T-49-12-05

# SIGNAL PROCESSING TECHNOLOGIES

# DIGITAL SIGNAL PROCESSING

## DASP-HDSP66110
Digital Array Signal Processor

## PAC-HDSP66210
Programmable Array Processor

### TABLE OF CONTENTS



6

## 1.0 INTRODUCTION

Honeywell's HDSP66 device family is a new generation of digital signal processing (DSP) VLSI integrated circuits targeted for high performance military and commercial applications. Honeywell is a proven leader in very high performance integrated circuit technology which includes a variety of CMOS, GaAs, Bipolar and BEMOS (Bipolar Enhanced CMOS) semiconductor processes. Several Honeywell products, based on one micron class technologies, have been in production for several years. In addition, an extensive amount of research is being conducted in sub-micron VLSI technologies.

The HDSP66 family of DSP devices is produced by combining innovative architectures with high performance, production-proven processes. This combination yields extremely powerful devices which provide great advantages in terms of speed, flexibility, power dissipation, level of integration and cost.

### 1.1 DASP and PAC Chip Set

The Digital Array Signal Processor (DASP) (HDSP66110) and the Programmable Array Controller (PAC) (HDSP66210) are the first 1.2 micron CMOS chip set in the HDSP66 family. These devices are optimized for DSP applications based upon the Fast Fourier Transform (FFT) algorithm [References 1, 2, 3]. The DASP and the PAC implement DSP systems which process data rates up to 100 MHz in real time and perform Discrete Fourier Transforms, spectrum analysis, digital filters, correlations, convolutions and adaptive filters based upon FFT techniques. Generally, it is easier and more efficient to process signals in the frequency domain than the time domain [Reference 1 (p.110), Reference 2 (p. 198), Reference 3 (p.635 and p.689), Reference 4].

The DASP and the PAC chips use a new architectural approach that allows for a variety of FFT based DSP system configurations. The resulting systems require only 10 to 20 PAC instructions and little hardware yet perform at rates five to 50 times faster than other present solutions. Some of the applications of the DASP and the PAC are listed in Table 1. Additional applications of the DASP chip, beyond DSP, such as graphics processing, are possible due to the inclusion of general purpose functions on the DASP.

The DASP chip performs about 500 million arithmetic operations (16 bits or more) per second and operates at an I/O rate of about 5 billion bits per second. An operation is defined to be a multiply, add or equivalent operation. Unlike traditional DSP microprocessors, the DASP processes arrays of data values rather than single data values. The DASP is capable of performing functions, composed of multiple arithmetic operations, on two sets of four complex values or two sets of eight real values every machine cycle (80 nanoseconds). A variety of general purpose and FFT specific functions are supported.

**TABLE 1: APPLICATION AREAS**

T-49-12-05

- Medical Electronics
- High End Instrumentation
- Robotics
- Radar
- Graphics Systems
- Sonar
- Telemetry
- Electronic Warfare
- Electronic Counter Measures
- Navigation and Guidance
- Very-High-Speed Modems
- Satellite Telecommunications
- Image Processing
- Transmultiplexers
- Spread Spectrum Communications
- Digital Radio

The Programmable Array Controller (PAC) can be used as a companion device to the DASP in FFT based DSP systems. Stand-alone DSP systems, as shown in Figure 1, can be designed by combining the DASP and the PAC with off-the-shelf single-port memories. The PAC is responsible for managing the complete system, based upon the program downloaded to its program memory during initialization. The programmability of PAC makes it applicable in various hardware and algorithmic configurations. The program, typically 10 to 20 instructions, defines the algorithm to be executed on the DSP system (e.g., spectrum analysis, digital filtering, etc.). DSP microprocessors, on the other hand, usually require hundreds or thousands of instructions.

T-49-12-05



### 1.2 Benchmarks

The ability to define an algorithm and various hardware configurations in the PAC allows the chip set extreme versatility in a variety of applications. The system performance can be balanced against the hardware cost by choosing the number of DASP/PAC stages and the number of memory devices in a given system. An extremely high performance DSP system can be designed by cascading multiple DASP/PAC stages. A mid-performance system can be designed by using dual memory sets around one DASP/PAC stage. A lower performance system can be based upon one memory set and one DASP/PAC stage. The benchmarks, for various sized FFT's on different system configurations, are shown in Table 2. Some benchmarks for digital filters (fast convolution), implemented via the frequency domain, are given in Table 3. It is assumed that the incoming data frames overlap by 50%. The amount of data frame overlap is programmed into the PAC by the user. Table 3 illustrates the efficiency of using frequency domain techniques for digital filtering when the number of points in the frequency response of the filter is large. In the time domain, for example, over 350, 100-nS multiplier/accumulators are necessary to implement a 2048 pole digital filter. This can be achieved in the frequency domain with a single DASP/PAC recursive stage.

