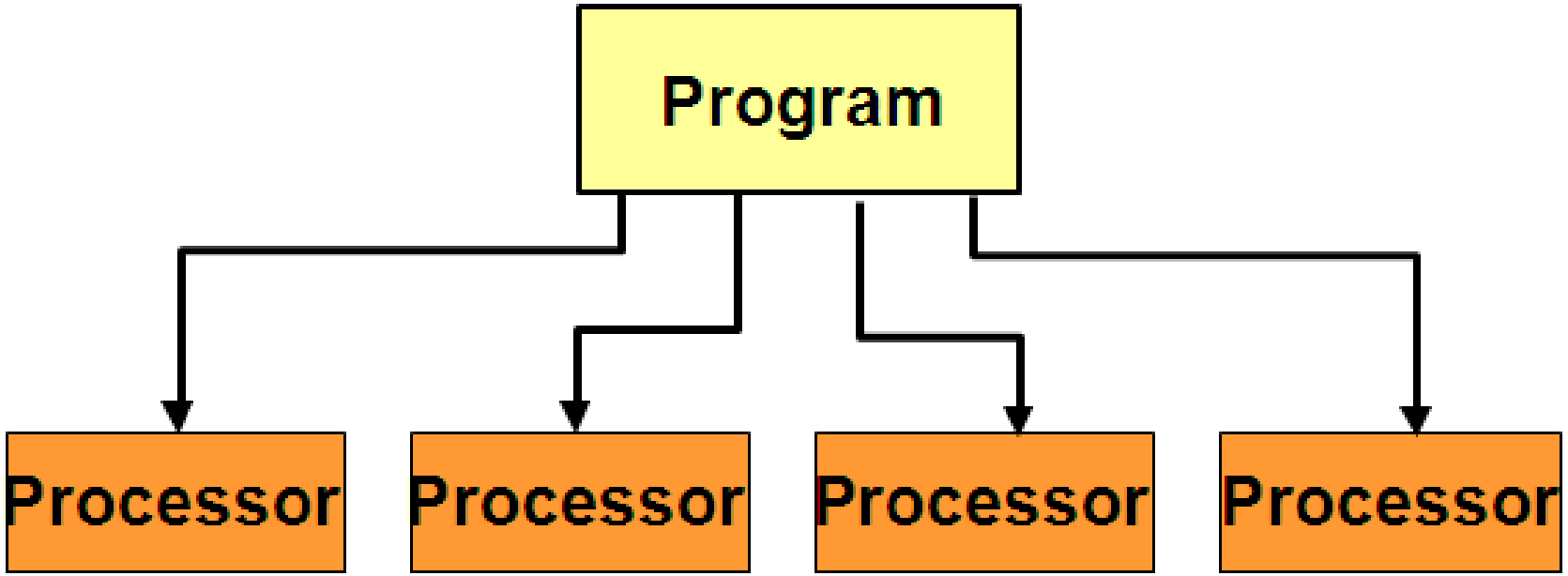
INPUT				SHIFT (SWITCH)	OUTPUT (B_3 B_2 B_1 B_0)			
A_3	A_2	A_1	A_0	0 (S_0)	A_3	A_2	A_1	A_0
A_3	A_2	A_1	A_0	1 (S_1)	A_3	A_3	A_2	A_1
A_3	A_2	A_1	A_0	2 (S_2)	A_3	A_3	A_3	A_2
A_3	A_2	A_1	A_0	3 (S_3)	A_3	A_3	A_3	A_3

Fig Implementation of a 4 bit Shift Right Barrel Shifter

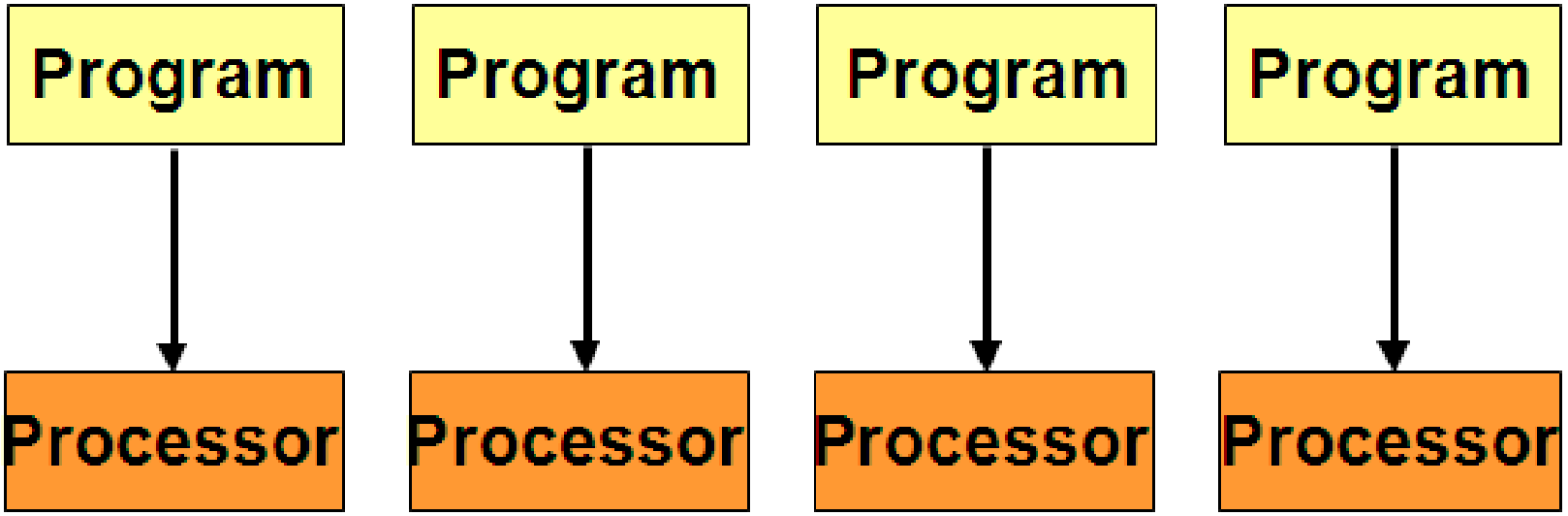
MAC UNIT


X
Y
n
n
Multiplier
Product Register
2n
ADD / SUB
2n
Accumulator
2n

PROCESSOR ARCHITECTURES

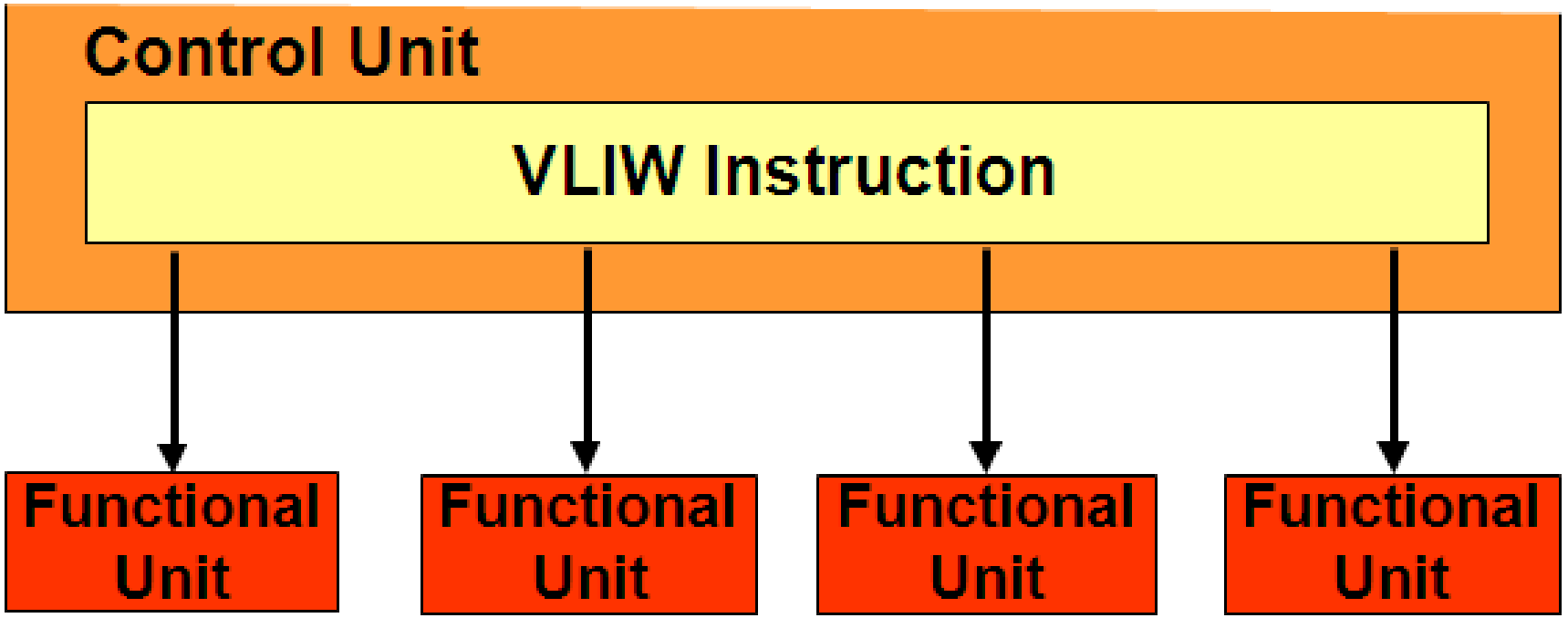


Processor Architectures Cont..



Processor Architectures Cont..

- VLIW – Very Long Instruction Words



VLIW ARCHITECTURE

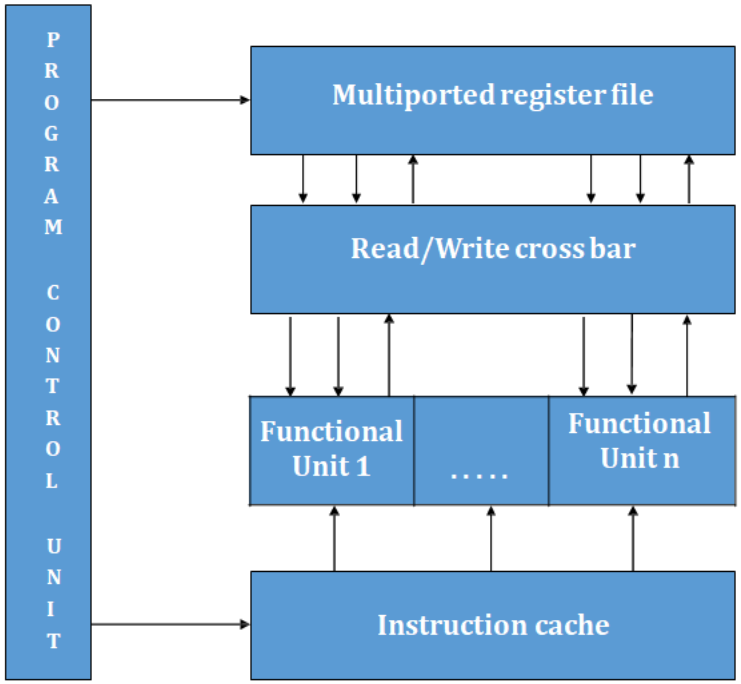
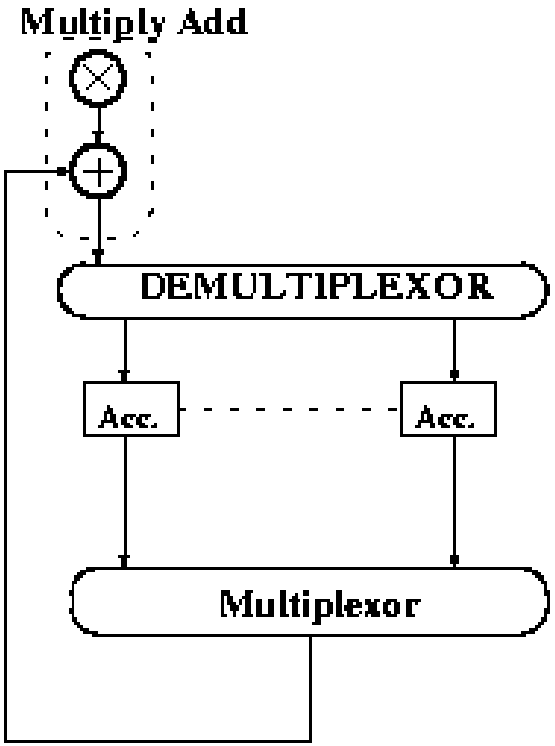


Fig: Block Diagram of the VLIW Architecture

Low Power MMAC Multiplier Multiple Accumulator

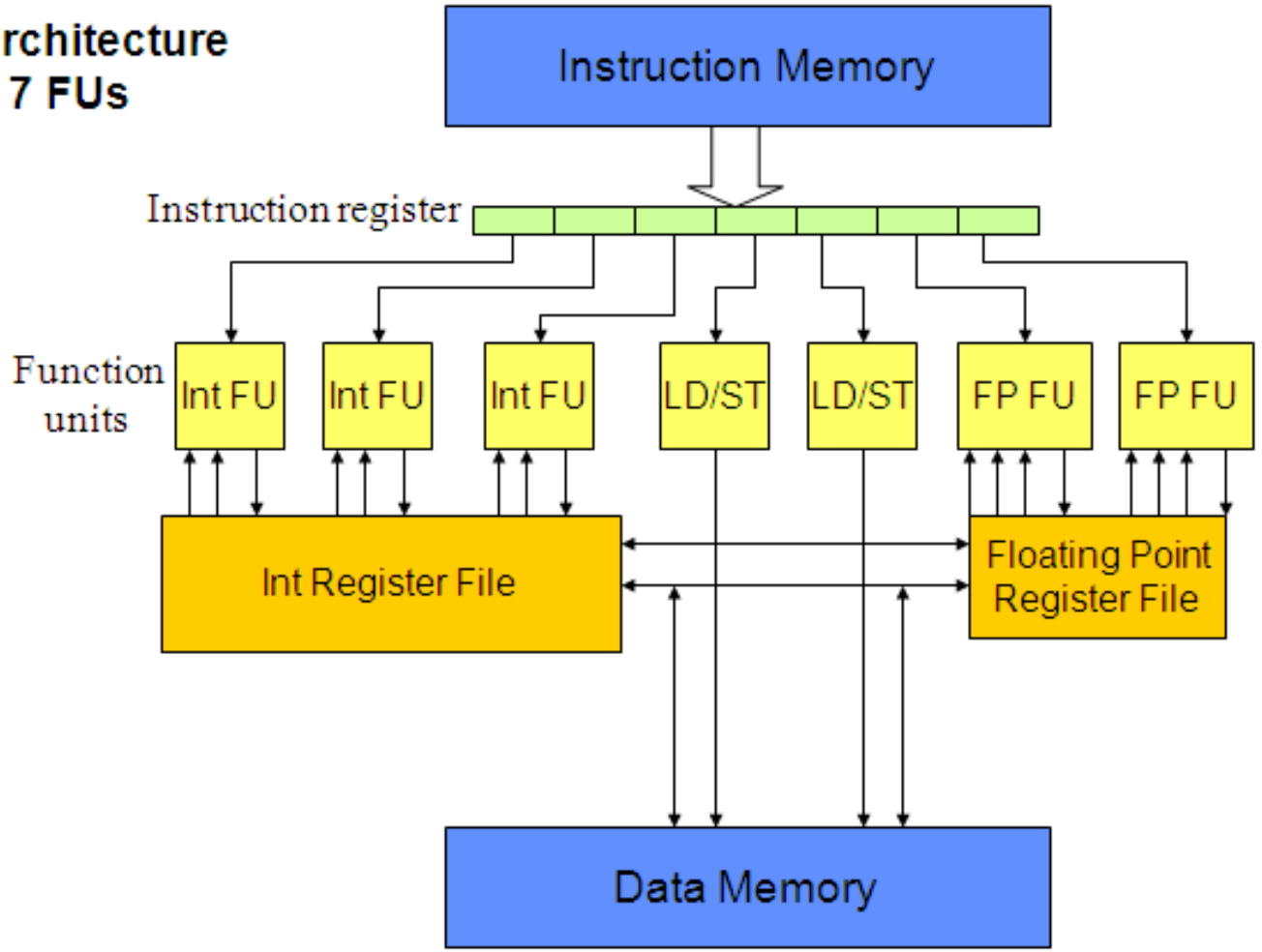


MMAC architecture: the number of output iterations that can coexist is equal to the number of Accumulators

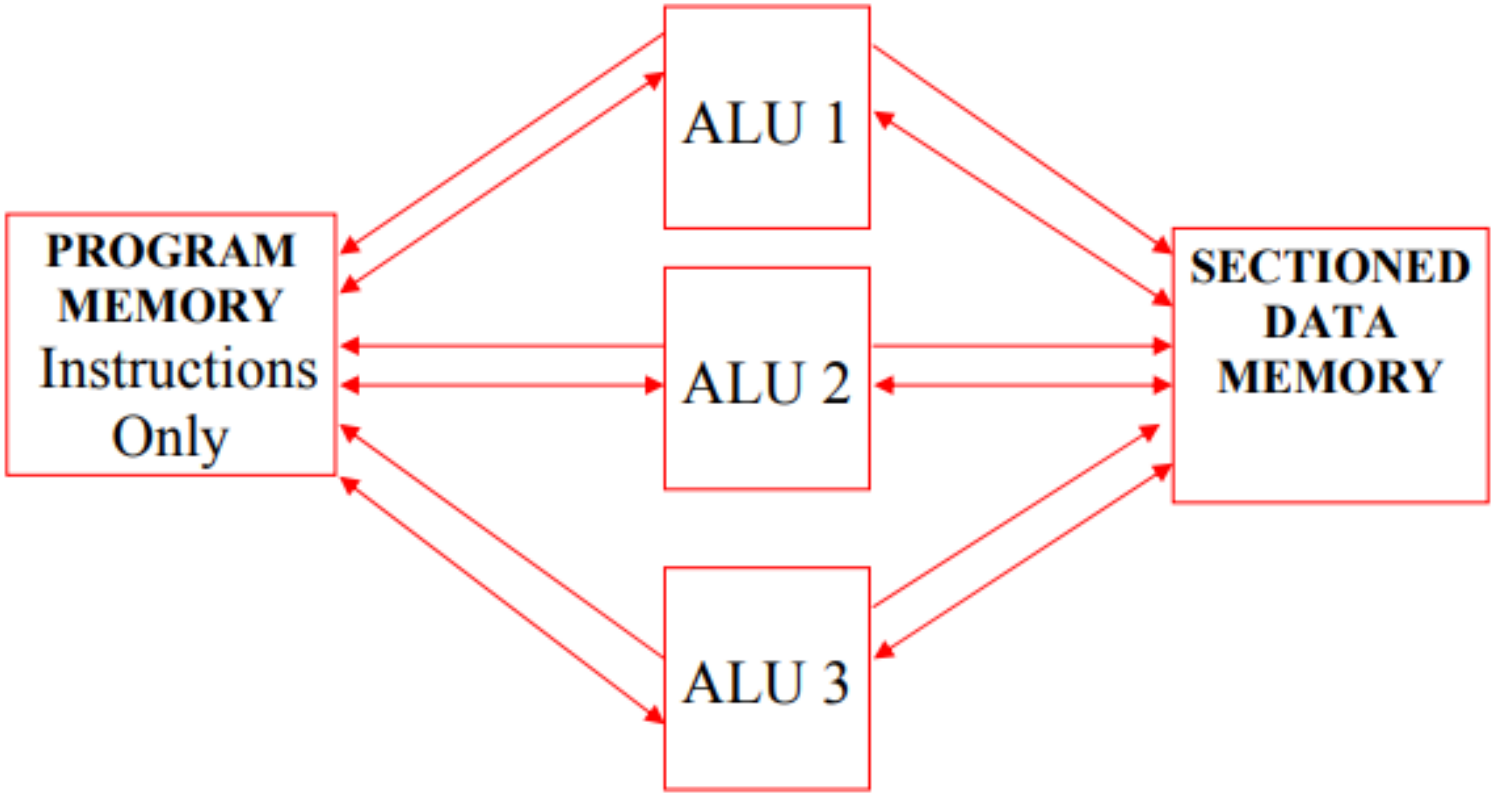
- By using anti-aliasing filters.

