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

$$y(m) = w(mM) = \sum_{k=-\infty} b_k x(mM - k)$$

where $w(n) = \sum_{k=-\infty}^{\infty} b_k x(n-k)$

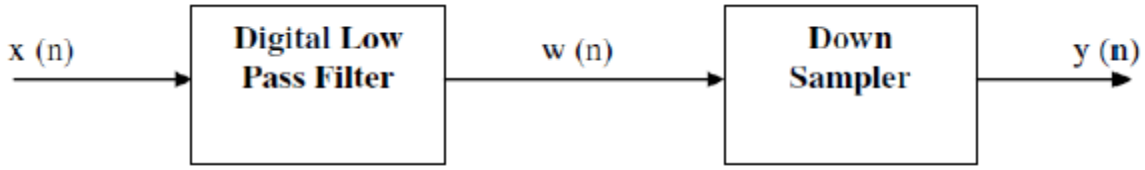


Figure 1.1 : Decimation process

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

$$\sum_{k=-\infty}^{\infty} b_k w(m-k) \quad \text{where } w(n) = \begin{cases} x\left(\frac{m}{L}\right), m = 0, \pm 1L, \pm 2L, \pm 3L \ldots\ldots \\ 0, otherwise \end{cases}$$

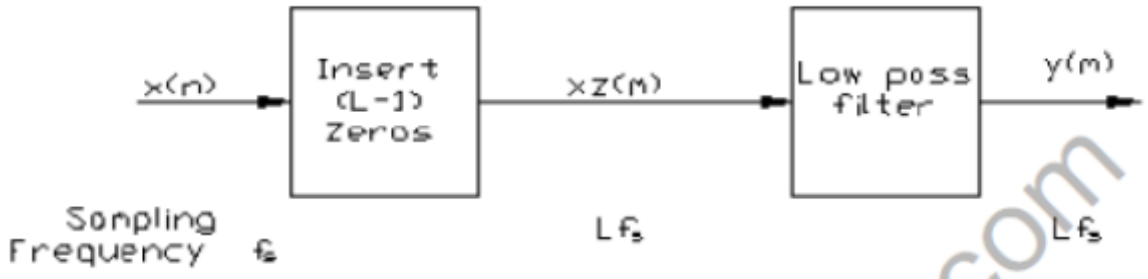


Figure 1. 2: Interpolation process

X(n) = {0, 2, 4, 6, 8 }

L=2 bk ={0.5, 1, 0.5}

Insert L-1 = 2-1 =1 zero ➔ w(n) = {0, 0, 2, 0, 4, 0, 6, 0, 8, 0}

y(m) = w(n) * bk

= {0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 4, 0}

1 b. DSP is a technique of performing the mathematical operations on the signals in digital domain. As real time signals are analog in nature we need first convert the analog signal to digital, then we have to process the signal in digital domain and again converting back to analog domain. Thus ADC is required at the input side whereas a DAC is required at the output end. A typical DSP system is as shown in figure 1.3.

Figure 1.3 : A typical DSP system

A computer or a processor is used for digital signal processing. Antialiasing filter is a LPF which passes signal with frequency less than or equal to half the sampling frequency in order to avoid Aliasing effect. Similarly at the other end, reconstruction filter is used to reconstruct the samples from the staircase output of the DAC (Figure 1.4).

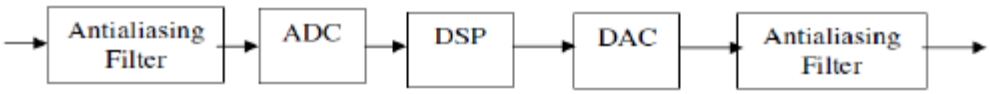


Figure 1.4: The block diagram of a DSP system

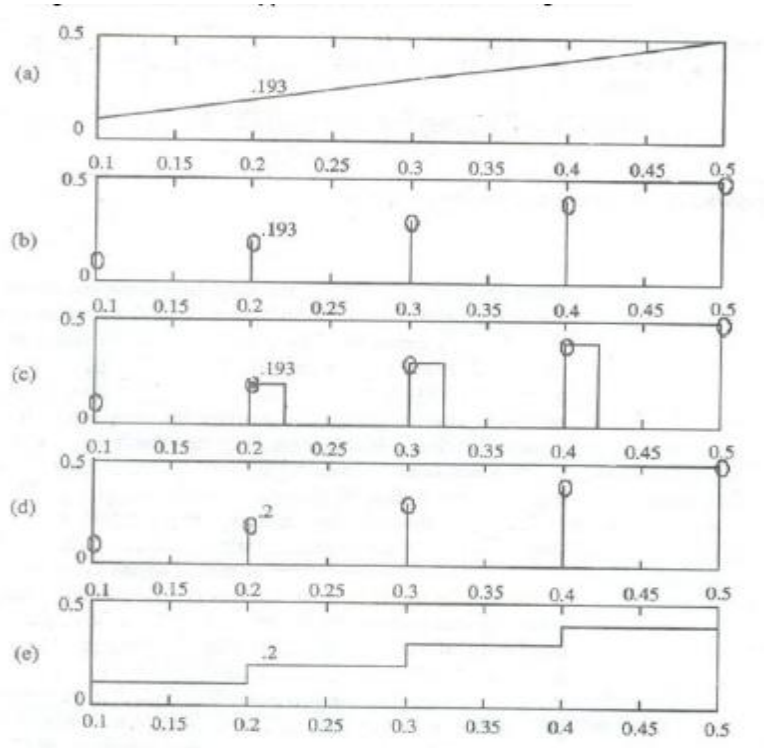


Figure 1.5 : Signals that occur in a DSP system

1 C. The number of complex multiplications = $(\frac{N}{2})\log_2 N$ = (1024/2 X $\log_2$ 1024)

= 5120

The number of Real multiplications = 4 x 5120 = 20480

2 a.

In order to implement the above operation in a DSP, the architecture requires the following features

i. A RAM to store the signal samples x (n)

ii. A ROM to store the filter coefficients h (n)

iii. An MAC unit to perform Multiply and Accumulate operation

iv. An accumulator to store the result immediately

v. A signal pointer to point the signal sample in the memory

vi. A coefficient pointer to point the filter coefficient in the memory

vii. A counter to keep track of the count

viii. A shifter to shift the input samples appropriately

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

Circular Addressing mode requires three registers viz

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

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

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

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

a. SAR < EAR & updated PNTR > EAR

b. SAR < EAR & updated PNTR < SAR

c. SAR >EAR & updated PNTR > SAR

d. SAR > EAR & updated PNTR < EAR

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

The pointer updating algorithm for the circular addressing mode is as shown below.

```
; Pointer Updating Algorithm

Updated PNTR ◄—— PNTR ± increment

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

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

Else
        New PNTR ◄—— Updated PNTR
```

