Introduction to Programmable DSP Processor

A digital signal processor (DSP) is a specialized microprocessor designed specifically for digital signal processing, generally in real time computing. They contain special architecture and instruction set so as to execute computation-intensive DSP algorithms more efficiently. The programmable DSPs (P-DSPs) can be divided into two broad categories. They are

(i) General purpose DSPs
(ii) Special purpose DSPs.

General purpose DSPs:

These are basically high speed microprocessors with architecture and instruction sets optimized for DSP operations.

They include fixed point processors such as Texas instruments TMS320C5X, TMS320C54X and Motorala DSP563X and floating point processors such as Texas instruments TMS320C4X, TMS320C67XX, and analog devices ADSP21XXX.

Special purpose DSPs:

This type of processors consist of hardware (i) designed for specific DSP algorithms such as FET, (ii) designed for specific applications such as PCM and filtering.

Examples for special purpose DSPs are MT93001, PDSP16515A and UPDSP16256.

Advantages of DSP Processors over Microprocessors:

- The architecture of DSP processors supports fast processing of arrays.
- The DSP processors require single clock cycle to execute instructions.
- The DSP processors support parallel execution of instructions.
- The DSP processors have separate data and program memories.
- The DSP processors support simultaneous fetching of multiple operands.
- The DSP processors have three separate computational units. Arithmetic logic unit, multiplier and accumulator, and shifter.
- The DSP processors consist of powerful interrupt structure and timers.
- The DSP processors have multiprocessing ability.
- The DSP processors have on chip program memory and data memory.

Multiplier and Multiplier Accumulator (MAC)

Array multiplication is one of the most common operations required in digital signal processing applications. An example is: convolution and correlation which require array multiplication operation.

One of the important requirements of these array multipliers is that they have to process the signals in real time. The array multiplication should be completed before the next sample of the input signal arrives at the input to the array. This requires the multiplication as well as accumulation to be carried out using hardware elements. There are two approaches to solve this problem.

- A dedicated MAC unit may be implemented in hardware, which integrates multiplier and accumulator in a single hardware unit.
- Have separate multiplier and accumulator. In this approach, the output of the multiplier is stored into the product register and the content of the product register is added to accumulator register in the central ALU

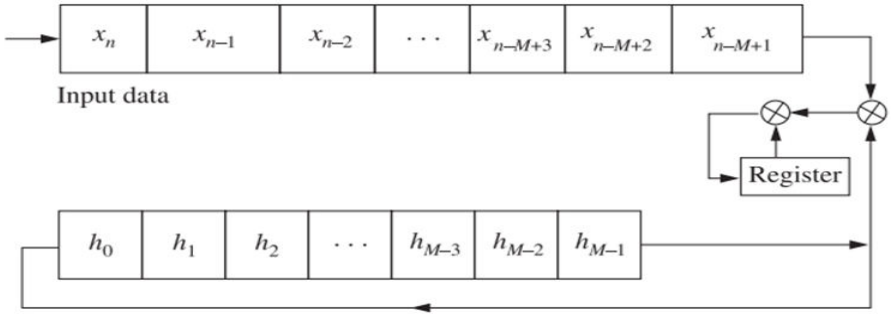


Fig: Implementation of convolver with single Multiplier/adder

Implementation of the convolver with single multiplier/adder is shown in above Figure. In the above figure, the array corresponding to the present and the past $M - 1$ samples of the input is given by

$$x_n = \{x_n x_{n-1} x_{n-2} \ \ldots \ x_{n-M+3} x_{n-M+2} x_{n-M+1}\}$$

and the array corresponding to the impulse response of the sequence is given by

$$h = \{h_0 h_1 h_2 \ \ldots \ h_{M-3} h_{M-2} h_{M-1}\}$$

The output at the n^{th} sampling instant, y_n is obtained by multiplying the array x_n with the array h. To obtain y_{n+1}, the input signal array x_{n+1} is multiplied with the array h. The vector x_{n+1} is obtained by shifting the array x_n towards right so that the $(n + 1)^{th}$ sample of the input data x_{n+1} becomes the first element and all the elements of x_n are shifted right by one position so that the i^{th} element of x_n becomes $(i + 1)^{th}$ element of x_{n+i} . The content of the product register is added to the accumulator before the new product is stored.

Modified Bus Structures and Memory Access Schemes in PDSPs :

The MACD instruction, that is, the MAC operation with data move requires four memory accesses per instruction cycle. The four memory accesses/clock periods required for the MACD instruction are as follows:

- Fetch the MACD instruction from the program memory.
- Fetch one of the operands from the program memory.
- Fetch the second operand from the data memory.
- Write the content of the data memory with address ‘dma’ into the location with the address ‘dma + 1’

The Below Figure shows the Von Neumann architecture. This architecture consists of 3 buses: The data bus, the address bus and the control bus.

