1. Write the applications of barrel shifter

The barrel shifter is also used for scaling operations such as:

i. Prescaling an input data-memory operand or the accumulator value before an ALU operation.

ii. Performing a logical or arithmetic shift of the accumulator value.

iii. Normalizing the accumulator.

iv. Post scaling the accumulator before storing the accumulator value into data memory.

2. Define Overflow Handling

The ALU saturation logic prevents a result from overflowing by keeping the result at a maximum (or minimum) value. This feature is useful for filter calculations. The logic is enabled when the overflow mode bit (OVM) in status register ST1 is set.

When a result overflows:

1. If OVM = 0, the accumulators are loaded with the ALU result without modification.

If OVM = 1, the accumulators are located with either the most positive 32-bit value (00 7FFF FFFFh) or the most negative 32-bit value (FF 8000 0000h), depending on the direction of the overflow.

3. Write the operations of Compare, Select, and Store Unit (CSSU)

The compare, select, and store unit (CSSU) is an application-specific hardware unit dedicated to add/compare/select (ACS) operations of the Viterbi operator. Fig.11.13 shows the CSSU, which is used with the ALU to perform fast ACS operations.

The CSSU allows the C54x device to support various Viterbi butterfly algorithms used in equalizers and channel decoders.

The add function of the Viterbi operator (see fig.11.13) is performed by the ALU. This function consists of a double addition function (Met1+D! and Met2+D2). Double addition is completed in one machine cycle if the ALU is configured for dual 16-bit mode by setting the C16 bit in ST1. With the ALU configured in dual 16-bit mode, all the long-word (32-bit) instructions become dual 16-bit arithmetic instructions.
4. Write the Onchip peripherals of dsp processors

The C54x DSP has the following on-chip peripherals

i. General-purpose I/O pins; XF AND BIO

ii. Timer

iii. Clock generator

iv. Host port interface (HPI)

v. Synchronous serial port

vi. Buffered serial port (BSP)

vii. Multichannel buffered serial port (McBSP)

viii. Time-division multiplexed (TDM) serial port

Software-programmable wait-state generator

5. Describe Serial Ports of TMS320C54x processors

TMS320C54x DSP core CPU:

Standard synchronous serial port interface

Buffer serial port interface

Multichannel buffered serial Port (McBSP) interface

Time-division multiplexed serial port interface

6. Explain-Wait-State Generator

The software-programmable wait-state generator can extend external bus cycles by up to seven machine cycles (14 machine cycles on C5402, C5409, C5410, and C5420 devices), providing a convenient means to interface the C54x DSP to slower external devices. Devices that require more than seven wait states can be interfaced using the hardware READY line. When all external accesses are configured for zero wait states, the internal clocks to the wait-state generator are shut off. Shutting off these paths from the internal clocks allows the device to run with lower power consumption.
7. Write the steps of pipelining
The basic action of any microprocessor can be broken down into a series of four simple steps. They are
1. **The Fetch** phase (F) in which the next instruction is fetched from the address stored in the program counter.
2. **The decode** phase (D) in which the instruction in the instruction register is decoded and the address in the program counter is incremented.
3. **Memory read** (R) phase reads the data from the data buses and also writes data to the data buses.
4. **The Execute phase** (X) executes the instruction currently in the instruction register and also completes the write process.

8. Write the **Status and control registers**

   The C54x DSP has three status and control registers:
   - I. Status register 0 (ST0),
   - II. Status register 1 (ST1),
   - III. Processor mode status register (PMST).

   ST0 and ST1 contain the status of various conditions and modes; PMST contains memory-setup status and control information.

9. Define **Host Port Interface**

   The Host Port Interface (HPI) is an 8-bit parallel port that interfaces a host device or host processor to the C54xE DSP. Information is exchanged between the C54x DSP and the host device through on-chip C54x DSP memory that is accessible by both the host and the C54x DSP.