VLIW: GENERAL CONCEPT

A VLIW architecture with 7 FUs



Basic structure of VLIW Architecture



VLIW characteristics

- Multiple operations per instruction
- One instruction per cycle issued (at most)
- Compiler is in control
- Only RISC like operation support
  - Short cycle times
  - Easier to compile for
- Flexible: Can implement any FU mixture
- Extensible / Scalable

However:

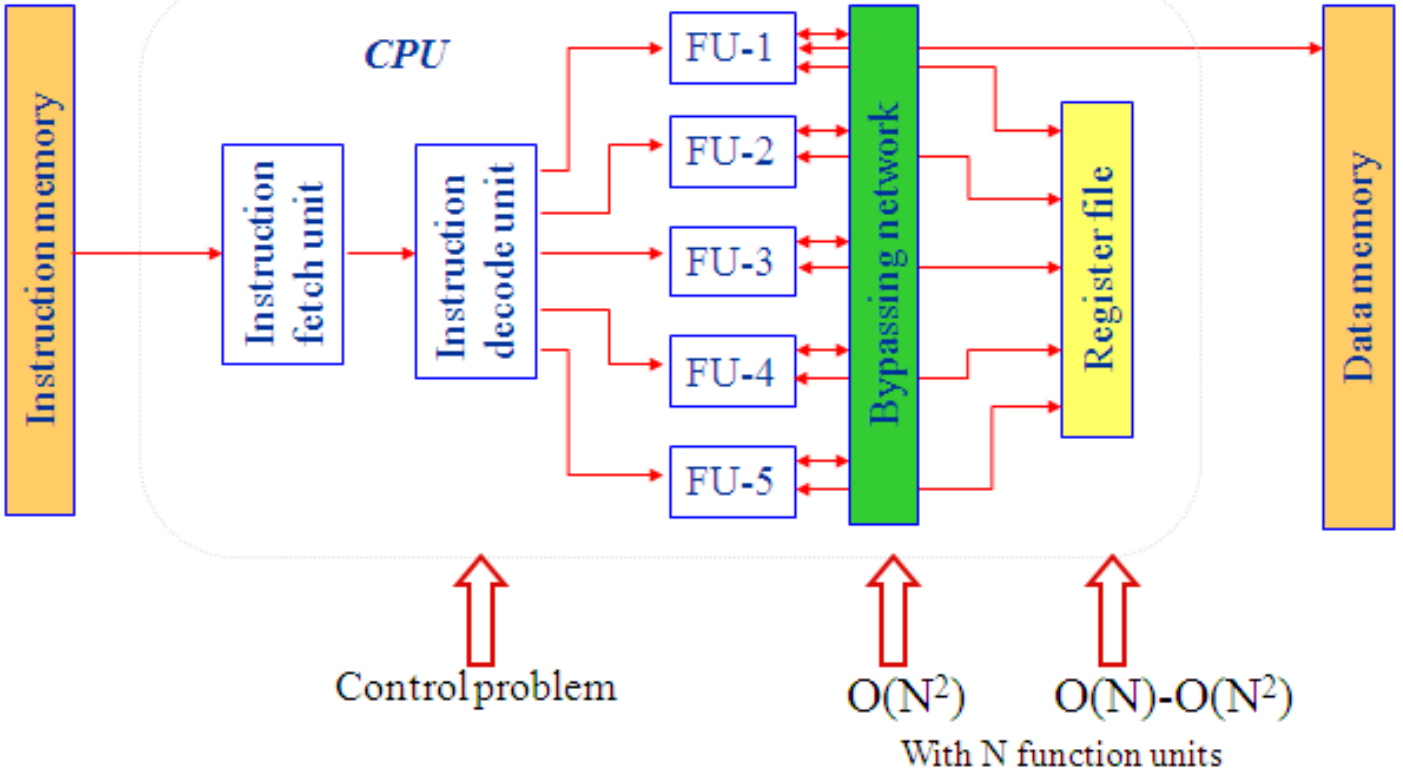
- tight inter FU connectivity required
- ***not binary compatible !!***
  - (new long instruction format)
- low code density

VLIW EXAMPLE: TMS320C62

TMS320C62 VelociTI Processor

- 8 operations (of 32-bit) per instruction (256 bit)
- Two clusters
  - 8 Fus: 4 Fus / cluster : (2 Multipliers, 6 ALUs)
  - 2 x 16 registers
  - One bus available to write in register file of other cluster
- Flexible addressing modes (like circular addressing)
- Flexible instruction packing
- All instruction conditional
- Originally: 5 ns, 200 MHz, 0.25 um, 5-layer CMOS
- 128 KB on-chip RAM

VLIW EVALUATION



MULTIPLE ACCESS MEMORY

- The number of memory accesses/clock period can also be increased by using a high speed memory than one memory accesses/clock period.
- The concept of DARAM
- Dual-access RAM blocks can be accessed twice per machine cycle. This memory is intended primarily to store data values; however, it can be used to store program as well.
- At reset, the DARAM is mapped into data memory space

MULTIPORTED MEMORY

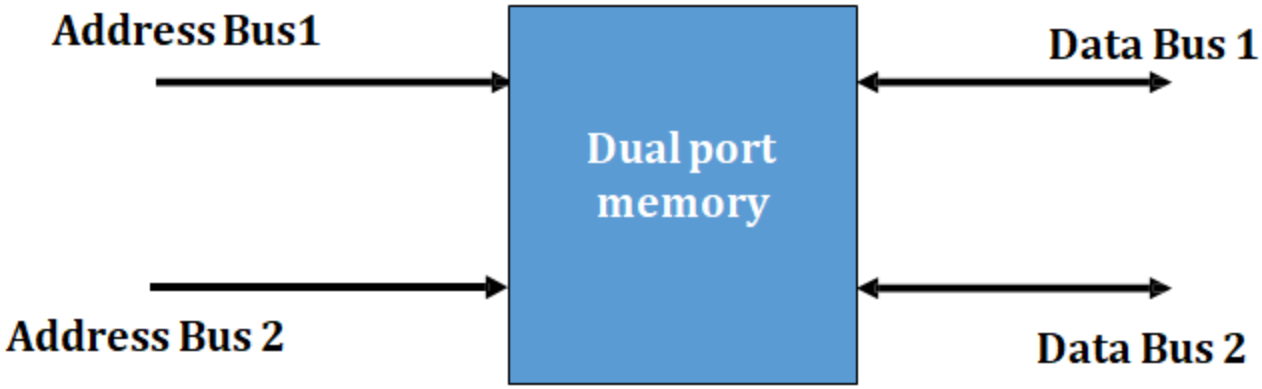


Fig: Block Diagram of a dualported memory

PIPELINING

- One of the approach for increasing the efficiency of P-DSPs and Advanced Microprocessors.
- An instruction cycle starting with the fetching of an instruction & ending with the execution of the instruction including the time storage of the results can be split into a number of microinstructions.
- Pipelining is a technique where multiple instructions are overlapped during execution. Pipeline is divided into stages and these stages are connected with one another to form a pipe like structure. Instructions enter from one end and exit from another end. Pipelining increases the overall instruction throughput
- Different stages of pipelining are Instruction fetch ,Instruction decode, register fetch ,Execute ,Memory access.

PIPELINING

Fig: Instruction cycles of processor with no pipelining

Value of T	Fetch	Decode	Read	Execute
1	I1			
2		I1		
3			I1	
4				I1
5	I2			
6		I2		
7			I2	
8				I2
9	I3			
10		I3		
11			I3	
12				I3

Fig: Instruction cycles of processor with pipelining

Value of T	Fetch	Decode	Read	Execute
1	I1			
2	I2	I1		
3	I3	I2	I1	
4	I4	I3	I2	I1
5	I5	I4	I3	I2
6	I6	I5	I4	I3
7	I7	I6	I5	I4
8	I8	I7	I6	I5
9	I9	I8	I7	I6
10		I9	I8	I7
11			I9	I8
12				I9

SPECIAL ADDRESSING MODES IN P-DSPS

1) Short Immediate Addressing
2) Short direct Addressing
3) Memory-mapped Addressing
4) Indirect Addressing
5) Bit Reversed Addressing Mode
6) Circular Addressing

SPECIAL ADDRESSING MODES IN P-DSPS

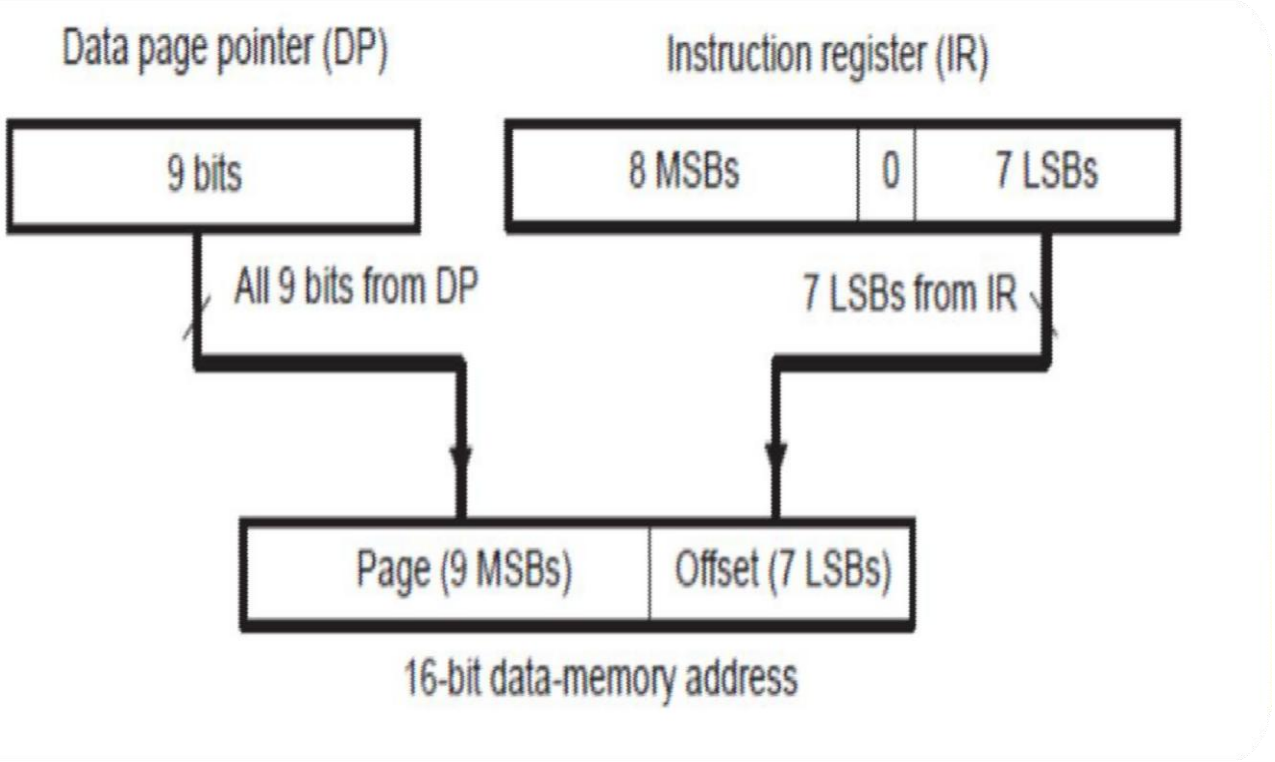
1) Short Immediate Addressing

- Permits the operand to be specified using a short constant that forms part of a single word instruction.
- The length of the short constant depends on he instruction type & P-DSP.
- Short immediate values can be 3, 5, 8, or 9 bits in length. Mode

2) Short direct Addressing

- Permits the lower order address of the operand of an instruction to be specified in the single word instruction.
- In TI TMS320 DSPs, the higher order 9 bits of the memory are stored in the data page pointer & only the lower 7 bits are specified as a part of the instruction.

SPECIAL ADDRESSING MODES IN P-DSPS



Generation of Data Addresses in Direct Addressing Mode

3) Memory-mapped Addressing

- The CPU registers & I/O registers of P-DSPs are also accessible as memory location.
- This is achieved by storing them in either the starting page or the final page of the memory space.
- For Eg. In TMS320C5X, page 0 corresponds to CPU registers & I/O registers.
- When these registers are accessed using memory mapped addressing modes, the higher address bits are not taken from the data page pointer & instead made to be 0 in case of TI DSPs & 1 in Motorola DSPs

4) Indirect Addressing

- In indirect addressing, any location in the 64K-word data space can be accessed using the 16-bit address contained in an auxiliary register.
- The C54x DSP has eight 16-bit auxiliary registers (AR0–AR7).
- Indirect addressing is used mainly when there is a need to step through sequential locations in memory in fixed-size steps.
- In TI, offset register is called as INDX register.
- In Analog devices, called as Modifier Register.
- Contents can also be updated by a constant using Bit Reversed Addressing Mode.
- Pre-increment / decrement & Post-increment / decrement.

BIT REVERSED ADDRESSING MODE

- The binary pattern corresponding to a particular decimal number is obtained by writing the natural binary equivalent of the number in the reverse order so that the MSB of the natural binary becomes the LSB of the bit reversed number & vice-versa.

BIT REVERSED ADDRESSING MODE

Decimal Number	Natural Binary Number	Bit Reversed Number
0	0000	0000
1	0001	1000
2	0010	0100
3	0011	1100
4	0100	0010
5	0101	1010
6	0110	0110
7	0111	1110
8	1000	0001
9	1001	1001
10	1010	0101
11	1011	1101
12	1100	0011
13	1101	1011
14	1110	0111
15	1111	1111

CIRCULAR ADDRESSING

- Memory can be organized as a circular buffer with the beginning memory address & the ending memory address corresponding to this buffer defined by the programmer.
- In this, when the address pointer is incremented, the address will be checked with the ending memory address of the circular buffer.
- If it exceeds that, the address will be made equal to the beginning address of the circular buffer

ON CHIP PERIPHERALS

- Clock Generator
- Hardware Timer
- Software-Programmable Wait-State Generators

  Parallel I/O Ports
- Host Port Interface (HPI)
- Serial Port
- TDM Serial Port
- Buffered Serial Port
- User Maskable Interrupts

UNIT-3

Architecture of TMS320C54XX DSPs, addressing modes, memory space of TMS320C54XX processors. Program control, instruction set and programming, on-chip peripherals, interrupts of TMS320C54XX processors, pipeline operation.

UNIT-3 CLO'S

CLO 9	Understand the architecture of TMS320C54XX DSPs.
CLO 10	Understand the addressing modes and memory space of TMS320C54XX DSPs.
CLO 11	Understand the various interrupts and pipeline operation of TMS320C54XX processors.
CLO 12	Understand the special addressing modes in PDSPs
CLO 13	Understand the concept of on-chip Peripherals

Architecture of TMS320C54XX

- Fixed Point Processor
- Advanced Harvard Architecture ,CICS processor

  Separate memory bus structures for program and data
- High Degree of parallelism

  Multiply ,load/store , add/sub to/from ACC and new address generation can be done simultaneously
- Powerful Instruction set & most of operations are of single cycle
- Targeted for portable devices (cellular phones,MP3 players , digital cameras)

INTRODUCTION

This unit provides the architectural overview of TMS320C54XX which comprises of :-

- CPU
- On Chip Memory
- On Chip Peripherals
- Addressing Modes
- Interrupts
- Program Control
- Internal Memory Bus Organization
- Buses
- Pipelining

INTRODUCTION

- The C54XX DSP uses modified harvard architecture that maximizes processing power eight buses.
- Separate Program & Data buses allow simultaneous access to program & data providing high degree of parallelism.
- Data can be transferred between program & data memory.

EFFICIENT DATA/PROGRAM FLOW

#1: CPU designed for efficient DSP processing

- MAC unit, 2 Accumulators, AdditionalAdder, Barrel Shifter

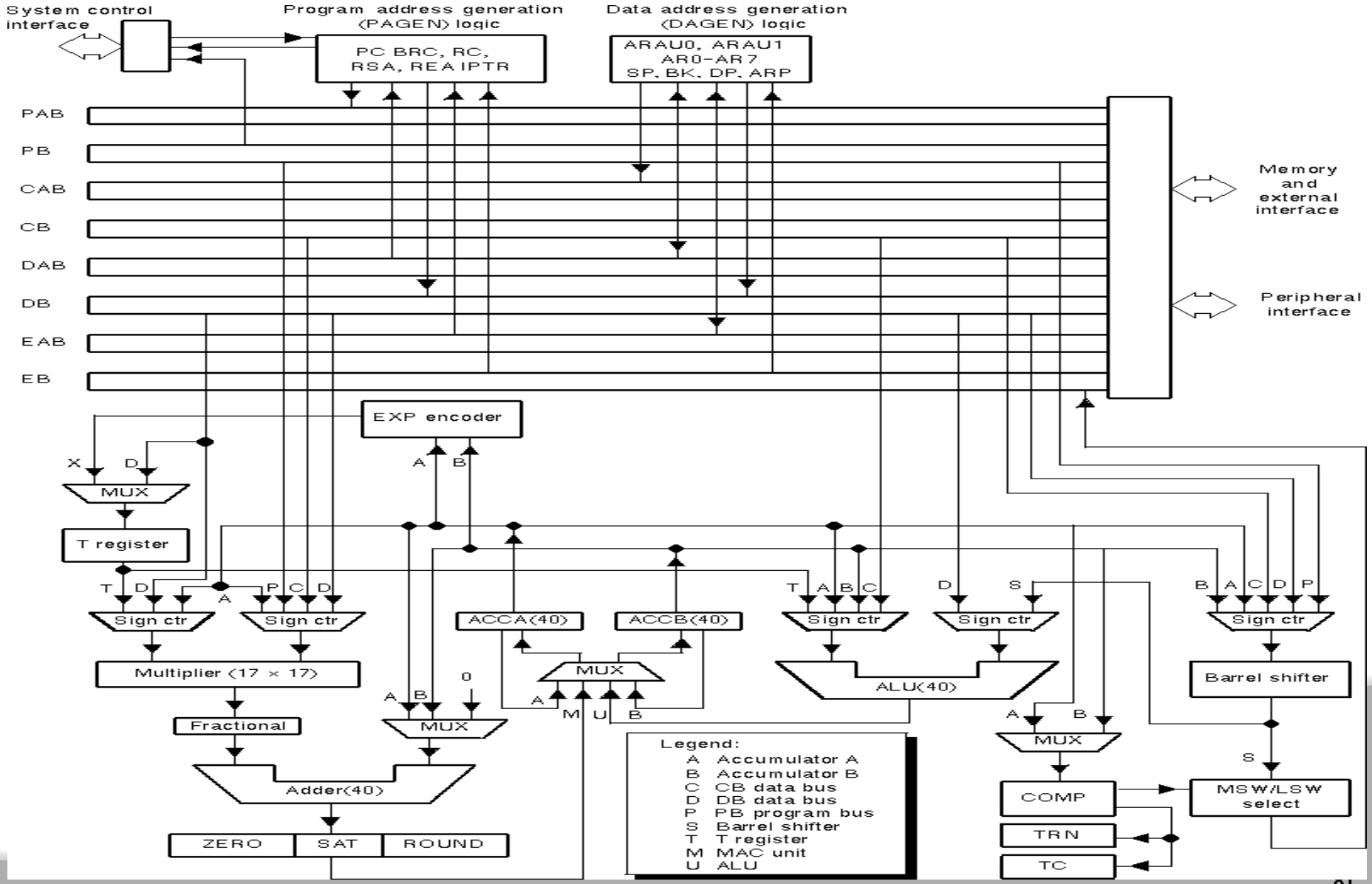
#2: Multiple busses for efficient data and program flow

- Four busses and large on-chip memory that result in sustained performance near peak

#3: Highly tuned instruction set for powerful DSP computing

- Sophisticated instructions that execute in fewer cycles, with less code and low power demands