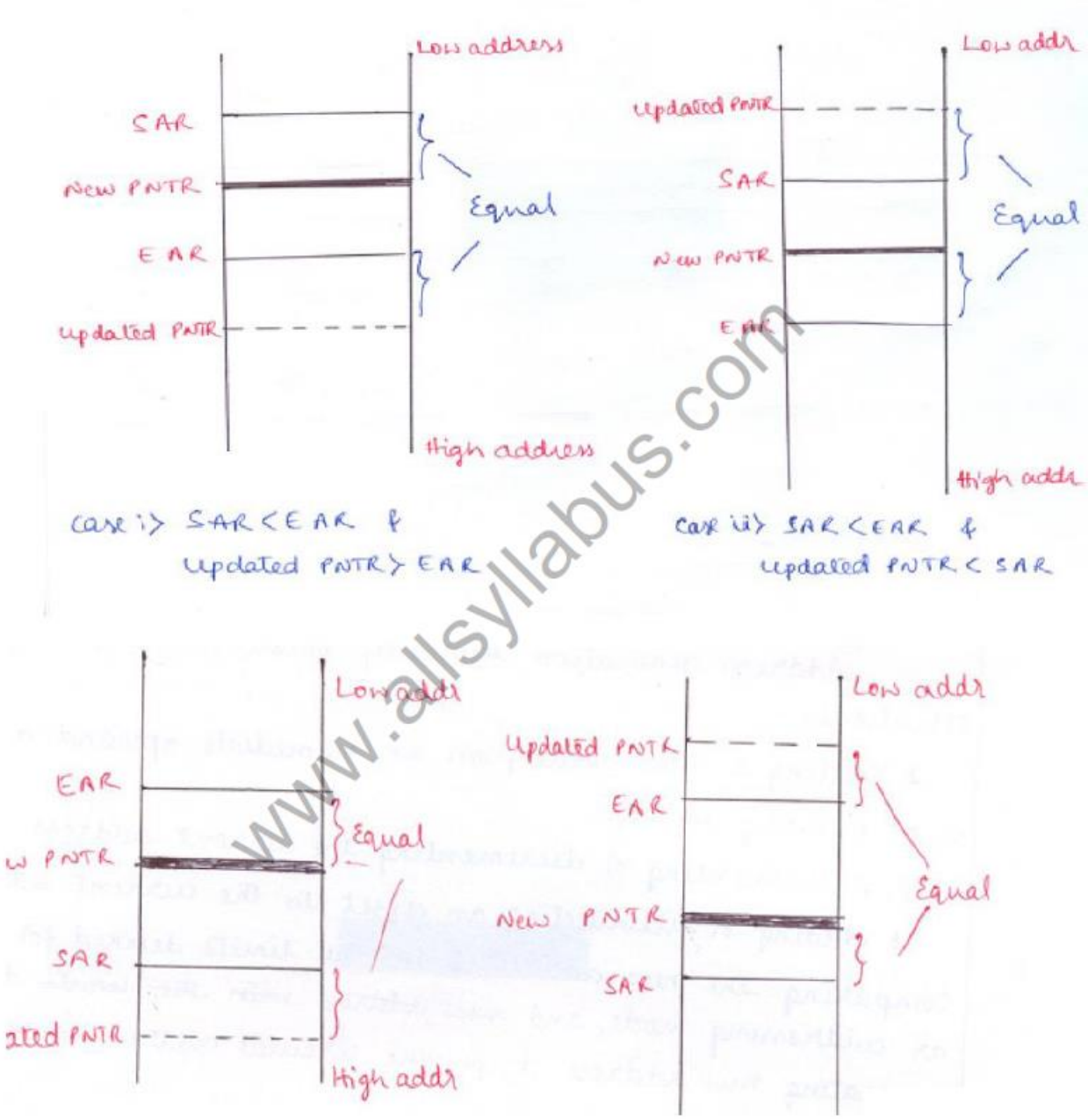


Figure 2.1 : Special cases in circular addressing mode

2 c. A typical DSP device should be capable of handling arithmetic instructions like ADD, SUB, INC, DEC etc and logical operations like AND, OR , NOT, XOR etc. The block diagram of a typical ALU for a DSP is as shown in the figure 2.2. It consists of status flag register, register file and multiplexers.

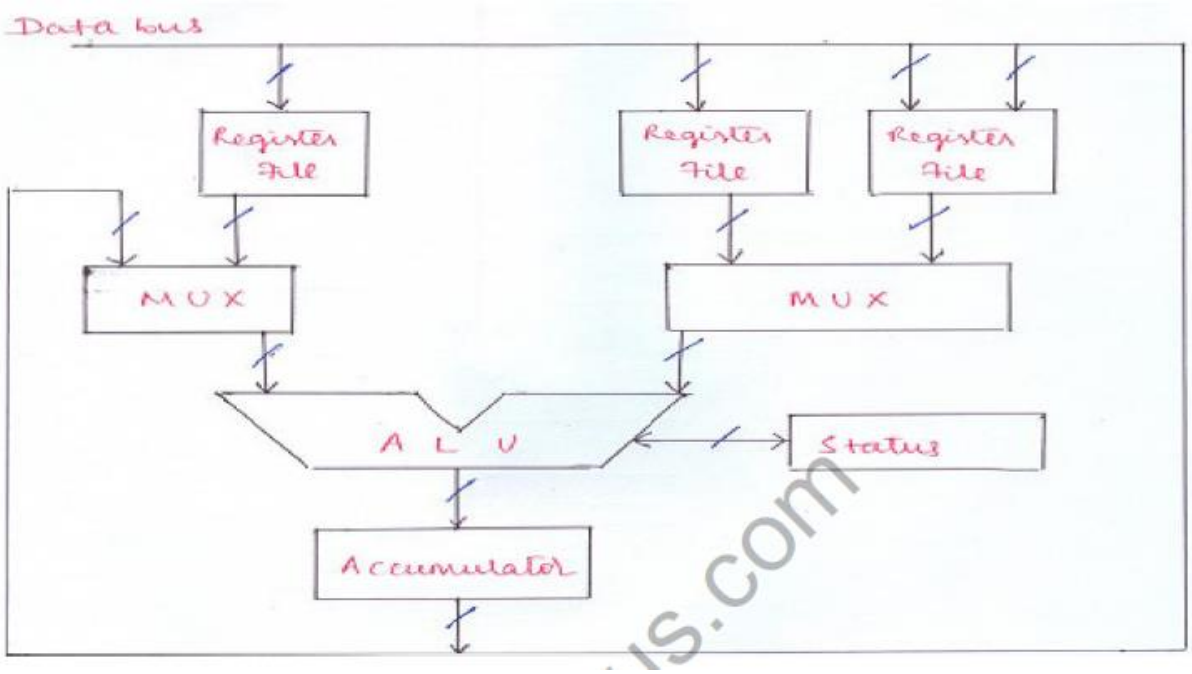


Figure 2. 2: ALU

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

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

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

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

a. As many DSP algorithms require instructions to be executed repeatedly, the instruction can be stored in the external memory, once it is fetched can reside in the instruction cache.

b. The access times for memories on-chip should be sufficiently small so that it can be accessed more than once in every execution cycle.

c. On-chip memories can be configured dynamically so that they can serve different purpose at different times.

3 a.

Architectural Feature	TMS320C25	DSP 56000
Data representation format	16-bit fixed	24-bit fixed point
Hardware multiplier	16 x 16	24 x 24
ALU	32 bits	56 bits
Internal buses	16-bit program bus 16-bit data bus	24-bit program bus 2 x 24-bit data buses 24-bit global databus
External buses	16-bit program/data bus	24-bit program/data bus
On-chip Memory	544 words RAM 4K words ROM	512 words PROM 2 x 256 words data RAM 2 x 256 words data ROM
Off-chip memory	64 K words program 64k words data	64K words program 2 x 64K words data
Cache memory	-	-
Instruction cycle time	100 nsec	97.5 nsec.
Special addressing modes	Bit reversed	Modulo Bit reversed
Data address generators	1	2
Interfacing features	Synchronous serial I/O DMA	Synchronous and Asynchronous serial I/O DMA

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

No overhead is required to implement the shift needed for the scaling operations. The'54xx barrel shifter can produce a left shift of 0 to 31 bits or a right shift of 0 to 16 bits on the input data. The shift count field of status registers ST1, or in the temporary register T. Figure 3.1 shows the functional diagram of the barrel shifter of TMS320C54xx processors.

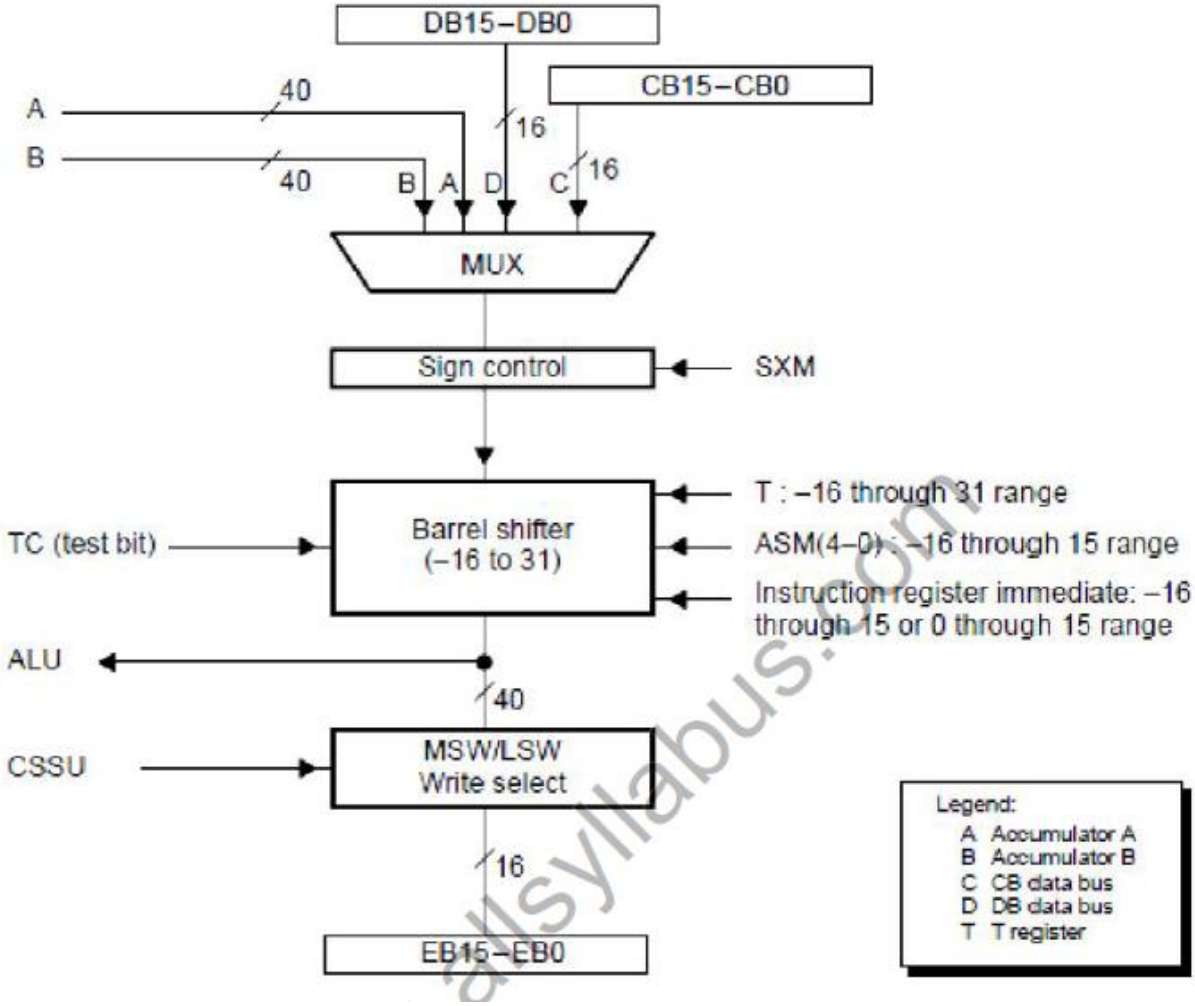


Figure 3.1 : Functional diagram of Barrel shifter

The barrel shifter and the exponent encoder normalize the values in an accumulator in a single cycle. The LSBs of the output are filled with 0s, and the MSBs can be either zero filled or sign extended, depending on the state of the sign-extension mode bit in the status register ST1. An additional shift capability enables the processor to perform numerical scaling, bit extraction, extended arithmetic, and overflow prevention operations.