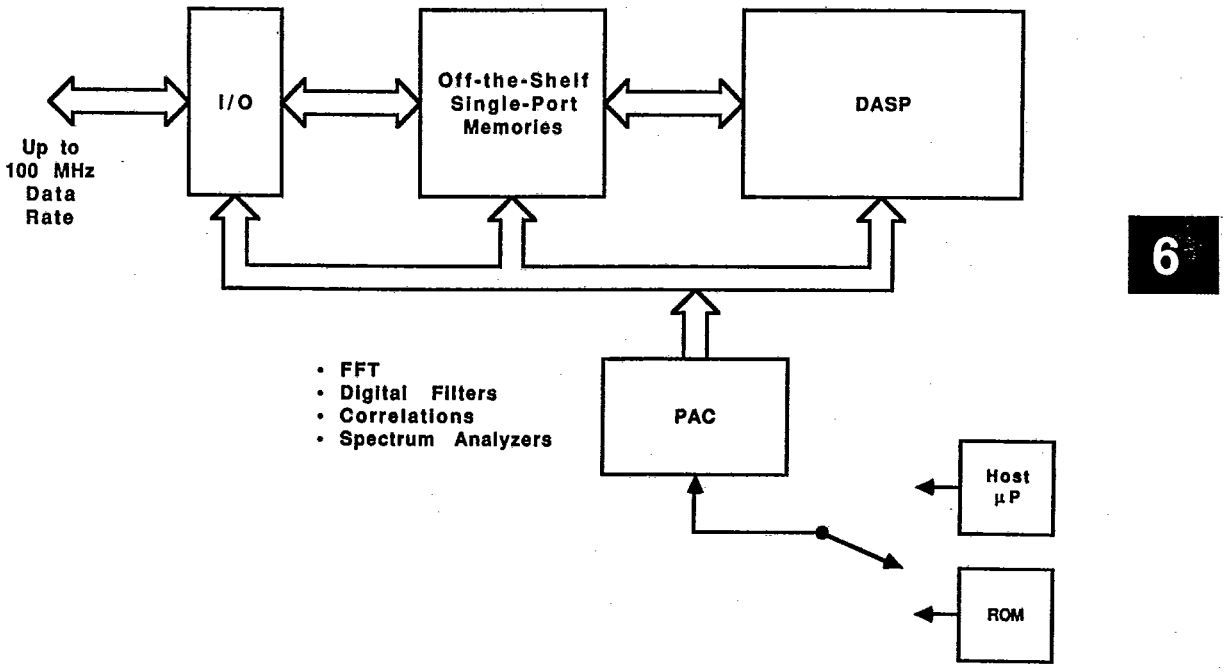


**FIGURE 1: DASP/PAC BASED DSP SYSTEM**

T-49-12-05

TABLE 2: FFT BENCHMARKS

| NUMBER OF POINTS | SYSTEM TYPE: CASCADED MULTIPLE STAGES | SYSTEM TYPE: DUAL MEMORY RECURSIVE (ONE STAGE) | SYSTEM TYPE: ONE MEMORY RECURSIVE (ONE STAGE) |
|---|---|---|---|
| 64 Real | 880 nS (4 Stages) | 3.52 µS | 10.56 µS |
| 64 Complex | 1.76 µS (3 Stages) | 5.28 µS | 10.56 µS |
| 512 Real | 5.36 µS (6 Stages) | 32.16 µS | 107.2 µS |
| 512 Complex | 10.72 µS (5 Stages) | 53.6 µS | 107.2 µS |
| 1024 Real | 10.48 µS (6 Stages) | 62.88 µS | 209.6 µS |
| 1024 Complex | 20.96 µS (5 Stages) | 104.8 µS | 209.6 µS |
| 64K Real | 0.65 mS (9 Stages) | 5.85 mS | 20.96 mS |
| 64K Complex | 1.31 mS (8 Stages) | 10.48 mS | 20.96 mS |

TABLE 3: DIGITAL FILTER BENCHMARKS (50% OVERLAP)

| NUMBER OF POINTS IN IMPULSE RESPONSE OF FILTER | INPUT COMPLEX DATA RATES: CASCADED SYSTEM (MULTIPLE STAGES) | INPUT COMPLEX DATA RATES: DUAL MEMORY RECURSIVE SYSTEM (ONE STAGE) |
|---|---|---|
| 32 | 25 MHz (7 Stages) | 3.57 MHz |
| 128 | 25 MHz (9 Stages) | 2.78 MHz |
| 512 | 25 MHz (11 Stages) | 2.27 MHz |
| 2048 | 25 MHz (13 Stages) | 1.92 MHz |

T-49-12-05

## 2.0 DASP (HDSP66110)

The DASP is a very-high-speed, block floating-point array processor that is capable of performing FFT specific and general purpose operations on arrays of data. The key features of the DASP are shown in Table 4.



TABLE 4: DASP KEY FEATURES

- Arithmetic throughput up to 500 million operations per second.
- Data I/O rate up to 5 billion bits per second.
- 16 FFT-specific and general purpose functions.
- 16/20 bit fixed point arithmetic with block floating point support for FFT functions.
- 16 bit parallel I/O buses to support static RAMs.
- Two interface options (Dual, Quad) for system trade-offs.
- 1.2 micron CMOS; typical dissipation – 2 watts.
- 269 pin ceramic PGA package.
- Performance examples:

| | |
|---|---|
| Radix-4 Butterfly | – 80 nS |
| Radix-2 Butterfly | – 40 nS |
| Complex Multiply | – 20 nS |
| Real Multiply | – 10 nS |
| ALU Operation | – 10 nS |

## 2.1 DASP: Architecture

DSP

The DASP is capable of processing sixteen,16-bit real values (or 8 complex values), and producing eight, 16-bit real values (or 4 complex values) every machine cycle (Tm). The lower limit on Tm is specified at 80 nano seconds. The block diagram of the DASP below (Figure 2) shows the internal data paths and input/output values. All input/output values are transferred every machine cycle. These values, designated as r0....r3, i0....i3, form the input data that is fed to the DASP every machine cycle. In complex arithmetic instructions (such as FFT), these values represent a set of 4 complex numbers: (r0, i0), (r1, i1), (r2, i2), (r3, i3). Another set of eight 16-bit input operands, which is called auxiliary data, is designated as c0....c3, s0....s3. In complex arithmetic instructions, these values are complex numbers [(c0, s0), (c1, s1), (c2, s2), (c3, s3)] representing auxiliary data values such as window coefficients or trigonometric coefficients. After operating upon the input operands, the device produces a set of eight 16-bit values that are termed x0....x3, y0....y3. Once again, these values represent a set of four complex numbers [(x0, y0), (x1,y1), (x2, y2), (x3, y3)] for complex arithmetic instructions. In summary, there are three I/O ports (input data, input auxiliary data and output data) on the DASP for data transfers.

First, the input data set is passed through an array of complementers to perform conjugating or complementing, if so desired. Then, the values are passed through an array of adders which can allow data to grow to 18-bits without producing an overflow. These adders are used to perform pre-multiplication additions associated with the Decimation-In-Frequency (DIF) FFT Butterfly operations. The added values are right shifted, if so desired, then rounded to 16-bits and fed to an array of multipliers. At this stage, the multiplier array also receives a scaled set of values from the input auxiliary data port. After multiplications, the 20 most significant bits of each resulting product are retained. Next, the 20-bit product values are operated upon by the ALU array. These ALU's are used to perform post-multiplication additions associated with the complex multiplications. These ALU's can also perform general purpose arithmetic and logical operations. The 20-bit output values from the ALU's are rounded to 16-bits. These values could be complemented or conjugated, if necessary. The final values, which are fed to the data output port, are also monitored by the on-chip scale-factor generator to implement the block floating point arithmetic (discussed later). The data path operation is controlled by a function code which is externally applied. After setting up a function code, the sets of I/O values can be continuously passed through the DASP (one set every machine cycle, Tm). The DASP introduces a latency of four machine cycles from the data-inputs/auxiliary-data-inputs to the outputs. The impact of latency is virtually insignificant since the DASP primarily deals with data-arrays and a given function is typically applied to the whole array.

