DSP architecture

Introduction

In general, DSP processors can be classified into two broad categories as general purpose and special purpose. Fixed-point devices such as Texas Instruments TMS320C54x, and Motorola DSP563x processors, and floating- point processors such as Texas Instruments TMS320C4x and Analog Devices ADSP21xxx SHARC processors are included in DSP Processors.

Special purpose hardware are divided into two categories,

1. One type of special- purpose hardware is sometimes called an algorithm-specific digital signal processor. Hardware designed for efficient execution of specific DSP algorithms such as digital filters, Fast Fourier Transform comes under this category.

2. Another type of hardware is sometimes called an application-specific digital signal processor. Hardware designed for specific applications: for example telecommunications, digital audio, or control applications comes under this category.

In most cases application-specific digital signal processors execute specific algorithms, such as PCM encoding/decoding, but they are also required to perform other application-specific operations. Examples of special-purpose DSP processors are Cirrus's processor for digital audio sampling rate converters (CS8420), Intel's multi- channel telephony voice echo canceller (MT9300), FFT processor (PDSPI65I5A) and programmable FIR filter (VPDSP 16256).

Both general-purpose and special-purpose processors can be designed with single chips or with individual blocks of multipliers, ALUs, memories, and so on. First, let us discuss the architectural features of digital signal processors that have made real-time DSP in many possible areas.