3 c. i) * AR3 – 0 = 200h – 40h = 1C0h

ii) * AR3+ = 200h + 1 = 201h

iii) *+AR3 (50h) = 200h + 50h = 250h

iv) *AR3 – 0B = 200h - 40h (with reverse carry propagation) = 27Fh.

4 a. **Pipeline operation of TMS320C54xx Processors:**

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

1 During program prefetch, the program address bus, PAB, is loaded with the address of the next instruction to be fetched.

2 In the fetch phase, an instruction word is fetched from the program bus, PB, and loaded into the instruction register, IR. These two phases from the instruction fetch sequence.

3 During the decode stage, the contents of the instruction register, IR are decoded to determine the type of memory access operation and the control signals required for the data-address generation unit and the CPU.

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

5 In the read phase the data operand(s), if any, are read from the data buses, DB and CB. This phase completes the two-phase read process and starts the two phase write processes. The data address of the write operand, if any, is loaded into the data write address bus, EAB.

6 The execute phase writes the data using the data write bus, EB, and completes the operand write sequence. The instruction is executed in this phase.

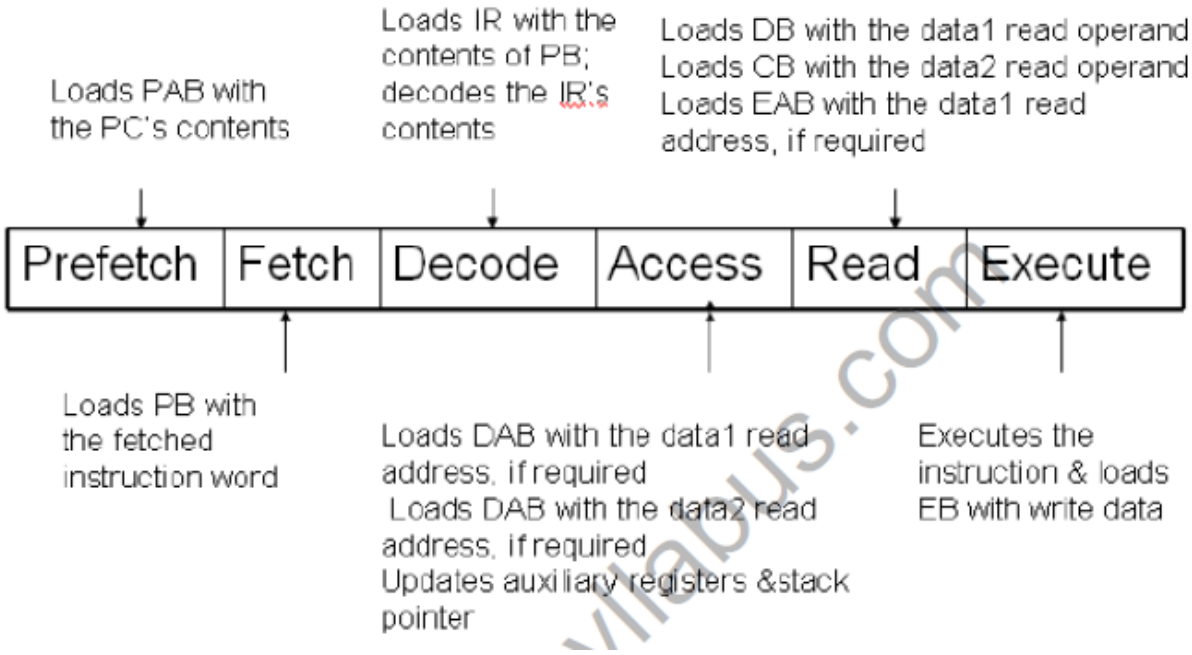


Figure 4. 1 : Pipeline operation of TMS320C54XX processor

```
LD    *AR3+, A
ADD   #1000h, A
STL   A, *AR3+
```

Cycles	Prefetch	Fetch	Decode	Access	Read	Exe& Write	AR3	A
1	LD						85	
2	ADD	LD					85	
3	STL	ADD	LD				85	
4		STL	ADD	LD			86	
5			STL	ADD	LD		86	5h
6				STL		LD	87	5h
7					STL	ADD	87	1005h
8						STL	87	1005h

4 b.

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

Figure 4. 2 : Functions of various bits in TCR register

4 c.

```
.global _c_int00

X       .usect "Input Samples", 3
Y       .usect "outout", 2
h       .usect "coefficient", 3
        .text

_c_int00:

        SSBX  SXM       ;Select sign extension mode
        LD    #h, DP    ;Select the data page for coefficients
        LD    @h, T     ;get the coefficient h(0)
        LD    #x, DP    ;select the data page for input samples
        MPY   @x, A     ; A = x(n) * h(0)
        LD    #h, DP    ; select the data page for coefficients
        LD    @h+1, T         ; get the coefficient h(1)
        LD    #x, DP    ;select the data page for input singals
        MPY   @x+1, B         ; B = x(n-1) * h(1)
        ADD   A, B            ; B = x(n)*h(0) + x(n-1)*h(1)
        LD    #h, DP    ; select the data page for coefficients
        LD    @h+2, T         ; get the coefficient h(2)
        LD    #x, DP    ;select the data page for input samples
        MPY   @x+2, B         ; B = x(n-2) * h(1)
        ADD   A, B            ; B = x(n)*h(0)+ x(n-1)*h(1) + x(n-2) * h(2)


        LD    #y, DP    ; select the data page for outputs
        STL   B, @y     ; save low part of output
        STH   B, @y+1         ; save high part of output
        NOP             ; No operation
        .end
```

6 a. i) $\log_2 N = 7$

ii) $\frac{N}{2} = 64$

iii) $\frac{N}{2} \log_2 N = 7 \times 64 = 448$

iv) Nil

6 b.

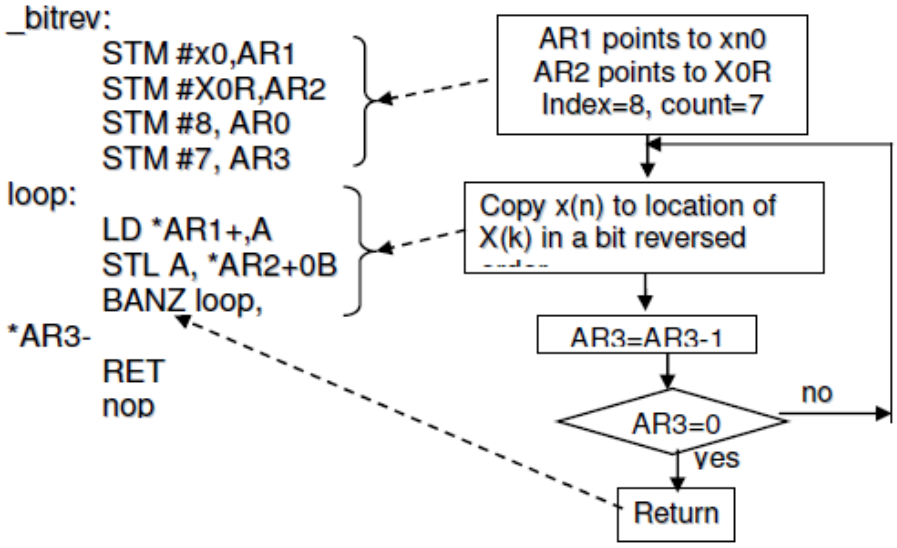
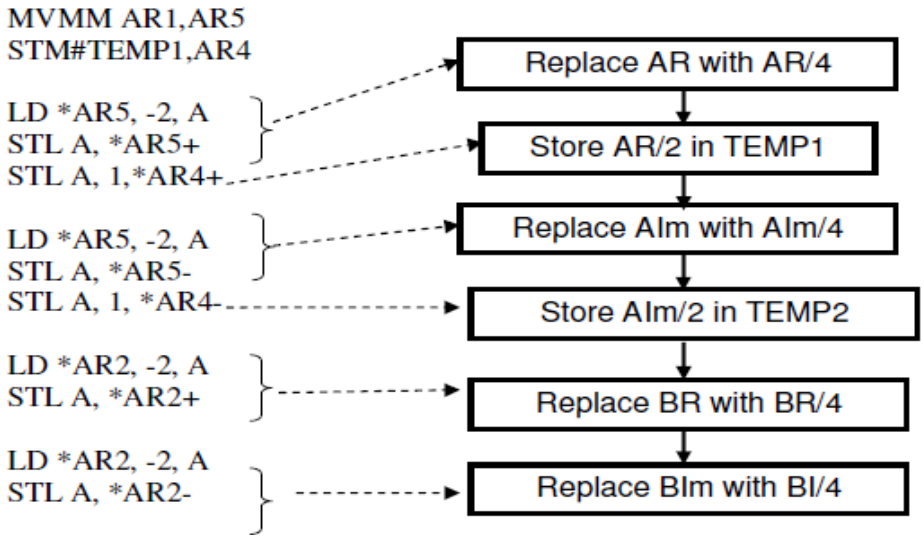
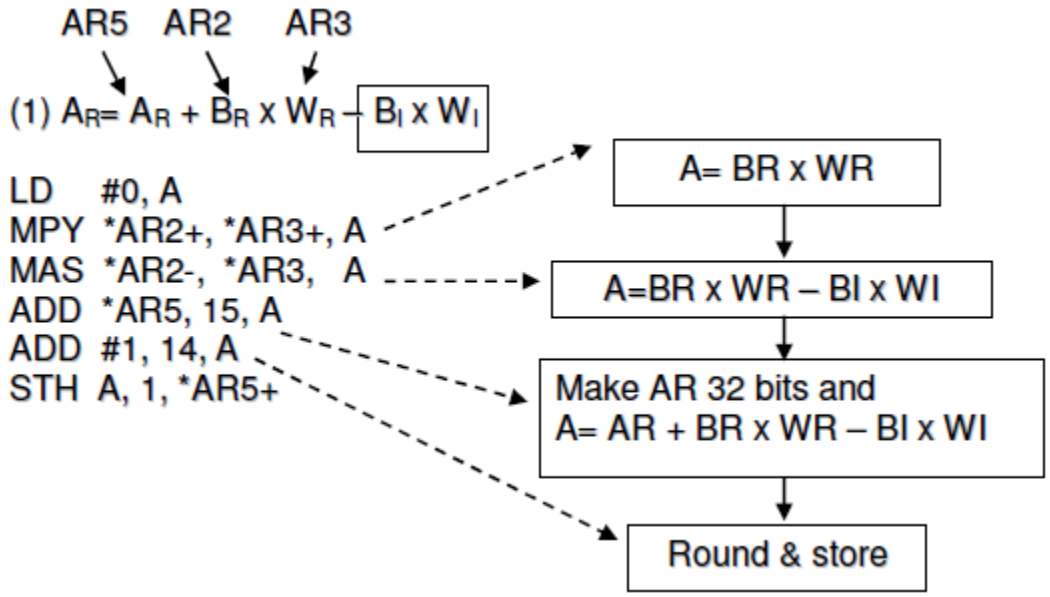