10. Write the **Addressing modes of DSP processors**:

    The addressing mode in TMS 32050 are:
    - (i) immediate addressing
    - (ii) indirect addressing
    - (iii) register addressing
    - (iv) memory mapped register addressing.
    - (v) direct addressing.
    - (vi) circular addressing.
1. Explain about Addressing mode of dsp processors

The addressing mode in TMS 32050 are,
(i)immediate addressing
(ii)indirect addressing
(iii)register addressing
(iv)memory mapped register addressing.
(v) direct addressing .
(vi) circular addressing

1. **immediate addressing**:

   immediate addressing is used to handle constant data. It allows the programme to operate on an actual value. The data can be either a 16-bit constant or constant length 7.9 or 13. Depending on the length of the data, the addressing mode is referred to as long immediate or short immediate addressing mode. In long immediate addressing, the data is contained in an proportion of the bits in a single word instruction. At the assembly code level, the developer uses a ‘#’ prefix to specify immediate addressing.

   Example:
   
   LD#80h,A : The instruction loads an immediate value 80 in to the accumulator.

2. **indirect addressing**:

   The indirect address mode uses the auxiliary register (ARS) to hold the address of operand in memory. In direct addressing, any location in the 64-k word data memory space can be accessed using a 16-bit address contained in AR. Each auxiliary register (AR0-AR7) provides flexible and powerful indirect addressing. To select a specific auxiliary register, the auxiliary register pointer (ARP) is loaded with a value from 0 to 7 for ARO through AR7 respectively. There are seven types of indirect addressing.

   a. auto increment
   b. auto decrement
   c. post indexing by adding the contents of ARO
   d. Post indexing by subtracting the contents of ARO
   e. Single indirect addressing with no increment
   f. Single indirect addressing with no decrement
3. Register addressing:

The register addressing mode uses operands in CPU registers either explicitly, such as with a direct reference to a specific register, or implicitly, with instructions that intrinsically refer to certain registers. That is, this addressing mode the address comes from one of two special purpose memory mapped registers in CPU: the block move address register (BMAR) and the dynamic bit manipulation register (DBMR). In either case, operand reference is simplified because 16-bit values can be used without specifying a full 16-bit operand address or immediate value.

For example, the instruction BLDP, BLPD, MADD, and MADS instructions use the BMAR to address an operand in program memory.

Memory mapped register addressing:

Memory mapped register addressing is used to access efficiently the CPU and on-chip peripheral registers. It operates like the direct addressing except that the upper 9-bits of the address that is accessed are assumed to be 0s. This allows us to access the memory mapped register of data page 0 directly without the overhead of changing the DP or auxiliary register. Only the seven lower bits of the complete code, including opcode and operand, can be represented using a single 16-bit word.

The following instructions operate in the memory mapped register addressing mode.

- LAMM - Load accumulator with memory mapped register
- LMMR - Load memory mapped register
- SAMM - Store accumulator in memory mapped register
- SMMR - Store memory mapped register.

**DIRECT ADDRESSING MODE:**

Direct addressing allows the CPU to access operand by specifying an offset from a base address that is defined in data pointer, DP.
(data pointer) is a 9-bit field contained in the status register (ST0). In this mode, the address of the operand is obtained by concatenating the 7-bit data memory address (dma) with the 9-bit of the data page pointer. The 16-bit data memory address is placed on an internal direct data memory address bus. Since data pointer is a 9-bit field, it points to one of 512 possible data memory pages, and the 7-bit address in the instruction points to one of 128 words within that data memory pages.

**Circular addressing mode:**

Circular addressing is the most sophisticated c5x addressing modes. Any algorithm such as convolution, correlation, and FIR filtering can be used circular buffer in memory to implement a sliding window, which contains most recent data to be processed. Five dedicated registers are allocated for implementation of circular addressing. They are:

- CBSR1 - CIRCULAR BUFFER 1 START REGISTER
- CBSR2 - CIRCULAR BUFFER 2 START REGISTER.
- CBER1 - CIRCULAR BUFFER 1 END REGISTER.
- CBER2 - CIRCULAR BUFFER 2 END REGISTER.
- CBCR - CIRCULAR BUFFER CONTROL REGISTER.

The register CBSR1 and CBSR2 are used to load the starting address of the circular buffer, and the register CBER1 and CBER2 are used to load the end address of the circular buffer. The 8-bit CBER enables and disables circular buffer operation. Additionally, one of the auxiliary register (ARS) is used as the pointer into the circular buffer.

