Design Digital Data Acquisition and Processing Systems for Embedded System

Maksym Antonyuk, Mykhaylo Lobur, Volodymyr Antonyuk

Abstract: **This paper describe main way to design digital data acquisition and processing systems. Represents architecture of two systems and main advantages and problems by design such system.**

Keywords: **Digital Processing Systems (DPS), Digital signal acquisition systems (DAQ), Analog-to-Digital Converter, Digital-to-Analog Converter, Harvard Architecture, Von Neumann System Architecture**

I. INTRODUCTION

Currently, digital processing systems support almost every human activity, as extremely efficient, fast and precise command and control tools. From bank transactions and modern digital telecommunication services to air and ground traffic control, or from quick and easy office management to the advanced technology of satellites, digital control systems can be found as key components.

Given the huge number of applications involving digital processing systems, new types of problems emerged, including:

- interfacing the digital systems to the application environment
- developing efficient algorithms for data and signal processing
- embedding digital control and processing units into products
- providing real-time system operating capabilities
- designing effective data communication architectures and protocols

Thus, a key problem is how digital systems interact with the environment in an efficient manner. The general configuration of a digital signal processing system is presented in Figure 1-1. Three main components appear, in close relationship with each other [2]:

- Environment, operating with continuous signals, like sound, light, movement, pressure, fluid flow and so on;
- Digital Signal Processing System (DSP), operating with digital (discrete) signals and data, and
- Data Acquisition and Conditioning System (DAQ), with the main function of interfacing the analog (continuous) signals to the digital counterparts.

CAD/CAM Department, Lviv Polytechnic National University, S. Bandery Str., 12, Lviv, 79046, UKRAINE,
E-mail: AM2003@ukr.net

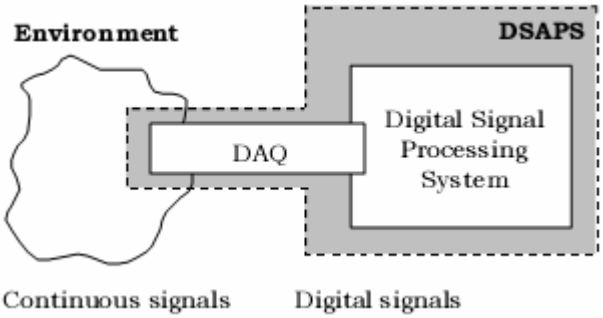

Figure 1-1. General configuration of a digital signal processing system

The structure generally referred to as Digital Signal Acquisition and Processing System (DSAPS) includes both the data acquisition block and the signal processing unit. This paper gives a detailed presentation of the internal architecture and operating principles of DSAPS.

GENERAL DESCRIPTION OF DSAPS

Digital signal acquisition and processing systems are found in a large number of real-life applications, including:

- Computer aided engineering
- Technologic engineering
- Computer aided testing of equipment
- Industrial automation
- Robotics
- Digital telecommunications
- Computer aided laboratory instrumentation
- Multimedia systems
- Digital audio and video processing
- Artificial intelligence systems
- Movie production
- Show-business

As depicted in Figure 1-1, DSAPS provide two main components to solve (a) the problem of interfacing continuous (analog) signals to digital data – the Data Acquisition System (DAQ), and (b) the problem of processing the acquired data in an efficient manner – the Digital Signal Processing (DSP) system.

The problem of interfacing signals can be further detailed as in Figure 1-2. Transducers transform the analog signals (e.g. sound, light, movement, pressure, fluid flow, biological signals, radar, sonar, seismic waves, audio/video communication signals and many others) into electrical signals, which are further transformed into digital with the analog-to-digital converters (ADC).

Authorized licensed use limited to: University of Durham. Downloaded on November 11,2024 at 12:37:04 UTC from IEEE Xplore. Restrictions apply.

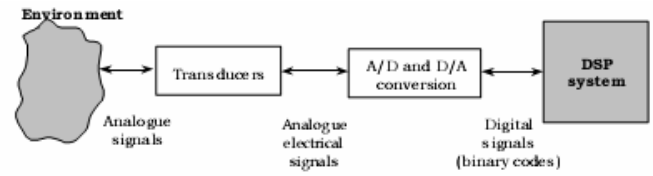

Figure 1-2. Signal interface configuration

Figure 1-2 also shows another important characteristic of DSAPS – the signal flow between the DSP system and the environment is bidirectional:

(a) Data coming from the environment is acquired by the DSP for further analysis and processing. This operation employs signal conversion from analog to digital (Analog-to-Digital Conversion, ADC or A/D).