In this architecture, the CPU can be either reading an instruction or reading/writing data from/to memory. Both cannot occur at the same time since the instruction and data use the same signal path ways and memory. The execution of each instruction requires four clock cycles because there is a single address busand there is a single data bus for accessing the program as well as data memory area.

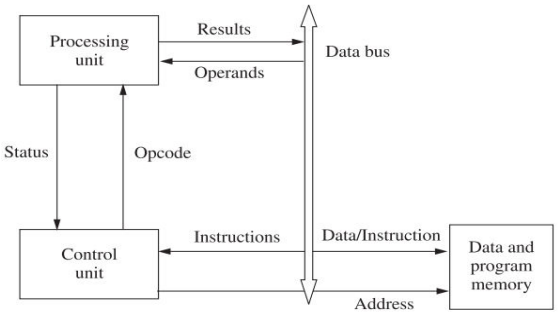


Fig: Von Neumann architecture.

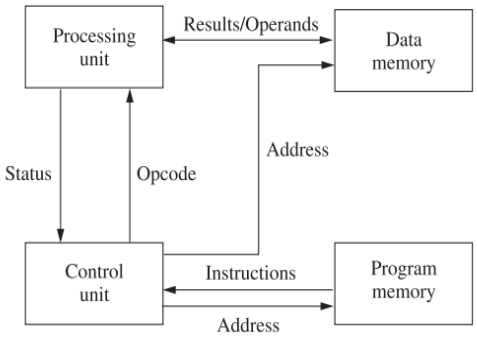


Fig: Harvard architecture.

The number of clock cycles required for the memory access can be reduced by using more than one bus for both address and data.

In the Harvard architecture, there are two separate buses for the program and data memory. Hence the content of program memory and data memory can be accessed in parallel. The instruction code is fed from the program memory to the control unit and the operand is fed to the processing unit from the data memory.

It is less flexible and needs two independent memory banks. These two resources are not interchangeable. The processing unit consisting of the registers and processing elements such as MAC units, multiplier, ALU, shifters, etc., are also referred to as data path.

The below figure shows the modified Harvard architecture. One set of bus is used to access a memory that has both program and data and another set of bus is used to access data alone. In modified Harvard architecture, data can also be transferred from one memory to another memory. This architecture is used in several P-DSPs. It may also have multiple bus system for program memory alone or for data memory alone. These multiple bus system increases complexity of CPU, but allow it to access several memory locations simultaneously, thereby increasing the data throughput between memory and CPU.

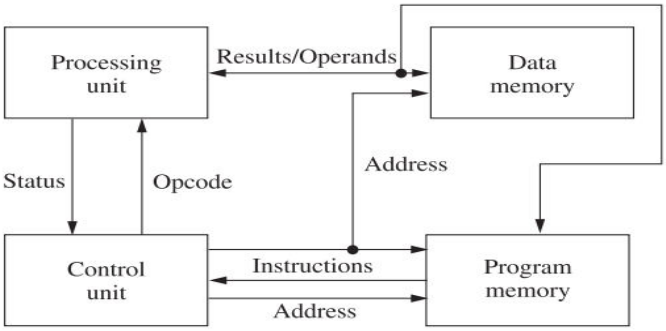


Fig: Modified Harvard architecture.

Multiple Access Memory and Multiported Memory:

The different techniques adopted for increasing the number of memory accesses/instruction cycle are:

(i) Multiple access memory

(ii) Multiported memory.

Multiple Access Memory:

The number of memory accesses/clock period can be increased by using a high speed memory that permits more than one memory access/clock period.

The memory that permits more than one access/clock period is called multiple access memory.

Two memory accesses per clock period can be achieved using the DRAM, the dual access RAM. By using the Harvard architecture, multiple access RAM may be connected to the processing unit of the P-DSP. Four memory accesses/clock period can also be achieved with dual access RAM, when it is connected to a programmable DSP with two independent address and data buses.

Multiported Memory:

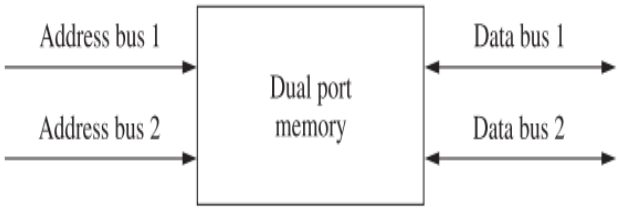


Fig: Dual Port Memory

The number of accesses/clock period can also be increased by using multiported memory.