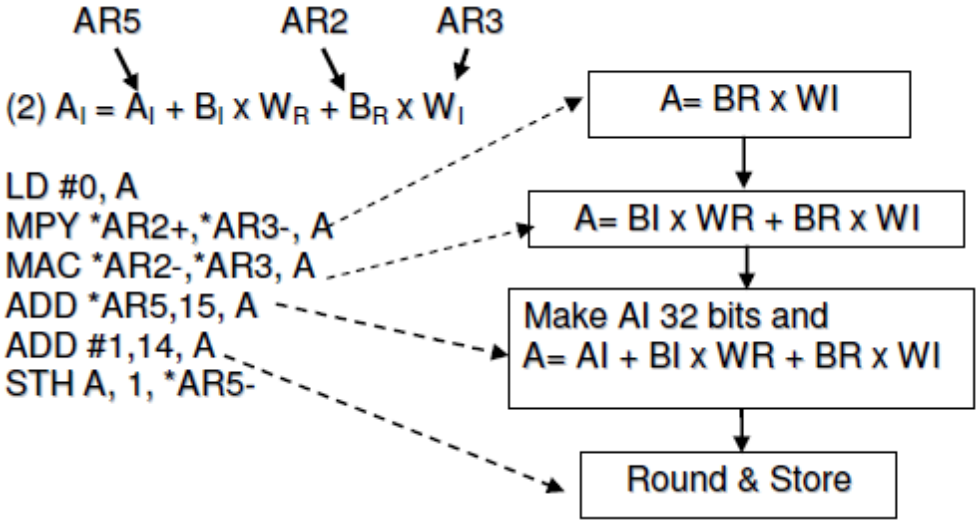
Figure 6.1 : Subroutine program for bit reversed address generation

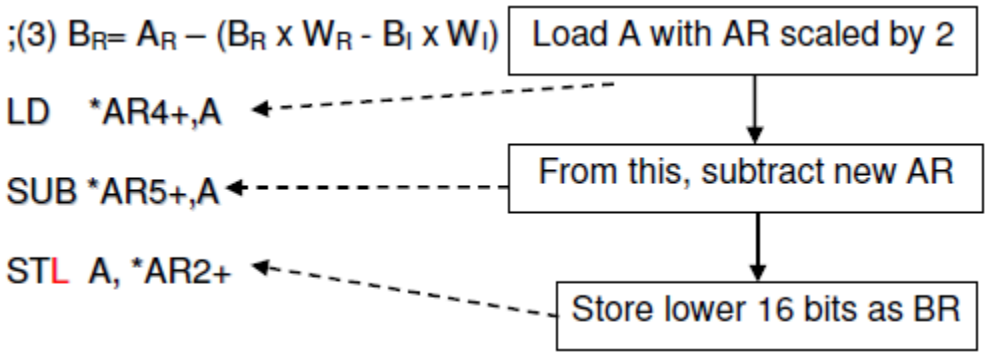
Here, AR1 is used as pointer to x(n). AR2 is used as pointer to X(k) locations. AR0 is loaded with 8 and used in bit reverse addressing. Instead of N/2 =4, it is loaded with N=8 because each X(k) requires two locations, one for real part and the other for imaginary part. Thus, x(n) is stored in alternate locations, which are meant for real part of X(k). AR3 is used to keep track of number of transfers.

6 c .




AR5
AR2
AR3
(1) $A_R = A_R + B_R \times W_R - B_I \times W_I$
LD #0, A
MPY *AR2+, *AR3+, A
MAS *AR2-, *AR3, A
ADD *AR5, 15, A
ADD #1, 14, A
STH A, 1, *AR5+
A= BR x WR
A=BR x WR – BI x WI
Make AR 32 bits and
A= AR + BR x WR – BI x WI
Round & store


AR5
AR2
AR3
(2) $A_I = A_I + B_I \times W_R + B_R \times W_I$
LD #0, A
MPY *AR2+,*AR3-, A
MAC *AR2-,*AR3, A
ADD *AR5,15, A
ADD #1,14, A
STH A, 1, *AR5-
A= BR x WI
A= BI x WR + BR x WI
Make AI 32 bits and
A= AI + BI x WR + BR x WI
Round & Store


;(3) $B_R = A_R – (B_R \times W_R - B_I \times W_I)$
LD *AR4+,A
SUB *AR5+,A
STL A, *AR2+
Load A with AR scaled by 2
From this, subtract new AR
Store lower 16 bits as BR

;(4) $B_I = A_I - (B_I \times W_R + B_R \times W_I)$

```
LD   *AR4 -, A
SUB *AR5-, A
STL  A, *AR2-

RET
nop
nop
```

7 a.

The timing sequence of memory access is shown in fig. 7.1. 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.

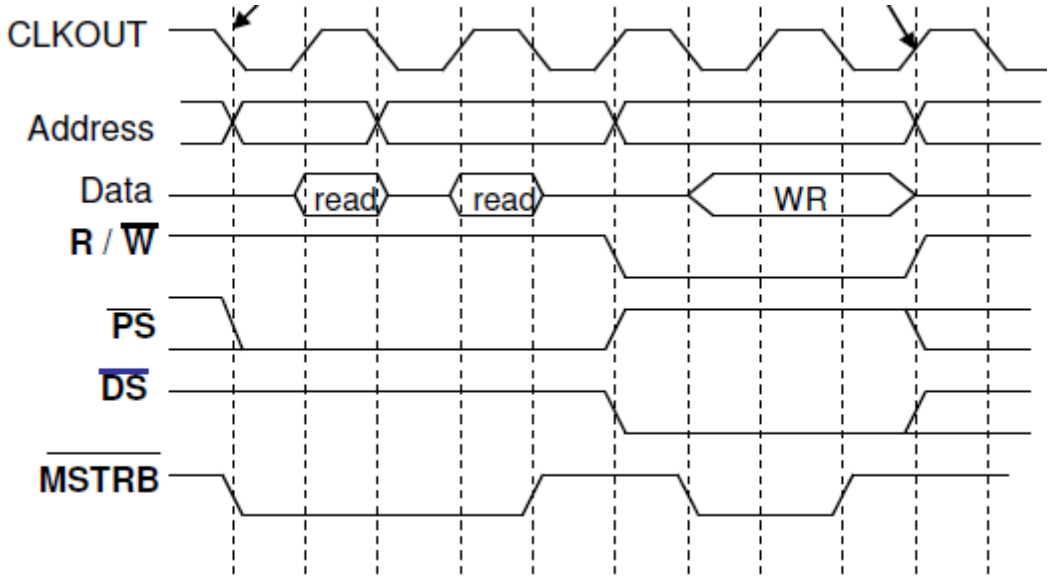


Figure 7.1 : Memory interface signals for read- read – write operations

7 b.