(b) The DSP system controls the environment according to its processing results.

II. DATA ACQUISITION SYSTEMS

DAQ systems are hybrid electronic devices (analog and digital) with the main role of interfacing the digital signal processing systems to the environment [2]. The key functions of a DAQ system, briefly described below, are:

(i) Signal conditioning
(ii) Analog-to-digital conversion (ADC)
(iii) Digital-to-analog conversion (DAC)
(iv) Digital I/O
(v) Data communication with the DSP system

(i) Signal conditioning

Non-electrical signals coming from the environment are transformed in electrical signals (current and/or voltage) by transducers. Signal conditioning is further necessary to adapt the output scale range of the transducers to the input signal characteristics of the A/D converters. Programmable Gain Amplifiers (PGA) are usually used to adjust the scale range of the input electrical signal.

(ii) Analog-to-digital conversion (ADC)

Analog to digital conversion of signals is one of the main goals of a data acquisition system. A/D conversion is the set of operations that establish an exact correspondence between an analog electrical value (current, voltage) and a finite-length binary code.

(iii) Digital-to-analog conversion (DAC)

Digital-to-analog conversion is the procedure reciprocal to ADC. With digital-to-analog conversion, each binary code of b bits length at the input is related to an electrical value (current or voltage).

(iv) Digital I/O

At present, many applications use digital signals directly, without any need of A/D or D/A conversion. Some transducers transform the non-electric signals from the environment into digital pulses. Mouse-like rotation transducers for example, are based on wheels having small peripheral apertures that can block or pass the light from a LED to a photo-diode. As a result, the transducer outputs a digital impulse train to the processing system. Another example is the CCD-based optical sensors that provide the processing system with digital signals indicating the presence (logic high) or absence (logic low) of light on a particular dot of the sensor matrix.

In such cases the interface between transducers and the digital signal processing system generally consists on buffering the I/O signals.

The most common digital I/O provided by the DAQ systems consists of serial interface and data buffering.

(v) Data communication with the DSP system

An important problem related to DAQ systems is how data communication with the DSP system is implemented and managed. Several aspects of the communication problem can be stated as follows:

- Speed/data throughput. Major improvements of the communication speed were performed in recent years due to the increasing number of applications requiring high performance data manipulation and processing (e.g. multimedia systems, top digital telecommunication systems, and so on).
- Local/remote data link. Depending on the overall DSAPS architecture, DAQ can be placed locally to the DSP (on the same board) or remotely. Each type of approach requires specific communication techniques and protocols.
- Simple communication protocols. Special care must be taken regarding design and implementation of the communication interface in a simple and efficient manner [3]. The interface should also match the specifications and communication protocol of the DSP system which acts usually as master.
- Standardization. In order to ensure system portability and modularity, the communication interface must obey well defined standards, from its early specification and design stages. Examples of standard communication interfaces used in moderate-performance DSAPS are [4]: RS232 serial interface, Centronix parallel port (SPP), Enhanced Parallel Port (EPP) and Extended Capabilities Port (ECP).

GENERAL ARCHITECTURE OF A DATA ACQUISITION AND CONDITIONING SYSTEM

The general architecture of a data acquisition and conditioning system is presented in Figure 2-1 [2].

DSP-based peripheral port, to some common serial and parallel communication interface standards, respectively.

Authorized licensed use limited to: University of Durham. Downloaded on November 11,2024 at 12:37:04 UTC from IEEE Xplore. Restrictions apply.