The dual port memory shown in the above figure has two independent data and address buses and hence two memory accesses can be achieved in one clock period. Multiported memories dispense with the need for storing the program and data in two different memory chips in order to permit simultaneous access to both program and data memory.

One of the major limitations of the dual ported memory is the increase in the cost compared to two single port memories of the same total capacity. This is due to the increased number of pins and larger chip area required for the dual port memory. Larger and more expensive package and a larger die size is required for larger number of I/O pins.

VLIW Architecture:

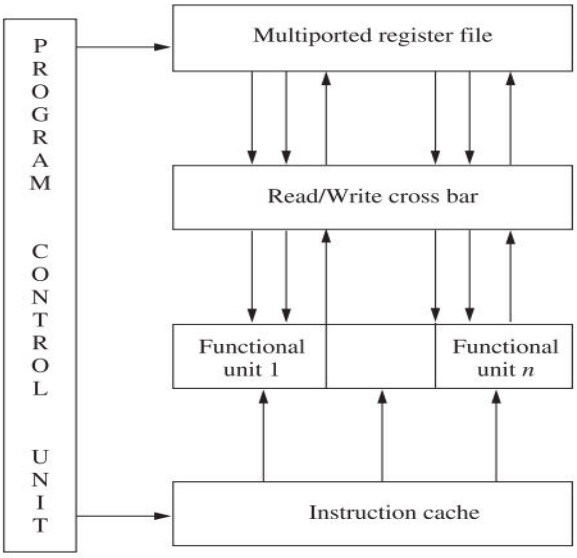


Fig: VLIW Architecture

Very long instruction word (VLIW) architecture is another architecture used for P-DSPs and the above figure shows VLIW architecture.

The VLIW processor consists of architecture that reads a relatively large group of instructions and executes them at the same time. These P-DSPs have a number of processing units (ALUs, MAC units, shifters etc). The VLIW is accessed from memory and is used to specify the operands and operations to be performed by each of the data paths. The VLIW processing increases the number of instructions that are processed per cycle. The multiple functional units share a common multiported register file for fetching the operands and storing the results.

The read/write cross bar provides parallel random access by multiple functional units to the multiported register file. Execution of the operations in the functional units is carried out concurrently with the load/store operation of data between a RAM and the register file.

The performance gains that can be achieved with VLIW architecture depends on the degree of parallelism in the algorithm selected for a DSP application and the number of functional units. The throughput will be higher only if the algorithm involves execution of independent operations.

Pipelining:

Instruction pipelining is a mechanism used for increasing the efficiency of the advanced microprocessors as well as P-DSPs.

Pipelining a processor means breaking down its instruction into a series of discrete pipeline stages which can be completed in sequence by specialized hardware.

An instruction cycle starting with the fetching of an instruction and ending with the execution of instruction including the time for storage of the results can be split into a number of microinstructions.

Execution of each of the microinstructions is also referred to as one phase of an instruction. For example, an instruction cycle requiring four microinstructions can be said to be in four phases as follows.

- **Fetch phase:** In this phase, the instruction is fetched from program memory.
- **Decode phase:** In this phase, the instruction is decoded.
- **Memory read phase:** In this phase, the operand required for the execution of the instruction may be read from the data memory.
- **Execution phase:** In this phase, the execution as well as storage of the results in either one of the register or memory is carried out.

In a modern processor, the above four steps get repeated over and over again till the program is finished executing. Each one of the above microinstructions may be carried out separately by four functional units. If we assume that each of the above phases take equal time for completion, then if there is no pipelining, each of the functional units is busy only for 25% of the time. The functional units can be kept busy almost all the time by using pipelining and processing a number of instructions simultaneously in the CPU.

If one clock cycle of the processor corresponds to T, in a period of 12T, only three instructions can be executed in a machine without pipelining, whereas in the same period, nine instructions can be carried out with pipelining. Hence the throughput is increased by a factor of 3 in this case.

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

Instruction cycles of processor with no 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

Instruction cycles of a processor with pipelining

The performance and programming simplification is achieved by pipeline operation with the elimination of pipeline interlocks; the control of pipeline is simplified.

The bottlenecks in the program fetch, multiply operations and data access are eliminated by increased pipelining.

The number of instructions that are processed simultaneously in the CPU, also referred to as depth of the instruction pipeline, differs in different families of DSPs.