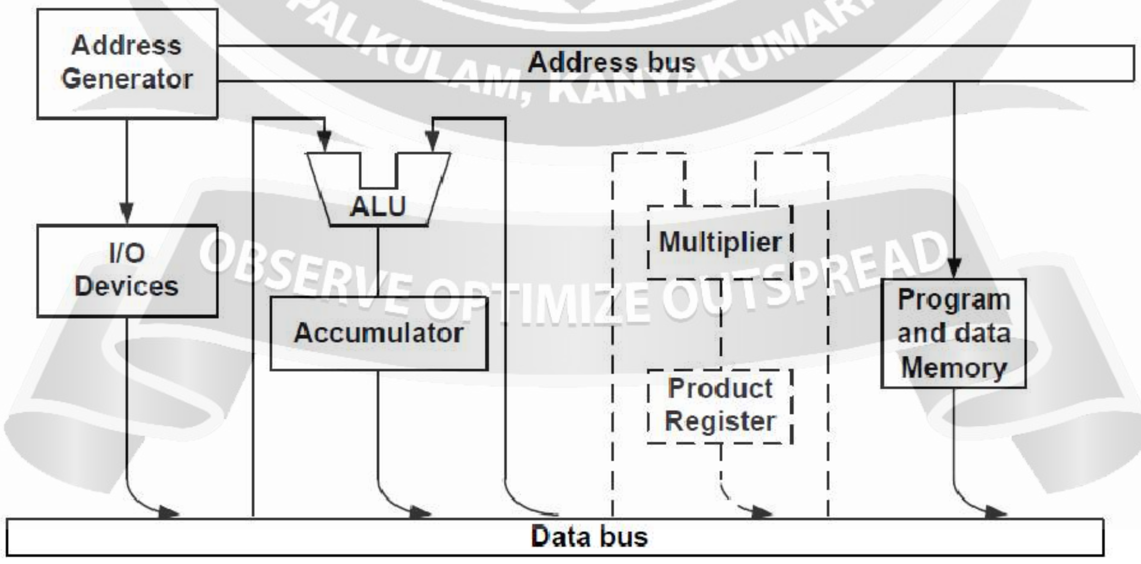

Figure 1. A simplified architecture for standard microprocessor

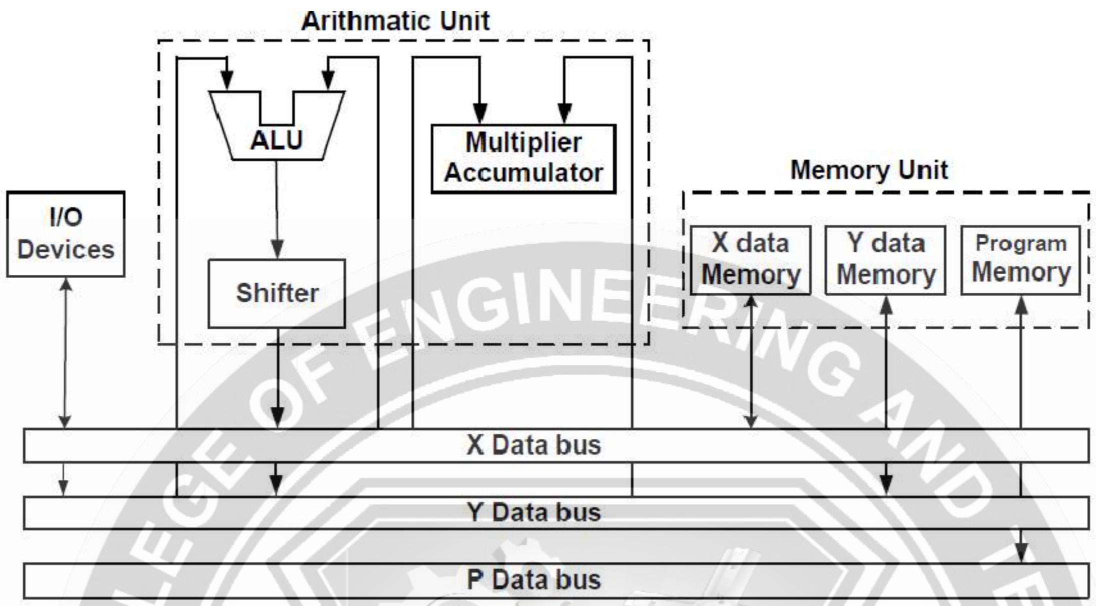

Figure 2. Basic generic hardware architecture for signal processing

Figure 2 shows generic hardware architecture suitable for real time DSP It is characterized by the multiple bus structure with separate memory space for data and program instructions. The data memories hold input data, intermediate data values and output samples, as well as fixed coefficients for example, digital filters or FFTs. The program instructions are stored in the program memory.

The I/O port provides a means of passing data to and from external devices such as the ADC and DAC or for passing digital data to other processors. Direct memory access (DMA), if available, allows for rapid transfer of blocks of data directly to or from data RAM, typically under external control.

Arithmetic units for logical and arithmetic operations include an ALU, a hardware multiplier and shifters (or multiplier--accumulator)

The main necessary of this architecture is that most DSP algorithms (such as filtering correlation and fast Fourier transform) involve repetitive arithmetic operations such as multiply, add, memory accesses, and heavy data flow through the CPU. The architecture of standard microprocessors is not suited for this type of activities. So an important goal in DSP hardware design is to optimize both the hardware architecture and the instruction set for DSP operations. In digital signal processors, this is achieved by making use of the concepts of parallelism. In particular, the following techniques are used:

1. Harvard architecture;
2. pipe-lining;
3. fast, dedicated hardware multiplier/accumulator;
4. special instructions dedicated to DSP;
5. replication;
6. on-chip memory/cache;
7. Extended parallelism — SIMD, VLIW and static superscalar processing.

For successful DSP design, it is important to understand these key architectural features.

Harvard architecture:

The principal feature of the Harvard architecture is that the program and data memories lie in two separate spaces, permitting a full overlap of instruction fetch and execution. Standard microprocessors, such as the Intel 6502, are characterized by a single bus structure for both data and instructions, as shown in Figure 1.

Suppose that in a standard microprocessor if a value op I at address ADR 1 in memory into the accumulator is to be read and then to be stored at two other addresses, ADR2 and *ADR3*. The instructions could be

LDA ADRI load the operand op1 into the accumulator from ADRI STA

ADR2 store op1 in address ADR2

STA ADR3 store op1 in address ADR3

Typically, each of these instructions would involve three distinct steps:

• instruction fetch;

• instruction decode;

• instruction execute.

In our case, the instruction fetch involves fetching the next instruction from memory, and instruction execute involves either reading or writing data into memory. In a standard processor, without Harvard architecture, the program instructions (that is, the program code) and the data (operands) are held in one memory space; see Figure 3. Thus the fetching of the next instruction while the current one is executing is not allowed, because the fetch and execution phases each require memory access.