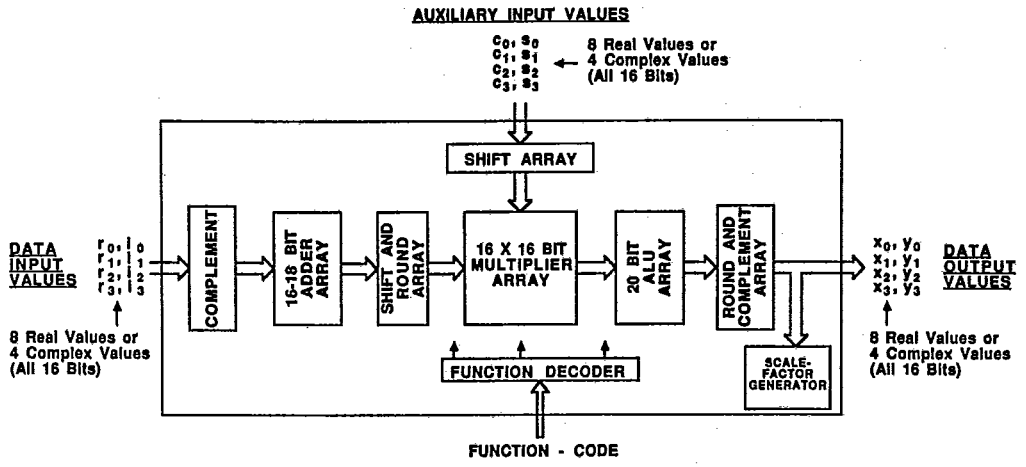


**FIGURE 2: DASP BLOCK DIAGRAM AND I/O VALUES**

## 2.2 DASP: Input/Output

Transferring the data in period Tm, as mentioned above, poses a challenge for the system architectural design. Therefore, two input/output modes have been devised for the DASP which offer trade-offs to the user. One of the modes, which is called the DUAL bus mode, is shown in Figure 3. In this scheme, each I/O port is served by two 16-bit parallel buses and transfers 8 values at intervals of Tm. Each bus, therefore, transfers a value at intervals of Tm/4. A clock signal at a frequency of 4/Tm is required to manage the DASP in the DUAL mode. The application of the DUAL bus DASP results in a simple system architecture since off-the-shelf single port memories can be used to serve each bus on the DASP. Also, the DASP input and output data ports (D1/D2, D3/D4) are bi-directional in the DUAL mode. This means that the roles of buses D1/D2 and D3/D4 can be interchanged. During a given data pass, the input data values could be fed through the D1/D2 ports and outputs produced on the D3/D4 ports (Figure 3). On the next pass, the inputs could be fed through the D3/D4 ports and outputs produced on the D1/D2 ports. As discussed later in the system applications section, the bi-directionality feature of the data ports makes the design of recursive systems very simple.

Another I/O scheme, which is shown in Figure 4, is called the QUAD bus mode and is chosen by applying a high level on the QUAD pin of the DASP. In this mode, there are four buses associated with each port of the DASP. Therefore, each bus is operated at intervals of Tm/2, requiring an input clock to the DASP at a frequency of 2/Tm. In the QUAD mode, the access time on external memories is relaxed. However, the memory architecture will generally be more complicated compared to the DUAL I/O mode. In the majority of applications, four port memories are required to feed each of the three ports. In addition, the data I/O buses (D1/D2/D5/D6, D3/D4/D7/D8) are not bi-directional in the QUAD mode causing some inconvenience in the recursive system designs.



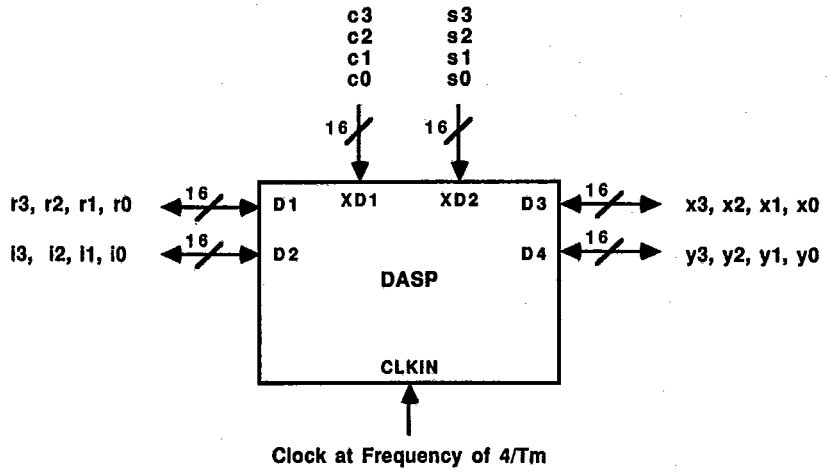


**FIGURE 3: DASP DUAL BUS MODE**

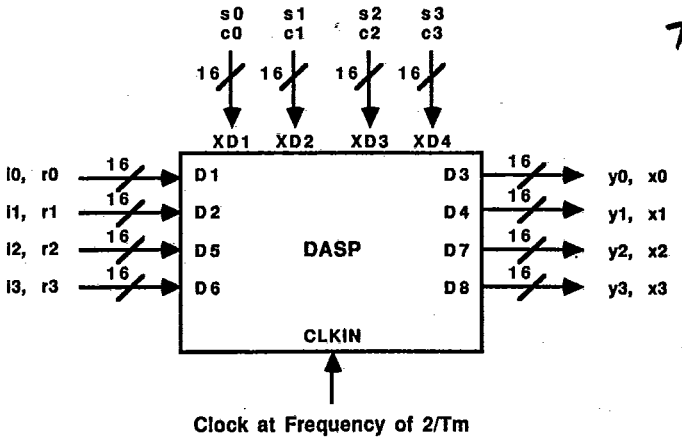


**FIGURE 4: DASP QUAD BUS OPTION**

### 2.3 DASP: Function Set

A total of 16 functions are supported on DASP (listed in Table 5). Note that the DASP is capable of implementing one function every machine cycle (period Tm). The function-code on pins FC(5:0) must be set up one machine cycle ahead of feeding the data to the processor. In typical array processing applications, such as FFT's, a function code is set up (e.g., BFLY4); then, the whole data array is clocked through the processor. The applied function will be, therefore, implemented on the whole array. As mentioned earlier, there is a latency of four machine cycles in implementing each function on the DASP.