To define a circular buffer, first we load the start and end addresses into the corresponding buffer register. Next, a value is loaded between the start and end register for the circular buffer into an AR and the corresponding circular buffer enable bit in the CBCR is set.
2. Explain about Instruction set of C50 processors:

Accumulator memory reference instruction:

- **ABS** - ABSOLUTE value of ACC; zero carry bit.
- **ADCB** - Add ACCB and carry bit to ACC
- **ADD** - Add data memory value, with left shift to ACC
- **ADDC** - Add data memory value and carry bit to ACC with sign extension suppressed.
- **ADDB** - Add ACCB to ACC
- **ANDB** - AND ACCB with ACC.
- **BSAR** - Barrel shift ACC right.
- **CMPL** - 1’s complement ACC.
- **CRGT** - Store ACCa in ACCB if ACC > ACCB
- **CERLT** - Store ACCa in ACCB if ACC < ACCB
- **ROL** - Rotate ACC left 1 bit.
- **ROLB** - Rotate ACCB and ACC left 1 bit.
- **SBB** - Subtract ACCB from ACC.
- **SFR** - Shift ACC right 1 bit.

Parallel logic unit (plu) instruction:

- **APL** - AND data memory value with DBMR, and store result in data memory location.
- **CPL** - Compare data memory value with DBMR.
- **LT** - Load data memory value to TREGO.
- **LPH** - Load data memory value to PREG high byte.
- **SPLK** - Store long immediate in data memory location.

Branch and call instruction:

- **B** - Branch unconditionally to program memory location.
- **BACC** - Branch to program memory location specified by ACCL.
- **BACCD** - Delay branch to program memory location specified by ACCL.
- **BANZ** - Branch to program memory location if AR
Multiply Accumulate Unit (MAC):

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

For FIR filters the output of the filter is given by

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

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

1. 16 * 16 bit 2’s complement inputs.
2. 16 * 16 bit multiplier with 32-bit product in 25 ns.
3. 32/40 bit accumulator.

In the TMS320C50, for example, the FIR equation can be efficient implemented using the instruction pair:

```
RPT    NMI
MACD HNMI, XNMI
```

The first instruction, RPTNMI, loads the (N-1) into the repeat instruction counter, and causes the multiply-accumulate with data move (MACD) instruction following it to be repeated N times. The MACD instruction performs a number of operations in one cycle:

1. Multiplies the data sample, \( x(n-k) \), in the data memory by the coefficient, \( h(k) \), in the program memory;
2. Adds previous product to the accumulator;
3. Implements the unit delay, symbolized by $z^{-1}$, by shifting the data sample, $x(n-k)$, up to update the tapped delay line.

**The Multiply-Accumulate (MAC) Function.**

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

A multiply-accumulate step performs the following:

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

The TMS320C54X multiply-accumulate (MAC) unit performs a $16 \times 16 \rightarrow 32$-bit fractional multiply-accumulate operation in a single instruction cycle. The multiplier supports signed/signed multiplication, signed/unsigned multiplication, and unsigned/unsigned multiplication. These operations allow efficient extended-precision arithmetic. Many instructions using the MAC unit can optionally specify automatic round-to-nearest rounding.

3. **Describe about PIPELINING**

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

The basic action of any microprocessor can be broken down into a series of four simple steps. They are
4. **The Fetch** phase (F) in which the next instruction is fetched from the address stored in the program counter.

5. **The decode** phase (D) in which the instruction in the instruction register is decoded and the address in the program counter is incremented.

6. **Memory read** (R) phase reads the data from the data buses and also writes data to the data buses.

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

In a modern processor, the above four steps get repeated over and over again until the program is finished executing. These are, in fact, the four stages in a classic RISC pipeline. Each of the above stages could be said to represent one phase in the “lifestyle” of an instruction. An instruction starts out in the fetch phase, moves to the decode phase, then to the memory read phase, and finally to the execute phase. Each phase takes a fixed, but by no means, equal amount of time.