TMS320C54X INTERNAL BLOCK DIAGRAM



CENTRAL PROCESSING UNIT (CPU)

The ’54xx CPU contains the following,

•40-bit Arithmetic Logic Unit (ALU)

•40-bit Accumulators (A and B)

•Barrel Shifter

•17 x 17-bit Multiplier

•40-bit Adder

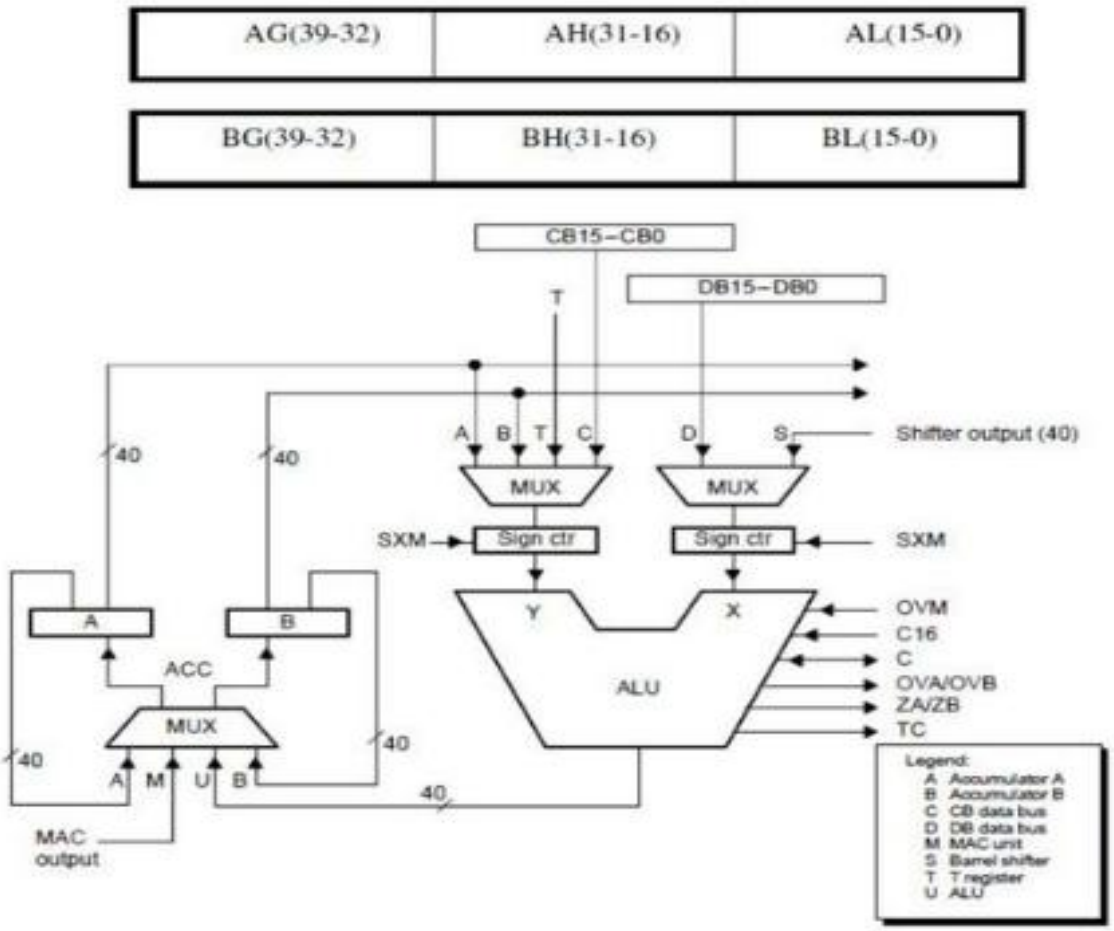
•Compare, Select And Store Unit (CSSU)

•Exponent Encoder(exp)

• Data Address Generation Unit (DAGEN) and

•Program Address Generation Unit (PAGEN).

FUNCTIONAL DIAGRAM OF CPU OF TMS320C54xx



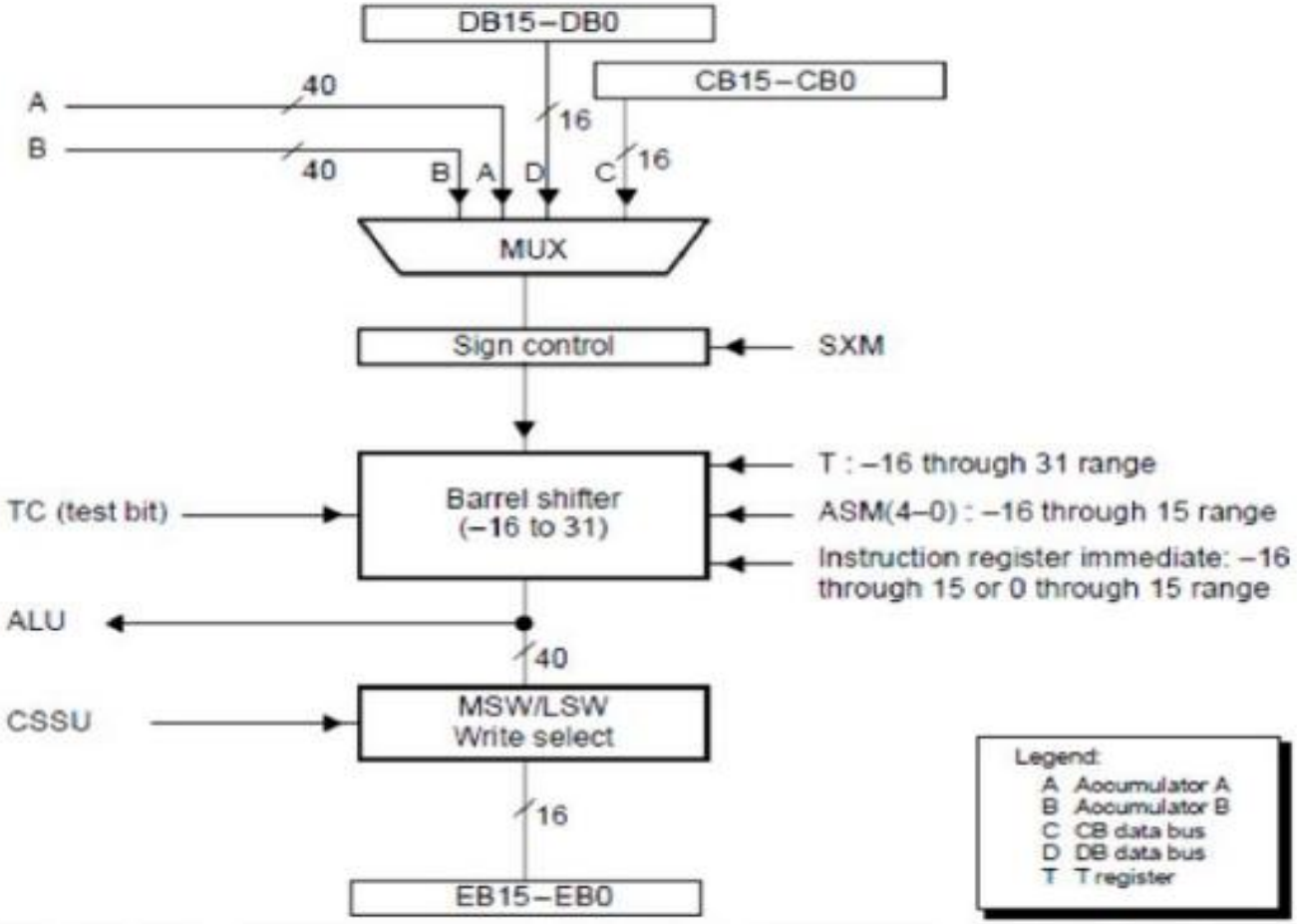
ALU

- The ALU performs 2’s complement arithmetic operations and bit-level Boolean operations on 16, 32, and 40-bit words.
- ACCUMULATORS A AND B:
- 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),Low-order Word (Bits 15-0)
- Each accumulator is memory-mapped and partitioned.
- It can be configured as the destination registers.

BARREL SHIFTER

- 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 16bits on the input data.
- The shift count field of status registers ST1, or in the temporary register T.
- 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.

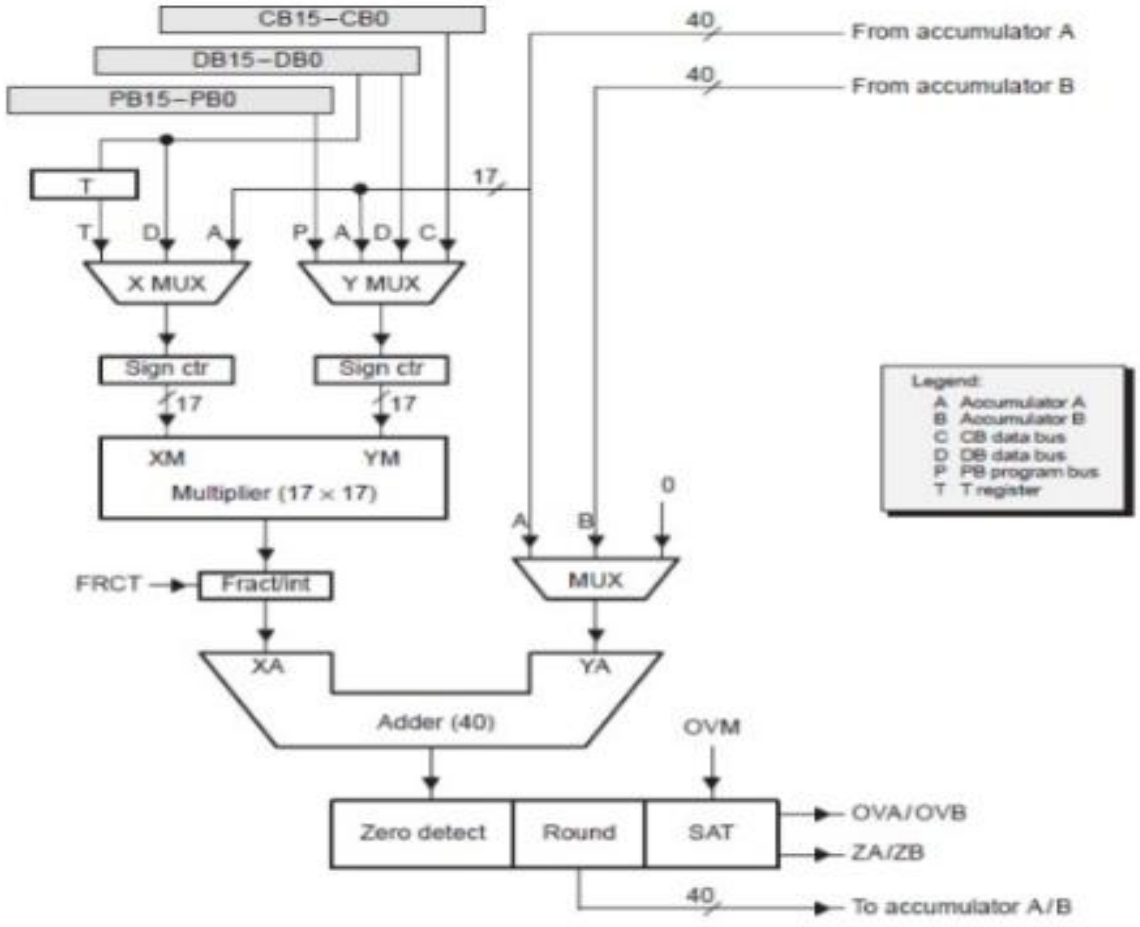
FUNCTIONAL DIAGRAM OF BARREL SHIFTER OF TMS320C54xx



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.
- The compare, select, and store unit (CSSU) is a hardware unit specifically incorporated to accelerate the add/compare/select operation.
- 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.

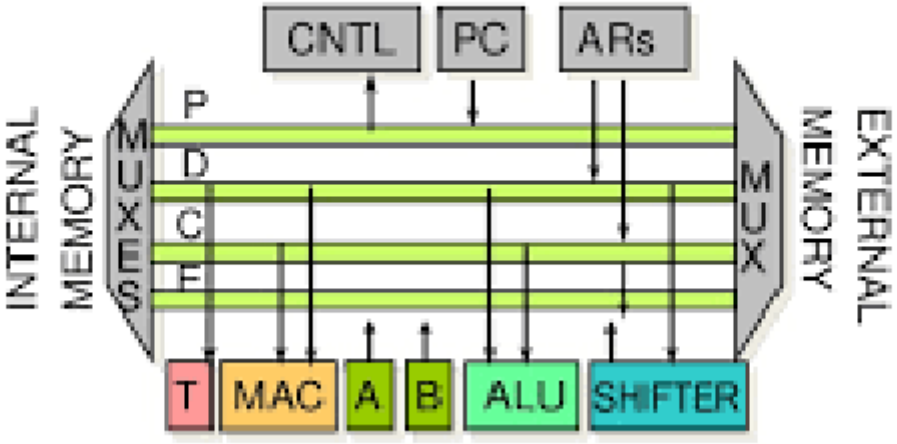
FUNCTIONAL DIAGRAM OF MULTIPLIER/ADDER OF TMS320C54xx



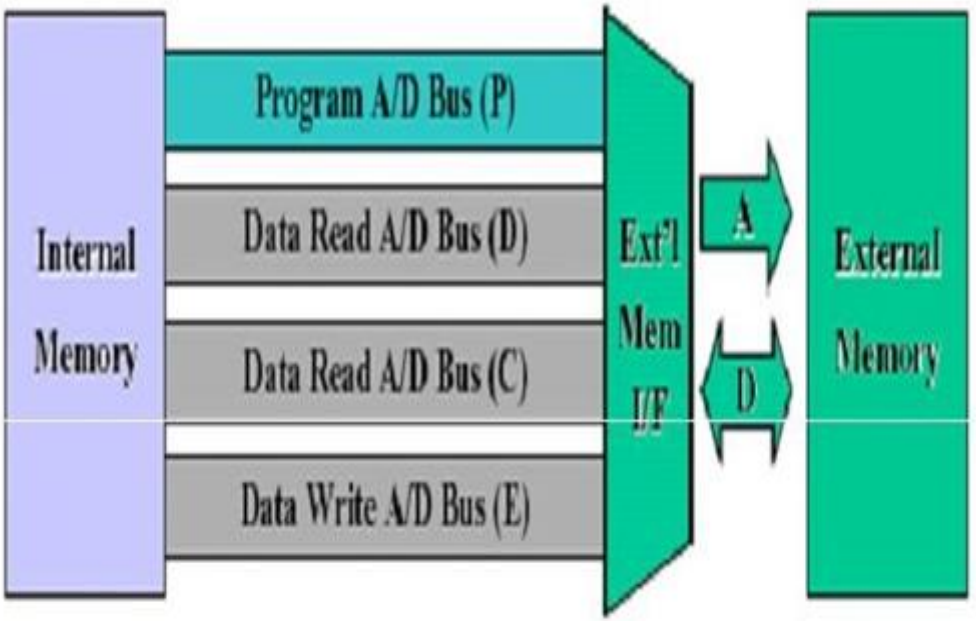
BUSES IN C54XX

- The C54XX architecture is built around 8 major 16 bit buses.
- The Program Bus carries the instruction code & immediate operands from program memory.
- Three data buses (CB,DB,EB) interconnect to various elements such as CPU, Data address generation logic ,on chip Peripherals & data memory.
- The CB & DB carry the data operands that are read from memory.
- The EB carries the data to be written to memory.
- Four address buses (PAB, CAB, DAB, and EAB) carry the addresses needed for instruction execution.

BUSES USAGE

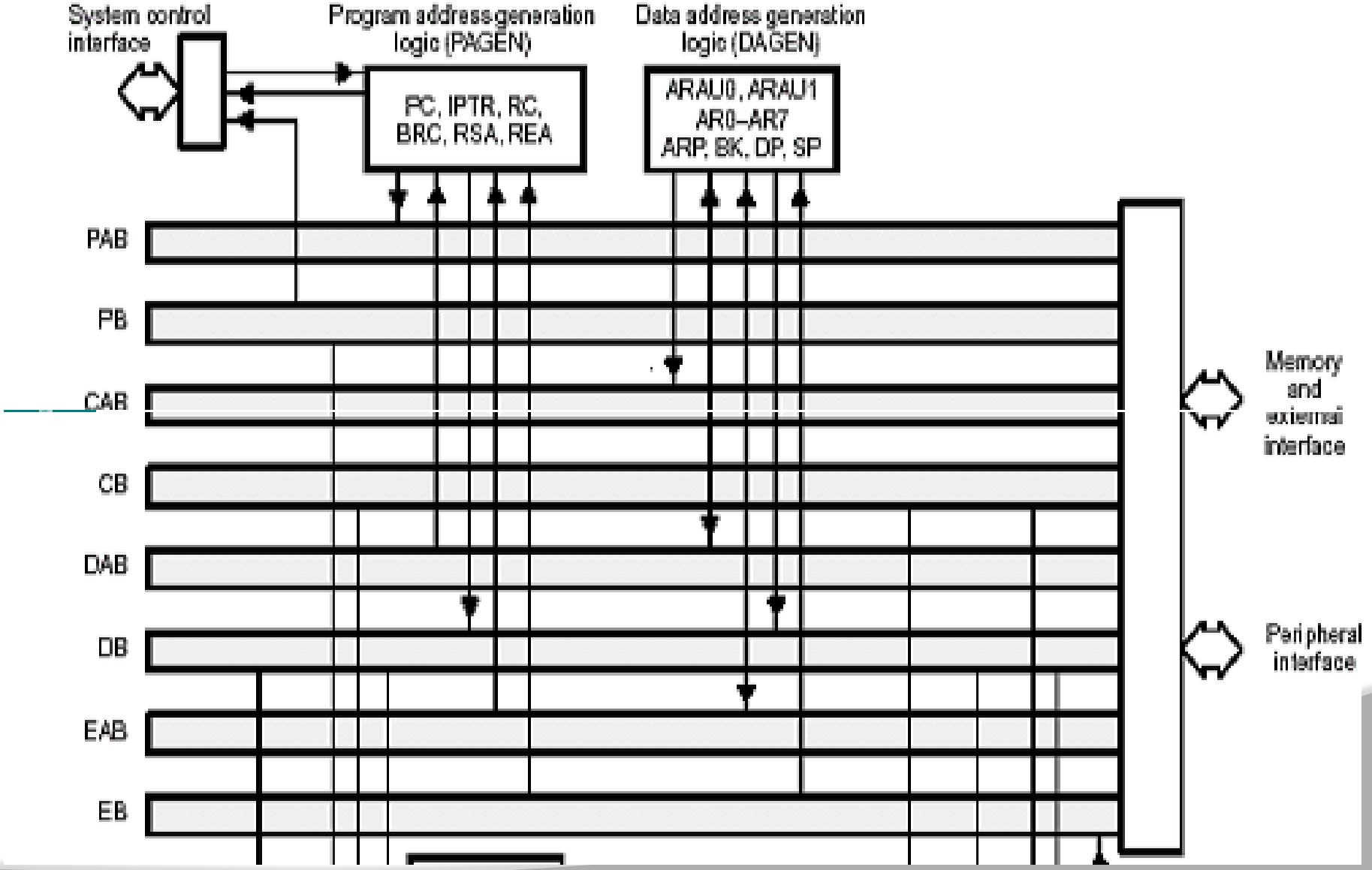

CNTL
PC
ARs
INTERNAL MEMORY
MUXES
P
D
C
E
MUX
EXTERNAL MEMORY
T
MAC
A
B
ALU
SHIFTER

BUSES



• All CPU registers ,peripheral registers and I/O ports occupy data memory space

BUSES


System control interface
Program address generation logic (PAGEN)
PC, IPTR, RC, BRC, RSA, REA
Data address generation logic (DAGEN)
ARAU0, ARAU1 AR0–AR7 ARP, BK, DP, SP
PAB
PB
CAB
CB
DAB
DB
EAB
EB
Memory and external interface
Peripheral interface

BUSES

- The C54x DSP can generate up to two data-memory addresses per cycle using the two auxiliary register arithmetic units (ARAU0 and ARAU1).
- The PB can carry data operands stored in program space to the multiplier and adder for multiply/accumulate operations or to a destination in data space for data move instructions.
- The C54x DSP also has an on-chip bidirectional bus for accessing on-chip peripherals. This bus is connected to DB and EB through the bus exchanger in the CPU interface

INTERNAL MEMORY ORGANIZATION

- Minimum address range of 192k words

  64K words for program space

  64K words for data space

  64K words for I/O space

- ROM , DARAM ,SARAM , two array shared RAM
- On-chip Memory Security option
- MMR : 26 CPU regs ,peripheral regs and scratch pad RAM block located on data page 0(DP0)
- The C54XX DSP memory is organized into three individually selectable spaces: program, data, and I/O space.