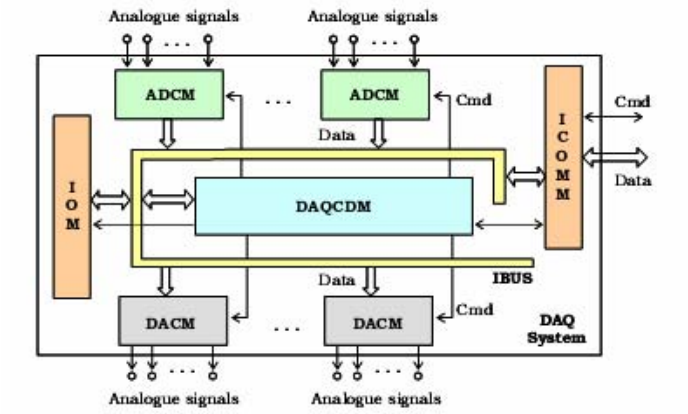

Figure 2-1. General architecture of a data acquisition system

As presented in the figure above, common data acquisition architectures feature the following components:

- one or more analog-to-digital signal conversion modules (ADCM)
- one or more digital-to-analog signal conversion modules (DACM)
- digital I/O module (IOM)
- data acquisition command module (DAQCDM)
- internal bus for data, address and command lines (IBUS)
- communication interface with the controlling system (ICOMM).

The A/D conversion module (ADCM) performs analog-to-digital conversion and optionally signals conditioning functions. Its main device is the A/D converter (ADC), as shown in Figure 2-2 below. For cost/performance reasons, a single ADCM block can process multiple analog input signal lines ($x^0(t)$ .. $x^{N-1}(t)$, thus, a total of N input lines). The input signals are properly conditioned either inside the ADCM or with a distinct signal conditioning module directly connected to the transducers and to the ADCM as well.

In the case of multiple input signals the conditioning block (highlighted with gray color in Figure 2-2) must include an analog multiplexing device (MUX) to select a single input channel to the output at any time. Channel selection is controlled by the DAQCDM block through the multiplexer selection lines and must be synchronized with the signal sampling and A/D conversion operations. Therefore, selection of a new input channel occurs at fixed time intervals, TS (sampling period).

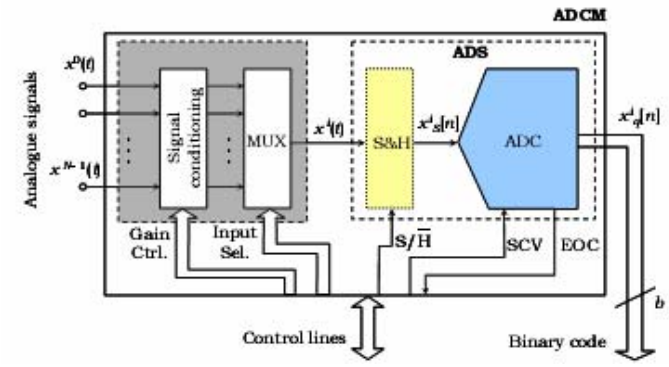

Figure 2-2. Analog-to-digital conversion module (ADCM)

Further on, the currently selected input signal, $x^i(t)$, is sampled with a "Sample-and-Hold" device (S&H), resulting the $x^i_s[n]$ signal. The input sample is then quantized and binary coded into xiq[n] by the ADC. The resulting code is of b-bit length, where b is the conversion resolution, as mentioned previously in this subsection. Modern A/D converters include both the ADC logic and the S&H device, thus being called "sampling A/D converters" or ADS.

ADCM operation is controlled through the following type of signals:

- Signal conditioning control lines, including Gain Control and multiplexer Channel Selection
- $S/\overline{H}$, to sample the input signal or hold the current sample value during the conversion
- SCV, to trigger a new signal conversion cycle on the ADC (ADS)
- EOC, conversion status line; high when the current conversion cycle finishes.

The D/A conversion module (DACM) has a symmetrical architecture to the ADCM (Figure 2-3) and performs function (iii) described above.

Modern D/A converter devices incorporate all the necessary logic for proper output signal generation (interpolation – zero-order hold or first-order linear) and for easy connection to microprocessor-type of data, address and control buses (data buffers/latches for input code, selection and enable input lines). The input data buffer plays two main roles:

(a) to store the input binary code $code_{IN}$, for the duration of the D/A conversion till the next code to be converted is supplied, and

(b) to implement the zero-order hold interpolation technique, common to most of the D/A signal conversion devices.

With the proper signal conditioning block, implemented as a component of the DACM or as an external device, multiple output analog signals can be obtained, each one of them correspondingly conditioned to the application needs (amplification/attenuation, post-filtering, supplementary interpolation and so on).