Number of memory mapped registers for DMA are 6x(5+4) and some common registers for all channels, amounting to total of 62 registers required. However, only 3 (+1 for priority related) are available. They are DMA Priority & Enable Control Register (DMPREC), DMA sub bank Address Register (DMSA), DMA sub bank Data Register with auto increment (DMSDI) and DMA sub bank Data Register (DMSDN). To access each of the DMA Registers Register sub addressing Technique is employed. The schematic of the arrangement is shown in fig. 7.13. A set of DMA registers of all channels (62) are made available in set of memory locations called sub bank. This avoids the need for 62 memory mapped registers. Contents of either DMSDI or DMSDN indicate the code (1's & 0's) to be written for a DMA register and contents of DMSA

refers to the unique sub address of DMA register to be accessed. Mux routes either DMSDI or MSDN to the sub bank. The memory location to be written

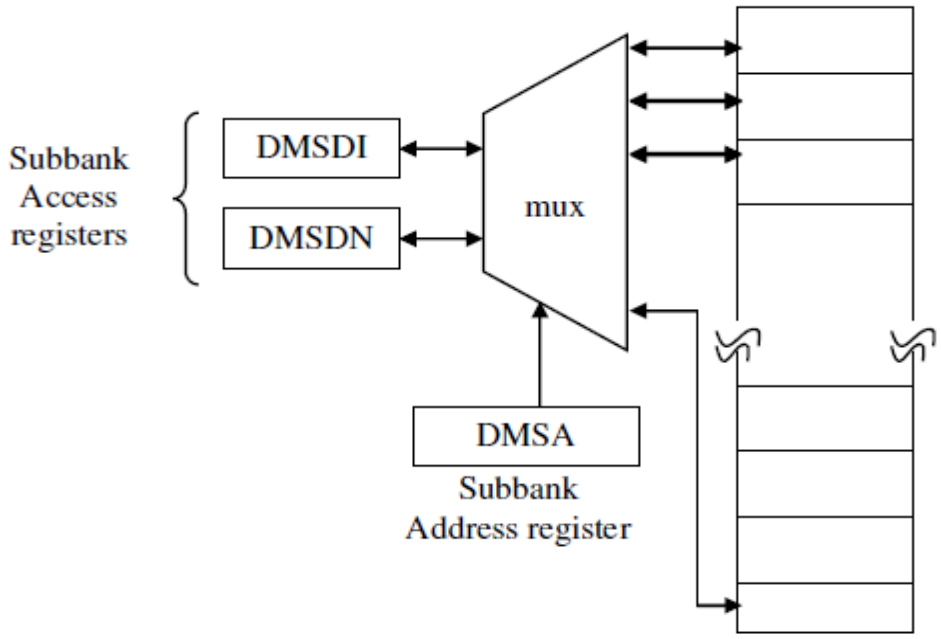


Figure 7. 2 : Register subaddress technique

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.

7 c.

Initialize processor with respect to desired speed, internal registers: PMST, BSCR, SWSR. Initialize internal timer for sampling period of ADC. Apply analog input signal. Send start conversion through TOUT. Continue with any other program execution. ADC interrupts DSP after conversion on INT1. DSP reads 10 bit data and converts to 8 bit by shifting to right. It then processes the sample and sends this data to DAC. DAC converts the data back to analog. The corresponding program is as follows.

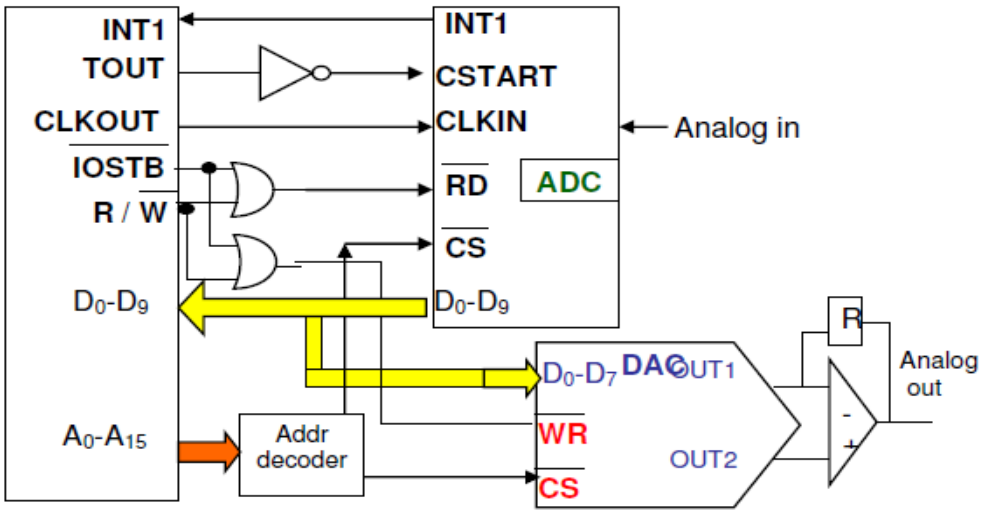


Figure 7.3 : Interfacing DSP to ADC and DAC

```
buffer: .bss sample,1                ;data buffer
    • text
_c_int00:

        STM #0X0500,SP               ;initialize Stack Pointer
        SSBX INTM                    ; disable all interrupts
        CALL init_DSP                ; initialize DSP processor
        CALL init_timer              ;initialize timer
        STM #0XFFFF,IFR              ;pending interrupts cleared
        ORM #0002H,IMR               ;INT 1 unmasked
        RSBX INTM                    ; enable all interrupts
;   Initialize DSP Processor
;init_DSP
        PMST_VAL          .set  00A0h; MC & OVLY, interrupt vector is set
        BSCR_VAL          .set  0000h ; bank switching reset, 64K only
        SWWSR_VAL          .set  2000h ;s/w wait, 2 clock wait states
```

```
.text
        init_DSP:
                LD #0,DP                ;data page initialized
                STM #0,CLKMD
                STM #0,CLKMD
              STM #0X4007,CLKMD         ;processor speed set
              STM #PMST_VAL, PMST       ;initialize PMST
              STM #BSCR_VAL, BSCR       ;initialize BSCR
              STM #SWWSR_VAL, SWWSR     ;initialize SWWSR
              SSBX OVM                  ; identify overflow
              SSBX SXM                  ;sign extension set
              RET                       ;return from subroutine
              NOP
              NOP


; Initialize Timer

        PRD_VAL         .set 9999       ; FSAMPLING  / FCPU
        TCR_VAL         .set 0000       ;start timer
        .text
        init_timer:
                STM PRD_VAL, PRD            ;initialize timer period register
                STM TCR_VAL,TCR             ;initialize timer count register
                RET                         ;return from subroutine
                NOP
                NOP
;ISR for ADC Read & DAC Write
                ADC_DATA_IN_addr    .set 05h    ;port address of ADC
                DAC_DATA_OUT_addr   .set 07h    ;port address of DAC
        .text
                PORTR ADC_DATA_IN_addr, sample ; read data from ADC
                LD sample,-2,A                  ; 10 bit data to 8 bit
                STL A, sample                   ; store in buffer
                PORTW sample, DAC_DATA_OUT_addr ; write to DAC
                RET                             ;return from subroutine
                NOP
                NOP
```

8 a.

Synchronous Serial Interface: There are certain I/O devices which handle transfer of one bit at a time. Such devices are referred to as serial I/O devices or peripherals. Communication with serial peripherals can be synchronous, with processor clock as reference or it can be synchronous. Synchronous serial interface (SSI) makes communication a fast serial

communication and asynchronous mode of communication is slow serial communication. However, in comparison with parallel peripheral interface, the SSI is slow. The time taken depends on the number of bits in the data word.

CODEC Interface Circuit: CODEC, a coder-decoder is an example for synchronous serial I/O. It has analog input-output, ADC and DAC. The signals in SSI generated by the DSP are DX: Data Transmit to CODEC, DR: Data Receive from CODEC, CLKX: Transmit data with this clock reference, CLKR: Receive data with this clock reference, FSX: Frame sync signal for transmit, FSR: Frame sync signal for receive, First bit, during transmission or reception, is in sync with these signals, RRDY: indicator for receiving all bits of data and XRDY: indicator for transmitting all bits of data. Similarly, on the CODEC side, signals are FS*: Frame sync signal, DIN: Data Receive from DSP, DOUT: Data Transmit to DSP and SCLK: Tx / Rx data with this clock reference. The block diagram depicting the interface between TMS320C54xx and CODEC is shown in fig. 8.1. As only one signal each is available on CODEC for clock and frame synchronization, the related DSP side signals are connected together to clock and frame sync signals on CODEC. Fig. 8.2 and fig. 8.3 show the timings for receive and transmit in SSI, respectively.