INTERNAL MEMORY ORGANIZATION

- The C54x devices can contain random access memory (RAM) and read-only memory (ROM).
- Among the devices, the following types of RAM are represented: dual-access RAM (DARAM), single-access RAM (SARAM), and two-way shared RAM.
- The DARAM or SARAM can be shared within subsystems of a multiple-CPU core device.
- We can configure the DARAM and SARAM as data memory
- or program/data memory.
- The C54x DSP also has 26 CPU registers plus peripheral registers that are mapped in data-memory space.

INTERNAL MEMORY AND MEMORY-MAPPED REGISTERS

- The amount and the types of memory of a processor have direct relevance to the efficiency and performance obtainable in implementations with the processors.
- The ‘54xx memory is organized into three individually selectable spaces: program, data, and I/O spaces.
- All ‘54xx devices contain both RAM and ROM. RAM can be either dual-access type (DARAM) or single-access type (SARAM).
- The ‘54xx processors have a number of CPU registers to support operand addressing and computations.
- The processors mode status (PMST) registers is used to configure the processor. It is a memory-mapped register located at address 1Dh on page 0 of the RAM.

INTERNAL MEMORY-MAPPED REGISTERS OF TMS320C54XX PROCESSORS.

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

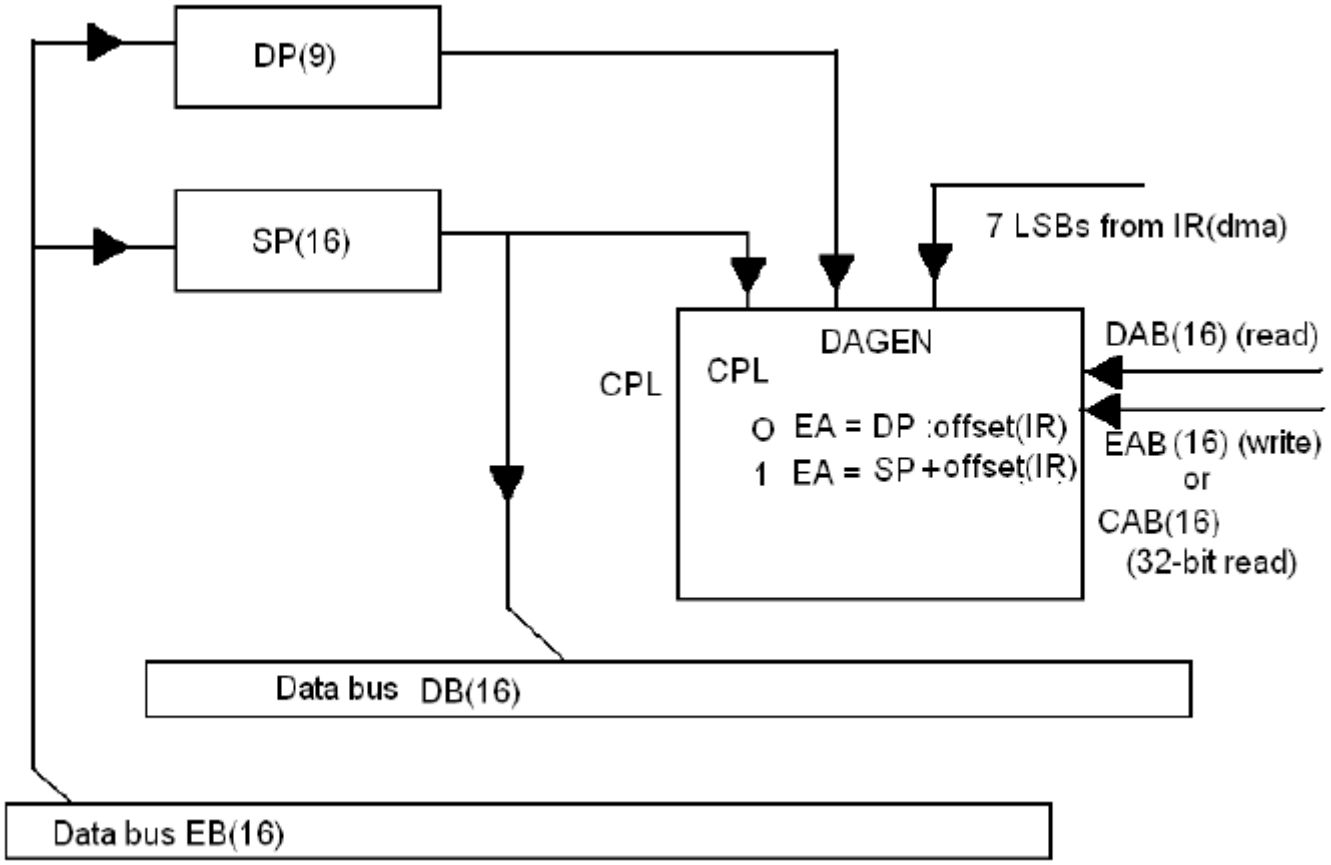
PERIPHERAL REGISTERS FOR THE TMS320C54XX PROCESSORS

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 Wait-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

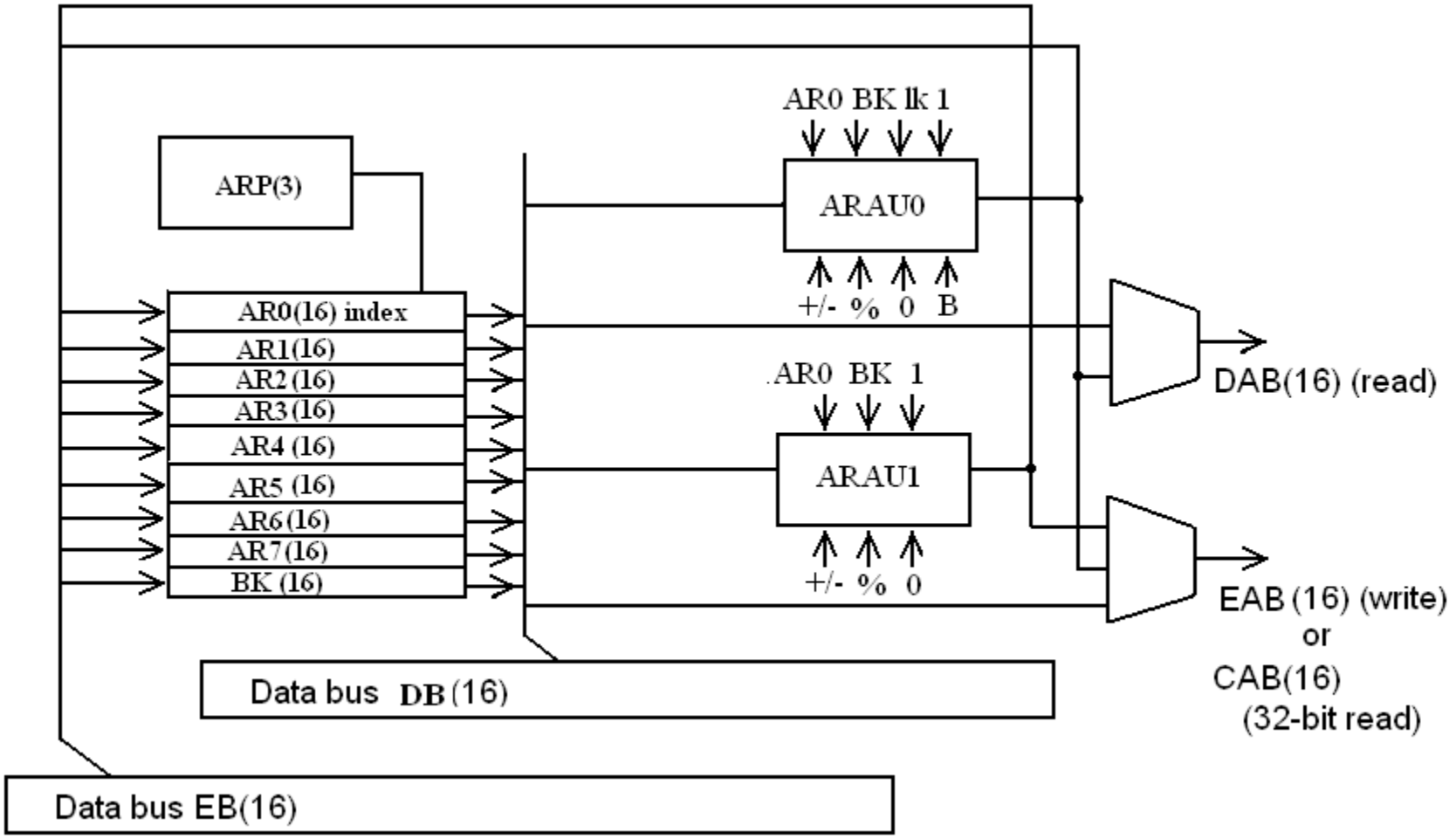
DATA ADDRESSING MODES OF TMS320C54X PROCESSORS:

- 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.

BLOCK DIAGRAM OF THE DIRECT ADDRESSING MODE FOR TMS320C54XX PROCESSORS



BLOCK DIAGRAM OF THE INDIRECT ADDRESSING MODE FOR TMS320C54XX PROCESSORS.



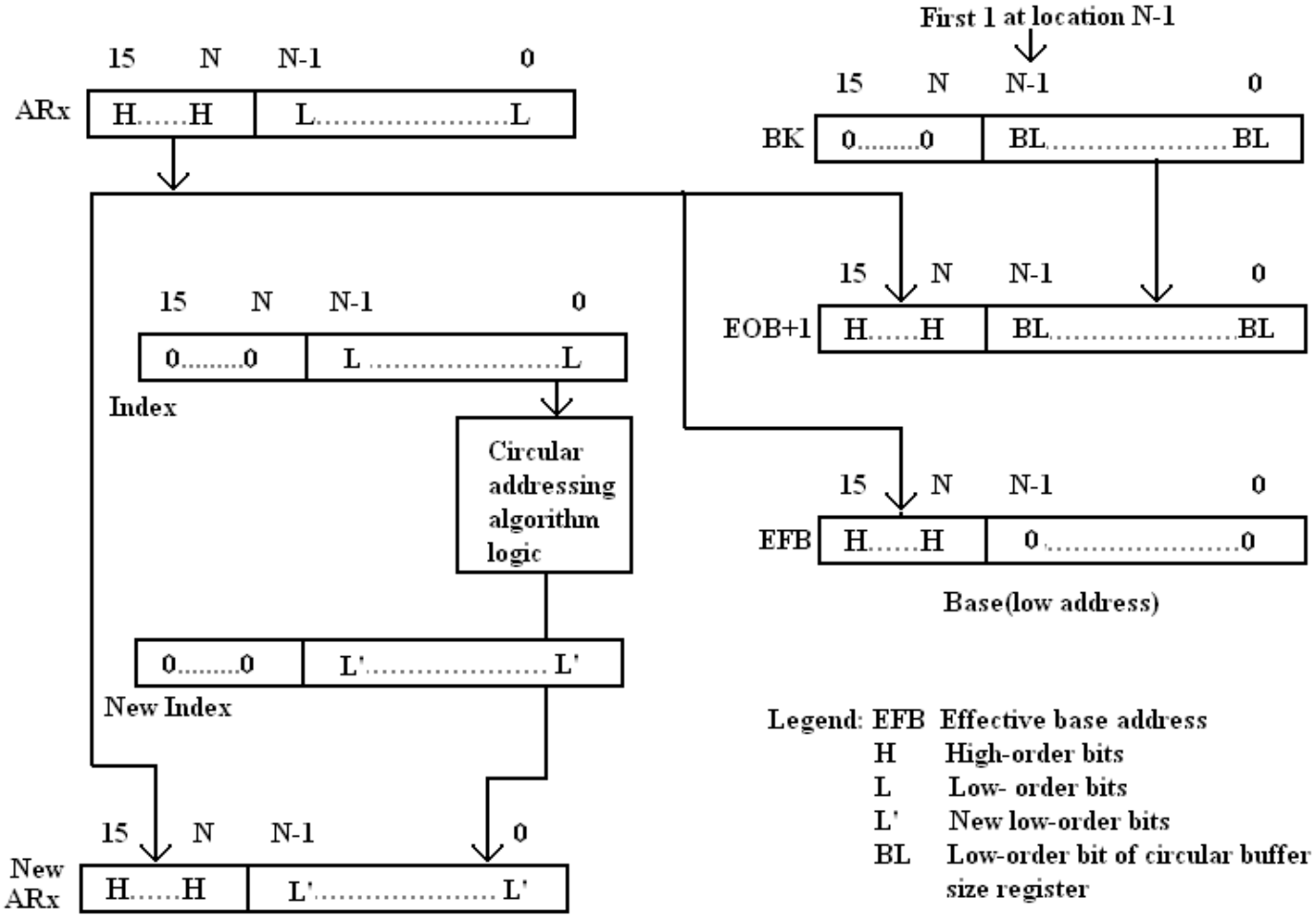
INDIRECT ADDRESSING OPTIONS WITH A SINGLE DATA –MEMORY OPERAND

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)

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.

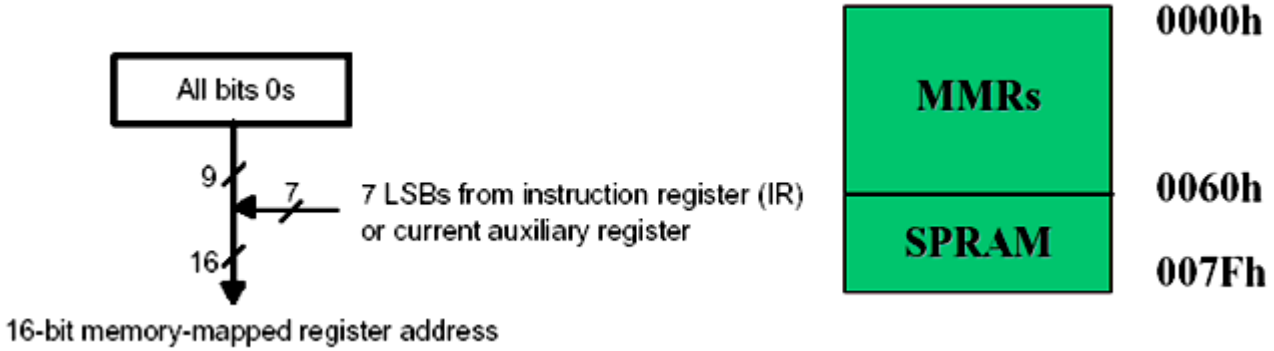
BLOCK DIAGRAM OF THE CIRCULAR ADDRESSING MODE FOR TMS320C54XX PROCESSORS



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

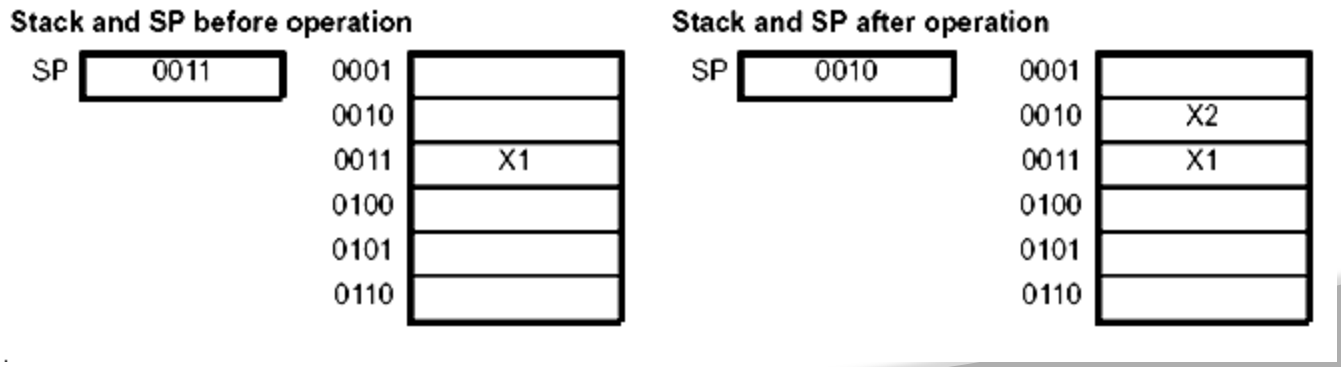
16 BIT MEMORY MAPPED REGISTER ADDRESS GENERATION



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

Values of stack &SP before and after operation

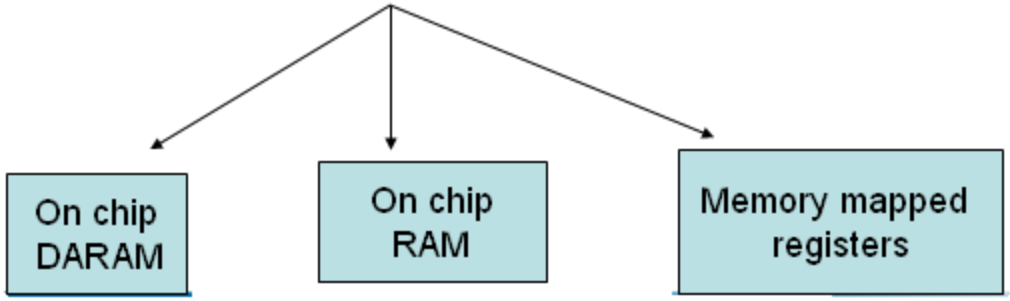


TMS320C54XX INSTRUCTIONS AND PROGRAMMING

- Load and Store Operations
- Arithmetic Operations
- Logical Operations
- Program-Control Operations
- Multiply Instruction
- Multiply and Accumulate Instruction
- Multiply and Subtract Instruction
- Multiply ,Accumulate , and Delay Instruction
- Repeat Instruction

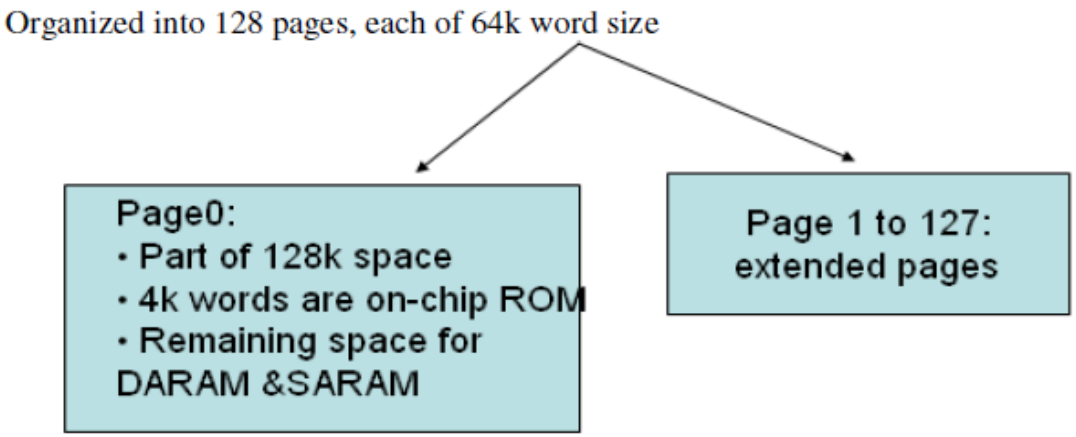
MEMORY SPACE OF TMS320C54XX PROCESSORS

- 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.