Figure 2-3 also presents the minimum command signals necessary to operate the DACM (EOC, SCV, Output channel selection, Gain controls – similar to the ADCM).

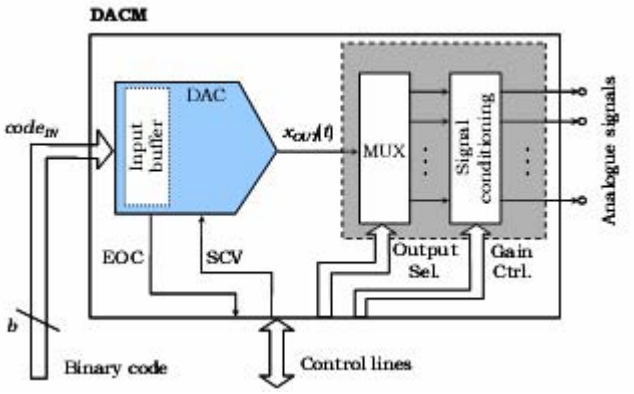

Figure 2-3. Digital-to-analog conversion module (DACM)

Authorized licensed use limited to: University of Durham. Downloaded on November 11,2024 at 12:37:04 UTC from IEEE Xplore. Restrictions apply.

The operation control of all functional blocks composing the DAQ system is performed by the DAQ command module (DAQCM). A typical DAQCM should feature the following functions:

- address decoding for selection of each device of the system;
- provide the application programs on the host/controlling system with special purpose data, command and status registers for flexible control of the acquisition process;
- provide synchronization of system components with programmable timer/counter logic;
- control the signal conditioning operations with appropriate circuits: input/output channel selection to the multiplexers, signal range adjustments (gain/attenuation) at the programmable gain amplifiers (PGA), etc.;
- interface the internal bus of the system (IBUS) to the corresponding devices connected to it (e.g. buffer/latch registers, bus driver circuits, Three-State logic);
- calibrate the analog devices involved in signal acquisition, in the case of high performance DAQ systems.

Figure 2-4 exemplifies the time diagram of the command flow employed by the DAQCM for signal acquisition with an ADCM block as described above. The diagram considers a total of N analog input channels into the ADCM, out of which only one is selected in a sequential manner by a multiplexer. Therefore, there are M channel selection lines as inputs for the multiplexer ($SEL_{0..M-1}$), where:

$$M = \log_2 N$$

The command flow presented in Figure 2-4 describes the auto-triggered type of signal acquisition, enabling the maximum acquisition rate the ADC device is capable of. First, the ACQCM selects the input channel (channel i) for the next acquisition process by properly asserting the $SEL_{0..M-1}$ lines ($SEL_{0..M-1}$ = i). Consecutively, the "Sample" signal is asserted ($S/\overline{H}$ = "1") to acquire a sample (x^i_s [n]) from the input analog signal $x^i(t)$.

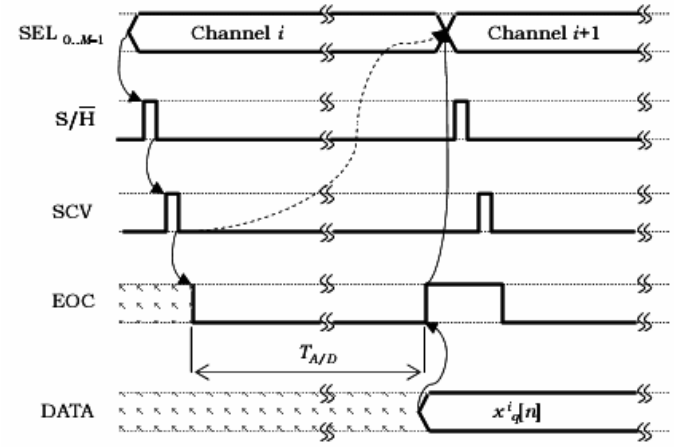

Figure 2-4. Example of a signal acquisition command flow

After sampling, the $S/\overline{H}$ line is set low to hold the current sample value for the A/D conversion cycle. Further on, the A/D conversion is started (SCV = "1"), resulting in the EOC line pulled low by the ADC device to indicate the start of a new conversion cycle. When the current A/D conversion completes (after the $T_{A/D}$ period), the ADC asserts the EOC line to indicate the end of conversion.