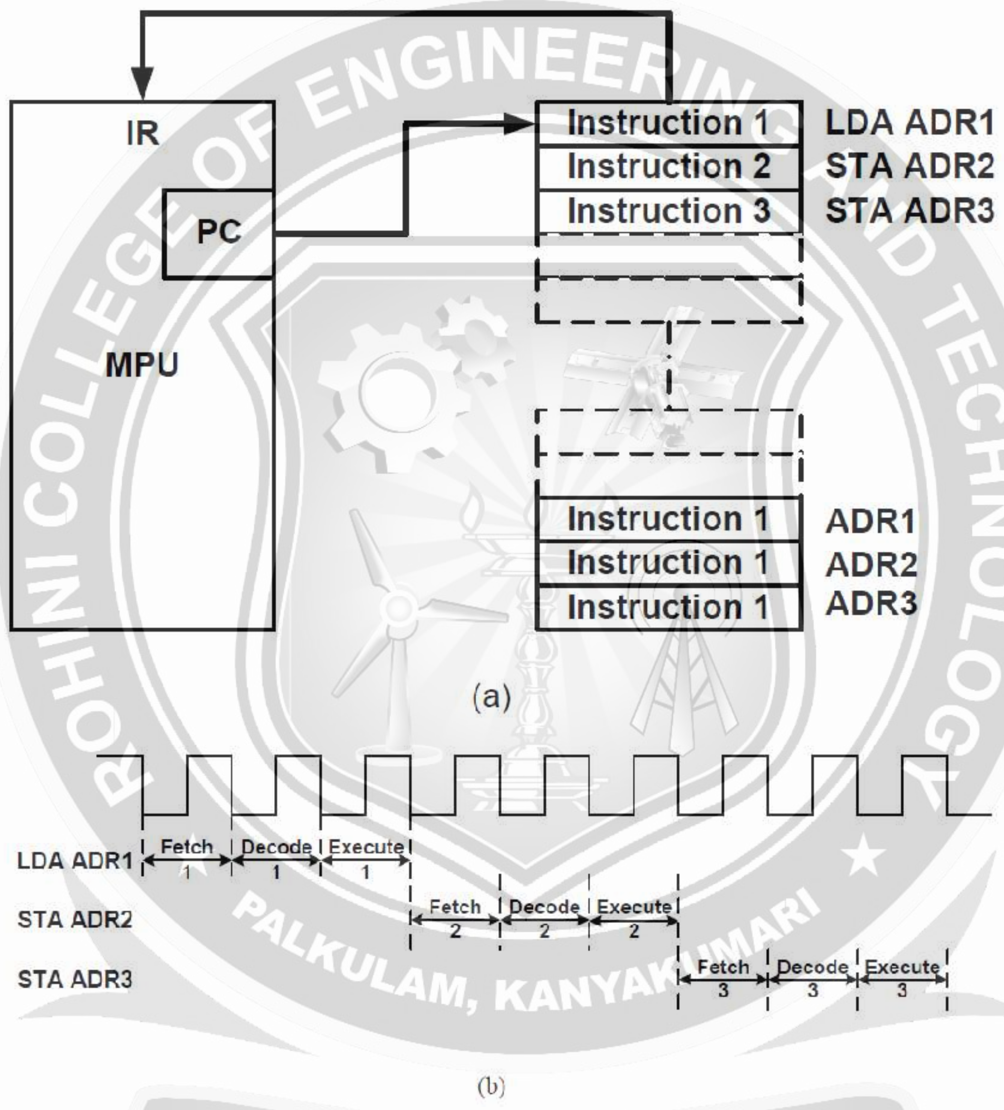

Figure 3. An illustration of instructions fetch, decode, and execute in a Non-Harward architecture with single memory space. (a) instruction fetch from memory (b) timing diagram

In a Harvard architecture (Figure 4), since the program instructions and data lie in separate memory spaces, the fetching of the next instruction can overlap the execution of the current instruction as shown in Figure 5. Normally, the program memory holds the program code, while the data memory stores variables such as the input data samples.