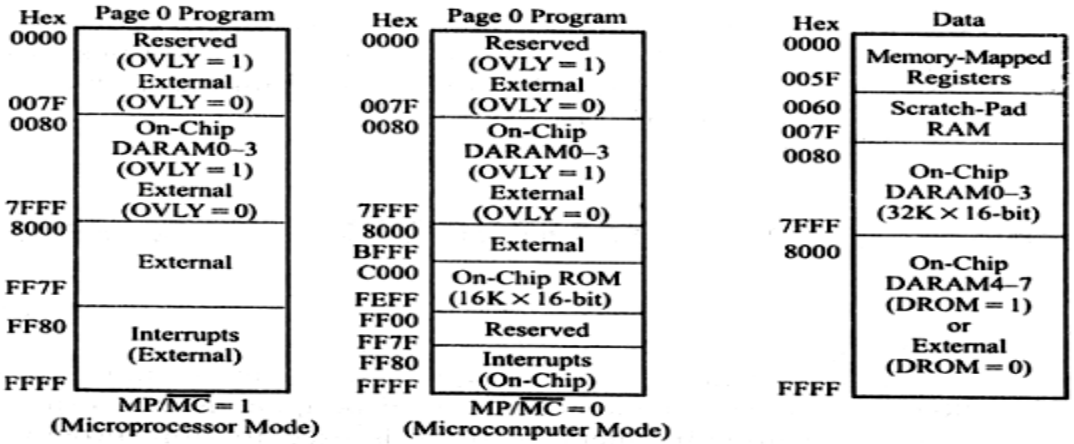
PROGRAM MEMORY

- To store program instructions &tables used in the execution of programs.
- Organized into 128 pages, each of 64k word size



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

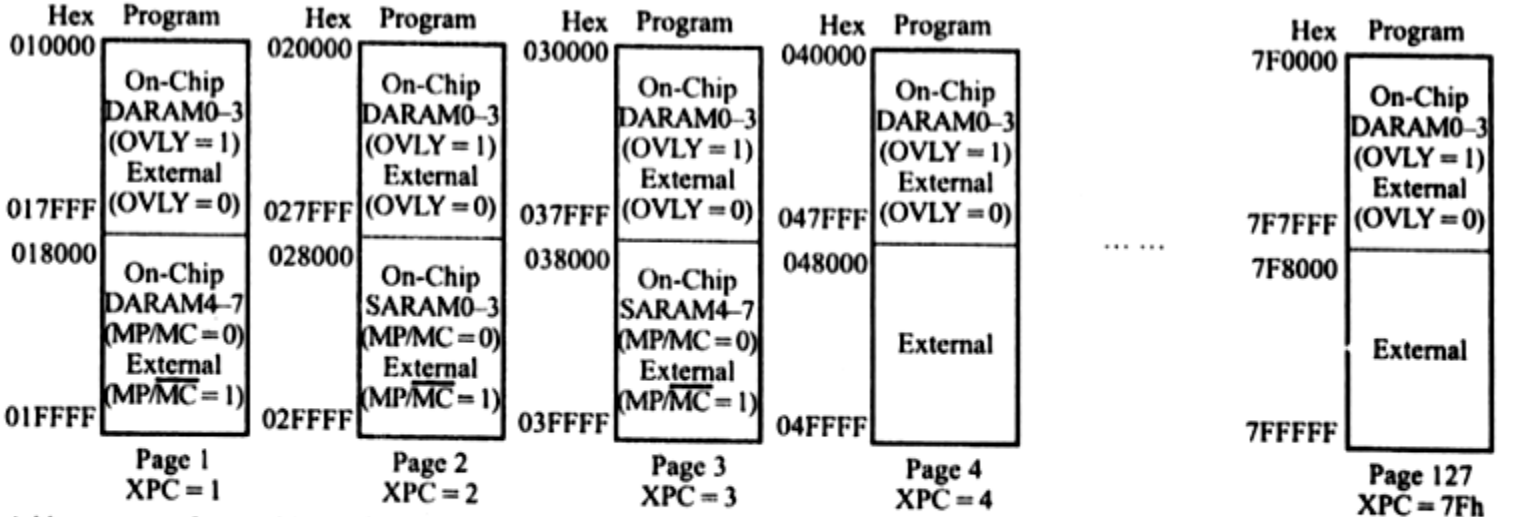


Address ranges for on-chip DARAM in data memory are:

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

(a)

MEMORY MAP FOR THE TMS320C5416 PROCESSOR



Address ranges for on-chip DARAM in program memory are:
DARAM4: 018000h–019FFFh; DARAM5: 01A000h–01BFFFh
DARAM6: 01D0000h–01DFFFh; DARAM7: 01E000h–01FFFFh

Address ranges for on chip SARAM in program memory are:
SARAM0: 028000h–029FFFh; SARAM1: 02A000h–02BFFFh
SARAM2: 02C000h–02DFFFh; SARAM3: 02E000h–02FFFFh
SARAM4: 038000h–039FFFh; SARAM5: 03A000h–03BFFFh
SARAM6: 03C000h–03DFFFh; SARAM7: 03E000h–03FFFFh

(b)

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

INTERRUPTS

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

INTERRUPTS

- It may also generated by software.
- Not all the interrupts are serviced by when they occur only those interrupts that are called non maskable are serviced when they occur.
- Other Interrupts which are called maskable interrupts are serviced only if they are enabled.
- There is also a priority to determine which interrupts gets serviced first if more than one interrupts occur simultaneously.

PIPELINE OPERATION of TMS320C54XX

- The C54xx DSP has a six-level deep instruction pipeline.
- The six stages of the pipeline are independent of each other,
- which allows overlapping execution of instructions.
- During any given cycle, from one to six different instructions can be active, each at a different stage of completion.

PIPELINE OPERATION OF TMS320C54XX

- The six levels and functions of the pipeline structure are:

Program Prefetch	• Program address bus (PAB) is loaded with the address of the next instruction to be fetched.
Program fetch	An instruction word is fetched from the program bus (PB) and loaded into the instruction register (IR). This completes an instruction fetch sequence that consists of this and the previous cycle.

PIPELINE OPERATION OF TMS320C54XX

Decode

- The contents of the instruction register (IR) are decoded to determine the type of memory access operation and the control sequence at the data-address generation unit (DAGEN) and the CPU.

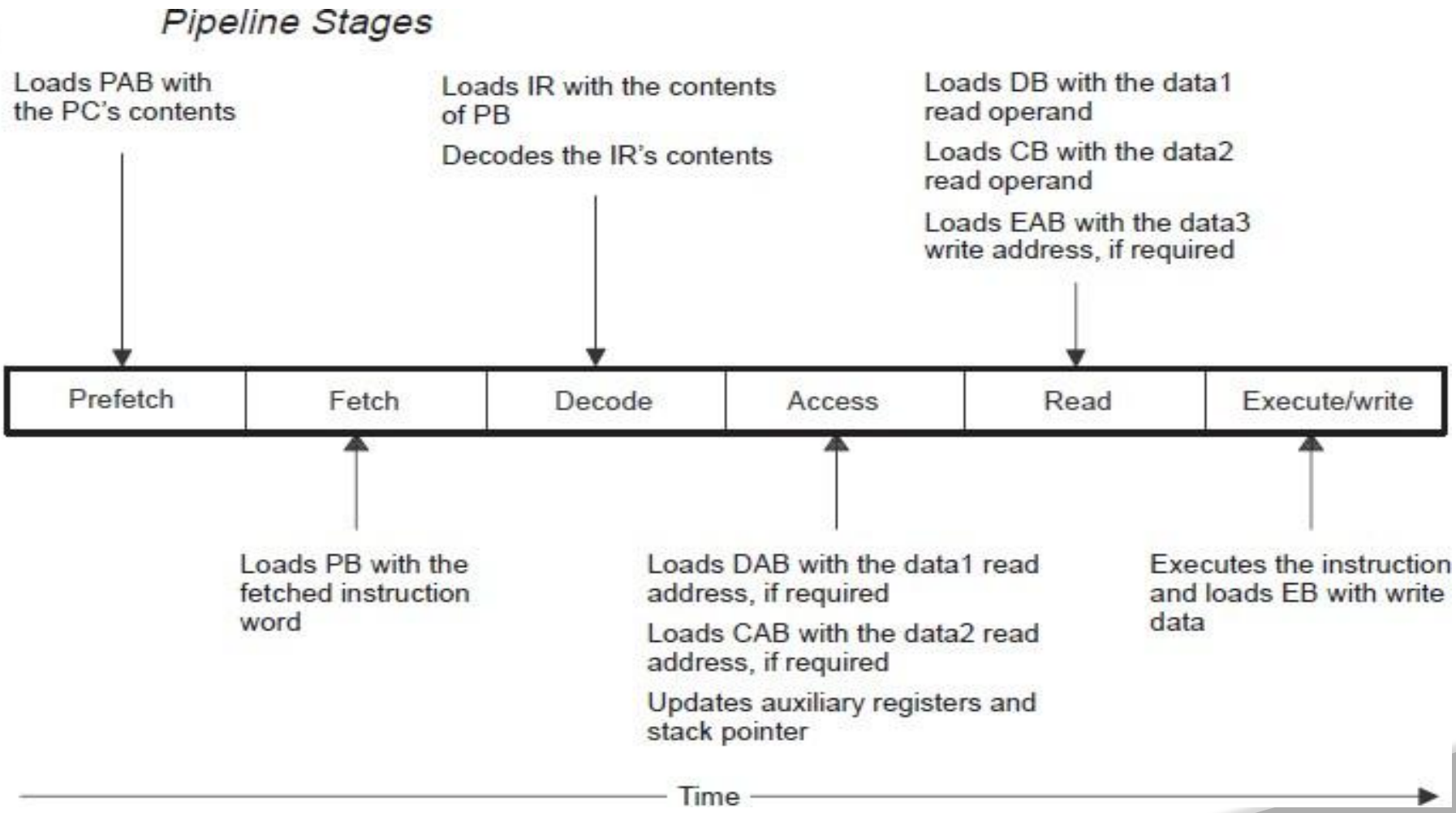
Access

- DAGEN outputs the read operand's address on the data address bus, DAB. If a second operand is required, the other data address bus, CAB, is also loaded with an appropriate address. Auxiliary registers in addressing mode and the stack pointer (SP) are also updated

PIPELINE OPERATION OF TMS320C54XX

Read	• The read data operand(s), if any, are read from the data buses, DB and CB. This completes the two-stage operand read sequence. At the same time, the two-stage operand write sequence begins. The data address of the write operand, if any, is loaded into the data write address bus (EAB).
Execute	• The operand write sequence is completed by writing the data using the data write bus (EB). The instruction is executed in this phase

PIPELINING STAGES



Cycle	Prefetch	Fetch	Decode	Access	Read	Execute & Write	AR3	A
1	LD						80	X
2	ADD	LD					80	X
3	STL	ADD	LD				80	X
4		STL	ADD	LD			81	X
5			STL	ADD	LD		82	1
6				STL	------	LD	82	0001h
7					STL	ADD	82	1001h
8						STL	82	1001h

ON CHIP PERIPHERALS

- General-purpose I/O pins: XF and BIO Timer
- Host port interface
- (HPI) Synchronous
- serial port Buffered
- serial port (BSP)
- Multichannel buffered serial port (McBSP)
- Time-division multiplexed (TDM) serial port
- Software-programmable wait-state generator
- Programmable bank-switching module

GENERAL-PURPOSE I/O

- The C54xx DSP offers general-purpose I/O through two dedicated pins that are software controlled. The two dedicated pins are the branch control input pin (BIO) and the external flag output pin (XF).
- BIO can be used to monitor the status of peripheral devices.
- XF can be used to signal external devices. The XF pin is controlled using software.
- It is driven high by setting the XF bit (in ST1) and is driven low by clearing the XF bit. The set status register bit (SSBX) and reset status register bit (RSBX) instructions can be used to set and clear XF, respectively.

SOFTWARE PROGRAMMABLE WAIT STATE GENERATOR

- Software Programmable wait state generator extends external bus cycle up to seven machine cycles to interface with slower off chip memory & devices./
- The Software wait state generator is incorporated without any external hardware.
- For off chip memory access from zero to seven wait states can be specified within the software wait state register.

HOST PORT INTERFACE

- The host port interface is an 8 bit parallel port that provides an interface with host processor.
- Information is exchanged between C54xx & host processor the C54xx on chip memory that is accessible to both C54xx & host processor.

HARDWARE TIMER

- The on-chip timer is a software-programmable timer that consists of three registers and can be used to periodically generate interrupts.
- The timer resolution is the CPU clock rate of the processor.
- The high dynamic range of the timer is achieved with a 16-bit counter with a 4-bit prescaler.

- **Timer Registers**:-

  The on-chip timer consists of three memory-mapped registers (TIM, PRD, and TCR).

TIMER REGISTERS

Timer 0 Address	Timer 1 Address (C5402 only)	Register	Description
0024h	0030h	TIM	Timer register
0025h	0031h	PRD	Timer period register
0026h	0032h	TCR	Timer control register

- Timer register (TIM):The 16-bit memory-mapped timer register (TIM) is loaded with the period register (PRD) value and decremented.
- Timer period register (PRD): The 16-bit memory-mapped timer period register (PRD) is used to reload the timer register (TIM).
- Timer control register (TCR):The 16-bit memory-mapped timer control register (TCR) contains the control and status bits of the timer.

UNIT-4

Memory space organization, external bus interfacing signals, memory interface, parallel I/O interface, programmed I/O, interrupts and I/O, direct memory access (DMA).

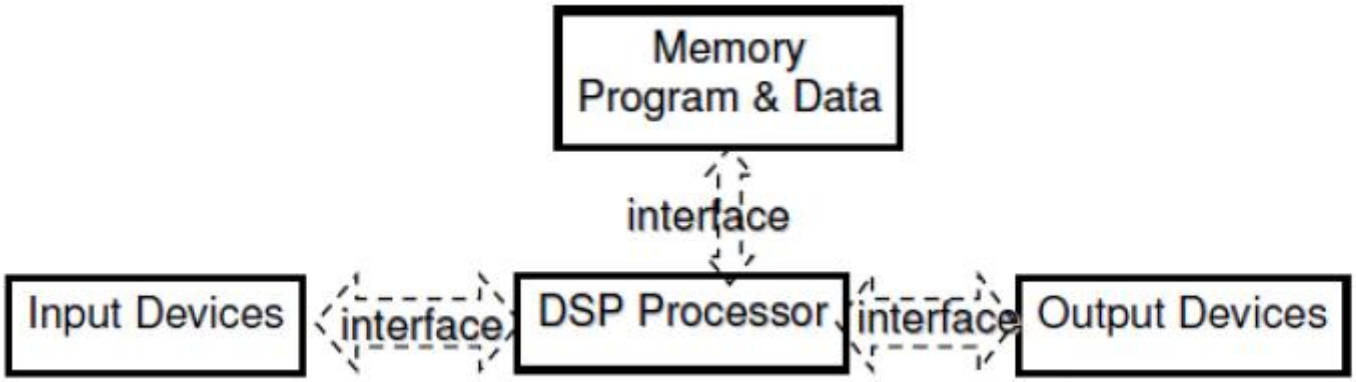
UNIT-4 CLO'S

CLO 14	Understand the significance of memory space organization.
CLO 15	Analyze external bus interfacing signals.
CLO 16	Explain about parallel I/O interface, programmed I/O.
CLO 17	Understand the significance of Interrupts and Direct Memory Access.

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 use DSP signals to generate the appropriate signals for setting up communication with the memory.

INTRODUCTION

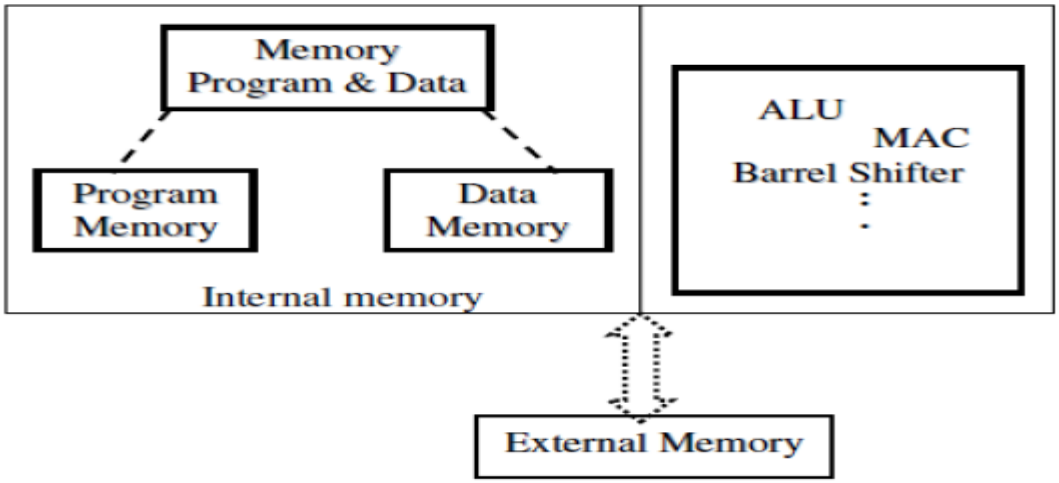


MEMORY SPACE ORGANIZATION

- 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. The actual 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.
- The On- Chip Memory are faster than External Memory. There are no interfacing requirements. 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.

MEMORY SPACE ORGANIZATION

- The purpose of such memory is to hold Program / Code / Instructions, to hold constant data such as filter coefficients / filter order, also to hold trigonometric tables / kernels of transforms employed in an algorithm

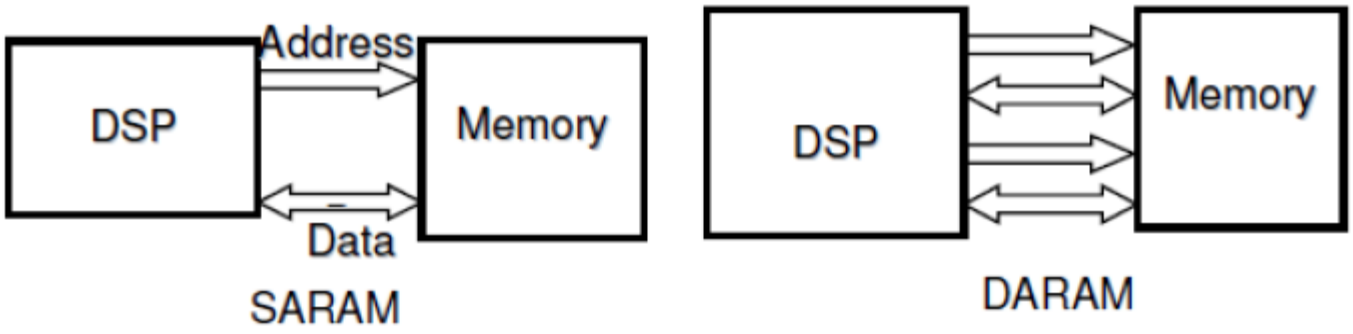


EFFICIENT DATA/PROGRAM FLOW

•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 memory space. 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) 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. The DSP can thus access two memory locations simultaneously.

EFFICIENT DATA/PROGRAM FLOW



DATA MEMORY

- 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 is enabled 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.

DATA MEMORY

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 third bit is DROM. It enables on-chip DARAM 4-7 to be mapped into data space. If this bit is 0, on-chip DARAM 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 on chip / off-chip.

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
- 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.

EXTERNAL BUS INTERFACING SIGNALS

- 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.
- 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.

EXTERNAL BUS INTERFACING SIGNALS

- 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.