The complex arithmetic functions support FFT processing (Table 5). The BFLY4 and BFLY2 functions can be used to implement radix-4, radix-2 or mixed-radix-4/radix-2 FFT algorithms. The functions FFTNN and FFT2N are useful for performing FFT's on real data [Reference 2 (p.166, 167)]. The FFTNN function can be used to process two frames of real-data simultaneously and obtain almost twice the performance of complex data FFT's. The BMUL function can be used to perform windowing on the input data or for general purpose complex multiplications encountered in demodulation processes. The general purpose functions, listed in Table 5, can be used to implement various arithmetic operations on real and complex data. The function BSQSM can be used to determine the magnitude-squared of the frequency spectrum for FFT applications. The general logic functions in Table 5 are useful for performing logic operations on arrays of data.

There are provisions,useful for implementing inverse-FFT's, to conjugate the inputs to, and the outputs from, the complex arithmetic functions. Similarly, the input and output data values can be complemented for the general arithmetic and logical functions. In addition, the input values can be shifted for all functions. These facilities lead to higher computational efficiency.

### 2.4 DASP: Block Floating Point Arithmetic

Primarily, the input and output values associated with the DASP are 16-bit fixed-point and are represented in 2's complement arithmetic (for arithmetic functions). However, on-chip intermediate values can grow to 20 bits but are rounded to 16 bits before they are transferred to the output buses. The growth in the number of bits helps to preserve an adequate signal-to-noise ratio. Still, there could be a severe loss in the signal-to-noise ratio on fixed-point arithmetic machines due to scaling, which is normally applied to prevent overflow. The block floating point scheme (Figure 5), which applies data dependent scaling, boosts the signal to noise ratio by extending the dynamic range of the fixed point arithmetic. When a BFLY4 or BFLY2 is implemented on the DASP, the magnitude of the values being produced at the output are monitored. After a complete pass through the data-array, the DASP produces a scale factor at the pins SFO(2:0) to scale the data on the next BFLY4/BFLY2 pass and prevent overflow. The scale factor is automatically applied to the input when the SFO output pins are tied to the

TABLE 5: DASP FUNCTIONS

T-49-12-05

| FUNCTION MNEMONIC | DESCRIPTION | NO. OF INPUT DATA VALUES | NUMBER OF INPUT AUX. DATA VALUES | NUMBER OF OUTPUT DATA VALUES |
|---|---|---|---|---|
| **COMPLEX ARITHMETIC CLASS** | | | | |
| BFLY4 | Radix-4 Decimation-in-Frequency Butterfly | 4 Complex | 4 Complex | 4 Complex |
| BFLY2 | Two Radix-2 Decimation-in-Frequency Butterflies | 4 Complex | 4 Complex | 4 Complex |
| FFTNN | Recombine N Complex-Point FFT to Two N Real Point FFTs | 4 Complex | X | 4 Complex |
| FFT2N | Recombine N Complex-Point FFT to 2N Real Point FFT | 4 Complex | 1 Complex | 1 Complex |
| BMUL | Block Multiply Two Sets of Complex Numbers | 4 Complex | 4 Complex | 4 Complex |
| **GENERAL ARITHMETIC CLASS** | | | | |
| AFLOW | Arithmetic Flow Through: Pass Data | 4 Complex or 8 Real | X | 4 Complex or 8 Real |
| BMULR | Block-Multiply Two Sets of Real Numbers | 8 Real | 8 Real | 8 Real |
| BMULRA | Block-Multiply Two Sets of Real Numbers and Partially-Add | 8 Real | 8 Real | 4 Real |
| BSQSM | Block Square and Sum a Set of Values | 4 Complex or 8 Real | X | 4 Real |
| BADD | Block - Add Two Sets of Values | 4 Complex or 8 Real | 4 Complex or 8 Real | 4 Complex or 8 Real |
| BSUB | Block - Subtract Two Sets of Values | 4 Complex or 8 Real | 4 Complex or 8 Real | 4 Complex or 8 Real |
| **GENERAL LOGIC CLASS** | | | | |
| LFLOW | Logical Flow Through: Pass Data | 8 Logical | X | 8 Logical |
| BAND | Block-AND Two Sets of Values | 8 Logical | 8 Logical | 8 Logical |
| BOR | Block-OR Two Sets of Values | 8 Logical | 8 Logical | 8 Logical |
| BXOR | Block-XOR Two Sets of Values | 8 Logical | 8 Logical | 8 Logical |
| BCONS | Generate Block of Logical Constants at the Outputs (Zeros or Ones) | X | X | 8 Logical |

T-49-12-05



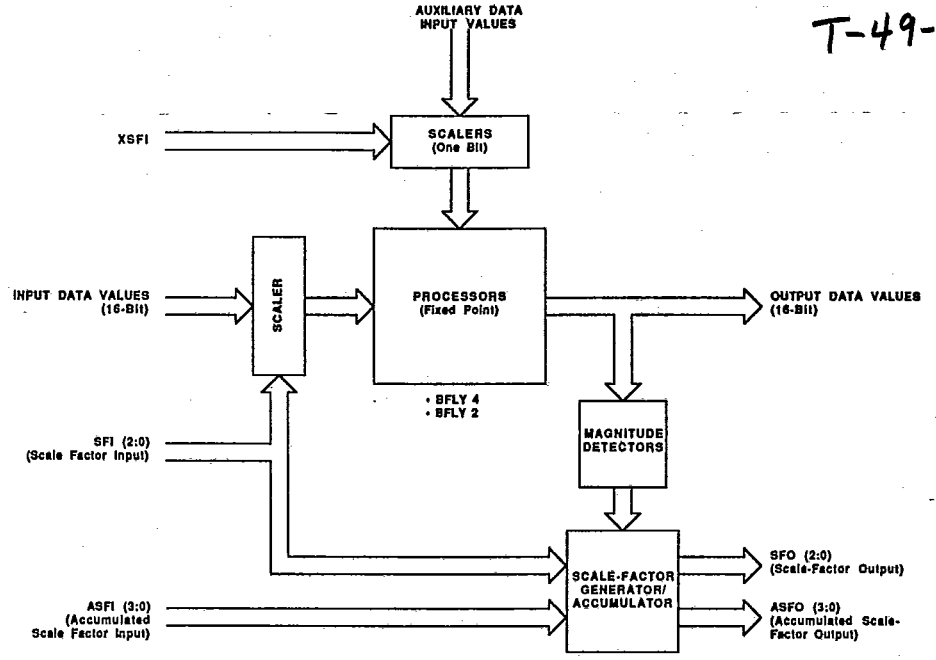


**FIGURE 5: DASP BLOCK FLOATING SCHEME**

corresponding SFI input pins of the same or of another DASP device (Figure 5). The DASP is also capable of accumulating various scale factors that may be applied by the processor at various stages of the algorithm. The accumulated scale factor is produced on the ASFO(3:0) pins. The ASFO from the final pass can be used to normalize the final processed data array.