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

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

The pipelining stages for different DSPs are shown in table 11.2. Note that TMS320C54x has two additional phases: pre-fetch (PF) phase which stores the address of the instruction to be fetched and the access phase (A) which reads the address of the operand and modify the auxiliary registers and stack pointer if required.

| Instruction 1 | F1 | D1 | R1 | X1 |
| Instruction 2 | F2 | D2 | R2 | X2 |
| Instruction 3 | F3 | D3 | R3 | X3 |
| Instruction 4 | F4 | D4 | R4 | X4 |

Table 11.2 Pipeline in different TMS320 Processors

<table>
<thead>
<tr>
<th>DSP Processor</th>
<th>Pipeline phases</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMS320C2000</td>
<td>F-D-R-X (4 levels)</td>
</tr>
<tr>
<td>TMS320C3x</td>
<td>F-D-R-X (4 levels)</td>
</tr>
<tr>
<td>TMS320C5x</td>
<td>F-D-R-X (4 levels)</td>
</tr>
<tr>
<td>TMS320C54x</td>
<td>PF-F-D-A-R-X (6 levels)</td>
</tr>
</tbody>
</table>
Pipelining leads to dramatic improvements in system performance. The more stages that we can break the pipeline into, the more theoretical speed we can get from it. For example, let’s suppose it takes 12 clock cycles to handle all the steps to process an instruction. In theory, if you use a 4-stage pipeline, your maximum throughput is 1 instruction every 3 cycles. But if you use a 6-stage pipeline, maximum throughput is 1 instruction every 2 cycles.

4. Explain the Architecture of TMS320C54x processor

The Texas Instruments TMS320C54x is a 16-bit fixed point digital signal processor. It was introduced in Japan in 1994. It is fabricated with an advanced modified Harvard architecture that has one program memory bus, three data memory buses, and four address buses. The fastest processor in the family runs at 160MHz with a 1.6-volt core supply voltage. The lowest-voltage family member runs at 120MHz and 1.5volts. The C54x DSP also has an on-chip bidirectional bus for accessing on-chip peripherals. The Program bus (PB) carries the instruction code and immediate operands from program memory. Three data buses (CB, DB, and EB) interconnect to various elements, such as the CPU, data address generation logic, program address generation logic, on-chip peripherals, and data memory.

Internal Memory Organization

The C54xDSP memory is organized into three individually selectable spaces: Program, data, and I/O space. The C54x devices can contain random access memory (RAM) and read-only memory (ROM). 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. Both the DARAM and SARAM can be configured as data memory or program/data memory.

On-Chip ROM

The on-chip ROM is part of the program memory space and, in some cases, part of the data memory space. On most devices, the ROM contains a boot loader that is useful for booting to faster on-chip or external RAM. On devices with large amounts of ROM, a portion of the ROM may be mapped into both data and program space.

On-Chip Dual-Access RAM (DARAM)

The DARAM is composed of several blocks. Each DARAM block can be accessed twice per machine cycle. The CPU and peripherals, such as a buffered serial port (BSP) and host-port interface (HPI), can read from and write to a DARAM memory address in the same cycle. The DARAM is always mapped in data space and is primarily intended to store data values. It can also be mapped into program space and used to store program code.
On-Chip Single-Access RAM (SARAM)

The SARAM is composed of several blocks. Each block is accessible once per machine cycle for either a read or a write. The SARAM always mapped in data space and is primarily intended to store data values. It can also be mapped into program space and used to store program code.

On-Chip Two-Way Shared RAM

The devices with multiple CPU cores include two-way shared RAM Blocks. All the shared memory is program write-protected or read only by the CPU, only the DMA controller can write to the shared memory. This shared RAM is most efficiently used when the two CPUs are executing identical programs. In this case, the amount of program memory required for the application is effectively reduced by 50% since both CPUs can execute from the same RAM.

Memory-Mapped Registers

The data memory space contains memory-mapped registers for the CPU and the on-chip peripherals. These registers are located on data page 0, simplifying access to them. The memory-mapped access provides a convenient way to save and restore the registers for context switches and to transfer information between the accumulators and the other registers.

Central Processing Unit

The C54x CPU contains a 40-bit arithmetic logic unit (ALU), two 40-bit accumulators, Barrel shifter, 17 17-bit multiplier, a 40-bit adder, Compare, select, and store unit (CSSU), an exponent encoder, a data address generation unit (DAGEN), and a program address generation unit (PAGEN).