Special Addressing Modes in PDSPs:

The special addressing modes in P-DSPs are as follows:

- Short immediate addressing
- Short direct addressing
- Memory mapped addressing
- Indirect addressing
- Bit reversed addressing
- Circular addressing

Short immediate addressing

In short immediate addressing mode, the operand is specified as a short constant that forms part of a single word instruction. The length of the short constant depends on the programmable DSP and the instruction type.

In TMS320C5XDSPs, an 8 bit constant can be specified as one of the operands in the single word instruction like AND, OR, addition, Subtraction etc.

Short direct addressing

In short direct addressing mode, the lower order address of the operand is specified in the single word instruction.

In TMS320DSPs, the higher order 9 bits of the memory are stored in the data page pointer and only the lower 7 bits are specified as part of the instruction

Memory mapped addressing

In this addressing mode, the CPU registers and the I/O registers are accessed as memory locations by storing them in either the starting page or the final page of the memory space.

For example, in TMS320C5X, page 0 corresponds to the CPU registers and I/O registers.

In the case of Motorola DSP5600X, the last page of the memory space containing 64 locations is used as the memory map for the CPU and I/O registers.

Indirect addressing

This addressing mode has a number of options in P-DSPs. This mode permits an array of data to be efficiently processed, fetched and stored. The address of the operands can be stored in one of the registers called indirect address registers.

In the case of TI processors, the indirect address registers are called auxiliary registers ARs. When the operands fetched by these registers are being executed, these registers can be updated. This is made possible by having an additional ALU in the CPU core specifically for the indirect address registers of ARs. The increment or decrement of ARs can be performed either in steps of 1 or in steps specified by the content of an offset register.

In P-DSPs from Texas instruments, the offset register is known as INDEX register and in the case of analog devices the offset register is known as modifier register. The content of the indirect address registers may also be updated by a constant using bit reversed addressing mode.

Bit reversed addressing

The binary pattern corresponding to a particular decimal number is obtained by writing the natural binary equivalent of the number in the reverse order. Therefore, the least significant bit of the bit reversed number becomes the most significant bit of the natural binary number and vice versa. In this addressing mode, the address is incremented or decremented by the number represented in the bit reversed form.

Circular addressing

In real time processing of signals, the input signal is continuously stored in memory. The processed data is stored in another memory space continuously and may be written on to the output device. In this case, the input as well as output program will be simple. However, since the input as well as the output memory space is finite it would be exhausted after processing the input signal for some time, if the data is written into the memory by using linear addressing mode.

This problem may be overcome by checking continuously whether the range of either the input or the output memory space is exceeded. In that case, the new data is to be stored starting from the beginning of the particular memory space. Checking this condition is an overhead that can be overcome using the circular addressing mode.

In this mode, the memory can be organized as a circular buffer with the beginning memory address and the ending memory address corresponding to this buffer designed by the programmer. In the circular addressing mode, 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.

Onchip Peripherals:

The P-DSPs have a number of on-chip peripherals that relieve the CPU from routine functions. Some of the on-chip peripherals in P-DSPs and their functions are as follows.

Onchip Timer

Two of the common applications of timers are generation of periodic interrupts to the DSPs and generation of the sampling clocks for A/D converters. The timer can be programmed by the P-DSPs.

Serial Port:

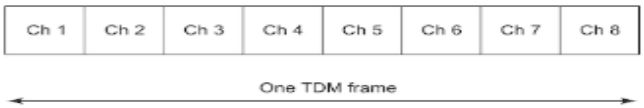
The serial port enables the data communication between the P-DSP and an external peripheral such as A/D converter, D/A converter or an RS232C device. These ports normally have input and output buffers so that the P-DSP writes or reads from the serial port in parallel form and the serial port sends and receives data to the peripherals in serial form and also inbuilt serial to parallel to serial and serial to parallel converters Serial port also has an ability to generate an interrupt if input buffer is full or output buffer is empty.

The serial port can operate in two modes

- Asynchronous Mode
  - Trasmit data and receive data lines are used
  - Bit clock is transmitted from either end
- Synchronous Mode
  - Bit clock and frame sync signal (indicates the beginning of first bit) transmitted from IO to serial port and vice versa

TDM Serial Port:

The TDM serial port is a special serial port the P-DSPs have. This port permits a P DSP to communicate with other devices or P-DSPs by using time division multiplexing. One of the devices can generate frame sync pulse that indicates beginning of a TDM frame and bit clock, duration for which a bit is to be transmitted. The following figure shows a TDM frame which is splitted into equal slots and each slot can be allotted for one of the devices.