The rising edge of the EOC signal can be used to trigger both the selection of the next input channel (channel i+1) and to fetch the result of the current conversion, $x^i_q[n]$, through the DATA bus.

To increase efficiency and to eliminate the specific delays of multiplexing, the next channel can be selected consecutively to the "Hold" command, i.e. right after the $S/\overline{H}$ line is set low (see the dotted arrow in Figure 2-4). The multiplexing delays that can be eliminated with this technique are: t_A (access time), t_{SET} (settling time) and t_{PD} (propagation delay) [2].

This type of data acquisition – the auto-triggered type – has the advantage of speed, employing the maximum conversion rate the ADC is capable of. Still, there is an important drawback, to this approach: the acquisition period cannot be guaranteed as constant and cannot be known exactly. All the ADC devices provide approximate (typical values of) conversion timing specifications. On the other hand, many applications require medium conversion speed but constant timing rather than maximum speed. Moreover, conversion of a signal sample from the input is very often triggered by the signal processing system that acts as master within the DSAPS.

A solution to the problems stated above is to provide the DAQ system with programmable timers to control the start of conversion cycles (i.e. the SCV signal is issued from the timer output line). This type of acquisition can be called programmed acquisition.

Another option is the DSP system to trigger and control directly the acquisition process as needed – the on-demand acquisition.

An important issue regarding data acquisition is the communication of the DAQ system with the DSP system (function (v) described above). The functional block of the DAQ that implements the communication interface is the ICOMM module (see Figure 2-1).

Data communication between the DAQ and the controlling system (DSP) solves a two-folded issue: (a) to provide a proper hardware/software infrastructure for data and command exchange and (b) to provide appropriate mechanisms for fetching the acquisition results when ready into the DSP.

There is a wide range of standard communication interfaces currently available for the ICOMM implementation, depending on the DSAPS system architecture and on the application requirements.

As standard communication interface for DSAPS we will be use a PCI (Peripheral Component Interconnect) those provide up to 132 MBps (with a 32-bit data bus) and 264 MBps (with a 64- bit data bus), at 33 MHz, multiplexed Data and Address bus (32-bit – AD31..0, or 64-bit – AD63..0). It is a Mezzanine (or intermediate) bus since it is not the processor's bus. Can drive other standard buses through a bridge cipset. Supports dual voltage cards (3.3 and 5 V), burst read/write cycles, bus mastering DMA, automatic

Authorized licensed use limited to: University of Durham. Downloaded on November 11,2024 at 12:37:04 UTC from IEEE Xplore. Restrictions apply.

configuration, and Plug-and-Play. Parity checking is performed for data and address lines.

III. Digital Signal Processing Systems

Digital signal processing (DSP) systems are the key component of any DSAPS. A DSP system is an autonomous digital electronic device with two main functions [1]:

- to process discrete-time signals according to well-defined algorithms (programs);
- to control and command the signal exchange with the environment through DAQ systems.

From a larger perspective, a DSP system is a set of particular, highly-specialized signal processing algorithms along with the digital automaton which implements these algorithms.

There are mainly four different approaches for implementing DSP systems:

(1) Finite-state automata
(2) Microcontroller-based systems
(3) Specialized processors (Digital Signal Processors, DSP)
(4) Software on personal computers (PC)

(1) Finite-state automata

The signal processing algorithms necessary to a particular application are implemented as a monolithic finite-state automaton, using digital devices at various integration scales. For example, FPGAs (Field Programmable Gate Arrays) can be programmed to implement the required finite-state machine.

Advantages of this approach include high processing speed and high reliability as well. Therefore, these devices are well suited for hard real-time applications.

There are major draw-backs though: severe lack of flexibility and scalability, restrictions on the number and complexity of signal processing algorithms that can be implemented, and the engineering of the system (design, implementation and testing) is rather difficult.

This approach is currently considered obsolete and is rarely used in today's practice.

(2) Microcontroller-based systems

Microcontrollers are still the most frequent solution for control units in embedded and real-time data processing systems. Their relatively high processing speed and reliability, combined with the large flexibility of implementing data processing algorithms as software units at low cost provide the system designers and programmers with an efficient platform for digital control.