Arithmetic logic unit (ALU)

The Figures shows the functional diagram of Arithmetic and logic unit. It implements a wide range of arithmetic and logical functions, most of which execute in a single clock cycle. After an operation is performed in the ALU, the result is usually transferred to a destination accumulator (accumulator A or B). The ALU can also function as two separate 16-bit ALUs and perform two 16-bit operations simultaneously.

ALU input takes several forms from several sources. The X input source to the ALU is either of two values: The shifter output (a 32-bit or 16-bit data-memory operand or a shifted accumulator value), A data-memory operand from data bus DB. The Y input source to the ALU is any of three values: The value in one of the accumulators (A or B), A data-memory operand from data bus CB or The value in the T register. When a 16-bit data-memory operand is fed through data bus CB or DB, the 40-bit ALU input is constructed in one of two ways:

1. If bits 15 through 0 contain the data-memory operand, bits 39 through 16 are zero filled (SXM=0) or sign-extended (SXM=1).
2. If bits 31 through 16 contain the data memory operand, bits 15 through 0 are zero filled, and bits 39 through 32 are either zero filled (SXM=0) or sign extended (SXM = 1)

**Overflow Handling**

The ALU saturation logic prevents a result from overflowing by keeping the result at a maximum (or minimum) value. This feature is useful for filter calculations. The logic is enabled when the overflow mode bit (OVM) in status register ST1 is set.

When a result overflows:

2. If OVM = 0, the accumulators are loaded with the ALU result without modification.

3. If OVM = 1, the accumulators are located with either the most positive 32-bit value (00 7FFF FFFFh) or the most negative 32-bit value (FF 8000 0000h), depending on the direction of the overflow.

**The Carry Bit**

The ALU has an associated carry bit that is affected by most arithmetic ALU instructions, including rotate and shift operations. It supports efficient computation of extended-precision arithmetic operations. Two conditional operands, C and NC, enable branching, calling, returning, and conditionally executing according to the status (set or cleared) of the carry bit.

**Dual 16-Bit Mode**

For arithmetic operations, the ALU can operate in a special dual 16-bit arithmetic mode that performs two 16-bit operations (for instance, two additions or two subtractions) in one cycle. You can select this mode by setting the C16 field of ST1. This mode is especially useful for the Viterbi add/compare/select operation.

**Accumulators**

A and B Accumulator A and accumulator B can be configured as the destination registers for either the multiplier/adder unit or the ALU. In addition, they are used for MIN and MAX instructions or for the parallel instruction LD||MAC, in which one accumulator loads data and the other performs computations. Each accumulator is split into three parts, as shown in fig. 11.10.

<table>
<thead>
<tr>
<th>39-32</th>
<th>31-16</th>
<th>15-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>AG</td>
<td>AH</td>
<td>AL</td>
</tr>
</tbody>
</table>
The guard bits are used as a head margin for computations. Head margins prevent some overflow in iterative computations such as autocorrelation. AG, BG, AH, BH, AL, and BL are memory-mapped registers that can be pushed onto and popped from the stack for context saves and restores by using PSHM and POPM instructions. These registers can also be used by other instructions that use memory-mapped registers (MMR) for page 0 addressing. The only difference between accumulators A and B is that bits 32-16 of A can be used as an input to the multiplier in the multiplier/ adder unit.

**Barrel Shifter**

The functional diagram of a barrel shifter is shown in fig.11.11. The 40-bit barrel shifter of C54 can perform arithmetic and logical shifts by up to 31 bits left or by up to 16 bits right in a single instruction cycle. Shifter inputs can come directly from data memory or from either of the two accumulators. Shifter outputs can be sent to the ALU or stored in memory. The shift count determines how many bits to shift. Positive shift values correspond to left shifts, whereas negative values correspond to right shifts. The shift count is specified as a 2s-complement value in several ways, depending on the instruction type.

The barrel shifter is also used for scaling operations such as:

v. Prescaling an input data-memory operand or the accumulator value before an ALU operation.

vi. Performing a logical or arithmetic shift of the accumulator value.

vii. Normalizing the accumulator.

viii. Post scaling the accumulator before storing the accumulator value into data memory.

**Multiplier/Adder Unit**

The TMS320C54x include a 17-bit * 17-bit multiplier, a dedicated 40-bit adder for nonpipelined MAC (multiply/accumulate) operation. The multiplier/adder unit is shown in Fig.11.12. The multiplier supports signed/signed multiplication, signed/unsigned multiplication and unsigned/unsigned multiplication. These operations allow efficient extended-precision arithmetic.