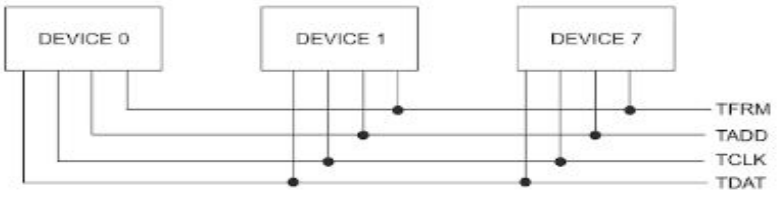


In the above figure, there are 8 slots/frame and is referred to as a TDM with eight channels. In each of the slots, a number of bits may be transmitted by a channel. The TDM serial port normally uses four lines for the purpose of Serial communication

TFRM: frame Sync signal

Tclock: The bit clock

TADD: the address of the serial device that is outputting data in a particulat TDM slot



Interconnecting 8 devices using TDM serial using 4-biPbus

TADD and TDAT signals are bidirectional and are tristate controlled so that only one of the devices transmit the data and address in these lines at a time. Any one of the devices can generate TFRM and clock signals and they are used by the other devices as a reference.

Parallel Port:

Parallel ports enable communication between the P-DSP and other devices to be faster compared to the serial communication by using a number of lines in parallel. In addition they have additional lines, which are for strobing or for hand shaking purpose.

Bit I/O Port:

These are additional I/O ports the P-DSPs have that are single bit wide. These port bits may be individually set, reset or read. These bits are normally used for control purposes, but they can also be used for data transfer.

HOST Port:

Host port is a special parallel port the P-DSPs have. This enables the P-DSPs to communicate with a microprocessor or a PC, which is called a host. In addition to data communication, the host can generate interrupts and also cause the P-DSP to load a program from ROM to the RAM on reset.

Comm Port:

These are parallel ports that are used for interprocess communication between a number of identical P-DSPs in a multiprocessor system.

On chip D/A and A/D Converters:

Some of the P-DSPs targeted towards voice applications such as cellular telephones and tapeless answering machines have A/D and D/A converters inside the P-DSPs.

PDSPS with RISC and CISC

P-DSPs may be implemented using either the RISC processor or the CISC processors. The relative advantages of each of these processors are given below.

Advantages of RISC Processors:

- In RISC processor, the control unit uses only around 20% of the chip area because of the reduced number of instructions. Hence the remaining area can be used for incorporating other features.
- The delayed branch and call instructions are used to improve the speed of the RISC processors.
- The execution time required for all the instructions of RISC processor is same because all the instructions are of uniform length.
- The RISC processors have smaller and simpler control units, which have fewer gates.
- The speed of the RISC processor is high because of smaller control unit and smaller propagation delays.
- Since a simplified instruction set allows for a pipelined superscalar design, RISC processors often achieve two to four times the performance of CISC processor using comparable semiconductor technology and the same clock rates.
- Because the instruction set of a RISC processor is simpler, it uses much less chip space than a CISC processor. Extra functional units such as memory management units or floating point arithmetic units can be placed on the same chip.
- The throughput of the processor can be increased by applying pipelining and parallel processing.
- Since RISC processors can be designed more quickly, they can take advantage of new technological developments sooner than corresponding CISC design.
- **High level language (HLL) support: -** The programs can be written in C and C++. It relieves the programmer from learning the instruction set of a P-DSP which in turn increases the throughput of the programmer.
- **Advantages of RISC Processors:**

Advantages of CISC Processors:

- The CISC processors have a very rich instruction set that even support high level language constructs similar to "if condition true then do", "for" and "while".
- The CISC processors have instructions specifically required for DSP applications such as MACD, FIRS, etc.
- The assembly language program of a CISC processor is very short and easy to follow.
- For RISC architecture compliers are essential. So this becomes costly. For CISC compilers are not required. Hence they are of low cost.
- New CISC processors are designed to be upward compatible with the older processors. This makes the learning curve steeper.
- Microprogramming is easier to implement and much less expensive than hardwiring a control unit.
- Since microprogram instruction sets can be written to match the constructs of high level languages, the complier need not be very complicated.
- As each instruction is more capable, fewer instructions could be used to implement a given task. This made more efficient use of the relatively slow main memory.