EXTERNAL BUS INTERFACING SIGNALS

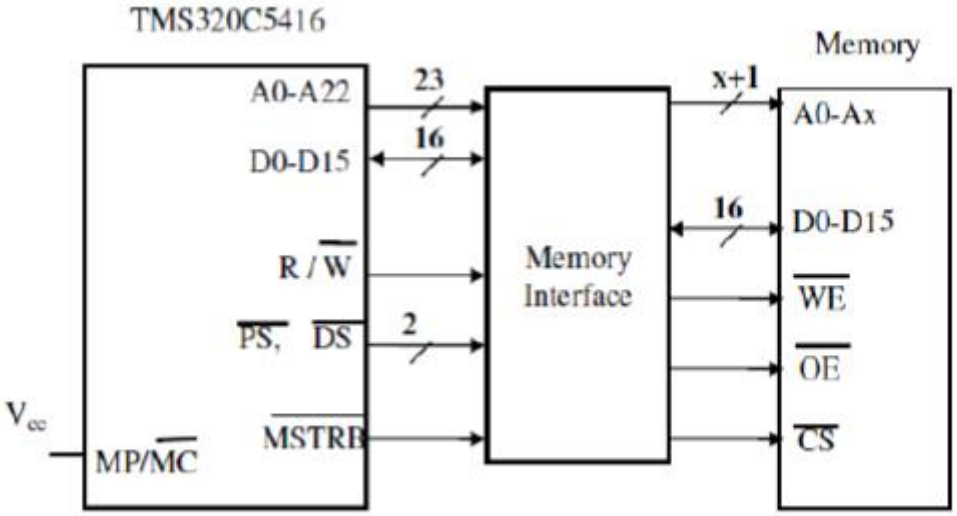
1	A0-A19	20 bit Address Bus
2	D0-D15	16 bit Data Bus
3	$\overline{DS}$	Data Space Select
4	$\overline{PS}$	Program Space Select
5	$\overline{IS}$	I/O Space Select
6	R / $\overline{W}$	Read/Write Signal
7	$\overline{MSTRB}$	Memory Strobe
8	$\overline{IOTRB}$	I/O Strobe

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

MEMORY INTERFACE

- Typical signals in a memory device are address bus to carry address of referred memory location. Data bus carries data to or from referred memory location. Chip Select Signal selects one or more memory ICs among many memory ICs in the system. Write Enable enables writing of data available on data bus to a memory location. Output Enable signal enables the availability of data from a memory location onto the data bus.
- The address bus is unidirectional, carries address into the memory IC 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.

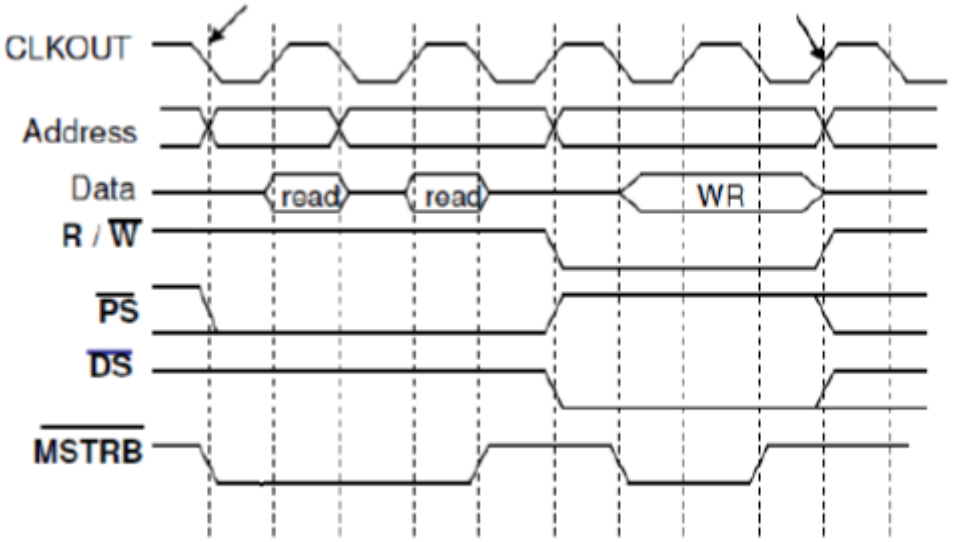
MEMORY INTERFACE



TIMING SEQUENCE FOR EXTERNAL MEMORY ACCESS

- The timing sequence of memory access is shown. 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.

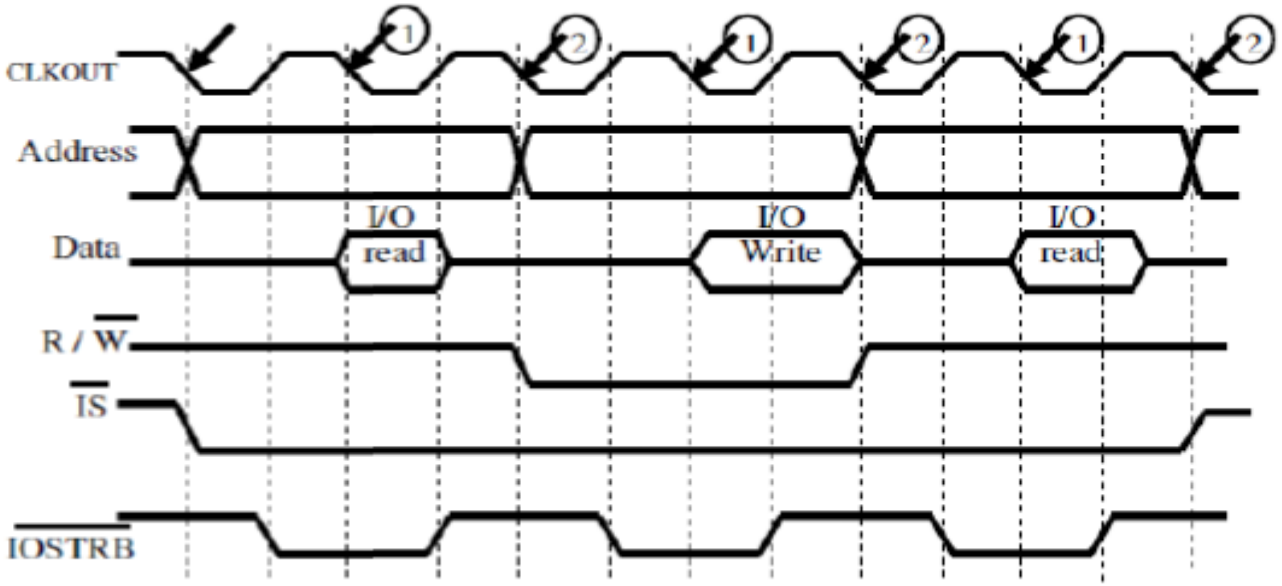
TIMING SEQUENCE FOR EXTERNAL MEMORY ACCESS



PARALLEL I/O INTERFACE

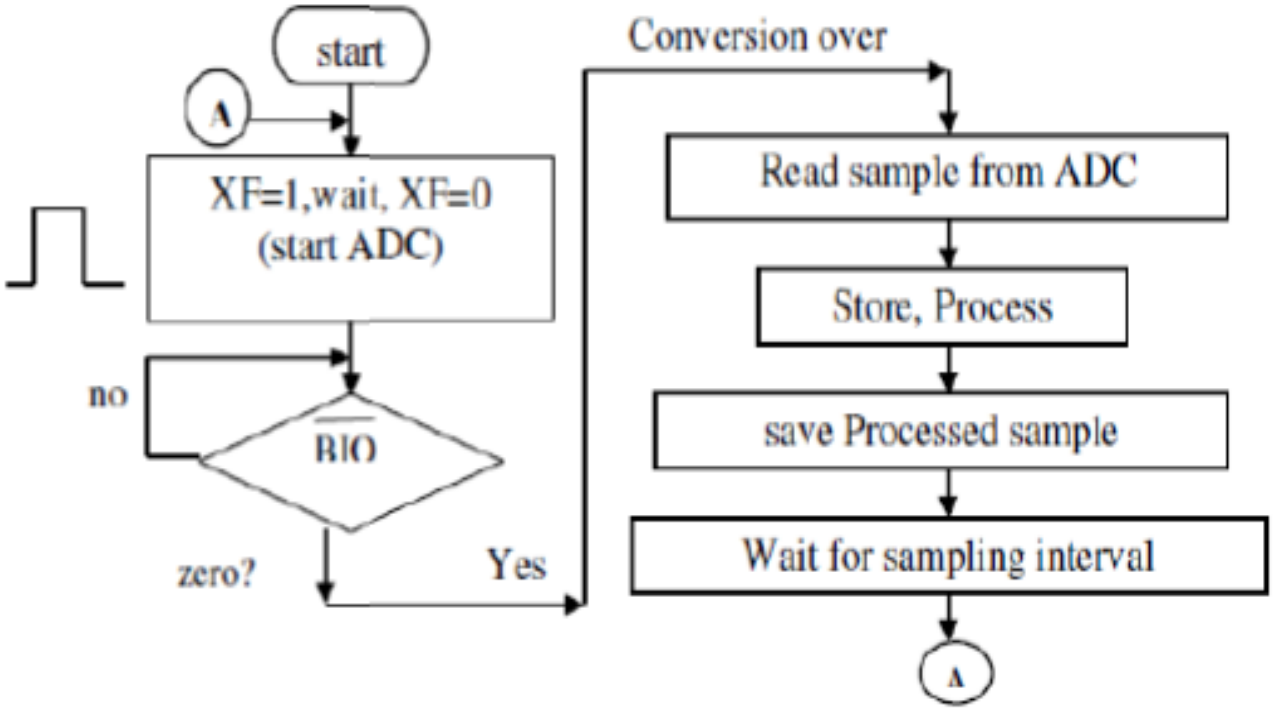
- I/O devices are interfaced to DSP using unconditional I/O mode, programmed I/O mode or interrupt I/O mode. Unconditional I/O does not require any handshaking signals. DSP assumes the readiness of the I/O and transfers the data with its own speed. Programmed I/O requires handshaking signals.
- DSP waits 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.

PROGRAMMED I /O INTERFACE



•The timing diagram in the case of programmed I/O is shown in figure. I/O strobe and I/O space select are issued by the DSP. Two clock cycles each are required for I/O read and I/O write operations.

PROGRAMMED I /O FLOWCHART



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.

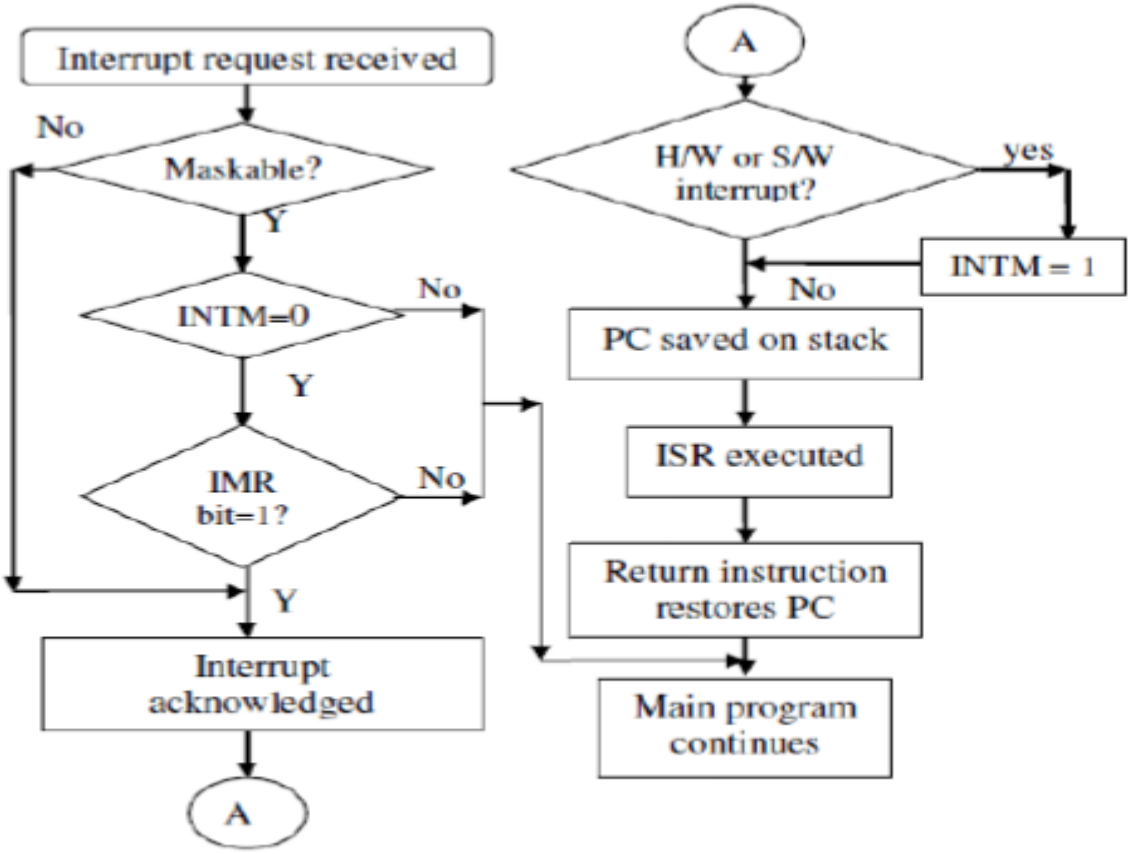
INTERRUPT I/O

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

INTERRUPT I/O

- 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 registers are Memory Mapped Registers.
- One flag, Global enable bit (INTM), in ST1 register is used to enable or disable all interrupts globally. If INTM is zero, all unmasked interrupts are enabled. If it is one, all maskable interrupts are disabled.

INTERRUPT I/O FLOWCHART



DIRECT MEMORY ACCESS (DMA) OPERATION

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

DIRECT MEMORY ACCESS (DMA) OPERATION

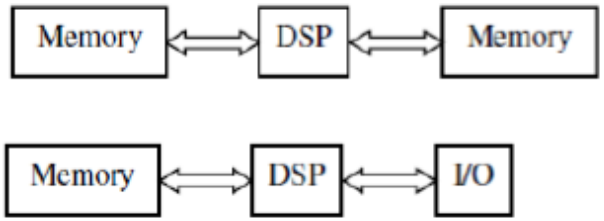


Fig. 7.11: Data transfer with intervention by DSP

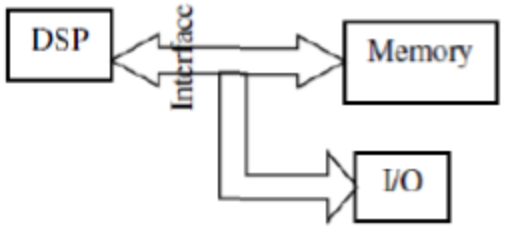


Fig. 7.10: Interface between DSP and external devices

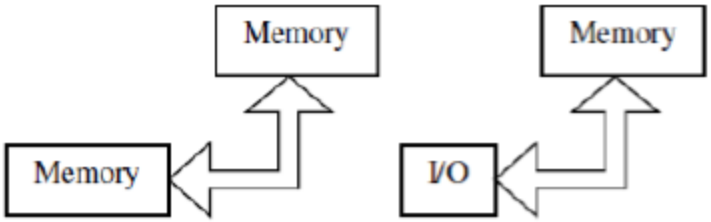
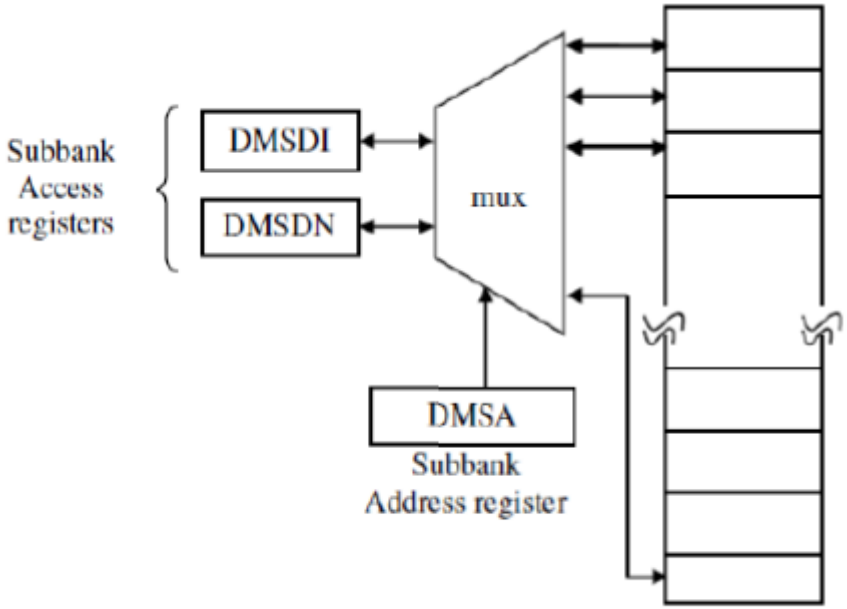


Fig. 7.12: data transfer without intervention by DSP

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.

UNIT-5

The Q-notation, convolution, correlation, FIR filters, IIR filters, interpolation filters, decimation filters, an FFT algorithm for DFT filters computation of the signal spectrum.

UNIT-5 CLO'S

CLO 18	Understand the basic concepts of convolution and correlation.
CLO 19	Compare the characteristics of IIR and FIR filters.
CLO 20	Analyze the concepts of interpolation and decimation filters.

Q-NOTATION

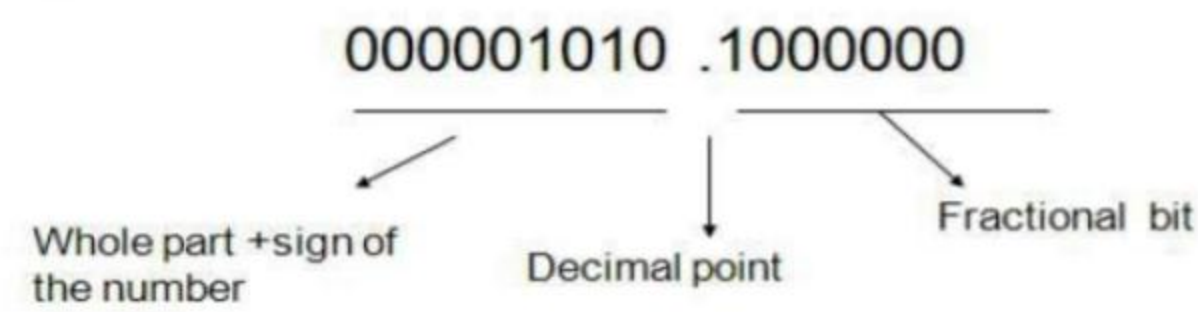
• DSP algorithm implementations deal with signals and coefficients. To use a fixed point DSP device efficiently, one must consider representing filter coefficients and signal samples using fixedpoint2's complement representation.

Ex: N=16, Range: -2N-1 to +2N-1 -1(-32768 to 32767).

• Typically, filter coefficients are fractional numbers. To represent such numbers, the Q-notation has been developed.

• The Q-notation specifies the number of fractional bits.

Ex: Q7



Q-NOTATION

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

$b_{15}.....b_1b_0$

$N= - b_{15}+ b_{14}2^{-1}+.....+b_02^{-15}$

Range:-1 to $1- 2^{-15}$

FIR FILTERS

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

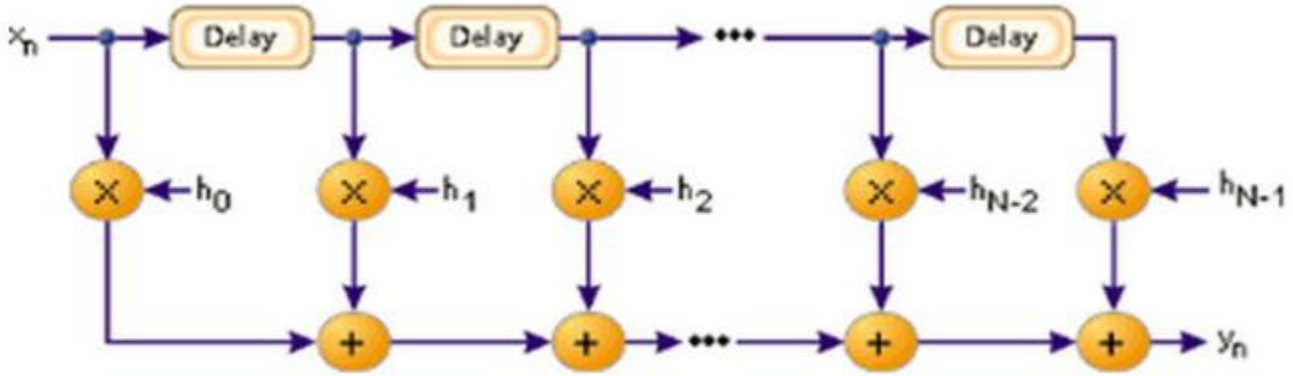
$$y[n] = \sum_{m=0}^{m=N-1} h(m)\,x(n-m)$$

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

- In digital signal processing, an FIR is a filter whose impulse response is of finite period, as a result of it settles to zero in finite time.
- Filters are signal conditioners and function of each filter is, it allows an AC components and blocks DC components.
- Due to efficiency and simplicity of the FIR filter, most commonly window method is used.
- The other method sampling frequency method is also very simple to use, but there is a small attenuation in the stop band.