The multiplier output can be shifted left by one bit to compensate for the extra sign bit generated by multiplying two 16-bit 2s-complement numbers in fractional mode. (Fractional mode is selected when the FRCT bit = 1 in ST1.) The adder in the multiplier/adder unit contains
a zero detector, a rounder (2s complement), and overflow/saturation logic. Rounding is performed in some multiply, MAC, and multiply/subtract (MAS) instructions when the suffix R is included with the instruction. The LMS instruction also rounds to minimize quantization errors in updated coefficients. The adder’s inputs come from the multiplier’s output and from one of the accumulators. Once any multiply operation is performed in the unit, the result is transferred to a destination accumulator (A or B).

*Compare, Select, and Store Unit (CSSU)*

The compare, select, and store unit (CSSU) is an application-specific hardware unit dedicated to add/compare/select (ACS) operations of the Viterbi operator. Fig.11.13 shows the CSSU, which is used with the ALU to perform fast ACS operations.

The CSSU allows the C54x device to support various Viterbi butterfly algorithms used in equalizers and channel decoders.

The add function of the Viterbi operator (see fig.11.13) is performed by the ALU. This function consists of a double addition function (Met1+D1 and Met2+D2). Double addition is completed in one machine cycle if the ALU is configured for dual 16-bit mode by setting the C16 bit in ST1. With the ALU configured in dual 16-bit mode, all the long-word (32-bit) instructions become dual 16-bit arithmetic instructions.

*Exponent Encoder*

The exponent encoder is an application-specific hardware device dedicated to supporting the EXP instructions in a single cycle. With the EXP instruction, the exponent value in the accumulator can be stored in T as a 2s-complement value within a -8 through 31 range. The exponent is defined as the number of leading redundant bits – 8, which corresponds to the number of shifts required in the accumulator to eliminate nonsignificant sign bits. This operation results in a negative value when the accumulator value exceeds 32 bits.

*Status and control registers*

The C54x DSP has three status and control registers:

IV. Status register 0 (ST0),

V. Status register 1 (ST1).

VI. Processor mode status register (PMST).

ST0 and ST1 contain the status of various conditions and modes; PMST contains memory-setup status and control information.

C54x also includes eight auxiliary registers and a software stack to enable a highly-optimized C compiler. The eight 16-bit auxiliary registers (AR0-AR7) can be accessed by the CPU and
modified by the auxiliary register arithmetic units (ARAU). The primary function of the auxiliary registers is to generate 16-bit addresses for data space.

**The C54x pipeline**

The C54x DSP has a six-level deep instruction pipeline. The six stages of the pipeline are independent of each other, which allow overlapping execution of instructions. During any given cycle, from one to six different instructions can be active, each at a different stage of completion. The six levels and functions of the pipeline structure are: Program prefetch, program fetch, decode access, read and execute.

**Onchip peripherals**

The C54x DSP has the following on-chip peripherals

ix. General-purpose I/O pins; XF AND BIO

x. Timer

xi. Clock generator

xii. Host port interface (HPI)

xiii. Synchronous serial port

xiv. Buffered serial port (BSP)

xv. Multichannel buffered serial port (McBSP)

xvi. Time-division multiplexed (TDM) serial port

xvii. Software-programmable wait-state generator.

The TMS320C54x provides three low-power modes invoked by the IDLE1, IDLE2 and IDLE3 instructions. In IDLE1 mode, on-chip peripherals (the serial port and timer) and interrupt lines remain active, and any unmasked interrupt wakes the processor. In IDLE2 mode, the on-chip peripherals are turned off, and only an interrupt on an external interrupt line wakes the processor. IDLE3 mode is similar to IDLE2 mode but it also turns off the on-chip crystal oscillator and PLL circuitry.

**General-Purpose I/O**

The C54xE 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. It is especially useful as an alternative to using an interrupt when time critical loops must not be disturbed.

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.
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.

The clock generator

The clock generator on the C54x devices consists of an internal oscillator and a phase locked loop (PLL) circuit. Currently, there are two different types of PLL circuits on C54x devices. Some devices have hardware-configurable PLL circuits while others have software-programmable PLL circuits.