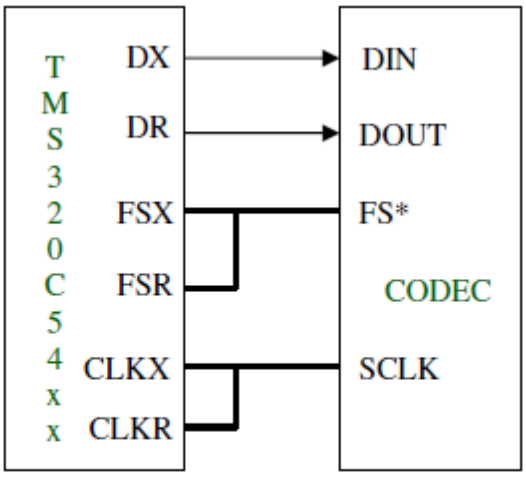


Fig. 8.1: SSI between DSP & CODEC

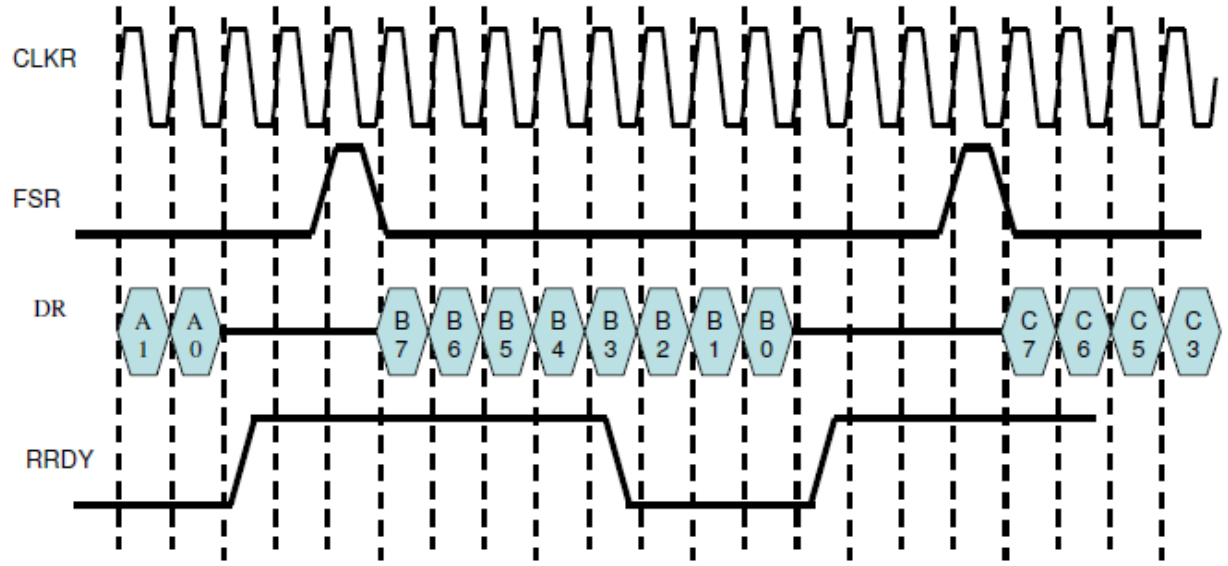


Fig. 8.2: Receive Timing for SSI

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

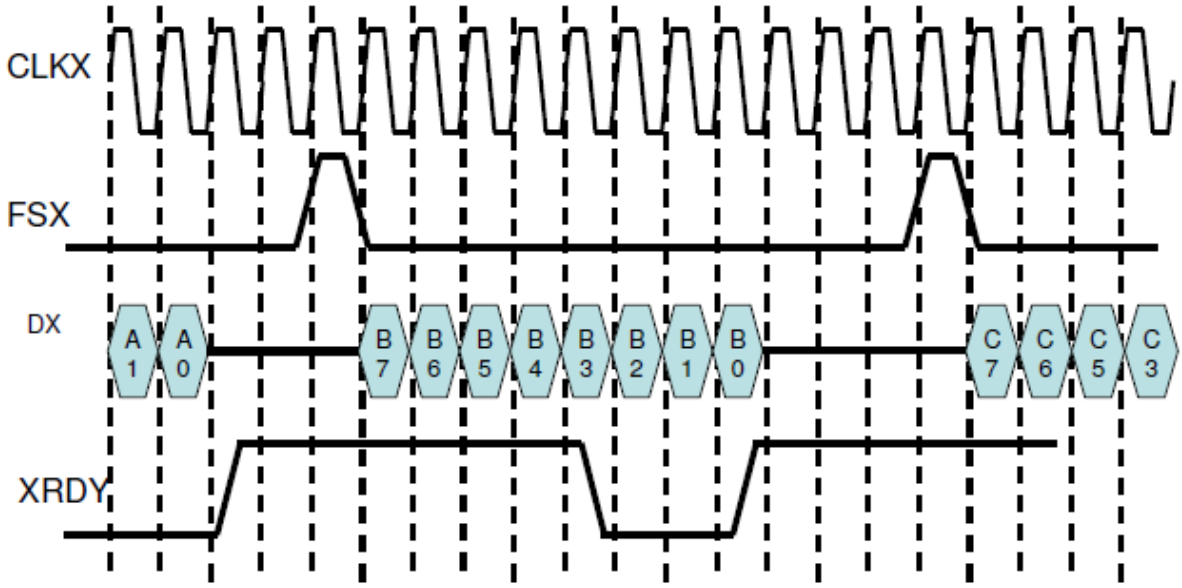


Fig 8.3: Transmit Timing for SSI

8 b.

DSP Based Biotelemetry Receiver:

Biotelemetry involves transfer of physiological information from one remote place to another for the purpose of obtaining experts opinion. The receiver uses radio Frequency links. The schematic diagram of biotelemetry receiver is shown in fig. below. The biological signals may be single dimensional signals such as ECG and EEG or two dimensional signals such as an image, i.e., X-ray. Signal can even be multi dimensional signal i.e., 3D picture. The signals at source are encoded, modulated and transmitted. The signals at destination are decoded, demodulated and analyzed.

An example of processing ECG signal is considered. The scheme involves modulation of ECG signal by employing Pulse Position Modulation (PPM). At the receiving end, it is demodulated. This is followed by determination of Heart beat Rate (HR). PPM Signal either encodes single or multiple signals. The principle of modulation being that the position of pulse decides the sample value. The PPM signal with two ECG signals encoded is shown in fig. below. The transmission requires a sync signal which has 2 pulses of equal interval to mark beginning of a cycle. The sync pulses are followed by certain time gap based on the amplitude of the sample of1st signal to be transmitted. At the end of this time interval there is another pulse. This is again followed by time gap based on the amplitude of the sample of the 2nd signal to be transmitted. After encoding all the samples, there is a compensation time gap followed by sync pulses to mark the beginning of next set of samples. Third signal may be encoded in either of the intervals of 1st or 2nd signal. With two signals encoded and the pulse width as tp, the total time duration is 5tp.

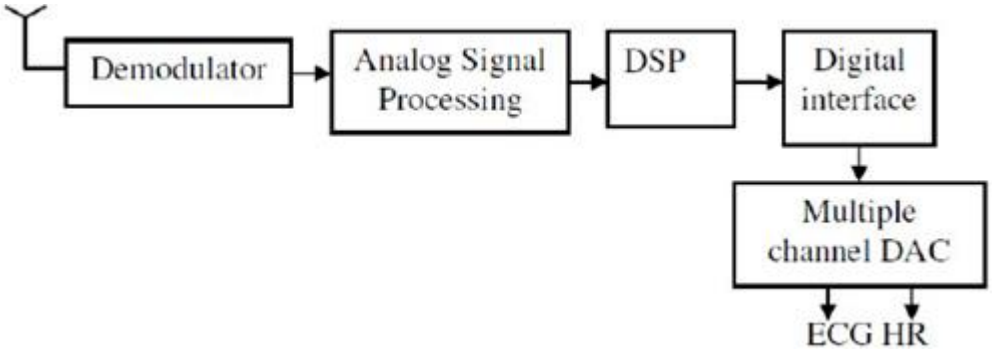


Figure 8.4 : Bio- telemetry receiver

An example of processing ECG signal is considered. The scheme involves modulation of

ECG signal by employing Pulse Position Modulation (PPM). At the receiving end, it is demodulated. This is followed by determination of Heart beat Rate (HR). PPM Signal either encodes single or multiple signals. The principle of modulation being that the position of pulse decides the sample value. The PPM signal with two ECG signals encoded is shown in fig. 8 .5.The transmission requires a sync signal which has 2 pulses of equal interval to mark beginning of a cycle. The sync pulses are followed by certain time gap based on the amplitude of the sample of 1st signal to be transmitted. At the end of this time interval there is another pulse. This is again followed by time gap based on the amplitude of the sample of the 2nd signal to be transmitted. After encoding all the samples, there is a compensation time gap followed by sync pulses to mark the beginning of next set of samples. Third signal may be encoded in either of the intervals of 1st or 2nd signal. With two signals encoded and the pulse width as tp, the total time duration is 5tp.