## 3.0 PAC (HDSP66210)

The Programmable Array Controller (PAC) combined with the DASP forms a compact, complete DSP system solution. The PAC provides addressing and control for the memories and other elements in the system. The key features of the PAC are listed in Table 6 and a block diagram is shown in Figure 6.

### 3.1 PAC: Address Sequences and Memory Control

The core of the PAC is the five address generators: IAS, OAS, RAS, WAS and XAS. These address generators encompass all addressing sequences which are necessary to implement various FFT-based DSP systems. The IAS and OAS generate address sequences for the input data collection memory and the output data dump memory. The RAS and WAS generate address sequences for the DASP read memory and the DASP write memory. The XAS sequencer addresses the auxiliary (coefficient) data memory. Each address generator outputs a new address every micro cycle (Tm/4) to support the I/O rate of the DASP. Therefore, five 16-bit addresses are generated every micro cycle on the external PAC address buses designated as ADRA, ADRB, ADRC, ADRD and ADRX. These address buses can control five distinct memories called A, B, C, D and X. The PAC provides full control of these memories by producing corresponding write strobes.

Typically, the address on each bus is used to address a pair of memories which hold Real and Imaginary data, respectively. Each memory pair will be assigned to one of the three DASP I/O ports, the input data collection buffer or the output data buffer. The PAC functions extremely well with the DUAL mode DASP and requires minimal external hardware. When the PAC is used with the QUAD mode DASP, the addresses from the PAC

T-49-12-05

**TABLE 6: PAC KEY FEATURES**

- Provides total system control and addressing for FFT-based DSP systems (Controls, I/O, Memories and DASP).
- Supports up to 64K point data frames (16-bit address buses).
- 32-word internal instruction memory to define user's algorithm (one instruction needed per pass).
- Optional external instruction memory space (unlimited).
- Generates various addressing sequences for FFT based applications on 5 address buses every micro cycle (Tm/4).
- On-chip multiplexers for assigning desired address sequences to the desired output ports.
- Compensates for pipeline latencies in the external memories.
- Control registers to define various parameters and hardware features of the system.
- Initialization by a host or by an autoboot from a ROM.
- Synchronization capability for multiple DASP and PAC devices.
- 1.2 micron CMOS process, typical dissipation — 1.2 watts.
- 180 pin ceramic PGA package.



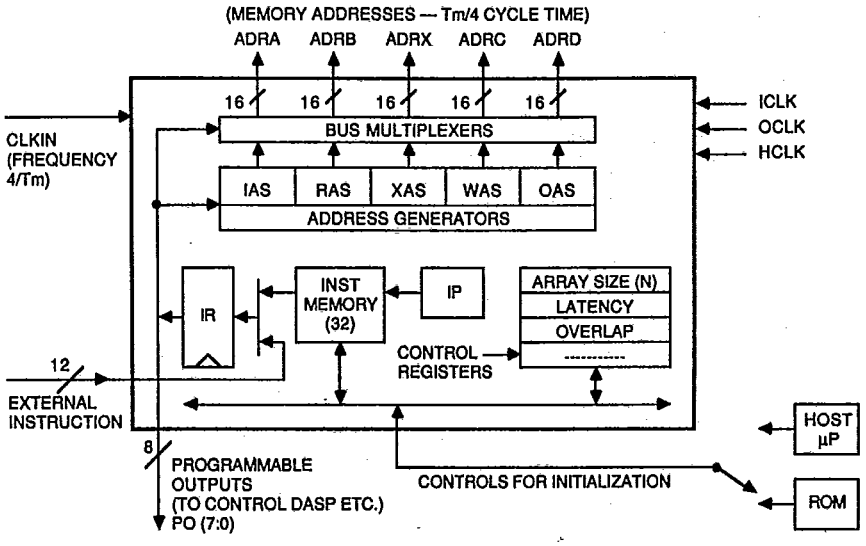


**FIGURE 6: PAC BLOCK DIAGRAM**

must be clocked into external registers (over a machine cycle) which; in turn, can be used to address four-port memories to feed data to the DASP I/O ports.

It is important to note that the bus multiplexers (Figure 6) have been included on the PAC so that various address sequences can be programmed to appear on the desired output buses. This allows the roles of memories in a system to be reallocated under program control. Various address sequences which are supported on the PAC are as follows:

| Mnemonic: | Sequence Description |
|---|---|
| **FFT0 to FFT15:** | FFT: Data/Coefficient address sequences associated with an in-place, Decimation-in-frequency (DIF) FFT algorithm. The sequences span from column 0 to column 15 of a FFT flowgraph, covering up to a 64K point data array. The radix for the algorithm is defined in the control registers. Digit reversed addresses can be optionally produced in the final FFT column. |
| **SEQ:** | Sequential: A normal binary sequence for the purpose of input, output, windowing etc. |
| **SSEQ:** | Symmetric Sequential: A normal binary sequence which is symmetrical in the middle to handle symmetric windows, etc. |
| **FFT2N:** | Supports DASP's FFT2N Function: Generates addressing sequences to handle data for the recombination associated with the implementation of 2N real point FFT on the N complex-point FFT machine. |
| **FFTNN:** | Supports DASP's FFTNN Function: Generates addressing sequences to handle data for the recombination associated with the implementation of two separate N real-point FFT's on the N complex-point FFT machine. |
| **FTRS:** | Filter Sequence: The overlap/save sequence to allow overlapping of data frames. |

### 3.2 PAC: Control Registers

The PAC also contains several user-loadable control registers which add to the versatility of the PAC (Figure 6). These registers supply various parameters to the PAC describing the configuration of the user system. Some of the registers are described below as examples.