TMS320C5X Processor

The TMS320 family consists of two types of single-chip DSPs: 16-bit fixed point and 32-bit floating-point. These DSPs possess the operational flexibility of high-speed controllers and the numerical capability of array processors. Combining these two qualities, the TMS320 processors are inexpensive alternatives to custom-fabricated VLSI and multichip bit-slice processors. The following characteristics make this family the ideal choice for a wide range of processing applications:

- Very flexible instruction set
- Inherent operational flexibility
- High-speed performance
- Innovative, parallel architectural design
- Cost-effectiveness

TMS320C5x Overview

The ’C5x generation consists of the ’C50, ’C51, ’C52, ’C53, ’C53S, ’C56,’C57, and ’C57S DSPs, which are fabricated by CMOS integrated-circuit technology. Their architectural design is based on the ’C25. The operational flexibility and speed of the ’C5x are the result of combining an advanced Harvard architecture (which has separate buses for program memory and data memory), a CPU with application-specific hardware logic, on-chip peripherals, on-chip memory, and a highly specialized instruction set. The ’C5x is designed to execute up to 50 million instructions per second (MIPS).

Architectural Overview

The ’C5x uses an advanced, modified Harvard-type architecture based on the ’C25 architecture and maximizes processing power with separate buses for program memory and data memory. The instruction set supports data transfers between the two memory spaces. Below Figure shows a functional block diagram of the ’C5x.

Bus Structure

Separate program and data buses allow simultaneous access to program instructions and data, providing a high degree of parallelism.

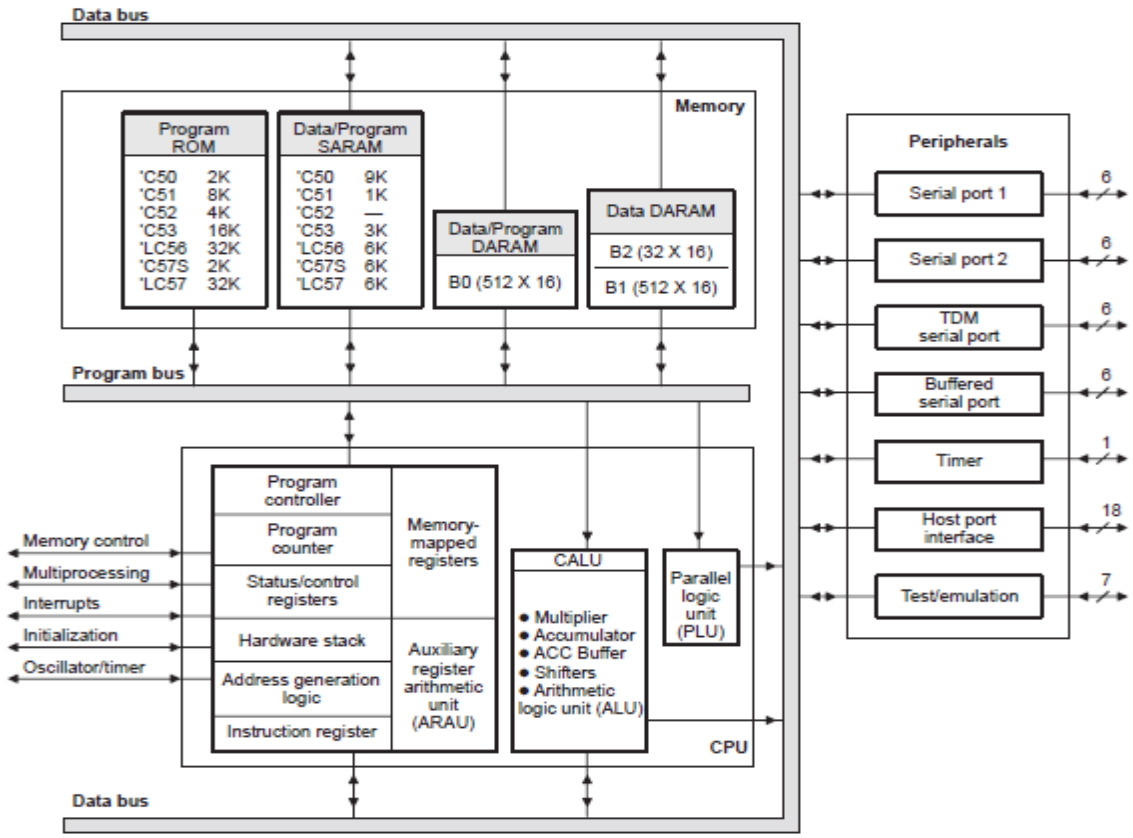


Fig: 'C5x Functional Block Diagram

For example, while data is multiplied, a previous product can be loaded into, added to, or subtracted from the accumulator and, at the same time a new address can be generated. Such parallelism supports a powerful set of arithmetic, logic, and bit-manipulation operations that can all be performed in a single machine cycle.The 'C5x architecture is built around four major buses:

- Program bus (PB)
- Program address bus (PAB)
- Data read bus (DB)
- Data read address bus (DAB)

The PAB provides addresses to program memory space for both reads and writes. The PB also carries the instruction code and immediate operands from program memory space to the CPU. The DB interconnects various elements of the CPU to data memory space. The program and data buses can work together to transfer data from on-chip data memory and internal or external program memory to the multiplier for single-cycle multiply/accumulate operations.