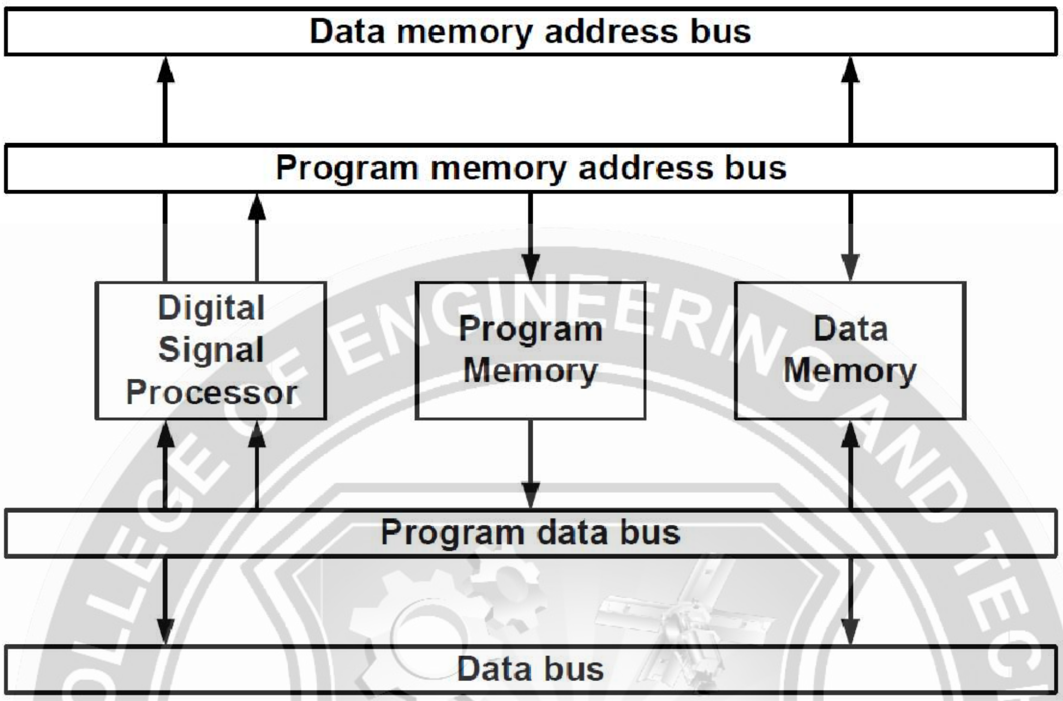

Figure 4. Basic Harvard architecture with separate data and program memory spaces

It may be seen from Figure 4 that data and program instruction fetches can be overlapped as two independent memories are used in the architecture. This is explained with the help of the timing diagram as shown in Figure 5 below.

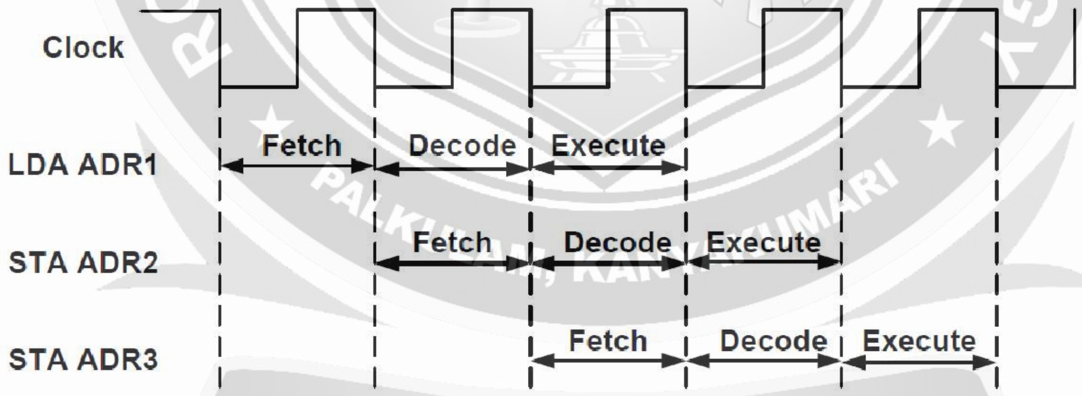

Figure 5. An illustration of instruction overlap made possible by the Harvard architecture

Strict Harvard architecture is used by some digital signal processors (for example Motorola D5P56000), but most use a modified Harvard architecture (for example, the TMS32O family of processors). For example in the modified architecture used by the TMS32O, separate program and data memory spaces are still maintained. But unlike the strict Harvard architecture, communication between the two memory spaces is permitted here.