The microcontroller provides all the resources needed by an autonomous data processing and control system: CPU, on-chip RAM and ROM, internal I/O ports, timers/counters, interrupts and DMA facilities. For all these reasons, microcontrollers are well suitable for DSP system implementation and provide in most cases the throughput required in real-time applications.

Internal architecture of microcontrollers is designed for general purpose data processing. Therefore, in digital signal processing applications, which employ complex data manipulation and arithmetic at high rates, the architecture of a microcontroller offers limited solutions. Another disadvantage is the lack of portability of programs designed on a particular microcontroller. High-level language compilers for microcontrollers are rather expensive and finding the right compiler for a specific microcontroller is sometimes a difficult problem.

(3) Specialized processors (Digital Signal Processors)

Lately, a new class of processors has been developed to provide efficient solutions to digital signal processing applications. Digital signal processors (DSP) derive from microcontroller architecture with further improvements to efficiently accommodate specific operations statistically employed by signal processing algorithms.

A typical signal processor can be viewed as a filter operating on a set of discrete-time input signals to obtain a desired output signal, according to a set of well-defined processing algorithms. The overall set of particular operations performed by the filter is called transformation. Therefore, considering x[n] the input signal and z[n] the resulting signal (i.e. the output), the filtering operation can be expressed as:

$$z[n] \equiv \Im\{x[n]\}, n \in Z \qquad (3.1)$$

, where $\Im$ denotes the transformation performed by the filter.

The transformation particular to a certain digital system (digital processor) depends on the characteristics of that system and represents in a unique manner the system. The characteristics of the filter can be modeled using its impulse response signal, y[n] [1] As a result the transformation performed by the system on the input signal to obtain the output signal (equation (3.1) above) can be rewritten using the impulse response and the convolution operator:

$$z[n] = x[n] * y[n] = \sum_k x[k] * y[n-k] \qquad (3.2)$$

, where $k \in Z$ scans the discrete time interval of the longest signal – x[n] or y[n].

Thus, convolution is the most frequent operation used in signal processing. As seen in (3.2), convolution involves a repetitive set of accumulated multiplications between two samples consecutively taken from each operated signal (i.e. x[k] and y[n – k]).

As a result, the architecture of any DSP is designed in such a manner to support as efficiently as possible the operations employed by the convolution.

All DSP processor architectures provide special-designed arithmetic and logic unit(s), supporting single-cycle complex operations, such as MAC – multiply and accumulate instructions. Further on, a high degree of instruction parallelism is implemented due to the Harvard architecture and instructions pipelines and caches.

Figure 3-1, Figure 3-2 and Figure 3-3 present the basic differences of the classic (Von Neumann), Harvard and the modified Harvard architectures, respectively.

Authorized licensed use limited to: University of Durham. Downloaded on November 11,2024 at 12:37:04 UTC from IEEE Xplore. Restrictions apply.

VON NEUMANN ARCHITECTURE

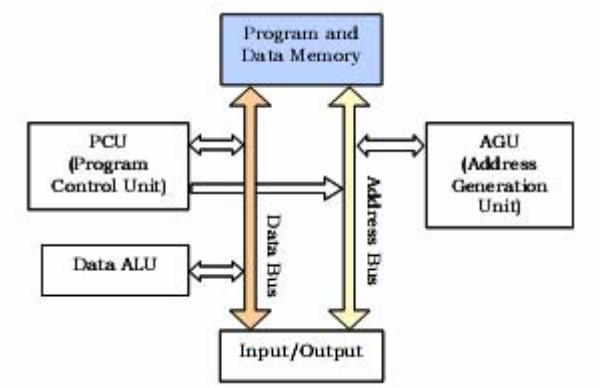

Figure 3-1. Von Neumann architecture

Von Neumann architecture specifies a common memory for instruction code (program) and for operands (data). Internal buses of a Von Neumann digital system consist of a Data Bus and an Address Bus (see Figure 3-1). Therefore, convolutions loop conforming to (3.2) can be programmed as in the Code Listing below:

Code Listing 3-1. Convolution loop on a Von Neumann architecture

```
mul     X,Y,B       ; multiply current values (initially 0)
                    ;; of ALU input data buffers into
                    ;; accumulator B
add     A,B         ; add product to current value in
                    ;; accumulator A
mov     (R0),X      ;load x[k] into X ALU input data buffer
mov     (R4),Y      ; load y[n-k] into Y ALU input data
                    ;; buffer
inc     R0          ; increment memory reference register
                    ;; to point to x[k+1] sample
dec     R4          ; decrement memory reference register
                    ;; to point to y[n-k-1] sample
                    ;; for the next loop
```

As seen in the example above, the convolution loop requires six instruction cycles: a multiplication and a summation performed by the Data ALU, two data moves from the data memory locations pointed by R0 and R4 registers into the ALU data input buffers performed by the PCU, and one register incrementation and one decrementation performed by the AGU.

HARVARD ARCHITECTURE

With the Harvard architecture, execution of the convolution loop can be improved by parallelizing arithmetic and logic instructions along with data/address moves. Harvard architectures (Figure 3-2) provide distinct program busses (PDB – Program Data-Bus, PAB – Program Address-Bus) and data busses (DDB – Data Data-Bus, DAB – Data Address-Bus), which can operate in parallel.

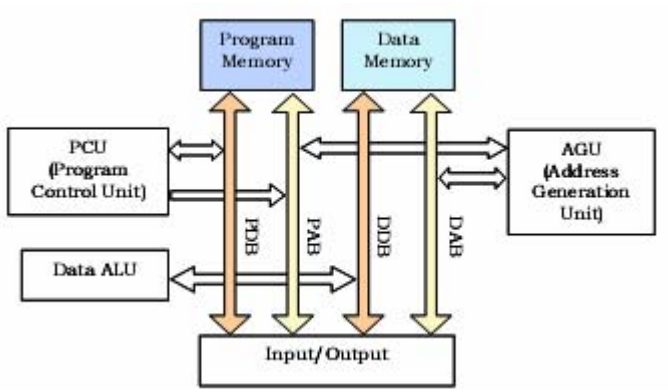

Figure 3-2. Harvard architecture

Code Listing 3-2 exemplifies the same convolution cycle on a Harvard architecture featuring a MAC (multiply and accumulate) unit into the Data ALU.

Code Listing 3-2. Convolution loop on Harvard architecture

```
mac     X,Y,A (R0),X    ; multiply operands and
                        ;; accumulate product into A,
                        ;; and load into X ALU input
                        ;; data buffer the x[k] sample
mov     (R4),Y R0+      ; load into Y ALU input data
                        ;; buffer the y[n-k] sample
                        ;; and increment memory
                        ;; reference register R0 to
                        ;; point to x[k+1]
                        ;; sample
dec     R4              ; decrement memory reference
                        ;; register R4 to point to the
                        ;; y[n-k-1] sample for the
                        ;; next loop
```

The example above demonstrates the improvement of the code execution on a Harvard architecture compared to the Von Neumann architecture: only three instruction cycles are needed this time.

Knowing that, generally most operations use two data operands, as does the convolution in particular – further improvements can be made on parallelizing code execution. The modified Harvard architecture features distinct program busses as well as two sets of data busses, thus dividing the data memory into two separate blocks: X: data memory block and Y: data memory block (Figure 3-3).

With a MAC unit that operates in a single instruction cycle and the three main units (PCU, ALU and AGU) operating autonomously, the convolution loop can be executed within a single instruction cycle, as shown in Code Listing 2-3.

Code Listing 3-3. Convolution loop on a modified Harvard architecture

```
mac     X,Y,A X:(R0)+,X Y:(R4)-,Y
```


Authorized licensed use limited to: University of Durham. Downloaded on November 11,2024 at 12:37:04 UTC from IEEE Xplore. Restrictions apply.