LOGICAL STRUCTURE OF FIR FILTER

- FIR filter is used to implement almost any type of digital frequency response. Usually these filters are designed with a multiplier, adders and a series of delays to create the output of the filter.
- The following figure shows the basic FIR filter diagram with N length. The result of delays operates on input samples. The values of hk are the coefficients which are used for multiplication. So that the o/p at a time and that is the summation of all the delayed samples multiplied by the appropriate coefficients.



Logical Structure of FIR Filter

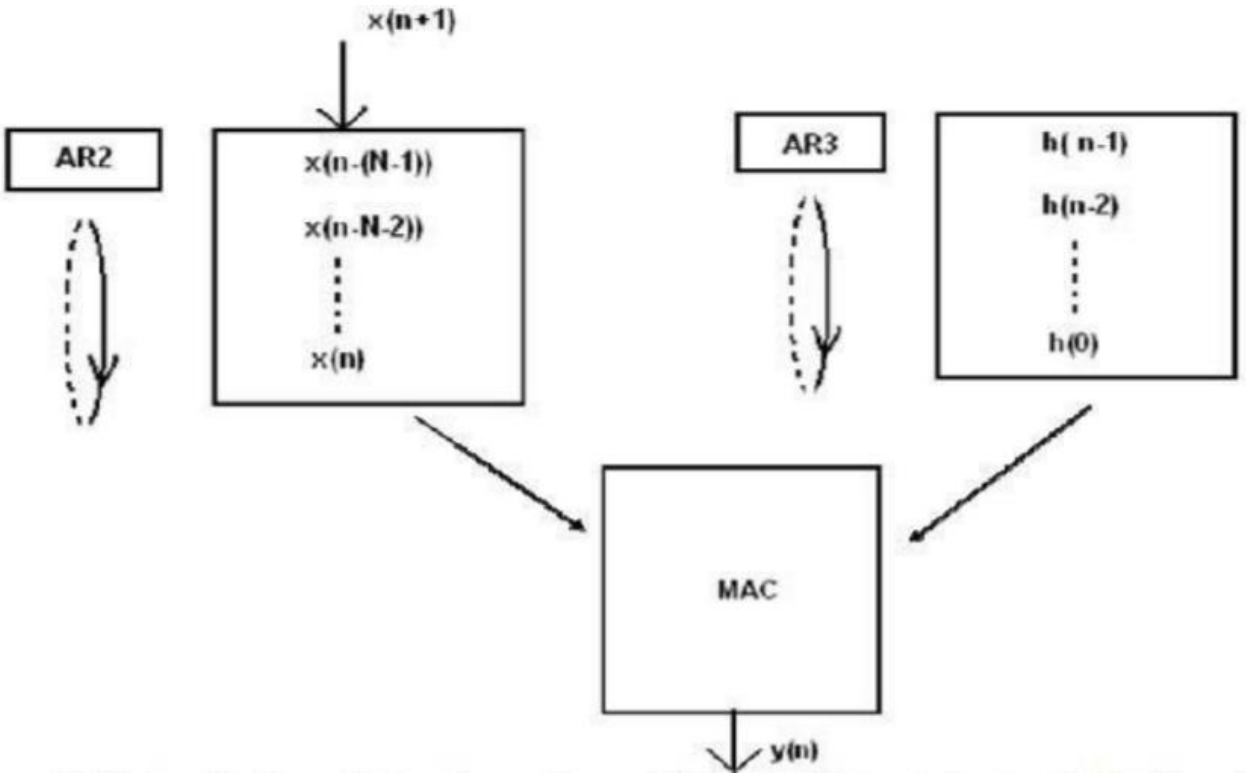
FIR FILTERS

•The implementation requires signal delay for each sample to compute the next output, y(n+1), is given as y(n+1)=h(N-1)x(n-(N-2))+h(N-2)x(n-(N-3))+ ...h(1)x(n)+h(0)x(n+1) Figure shows the memory organization for the implementation of the filter.

•The filter Coefficients and the signal samples are stored in two circular buffers each of a size equal to the filter. AR2 is used to point to the samples and AR3 to the coefficients.

• In order to start with the last product, the pointer register AR2 must be initialized to access the signal sample x(2-(N-1)), and the pointer register AR3 to access the filter coefficient h(N-1). As each product is computed and added to the previous result, the pointers advance circularly. At the end of the computation, the signal sample pointer is at the oldest sample, which is replaced with the newest sample to proceed with the next output computation.

FIR FILTERS



Organization of signal samples and filter coefficients in circular buffers for a FIR filter implementation.

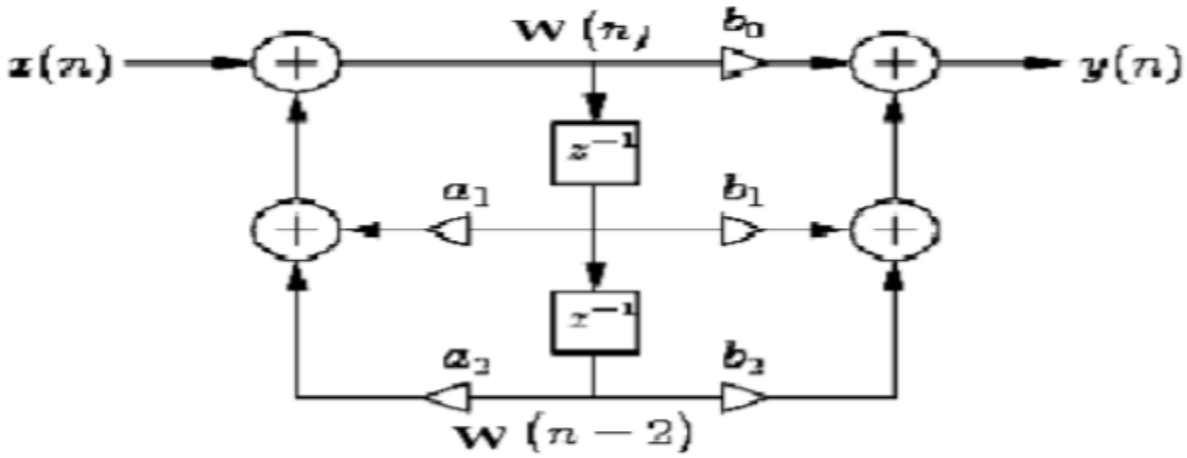
IIR FILTERS

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

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}}$$

IIR FILTERS

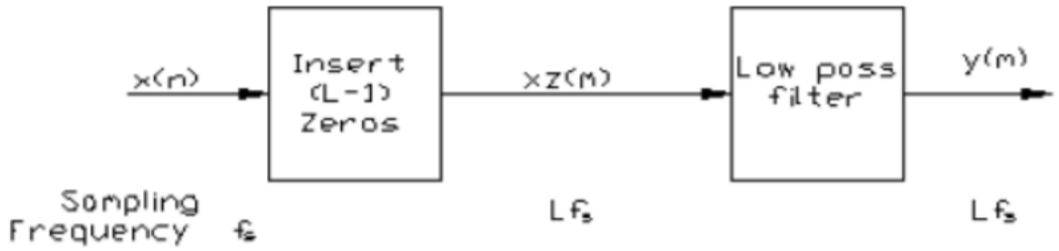
$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}}$$



$$w(n) = x(n) + a_1 w(n-1) + a_2 w(n-2)$$
$$y(n) = b_0 w(n) + b_1 w(n-1) + b_2 w(n-2)$$

INTERPOLATION FILTERS

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



INTERPOLATION FILTERS

Example:

X(n) = [0 2 4 6 8 10]	;input sequence
Xz(n) = [0 0 2 0 4 0 6 0 8 0 10 0]	;zero inserted sequence
h(n) = [0.5 1 0.5]	;impulse sequence
Y(n) = [0 0 1 2 3 4 5 6 7 8 9 10 5 0]	;interpolated sequence y(n)

The kind of interpolation carried out in the examples is called linear interpolation because the convolving sequence h(n) is derived based on linear interpolation of samples.

Further, in this case, the h(n) selected is just a second-order filter and therefore uses just two adjacent samples to interpolate a sample.

A higher-order filter can be used to base interpolation on more input samples.

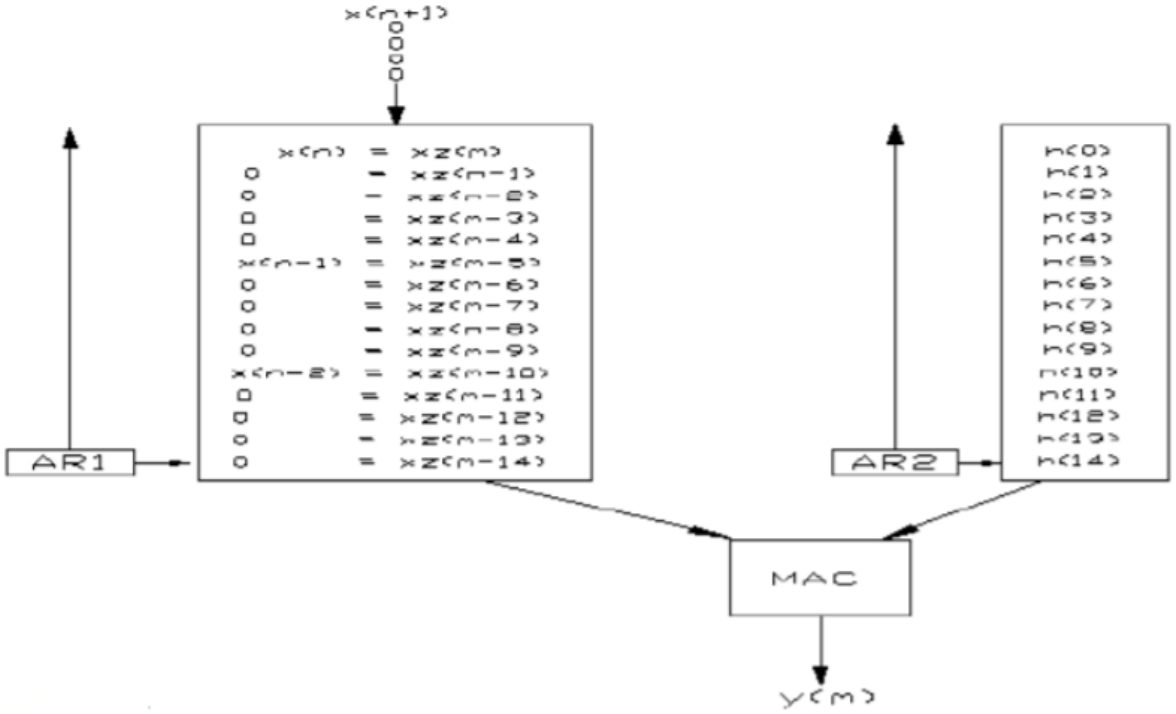
INTERPOLATION FILTERS

- To implement an ideal interpolation. Figure shows how an interpolating filter using a 15-tap FIR filter and an interpolation factor of 5 can be implemented. In this example, each incoming samples is followed by four zeros to increase the number of samples by a factor of 5.
- The interpolated samples are computed using a program similar to the one used for a FIR filter implementation.
- One drawback of using the implementation strategy depicted in Figure is that there are many multiplies in which one of the multiplying elements is zero. Such multiplies need not be included in computation if the computation is rearranged to take advantage of this fact.

INTERPOLATION FILTERS

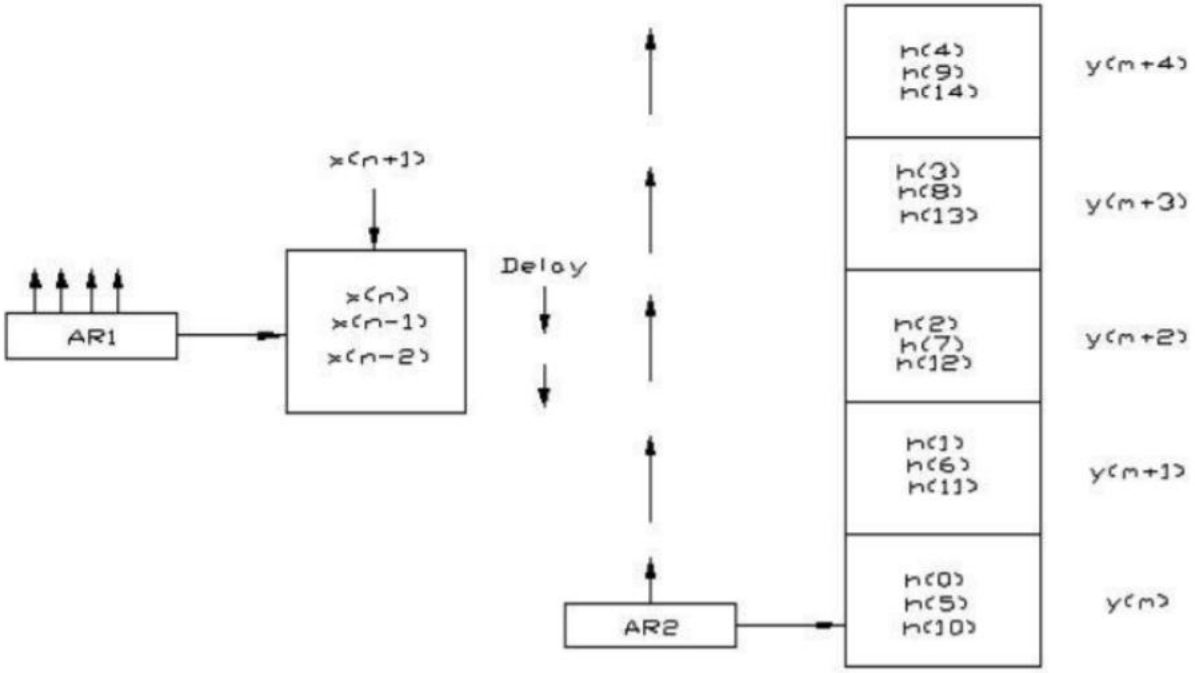
- One such scheme, based on generating what are called poly-phase sub-filters, is available for reducing the computation. For a case where the number of filter coefficients N is a multiple of the interpolating factor L, the scheme implements the interpolation filter using the equation.
- A scheme that uses poly-phase sub-filters to implement the interpolating filter using the 15-tap FIR filter and an interpolation factor of 5. In this implementation, the 15 filter taps are arranged as shown and divided into five 3-tap sub filters.
- The input samples x(n), x(n-1) and x(n-2) are used five times to generate the five output samples. This implementation requires 15 multiplies as opposed to 75 in the direct implementation of Figure

INTERPOLATION FILTERS



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

INTERPOLATION FILTERS



A scheme that uses poly-phase sub-filters to implement the interpolating filter

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

$$y(m+i) = \sum_{K=0}^{N/L-1} h(KL+i)x(n-K)$$

Where i = 0,1,2,.... (L-1) and m = nL.

DECIMATION FILTERS

- A decimation filter is used to decrease the sampling rate. The decrease in sampling rate can be achieved by simply dropping samples. For instance, if every other sample of a sampled sequence is dropped, the sampling the rate of the resulting sequence will be half that of the original sequence.
- The problem with dropping samples is that the new sequence may violate the sampling theorem, which requires that the sampling frequency must be greater than two times the highest frequency contents of the signal.
- To circumvent the problem of violating the sampling theorem, the signal to be decimated is first filtered using a low pass filter.

DECIMATION FILTERS

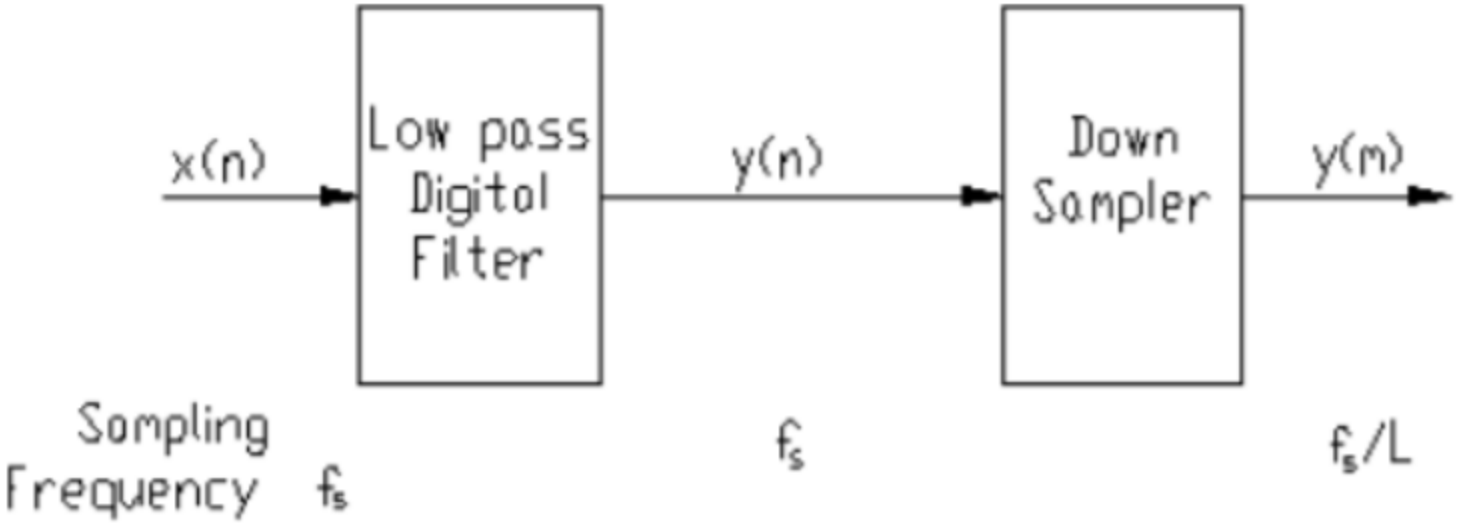
•The cutoff frequency of the filter is chosen so that it is less than half the final sampling frequency.

• The filtered signal can be decimated by dropping samples. In fact, the samples that are to be dropped need not be computed at all.

•Thus, the implementation of a decimator is just a FIR filter implementation in which some of the outputs are not calculated.

• Figure shows a block diagram of a decimation filter. Digital decimation can be implemented as depicted in Figure for an example of a decimation filter with decimation factor of 3. It uses a low pass FIR filter with 5 taps. The computation is similar to that of a FIR filter. However, after computing each output sample, the signal array is delayed by three sample intervals by bringing the next three samples into the circular buffer to replace the three oldest samples.