Central Processing Unit (CPU)

The ’C5x CPU consists of these elements:

- Central arithmetic logic unit (CALU)
- Parallel logic unit (PLU)
- Auxiliary register arithmetic unit (ARAU)
- Memory-mapped registers
- Program controller

The ’C5x CPU maintains source-code compatibility with the ’C1x and ’C2x generations while achieving high performance and greater versatility. Improvements include a 32-bit accumulator buffer, additional scaling capabilities, and a host of new instructions.

Central Arithmetic Logic Unit (CALU)

The CPU uses the CALU to perform 2s-complement arithmetic. The CALU consists of these elements:

- 16-bit ×16-bit multiplier
- 32-bit arithmetic logic unit (ALU)
- 32-bit accumulator (ACC)
- 32-bit accumulator buffer (ACCB)
- Additional shifters at the outputs of both the accumulator and the product register (PREG)

Parallel Logic Unit (PLU)

The CPU includes an independent PLU which operates separately and in parallel with ALU. The PLU performs Boolean operations or the bit manipulations required of high-speed controllers. The PLU can set, clear, test, or toggle bits in a status register, control register, or any data memory location. The PLU provides a direct logic operation path to data memory values without affecting the contents of the ACC or PREG. Results of a PLU function are written back to the original data memory location.

Auxiliary Register Arithmetic Unit (ARAU)

The CPU includes an unsigned 16-bit arithmetic logic unit that calculates indirect addresses by using inputs from the auxiliary registers (ARs), index register (INDX), and auxiliary register compare register (ARCR). The ARAU can auto index the current AR while the data memory location is being addressed and can index either by ±1 or by the contents of the INDX. As a result, accessing data does not require the CALU for address manipulation. So the CALU is free for other operations in parallel.

Memory-Mapped Registers

The 'C5x has 96 registers mapped into page 0 of the data memory space. All 'C5x DSPs have 28 CPU registers and 16 input/output (I/O) port registers but have different numbers of peripheral and reserved registers). Since the memory-mapped registers are a component of the data memory space, they can be written to and read from in the same way as any other data memory location. The memory-mapped registers are used for indirect data address pointers, temporary storage, CPU status and control, or integer arithmetic processing through the ARAU.

Program Controller

The program controller contains logic circuitry that decodes the operational instructions, manages the CPU pipeline, stores the status of CPU operations, and decodes the conditional operations. Parallelism of architecture performs three concurrent memory operations in any given machine cycle: fetch an instruction, read an operand and write an operand. The program controller consists of these elements:

- Program counter
- Status and control registers
- Hardware stack
- Address generation logic
- Instruction register

On-Chip Memory

The 'C5x architecture contains a considerable amount of on-chip memory to aid in system performance and integration:

- Program read-only memory (ROM)
- Data/program dual-access RAM (DARAM)
- Data/program single-access RAM (SARAM)

The 'C5x has a total address range of 224K words × 16 bits. The memory space is divided into four individually selectable memory segments:

- 64K-word program memory space
- 64K-word local data memory space
- 64K-word input/output ports
- 32K-word global data memory space.

Program ROM

All 'C5x DSPs carry a 16-bit on-chip maskable programmable ROM. The 'C50 and 'C57S DSPs have boot loader code resident in the on-chip ROM, all other 'C5x DSPs offer the boot loader code as an option. This memory is used for booting program code from slower external ROM or EPROM to fast on-chip or external RAM. Once the custom program

has been booted into RAM, the boot ROM space can be removed from program memory space by setting the MP/MC bit in the processor mode status register (PMST). The on-chip ROM is selected at reset by driving the MP/MC pin low. If the on-chip ROM is not selected, the 'C5x devices start execution from off-chip memory.

Data/Program Dual-Access RAM

All 'C5x DSPs carry a 1056-word×16-bit on-chip dual-access RAM (DARAM).The DARAM is divided into three individually selectable memory blocks:

- 512-word data or program DARAM block B0
- 512-word data DARAM block B1
- 32-word data DARAM block B2.