| | |
|---|---|
| **Array Size (N):** | Specifies the size of the data array to be processed. It can range from 4 points to 64K points. |
| **Processor Latency (PLAT):** | Specifies the latency of the DASP in terms of machine cycles. |
| **Memory Latency (MLAT):** | Specifies the latencies through the external memories associated with the DASP. Note that latency in a memory path is introduced if the address and/or data of a memory is being latched for high performance applications. The PAC compensates for these latencies when generating the addresses and write strobes for corresponding memories. |
| **Overlap Size (K):** | Specifies the size of the overlap in successive data frames. The overlapping is normally used for frequency-domain filtering. It can range from 0 to 50% of the array size. |
| **Configuration (RCONFIG):** | Contains system configuration-type information such as cascaded system, single-memory system, radix type. |

### 3.3 PAC: Instruction Memory

The PAC also has a 32 word (20-bit) user loadable instruction RAM which holds the user's DSP program. The concept of a PAC instruction is somewhat different than a microprocessor instruction. In microprocessors, an instruction typically manipulates one or a couple of data values. A PAC instruction controls a full pass of the N-Point data array through the DASP, where N is defined in the appropriate PAC control register. During the pass, a programmed DASP function will be applied

to the successive sets of data values in the N-Point data array. The DASP functions and the definition of a set of data values are defined in section 2.0. One instruction controls a complete pass of the data array through the DASP. A 1024 point FFT using a radix-4 algorithm, for example, takes five instructions because the corresponding FFT flowgraph consists of five FFT columns requiring five passes through the DASP. If additional passes such as windowing or magnitude-squaring are desired, one instruction will be needed for each additional pass. Typically, algorithms such as spectrum analysis, digital filtering or convolutions (via frequency domain) can be defined on the PAC by coding 10 to 20 instructions. Therefore, the on-chip 32-word memory suffices for the majority of applications, although unlimited external program memory may be used if necessary.

The format of the 20-bit PAC instruction is shown in Figure 7. The various fields of the instruction are described below.

**Programmable Outputs (PO):** The user-defined 8-bit value in this field is directly fed to the PO pins of the PAC. This can be used to control various system elements including the DASP function code.

**Bus Switch Code (BSC):** This field controls the PAC bus multiplexers (Figure 6) to assign the appropriate address generators to the address buses ADRA, ADRB, ADRC and ADRD. The BSC field is also output directly to the BSC pins.

**Shift X-Memory Address (XSHF):** This field defines the number of bit positions to left-shift the X-Memory (coefficient) address. This feature is useful for sharing coefficient memory for different sized FFT's, etc.

**Mixed-Radix Mode (MIXMD):** This bit indicates if a mixed-radix FFT (radix-4/radix-2) is being performed. By using the mixed-radix mode, it is possible to transform a data-array in which the number of points are an odd power of 2, almost at the speed of a radix-4 system.

**Node Sequence Type (NODE):** This field defines the type of addressing sequences to be produced by the five address generators of the PAC.

### 3.4 PAC: Initialization

T-49-12-05

The PAC control registers and instruction memory are memory mapped for ease of initialization. The PAC can be initialized by a host processor or it can autoboot itself from an external ROM. The initialization option is determined by the AUTOBOOT pin. After initialization, the processing is started by activating the GO pin. The PAC then manages the complete system. It executes the DSP algorithm by continuously looping through the program. The processing can be interrupted any time by handshake signals. The PAC can then be reinitialized for another process if so desired.

### 3.5 PAC: Clock Schemes

The PAC is operated by using a system clock (CLKIN) at the frequency of 4/Tm. However, additional clock inputs ICLK, OCLK and HCLK are also provided. The ICLK and OCLK manage the input data-collection and the output data-dump, respectively, at the desired rate. The HCLK manages the communication to the host processor (for initialization) at the desired rate. The independent control over these clocks provides great flexibility in designing real-time DSP systems. For very high performance cascaded systems the frequency of ICLK and OCLK approach 4/Tm. Therefore, it is possible to process complex-data to a rate of 4/Tm and real data at a rate up to 8/Tm. Multiple DASP's and PAC's can be synchronized via the SYNC pins for multiprocessing.

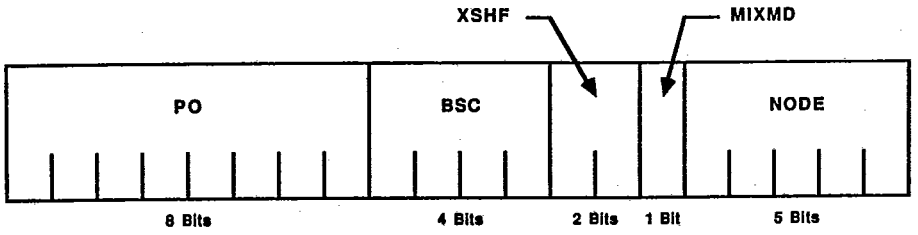


FIGURE 7: PAC INSTRUCTION FORMAT

## 4.0 SYSTEM ARCHITECTURES

The DASP and the PAC provide an effective means for the designer to solve many DSP problems. As mentioned earlier, the PAC is capable of handling a variety of DSP algorithms and system architectures. The designer may choose among several different system architectures supported by the PAC to achieve the desired performance. There are three basic architectures supported by PAC: recursive dual memory, recursive single memory and cascaded. The recursive dual memory architecture provides a cost effective solution with medium throughput and is extremely flexible in its programmability. The recursive single memory architecture provides the lowest throughput and the lowest cost solution. The cascaded architecture provides the maximum throughput by distributing the array processing task among multiple DASP/PAC stages. Variations on these basic architectures and user defined architectures can also be built. DASP/PAC based systems can be designed with very little effort and hardware since provisions in the device architectures were made to accommodate various system architectures.