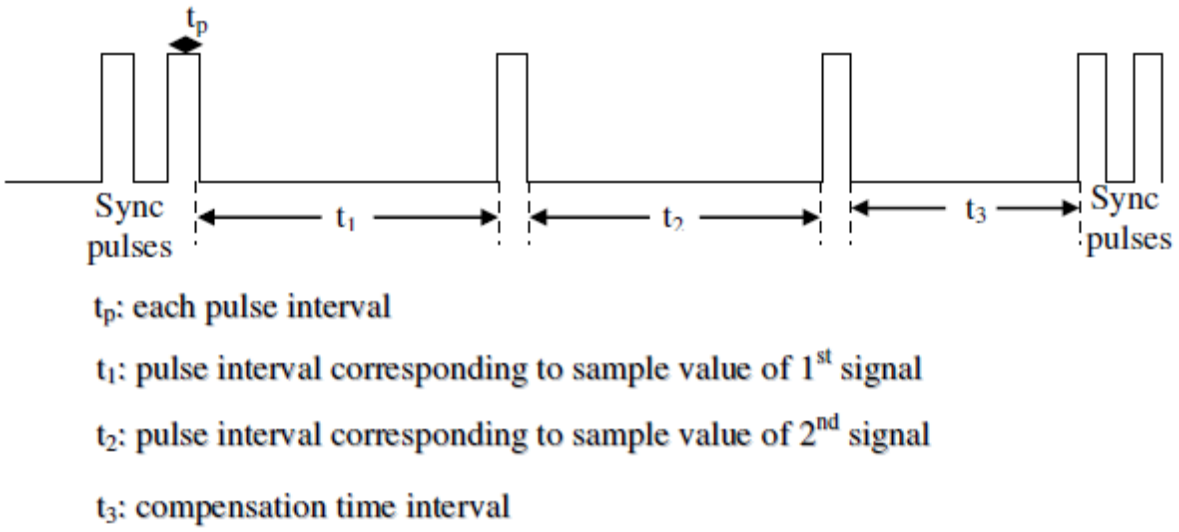


Figure 8.5: A PPM signal with 2 ECG signals

8 c.

An Image Processing System: In comparison with the ECG or speech signal considered so far, image has entirely different requirements. It is a two dimensional signal. It can be a color or gray image. A color image requires 3 matrices to be maintained for three primary colors-red, green and blue. A gray image requires only one matrix, maintaining the gray information of each pixel (picture cell). Image is a signal with large amount of data. Of the many processing, enhancement, restoration, etc., image compression is one important processing because of the large amount of data in image.

To reduce the storage requirement and also to reduce the time and band width required to transmit the image, it has to be compressed. Data compression of the order of factor 50 is sometimes preferred. JPEG, a standard for image compression employs lossy compression technique. It is based on discrete cosine transform (DCT). Transform domain compression separates the image signal into low frequency components and high frequency components. Low frequency components are retained because they represent major variations. High frequency components are ignored because they represent minute variations and our eye is not sensitive to minute variations. Image is divided into blocks of 8 x 8. DCT is applied to each block. Low frequency coefficients are of higher value and hence they are retained. The amount of high frequency components to be retained is decided by the desirable quality of reconstructed image. Forward DCT is given by eq (1).

$$f_{v,u} = \frac{1}{4} c_v c_u \sum_{x=0}^{7} \sum_{y=0}^{7} f_{x,y} \cos\left(\frac{(2x+1)u\pi}{16}\right) \cos\left(\frac{(2y+1)v\pi}{16}\right)$$

------ (1)

Since the coefficients values may vary with a large range, they are quantized. As already noted low frequency coefficients are significant and high frequency coefficients are insignificant, they are allotted varying number of bits. Significant coefficients are quantized precisely, with more bits and insignificant coefficients are quantized coarsely, with fewer bits. To achieve this, a quantization table as shown in fig. 8.20 is employed. The contents of Quantization Table indicate the step size for quantization. An entry as smaller value implies smaller step size, leading to more bits for the coefficients and vice versa.

The quantized coefficients are coded using Huffman coding. It is a variable length coding Huffman Encoding. Shorter codes are allotted for frequently occurring long sequence of 1's & 0's. Decoding requires Huffman table and dequantization table. Inverse DCT is taken employing eq (3). The data blocks so obtained are combined to form complete image. The schematic of encoding and decoding is shown in fig. 8.6.

$$f_{x,y} = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} c_u c_v f_{u,v} \cos\left(\frac{(2x+1)u\pi}{16}\right) \cos\left(\frac{(2y+1)v\pi}{16}\right)$$

------(2)

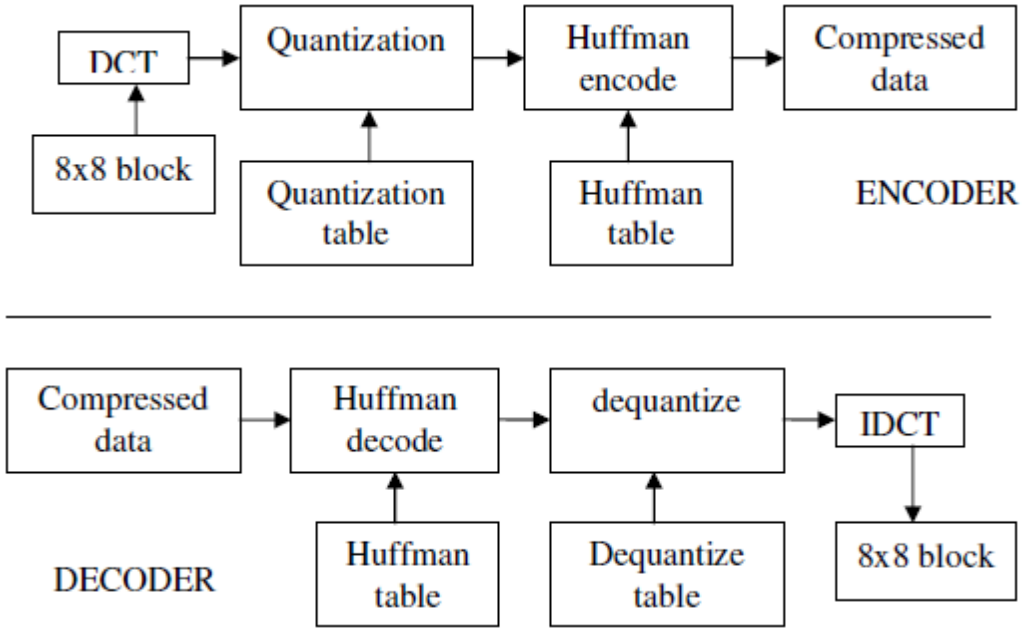


Figure 8.6 : JPEG encoder and decoder

5 a. i) .3125 as Q15 number

Q15 notation= .3125 x 2^{15} = 10,240 in decimal=2800h

ii)) -.3125 as Q15 number

Q15 notation= .3125 x 2^{15} = 10,240 in decimal=2800h

Since the number is negative (-.3125), hexadecimal equivalent of 10,240 (i.e) 2800h should be subtracted from FFFFh to get D7FFh

Hence -.3125 = - (10,240 decimal)= -(2800h) = FFFF-2800 = D7FFh

iii) 3.125 as Q7 number

3.125 X 2^7 = 190 h

5 b.

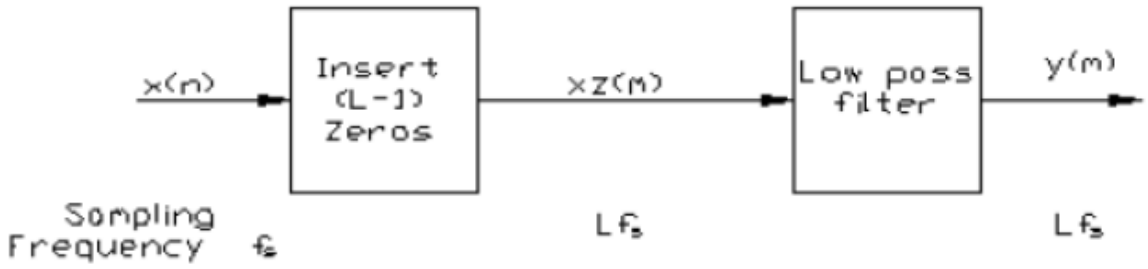


Figure 5.1 : Interpolation process

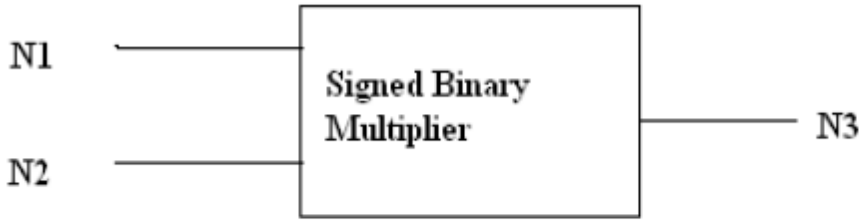
Example:

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)

```
                .mmregs
                .def _c_int00

                .sect "samples"
InSamples       .include "data_in.dat"      ; Incoming data (from a file)
InSampCnt       .set 50                     ; Input sample count
                .bss sample,3,1             ; Input samples: x(n),x(n-1),x(n-2)
OutSamples      .bss y,250,1                ; Allocate space for y(n)s
SampleCnt       .set 250                    ; Number of samples
Coeff           .sect "Coeff"
                .word 2560, 3072, 512       ; Filter coeffs h(4), h(9), h(14)
                .word 2048, 3584, 1024      ; Filter coeffs h(3), h(8),
h(13)
                .word 1536, 4096, 1536      ; Filter coeffs h(2), h(7),
h(12)
                .word 1024, 3584, 2048      ; Filter coeffs h(1), h(6),
h(11)
                .word 512, 3072, 2560       ; Filter coeffs h(0), h(5),
h(10)
CoeffEnd


Nm1             .set 2                      ; # of coeff/interp factor-1
IFm1        .set 4              ; interpolating factor-1

                .text
_c_int00:
                ssbx SXM            ; Select sign extension mode
                rsbx FRCT
                stm #InSamples,ar6          ; ar6 points to the input samples
                stm #InSampCnt-1,ar7        ; ar7 = input sample count - 1
                stm #OutSamples,ar5         ; ar5 points to the output samples
                rpt #SampleCnt-1            ; Reset ouput samples memory
                st #0,*ar5+
```

```
                stm #OutSamples,ar5          ; ar5 points to the output samples
                stm #sample,ar3        ; ar3 points to current in samples
                rpt #Nm1                     ; Reset the input samples
                st #0, *ar3+

INTloop1:

                stm #CoeffEnd-1,ar2          ; ar2 points to the last coeff
                stm #IFm1,ar4          ; ar4 = Interpolation factor -1

INTloop2:

                stm #sample+Nm1,ar3          ; ar3 points to last sample in use
                stm #Nm1,ar1           ; ar1 = samples for use
                ld #0,A                      ; A = 0

NXTcoeff:
                mac *ar2-,*ar3-,A                ;   Compute   interpolated
sample
                banz NXTcoeff,*ar1-
        banz INTloop2,*ar4-
        sth A,1,*ar5+                ; Store the interpolated sample
                stm #sample+Nm1-1, ar3       ; Delay the sample array
                rpt #Nm1-1
                delay *ar3-
                ld *ar6+,A                       ; Get the next sample
                stm #sample,ar2
                stl A,*ar2                       ;   Place   it   in   the   sample
buffer
                banz INTloop1,*ar7-              ;   Repeat   for   all   input
samples
                nop
        nop
                nop
        .end
```

5 c.