DECIMATION FILTERS

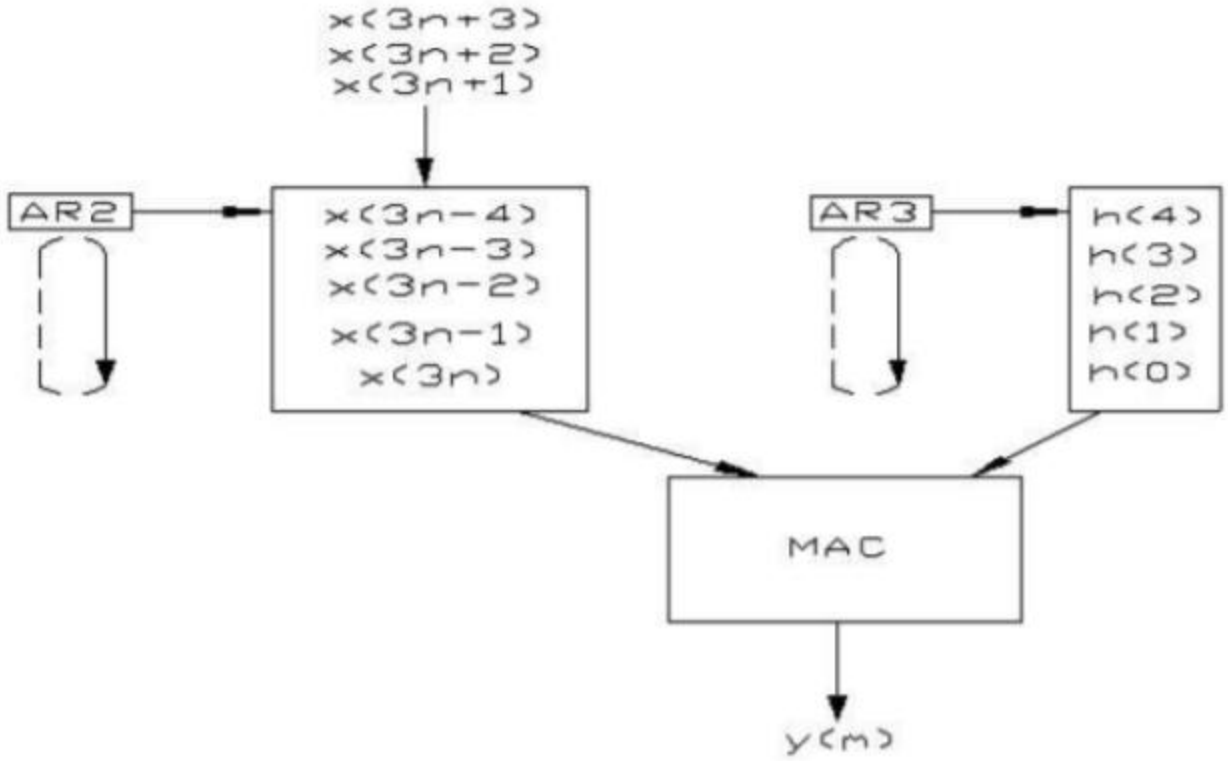


Decimation process

$$y(m) = y(nL) = \sum_{K=0} h(K)x(nL-K):$$

Where n = 0,1,2,.....
L=decimation factor
N=filter size

DECIMATION FILTER



Implementation of decimation filter

INTRODUCTION TO FFT

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

INTRODUCTION

Fig 6.1: DSP System

AN FFT ALGORITHM FOR DFT COMPUTATION

- As DFT / IDFT are part of signal processing system, there is a need for fast computation of DFT / IDFT. There are algorithms available for fast computation of DFT/ IDFT. There are referred to as Fast Fourier Transform (FFT) algorithms.
- There are two FFT algorithms: Decimation-In-Time FFT (DITFFT) and Decimation-In-Frequency FFT (DIFFFT). The computational complexity of both the algorithms are of the order of log2(N). From the hardware / software implementation viewpoint the algorithms have similar structure throughout the computation. In-place computation is possible reducing the requirement of large memory locations. The features of FFT are tabulated in the table 6.2.

AN FFT ALGORITHM FOR DFT COMPUTATION

Table 6.2: Features of FFT		
Features	**DITFFT**	**DIFFFT**
Sequence which is decimated by factor 2	Time domain sequence	DFT sequence
Input sequence	Bit reversed order	Proper order
Output sequence	Proper order	Bit reversed order

EXAMPLE OF COMPUTATION OF 2 POINT DFT

$$X(0) = x(0)W_2^0 + x(1)W_2^0 = x(0) + x(1)$$
$$X(1) = x(0)W_2^0 + x(1)W_2^1 = x(0) - x(1) \quad (6.3)$$

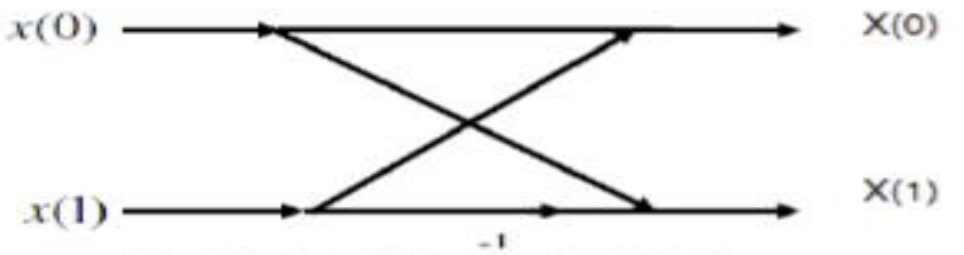


Fig 6.2: Signal Flow graph for N=2

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

BUTTERFLY STRUCTURE

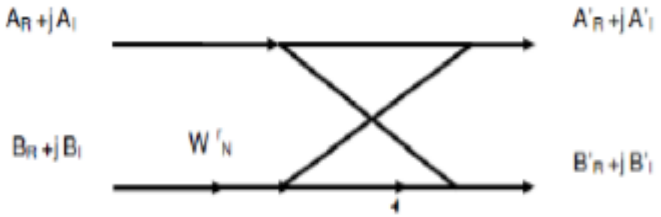


Fig 6. 3: Butterfly structure for N point DITFFT Computation

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

$$A'_R + jA'_I = A_R + jA_I + (B_R + jB_I)(W^r_R + jW^r_I) \quad (6.4)$$

$$B'_R + jB'_I = A_R + jA_I - (B_R + jB_I)(W^r_R + jW^r_I) \quad (6.5)$$

BUTTERFLY STRUCTURE

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

$$\left.\begin{aligned} A'_R &= A_R + B_R W_R^r - B_I W_I^r \\ A'_I &= A_I + B_I W_R^r + B_R W_I^r \\ B'_R &= A_R - B_R W_R^r + B_I W_I^r \\ B'_I &= A_I - B_I W_R^r - B_R W_I^r \end{aligned}\right\} \qquad (6.6)$$

SIGNAL FLOW GRAPH OF 8 POINT DITFFT COMPUTATION

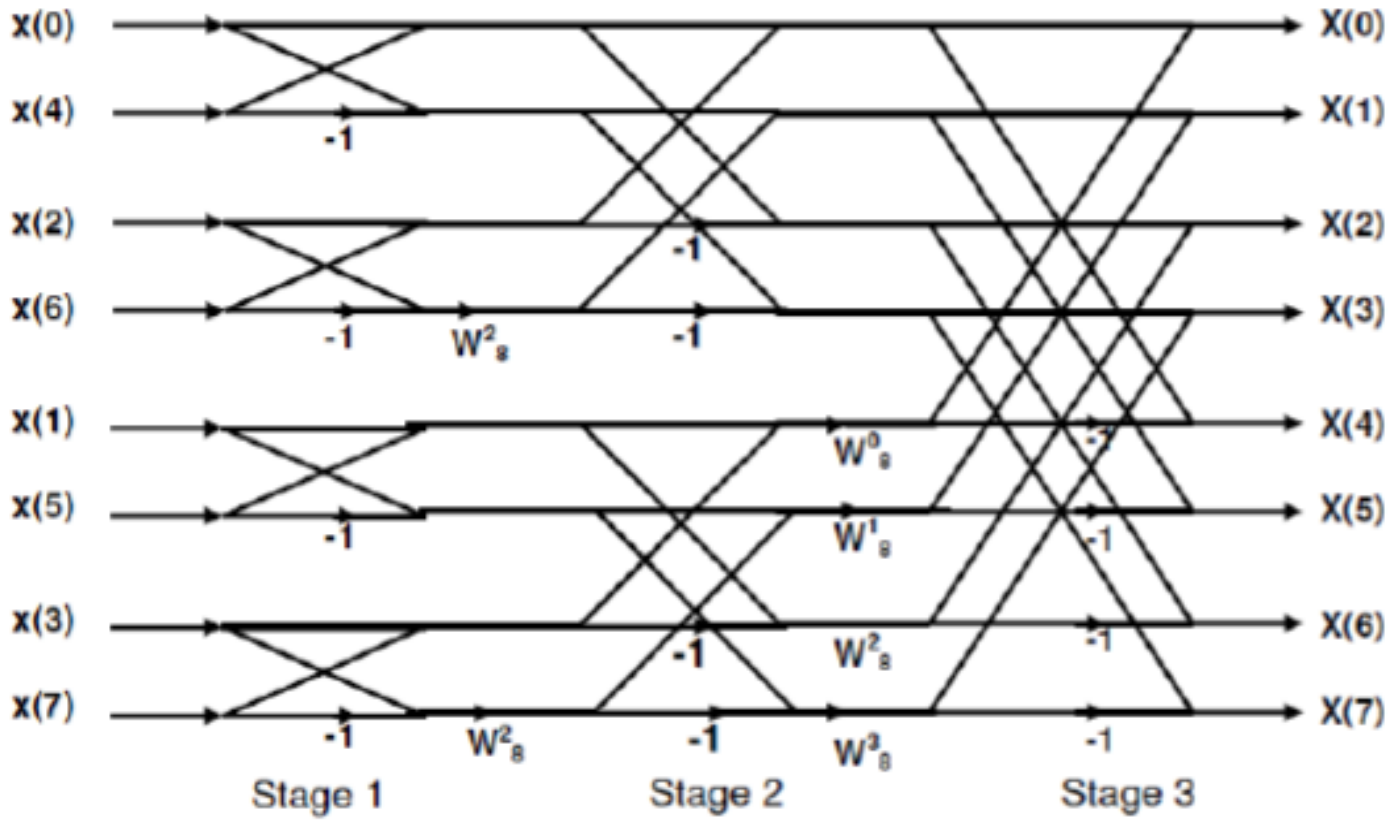


Fig 6.4. Signal flow graph of 8 point DITFFT Computation

SPECTRUM OF x(n)

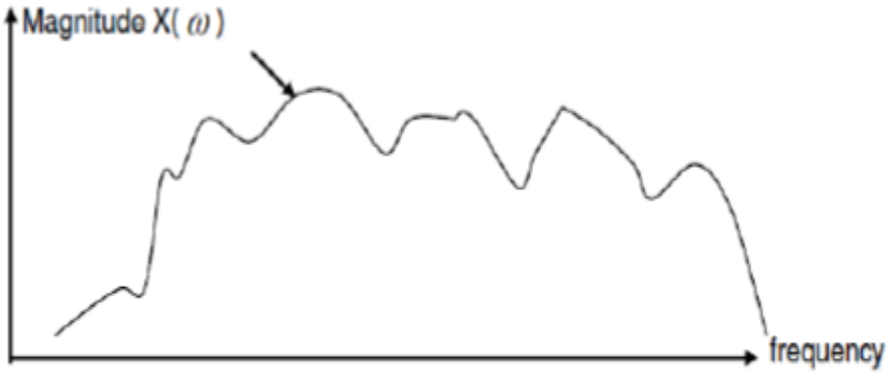


Fig 6.5: Spectrum of x(n)

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

•possible.

SPECTRUM

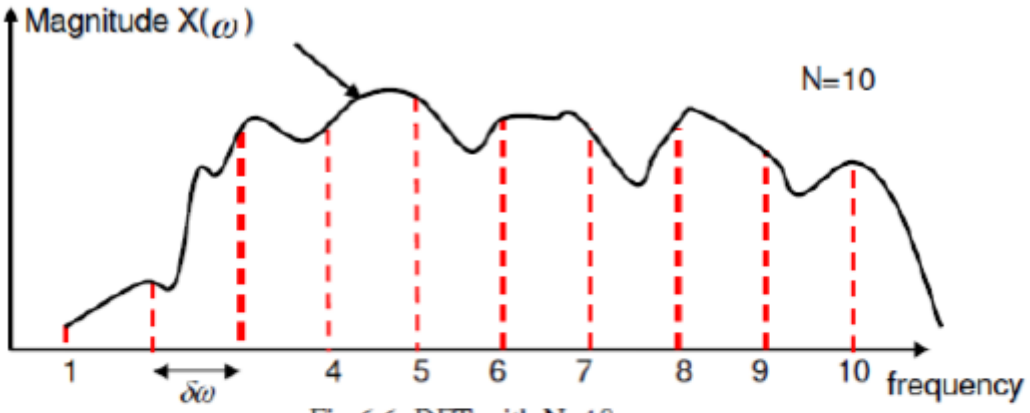


Fig 6.6: DFT with N=10

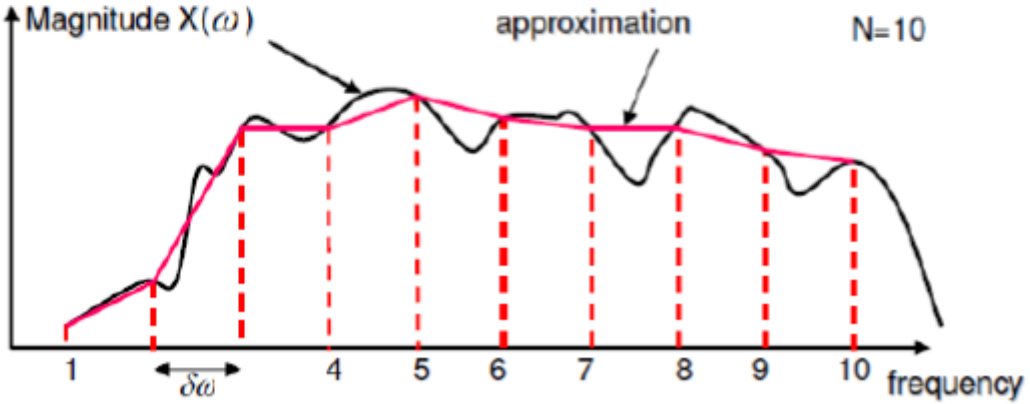
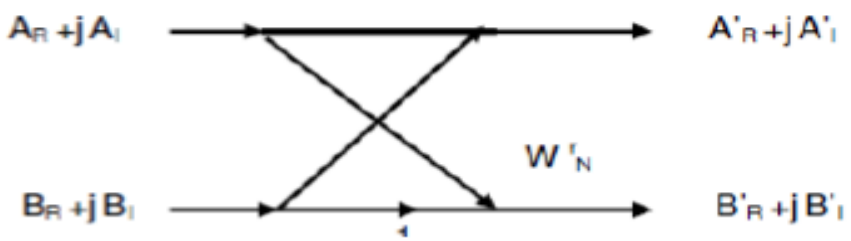


Fig 6.7: N=16 point DFT of x(n)

PROBLEM

- **Problem** : What minimum size FFT must be used to compute a DFT of 40 points? What must be done to samples before the chosen FFT is applied? What is the frequency resolution achieved?
- **Solution**: Minimum size FFT for a 40 point sequence is 64 point FFT. Sequence is extended to 64 by appending additional 24 zeros. The process improves frequency resolution from

$$\delta\omega = 2\pi/40 \text{ to } \delta\omega = 2\pi/64 \quad \text{(P6.1)}$$



OVERFLOW AND SCALING

- In any processing system, number of bits per data in signal processing is fixed and it is limited by the DSP processor used. Limited number of bits leads to overflow and it results in erroneous answer. InQ15 notation, the range of numbers that can be represented is -1 to 1. If the value of a number exceeds these limits, there will be underflow / overflow. Data is scaled down to avoid overflow.
- However, it is an additional multiplication operation. Scaling operation is simplified by selecting scaling factor of 2^-n. And scaling can be achieved by right shifting data by n bits. Scaling factor is defined as the reciprocal of maximum possible number in the operation. Multiply all the numbers at the beginning of the operation by scaling factor so that the maximum number to be processed is not more than 1. In the case of DITFFT computation

OVERFLOW AND SCALING - EXAMPLE

$$A_I' = A_I + B_I W_R^r + B_R W_I^r$$
$$= A_I + B_I \cos\theta + B_R \sin\theta \qquad (6.7)$$
$$\text{where } \theta = 2\pi kn/N$$

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

$$\frac{dA_I}{\partial\theta} = -B_I \sin\theta + B_R \cos\theta = 0$$
$$\Rightarrow B_I \sin\theta = B_R \cos\theta \qquad (6.8)$$
$$\Rightarrow \tan\theta = B_R/B_I$$
$$\therefore \sin\theta = \frac{B_R}{\sqrt{B_R^2 + B_I^2}} \qquad \text{Similarly,} \quad \cos\theta = \frac{B_I}{\sqrt{B_R^2 + B_I^2}}$$

Substituting them in eq(6.7).

$$A_I^{'} = A_I + \sqrt{B_R^2 + B_I^2}$$
$$A_{I,\max}^{'} = 1 + \sqrt{2} = 2.414$$

•Thus scaling factor is 1/2.414=0.414. A scaling factor of 0.4 is taken so that it can be implemented by shifting the data by 2 positions to the right. The symbolic representation of Butterfly Structure is shown in fig. 6.8. The complete signal flow graph with scaling factor is shown in fig. 6.9

OVERFLOW AND SCALING - EXAMPLE

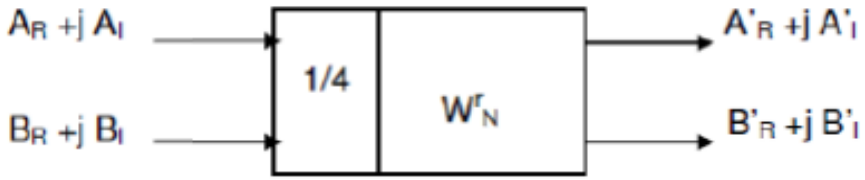


Fig 6.8: Symbolic representation of Butterfly structure with scaling factor

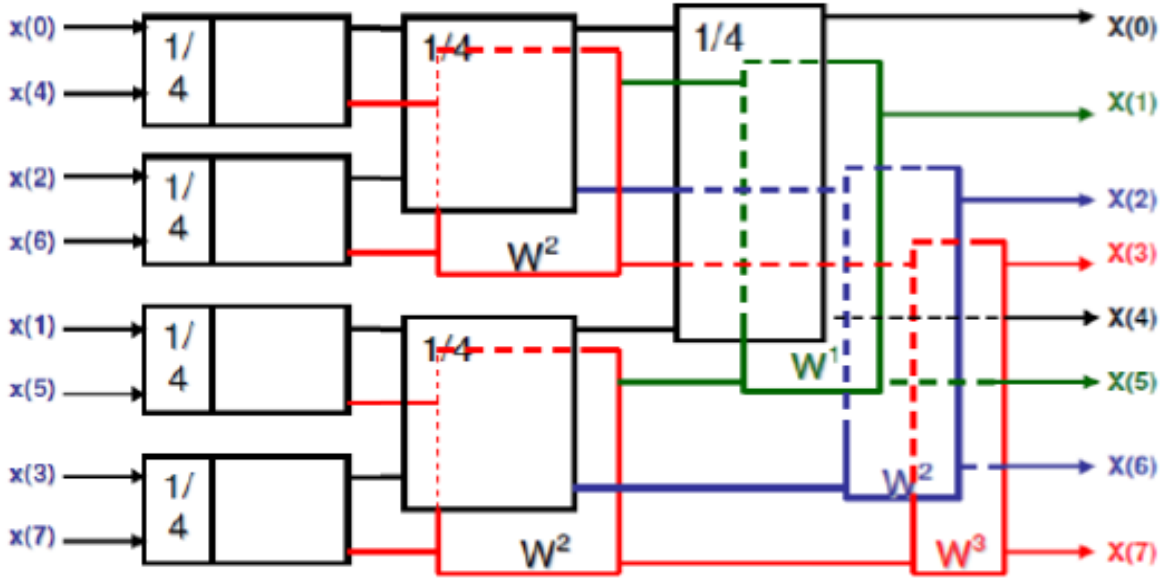


Fig 6.9: Signal flow graph with Scaling

BIT-REVERSED INDEX GENERATION

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

Table 6.3: Original & bit reverse indices

Original Index	Bit Reversed Index
000	000
001	100
010	010
011	110
100	001
101	101
110	011
111	111