The DARAM is primarily intended to store data values but, when needed, can be used to store programs as well. DARAM blocks B1 and B2 are always configured as data memory. DARAM On-Chip Memory block B0 can be configured by software as data or program memory. The DARAM can be configured in one of two ways:

- All 1056 words×16 bits configured as data memory
- 544 words×16 bits configured as data memory and 512 words × 16 bits configured as program memory

Data/Program Single-Access RAM

All 'C5x DSPs except the 'C52 carry a 16-bit on-chip single-access RAM (SARAM). Code can be booted from an offchip ROM and then executed at full speed, once it is loaded into the on-chip SARAM. The SARAM can be configured by software in one of three ways:

- All SARAM configured as data memory
- All SARAM configured as program memory
- SARAM configured as both data memory and program memory

On-Chip Peripherals

All 'C5x DSPs have different onchip peripherals connected to their CPUs. The 'C5x DSP on-chip peripherals available are:

- Clock generator
- Hardware timer
- Software-programmable wait-state generators
- Parallel I/O ports
- Host port interface (HPI)
- Serial port
- Buffered serial port (BSP)

- Time-division multiplexed (TDM) serial port
- User-maskable interrupts

Clock Generator

The clock generator consists of an internal oscillator and a phase-locked loop (PLL) circuit. The clock generator can be driven internally by a crystal resonator circuit or driven externally by a clock source. The PLL circuit can generate an internal CPU clock by multiplying the clock source by a specific factor.

Hardware Timer

A 16-bit hardware timer with a 4-bit prescaler is available. This programmable timer clocks at a rate that is between 1/2 and 1/32 of the machine cycle rate (CLKOUT1), depending upon the timer's divide-down ratio. The timer can be stopped, restarted, reset, or disabled by specific status bits.

Software-Programmable Wait-State Generators

Software-programmable wait-state logic is incorporated in 'C5x DSPs allowing wait-state generation without any external hardware for interfacing with slower off-chip memory and I/O devices. This feature consists of multiple wait state generating circuits. Each circuit is user-programmable to operate in different wait states for off-chip memory accesses.

Parallel I/O Ports

A total of 64K I/O ports are available, sixteen of these ports are memory-mapped in data memory space. Each of the I/O ports can be addressed by the IN or the OUT instruction. The memory-mapped I/O ports can be accessed with any instruction that reads from or writes to data memory. The IS signal indicates a read or write operation through an I/O port.

Host Port Interface (HPI)

The HPI available on the 'C57S and 'LC57 is an 8-bit parallel I/O port that provides an interface to a host processor. Information is exchanged between the DSP and the host processor through on-chip memory that is accessible to both the host processor and the 'C57.

Serial Port

Three different kinds of serial ports are available:

- General-purpose serial port
- Time-division multiplexed (TDM) serial port
- Buffered serial port (BSP).

Each 'C5x contains at least one general-purpose, high-speed synchronous, full-duplexed serial port interface that provides direct communication with serial devices such as CODECs, serial analog-to-digital (A/D) converters and other serial systems. The serial port is capable of operating at up to one fourth the machine cycle rate (CLKOUT1). The serial port

transmitter and receiver are double-buffered and individually controlled by maskable external interrupt signals. Data is framed either as bytes or as words.

Buffered Serial Port (BSP)

The BSP available on the ’C56 and ’C57 devices is a full-duplexed, double buffered serial port and an auto buffering unit (ABU). The BSP provides flexibility on the data stream length. The ABU supports high-speed data transfer and reduces interrupt latencies.

TDM Serial Port

The TDM serial port available on the ’C50, ’C51, and ’C53 devices is a full duplexed serial port that can be configured by software either for synchronous operations or for time-division multiplexed operations. The TDM serial port is commonly used in multiprocessor applications.

User-Maskable Interrupts

Four external interrupt lines (INT1–INT4) and five internal interrupts, a timer interrupt and four serial port interrupts, are user maskable. When an interrupt service routine (ISR) is executed, the contents of the program counter are saved on an 8-level hardware stack, and the contents of eleven specific CPU registers are automatically saved on a 1-level-deep stack. When a return from interrupt instruction is executed, the CPU registers’ contents are restored.

Test/Emulation

On the ’C50, ’LC50, ’C51, ’LC51, ’C53, ’LC53, ’C57S and ’LC57S, an IEEE standard 1149.1 (JTAG) interface with boundary scan capability is used for emulation and test. This logic provides the boundary scan to and from the interfacing devices. It can be used to test pin-to-pin continuity and to perform operational tests on devices that are peripheral to the ’C5x.