N1(16 bit)	N2(16 bit)	N3(16 bit)
Q0	Q0	Q0
Q0	Q15	Q15
Q15	Q15	Q30

Figure 5.1 : Multiplication of numbers represented using Q notations

Program to multiply two Q15 numbers

i.e N1×N2 = N1*N2

Where

N1 &N2 are 16-bit numbers in Q15 notation
N1×N2 is the 16-bit result in Q15 notation

```
                .mmregs                 ; memory mapped registers
                .data                   ; sequential locations
N1:             .word   4000h           ; N1=0.5 (Q15 numbers)
N2:             .word   2000h           ; N2=0.25 (Q15 numbers)
N1×N2           .space  10h             ; space for N1×N2
                .text
                .ref            _c_int00
                .sect           ".vectors"
RESET:  b       _c_int00                ; reset vector
        nop
```

```
                nop

_c_int00
                STM   #N1,AR2      ;AR2 points to N1
                LD    *AR2+, T     ;T  reg =N1
                MPY    *AR2+, A    ;A= N1 *N2 in Q30 notation
                ADD   #1, 14, A    ;round the result
                STH   A, 1, *AR2   ;save N1 *N2 as Q15 number
                NOP
                NOP
                .end
```

5 d. 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

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

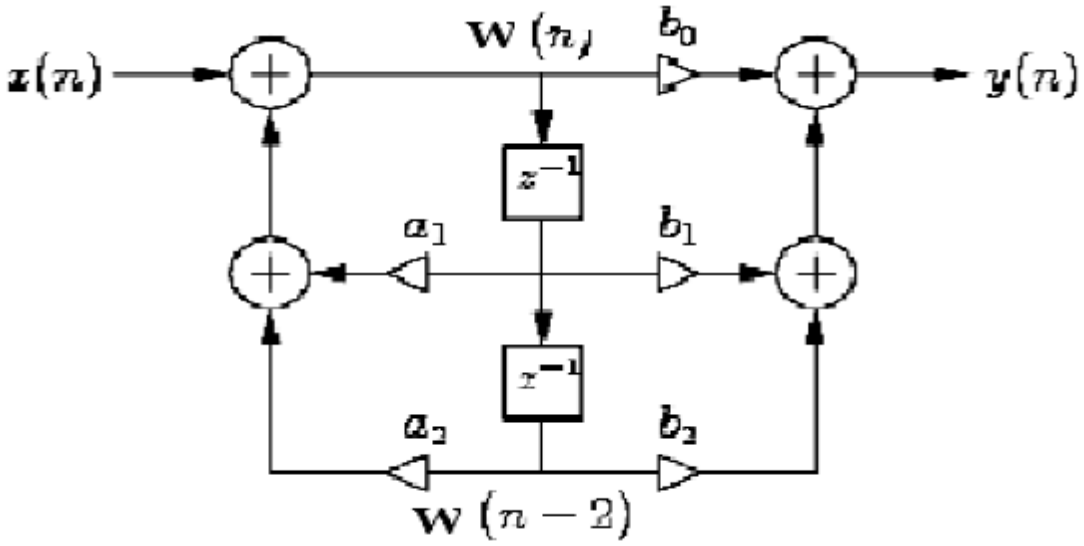


Figure 5.2 : Second order IIR filter

$w(n) = x(n) + a_1 w(n-1) + a_2 w(n-2)$

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

USN 4CR13EC174 10EC751

Seventh Semester B.E. Degree Examination, Dec.2017/Jan.2018
DSP Algorithms and Architecture

Time: 3 hrs. Max. Marks:100

Note: *Answer any FIVE full questions, selecting atleast TWO questions from each part.*

PART – A

1 a. With the help of block diagram and equations explain decimation and interpolation process. Also determine the interpolated sequence y(m), if the signal sequence $x(n) = \{0, 2, 4, 6, 8\}$ is interpolated using the interpolation filter sequence $b_k = \{0.5, 1, 0.5\}$. Interpolation factor L = 2. **(10 Marks)**

b. Explain with the block diagram of a DSP system. Also draw the typical signals in a DSP scheme. **(08 Marks)**

c. Assuming x(k) as a complex sequence, determine the number of complex and real mulitplies for computing DFT, using direct and radix – 2 FFT algorithms. assume N = 1024. **(02 Marks)**

2 a. Mention the basic features that should be provided in the DSP architecture to be used to implement the following N^{th} order FIR filter $y(n) = \sum_{i=0}^{N-1} h(i)x(n-i)$; n = 0, 1, 2, - - - . **(04 Marks)**

b. Explain the register pointer updating algorithm for circular buffer. **(06 Marks)**

c. With relevant block diagram, explain the various features of arithmetic and logic unit of DSP processor. **(06 Marks)**

d. Write a note on organization of the on-chip memory. **(04 Marks)**

3 a. Compare the architectural features of TMS320C25 and DSP56000. **(08 Marks)**

b. Draw the functional diagram of the barrel shifter of TMS320C54XX processor and explain the significance of each block. **(08 Marks)**

c. Assuming the current contents of AR3 to be 200h, what will be its contents after each of the following TMS320C54XX addressing modes is used. Assume that the contents of ARO are 40h i) *AR3 – 0 ii) *AR3 + iii) *+AR3(50h) iv) *AR3 – OB. **(04 Marks)**

4 a. Explain the pipeline operation of TMS320C54XX processor. Show the pipeline operation of the following sequence of instructions if the initial value of AR3 is 85h and the values stored in memory location 85h, 86h, 87h are 5, 6 and 7.

```
LD  * AR3 +, A
ADD # 1000h, A
STL A, * AR3 +.
```

b. Write the TCR register format and explain the functions of the various bits in the TCR register. **(08 Marks)**

c. Write a program to compute the sum of three product terms given by the equation : $y(n) = h_0x(n) + h_1x(n-1) + h_2x(n-2)$ where x(n), x(n – 1), x(n – 2) are data samples stored at three successive data memory locations h_0, h_1, h_2 are constants stored in data memory. Use direct addressing mode. **(06 Marks)**

PART – B

5 a. Represent each of the following as 16 – bit numbers in the desired Q – notation :
   i) 0.3125 as a Q_{15} number
   ii) –0.3125 as a Q_{15} number
   iii) 3.125 as a Q_7 number
   iv) –352 as a Q_6 number. **(04 Marks)**

b. Write a TMS320C54XX program for the implementation of an interpolating FIR filter of length 15 and interpolating factor 5. **(08 Marks)**

c. Write a program to multiply two Q_{15} numbers in TMS320C54XX processor. **(04 Marks)**

d. Briefly explain IIR filters. With the help of block diagram, explain second order IIR filters. **(04 Marks)**

6 a. Determine the following for a 128 – point FFT computations :
   i) Number of stages
   ii) Number of butterflies in each stage
   iii) Number of butterflies needed for the entire computation
   iv) Number of butterflies that need no twiddle factor. **(04 Marks)**

b. Write subroutine for bit reverse address generation and explain the same. **(06 Marks)**

c. Explain the butterfly computation in DIT FFT algorithm and write a subroutine that implements the butterfly computation. **(10 Marks)**

7 a. Draw the timing diagram of the memory interface signals for a read – read –write sequence of operations. Also explain the purpose of each signal. **(06 Marks)**

b. Explain the register sub-addressing technique for configuring DMA. **(04 Marks)**

c. Interface the TMS320C54XX to a 10 –bit ADC(TLC1550) and an 8 bit DAC (TLC7524). The sampled signal read from the ADC is to be written to the DAC after adjusting its size. The start of conversion is initiated by the TOUT signal. Write a flowchart for main program and interrupt service routine and also write the program. **(10 Marks)**

8 a. With a neat block diagram and timing diagram for transmit and receive operation, explain the signals involved in synchronous serial interface. **(08 Marks)**

b. With the help of block diagram, explain DSP based biotelemetry receiver system. **(06 Marks)**

c. Explain the image compression and reconstruction using JPEG encoder and decoder. **(06 Marks)**

* * * * *