### 4.1 Recursive Architecture: Dual Memory

In the recursive architecture, DSP algorithms on a single DASP/PAC stage are computed by multiple passes of the data array through a single DASP. A core stage of the recursive dual memory system is shown in Figure 8. The system employs three memory banks (B, C, X) which are all addressed and managed by the PAC. Each memory block (R and I) represents an N-word, complex data memory (Real and Imaginary Channels) where N is the size of the data array defined in the PAC. In the beginning of a process, a data frame is collected in memory B by invoking an addressing sequence on the PAC bus ADRB. The PAC begins processing by setting up a function code (e.g., BMUL, BFLY4, BFLY2) on DASP. In the first pass, the PAC transfers the data from memory B to memory C by providing the read address sequence on the bus ADRB and the write address sequence on bus ADRC. The write address sequence is appropriately staggered from the read address sequence to allow for DASP pipeline latency and any memory path latency due to the latching of address or data. During the data transfer pass, the chosen DASP function is applied to the entire N-point data array. The auxiliary data, which will be trigonometric coefficients or window coefficients in the case of FFT's, is automatically picked up from memory X using the address sequence on the bus ADRX. After the entire data array has been written into memory C, the PAC reverses the memory roles on the next pass by using its on-chip address multiplexers; memory C receives the read address sequence, the DASP receives the next function to perform, and memory B receives the staggered write address. Data flows in the opposite direction as shown in Figure 9. Note that if the DASP is operated in the dual bus mode, no tri-state buffers are required since the DASP data buses are bi-directional. The process of alternating data directions and recursively passing the data through the DASP is repeated as many times as necessary to implement the user's entire algorithm. After all computational passes are completed, the user would then perform an I/O pass. The input and output address generators would be routed to the appropriate memory banks, and data is simultaneously loaded and unloaded. One may then repeat this entire process on a new data array.

### 4.2 Recursive Architecture: Dual Memory, I/O Buffered

T-49-12-05

In the previous scheme, processing resources are idle while I/O is being performed making it unsuitable for a

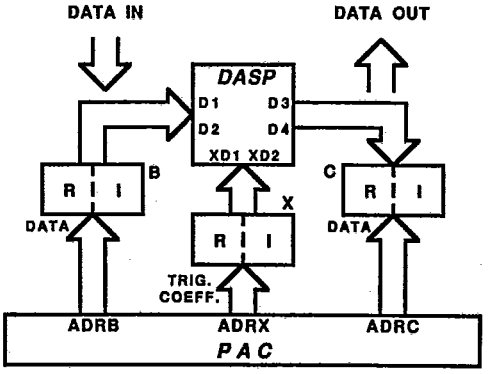


FIGURE 8: CORE DUAL MEMORY RECURSIVE STAGE

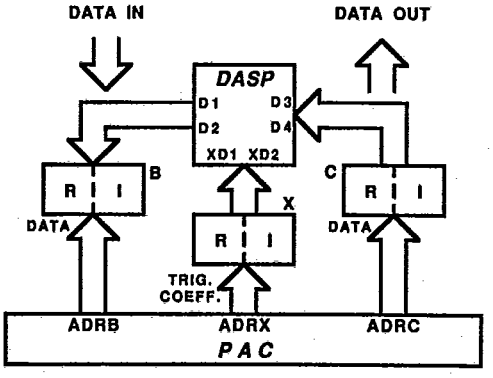


FIGURE 9: CORE STAGE - REVERSE DIRECTION

real-time system. A real-time DSP system can be configured by taking advantage of all five PAC address buses in a double buffered I/O configuration. As shown in Figure 10, two memories may be devoted to input and output while three other memories are used for processing. If the (K+1)th frame of data is being collected in memory A, the (K-1)th processed data frame is being dumped out from memory D. Memories B and C act as read and write memories for recursive execution of the DSP algorithm on the Kth frame, as discussed in the previous scheme. After the algorithm is executed, the processing can be held until the (K+1)th frame of data has been collected in memory A and dumping of the (K-1)th processed data frame from memory D has been completed. At this time, the roles of memories A and D are interchanged with those of memories B and C to begin processing of the (K+1)th frame, as shown in Figure 11. The PAC accommodates the new roles of memories by switching address sequencers among the address buses ADRA, ADRB, ADRC and ADRD. The system continues working in this fashion by reallocating the roles of memories every pass. Data may be processed in real-time by using this double buffered I/O scheme. Since the PAC has separate clocks for input (ICLK), output (OCLK) and processing (CLKIN), the clock speeds may be set so that data processing is completed before the next frame is acquired.

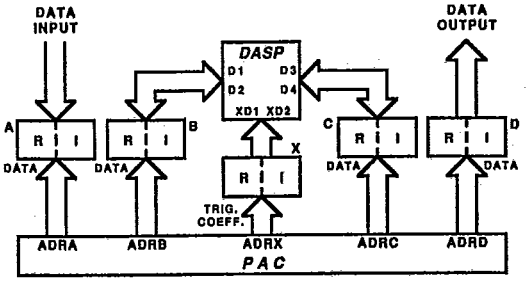


FIGURE 10: DUAL MEMORY RECURSIVE SYSTEM WITH I/0 BUFFERS

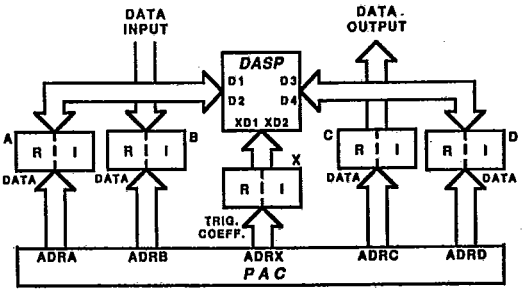


FIGURE 11: RECURSIVE SYSTEM WITH I/0 BUFFERS - MEMORY REALLOCATION

### 4.3 Recursive Architecture: Single Memory

The system cost, size and power can be vastly reduced by configuring a system around a single Data Memory, as shown in Figure 12. Since the PAC employs an in-place FFT algorithm, the read and write operations can be interleaved to one data memory. One of DASP's I/O ports is ignored and only two of PAC's address buses are utilized. The performance of such a system is halved due to the I/O bottleneck on the DASP data port. Because only one I/O port is used, it takes twice the number of cycles, compared to previous systems, to perform all the I/O operations on a single data port. Therefore, the internal computational rate of the DASP is automatically halved in order for such a system to function. As shown in Figure 12, a data frame will first be collected in memory A by invoking an address sequence on the PAC address bus ADRA. Then, the data will be processed by executing multiple passes through the DASP. During each pass, the PAC alternates between read and write cycles, providing the proper address value at the right time on address bus ADRA. Finally, the processed data is dumped out by invoking a sequential or digit-reversed sequence on the address bus ADRA.