Host Port Interface

The Host Port Interface (HPI) is an 8-bit parallel port that interfaces a host device or host processor to the C54xE DSP. Information is exchanged between the C54x DSP and the host device through on-chip C54x DSP memory that is accessible by both the host and the C54x DSP.

Serial Ports

TMS320C54x DSP core CPU:

Standard synchronous serial port interface

Buffer serial port interface

Multichannel buffered serial Port (McBSP) interface

Time-division multiplexed serial port interface.

These peripherals are controlled through registers that reside in the memory map. The serial ports are synchronized to the core CPU by way of interrupts.

Synchronous Serial Ports

Synchronous serial ports are high-speed, full-duplexed serial ports that provide direct communication with serial devices such as codec's, analog-to-digital (A/D) converters, and other serial systems. When more than one synchronous serial port resides on a C54x device, these ports are identical but independent. Each synchronous serial port can operate up to one-fourth the machine cycle rate (CLKOUT). The synchronous serial port transmitter and receiver are double buffered and individually controlled by mask able external interrupt signals. Data is framed either as bytes or as words.

Buffered Serial Ports
A buffered serial port (BSP) is a synchronous serial port that is enhanced with an auto buffering unit and is clocked at the full CLKOUT rate. It is full-duplexed and double-buffered to offer flexible data stream length. The auto buffering unit supports high-speed transfers and reduces the overhead of servicing interrupts.

**Multichannel Buffered Serial Ports (McBSPs)**

The McBSP is an enhanced buffered serial port that includes the following standard features: buffered data registers; full duplex communication, and independent clocking and framing for receive and transmit.

**TDM Serial Ports**

The time-division multiplexed (TDM) serial port is a synchronous serial port that is enhanced to allow time-division multiplexing of the data with up to seven other C54x devices with TDM ports. It can be configured for either synchronous operations or for TDM operations and is commonly used in multiprocessor applications.

**External Bus Interface**

The C54xE DSP can address up to 64K words of data memory, 64K words of program memory (up to 8M words in some devices), and up to 64K words of 16-bit parallel I/O ports. Accesses to either external memory or I/O ports take place through the external interface. Individual space-select signals, DS, PS, and IS, allow the selection of physically separate spaces.

The C54x DSP external interface consists of data buses, address buses, and a set of control signals for accessing off-chip memory and I/O ports.

**Wait-State Generator**

The software-programmable wait-state generator can extend external bus cycles by up to seven machine cycles (14 machine cycles on C5402, C5409, C5410, and C5420 devices), providing a convenient means to interface the C54x DSP to slower external devices. Devices that require more than seven wait states can be interfaced using the hardware READY line. When all external accesses are configured for zero wait states, the internal clocks to the wait-state generator are shut off. Shutting off these paths from the internal clocks allows the device to run with lower power consumption.

The software-programmable wait-state generator is controlled by the 16-bit software wait-state register (SWWSR), which is memory-mapped to address 0028h in data space.
5. Explain Memory Organization of dsp processors

The organization of a processor's memory subsystem can have a large impact on its performance. As mentioned earlier, the MAC and other DSP operations are fundamental to many signal processing algorithms. Fast MAC execution requires fetching an instruction word and two data words from memory at an effective rate of once every instruction cycle.

There are a variety of ways to achieve this, including multiported memories (to permit multiple memory accesses per instruction cycle), separate instruction and data memories (the “Harvard” architecture and its derivatives), and instruction caches (to allow instructions to be fetched from cache instead of from memory, thus freeing a memory access to be used to fetch data). Figures 3 and 4 show how the Harvard memory architecture differs from the “Von Neumann” architecture used by many microcontrollers.

Another concern is the size of the supported memory, both on- and off-chip. Most fixed-point DSPs are aimed at the embedded systems market, where memory needs tend to be small. As a result, these processors typically have small-to-medium on-chip memories (between 4K and 64K words), and small external data buses. In addition, most fixed-point DSPs feature address buses of 16 bits or less, limiting the amount of easily-accessible external memory.
FIGURE 3. A Harvard architecture, common to many DSP processors. The processor can simultaneously access the two memory banks using two independent sets of buses, allowing operands to be loaded while fetching instructions.