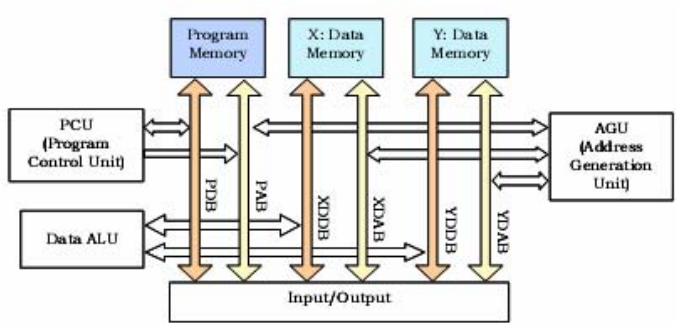

Figure 3-3. Modified Harvard architecture

As a result, modified Harvard architectures can significantly increase the instruction parallelism, particularly to code sequences that appear more frequently in digital signal processing algorithm, from the statistical point of view.

Examples of Harvard-based DSP architectures include the Texas Instruments TMS320Cxx family of processors: TMS320C1x, TMS320C2x and TMS320C2xx – fixed-point 16-bit DSPs (see Figure 2-16), and TMS320C4x – floating-point 32-bit DSP.

Examples of modified Harvard-based DSP architectures include the Motorola DSP56k family of processors: DSP568xx – fixed-point 16-bit DSPs and DSP563xx – fixed-point 24-bit DSPs (see Figure 2-17).

Digital signal processing architectures feature additional logic to improve DSP- specific algorithm programming and execution. For example, the Motorola DSP56k family of processors implements the following enhancements:

- register and addressing support for various types of buffering (linear and circular buffers) and table manipulation (e.g. step into tables at specified rates for waveform generation);
- bit-reversed addressing scheme, useful for addressing the twiddle factors in 2^k-point FFT addressing and to unscramble 2^k-point FFT data;
- hardware support (registers and additional logic) for loop programming (i.e. hardware DO loop instruction);
- most core registers can be programmed according to their specific functions or can be used as general purpose registers;
- due to the highly parallel internal bus architecture, all the main units of the DSP56k core can operate simultaneously and autonomously: PCU, Data ALU, AGU, DMA Controller, Expansion Port, Clock Logic. The peripheral interfaces of a particular DSP based on the '56k core can operate autonomously as well.

The high efficiency of digital signal processors (high performance, programming flexibility, reduced power consumption and low cost), along with the corresponding software tools (assemblers, linkers, debuggers and compilers), available to system and application programmers at moderate cost, recommend the DSPs as the most elegant solution for current signal processing systems.

(4) Software on personal computers (PC)

DSP algorithms can be implemented as programs running on personal computers. The solution has the advantage of maximum flexibility and portability as the programming language used can be one of the common high-level languages such as C or Java. Currently, the application programmer benefits from a very large variety of software tools for DSP algorithm design and implementation on the PC: assemblers for a wide range of target processors, linkers, debuggers, high-level language compilers, emulators and simulators, as well as fully integrated software development packages.

On the other hand, the relatively moderate-to-low execution speed that PCs running under common operating platforms (e.g. MS Windows, Linux, etc.) are capable of when executing specialized DSP applications does not recommend them for use on autonomous, industrial systems. Also, common operating systems for PC do not provide real-time operating capabilities, necessary for many signal processing applications.

IV. References

[1] E. C. Ifeachor, B. W. Jervis, "Digital Signal Processing: A Practical Approach", Addison-Wesley, 1993.

[2] M. V. Micea, "Signal Acquisition and Conditioning Systems: Laboratory Workshops", "Sisteme de achizitie numerica a datelor: Indrumator de laborator"), Comanda 270/2000, Centrul de multiplicare al Universitatii POLITEHNICA Timisoara, 2000.

[3] M. V. Micea, "Interfacing DAQ Systems to Digital Signal Processors", Transactions on Automatic Control and Computer Science, Special Issue Dedicated to the Fourth International Conference on Technical Informatics, CONTI'2000, October 12-13, Vol. 45 (59), No. 4, "Politehnica" University of Timisoara, 2000, pp. (23-28).

[4] M. V. Micea, V. Cretu, D. Chiciudean, "Communication Protocols Implementation on DSP-based Systems", Proceedings of the International Symposium on Systems Theory, SINTES 10, 10-th Edition, Automation, Computers, Electronics, May 25-26, University of Craiova, 2000, pp. (C 205- 208).

Authorized licensed use limited to: University of Durham. Downloaded on November 11,2024 at 12:37:04 UTC from IEEE Xplore. Restrictions apply.