T-49-12-05

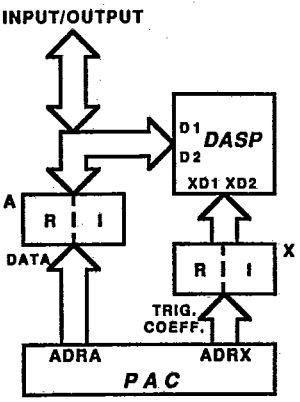


FIGURE 12: SINGLE MEMORY RECURSIVE SYSTEM

### 4.4 Cascaded Architecture

For extremely high throughputs, several DASP/PAC stages can be cascaded. With such architectures, complex data can be processed at 50 MHZ rates and real data can be processed at 100 MHZ rates. In a cascaded system, several core recursive stages(Figure 8) are interconnected in a cascaded fashion to achieve the desired performance. The resultant system architecture is depicted in Figure 13. In this system, each processing node executes the same instruction on successive data input arrays, and the algorithm is distributed over several processing nodes. For example, the M columns of an FFT algorithm are executed over M processing nodes to achieve the highest performance. The data is passed from one stage to the next by the reallocation of neighboring memories to the neighbor DASP devices. In Figure 13, let us assume that the Kth data frame is being collected in memory A1. The (K-1)th data frame is being processed on node 1 by transferring the data from memory B1 to C1. Concurrently, the previous frames are being processed on other stages. The last stage is processing the (K-p)th frame to complete the processing on that frame. The (K-p-1)th processed frame is being output from memory Dp. When each processing node has completed processing, which is primarily transferring the data from one memory to another through the DASP, the memories are reallocated, as shown in Figure 14. The memory B1 becomes the new input memory and the memory A1, which contains the Kth data frame, becomes the processing memory for the first node and so on. Note that when the system is in the mode shown in Figure 14, the PAC devices generate the write address sequence for its left neighbor and the read address sequence for its right neighbor. The PAC has two control registers to define the node type of its left and right neighbors (for a cascaded system) which enables the PAC to manage such a system. This eliminates any need for multiplexers, etc. in the address paths, enhancing the memory cycle time. The cascaded system will alternate between the configurations shown in Figure 13 and Figure 14 (at every data array pass) to execute the desired algorithm. Note that the example shown here illustrates an extremely high performance system where an algorithm, containing p passes, is distributed over p stages. A pass may be a window pass, FFT column, magnitude-square, FFTNN etc. or general purpose processing. Intermediate configurations between the cascaded system and the recursive system are also possible. A system may employ several cascaded stages, but each stage may recurse several times. In this way, the user may attain an ideal cost/performance trade-off for a given system.

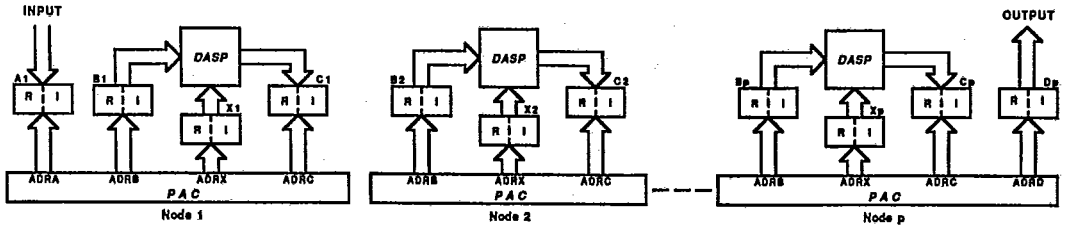


FIGURE 13: CASCADED SYSTEM

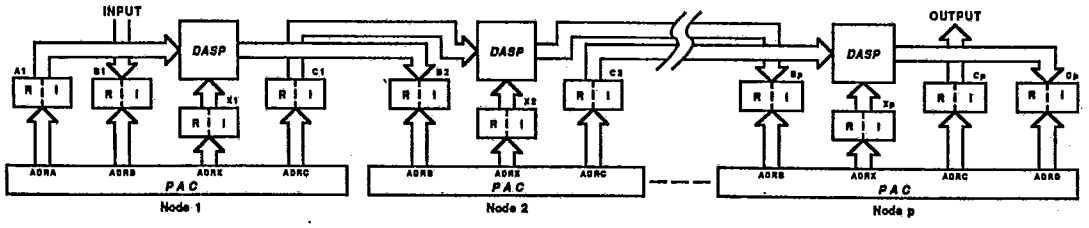


FIGURE 14: CASCADED SYSTEM - MEMORY REALLOCATION

### 4.5 FFT Systems

FFT systems can be easily implemented by using any of the above mentioned architectures. The algorithm is defined by programming the PAC instruction memory, and the hardware configuration is defined by appropriately programming the PAC control registers. For example, a 256 complex-point radix-4 FFT is implemented by four basic instructions of the PAC resulting in four data array passes as shown below.

- BFLY4 for FFT column 0
- BFLY4 for FFT column 1
- BFLY4 for FFT column 2
- BFLY4 for FFT column 3

This FFT algorithm can be turned into a spectrum analysis algorithm by adding two additional passes (two PAC instructions).

- BMUL for windowing
- BFLY4 for FFT column 0
- BFLY4 for FFT column 1
- BFLY4 for FFT column 2
- BFLY4 for FFT column 3
- BSQSM for magnitude-square

Note that memory for holding the window coefficients will be required in addition to the trigonometric coefficient memory. For the BMUL function, the PAC will address the window memory with a sequential or symmetric sequential sequence.

If the data is real, then two separate data sequences are combined in the input memory; one becoming real and the other imaginary. Then, a complex data FFT is performed in a normal fashion. At the end, the FFTNN function is applied to split the complex FFT into two separate real FFT's. For example, two 256-point real FFT's are executed by 5 data array passes (PAC instructions) as follows:

- BFLY4 for FFT column 0
- BFLY4 for FFT column 1
- BFLY4 for FFT column 2
- BFLY4 for FFT column 3
- FFTNN for recombining the complex FFT into two separate FFT's

At the end, two separate, transformed data frames will be left in the upper and lower part of the output data memory.

When the FFT is being performed on a data array of size N, where N is an odd power of 2, then a mixed-radix FFT can be performed, instead of a radix-2 FFT, to achieve higher performance. For example, the following passes (PAC instructions) will be required to execute a 512 point, complex FFT which also includes windowing and magnitude-square.

T-49-12-05

- BMUL for windowing
- BFLY2 for a radix-2 FFT column 0
- BFLY4 for a radix-4 mixed mode FFT column 0
- BFLY4 for a radix-4 mixed mode FFT column 1
- BFLY4 for a radix-4 mixed mode FFT column 2
- BFLY4 for a radix-4 mixed mode FFT column 3
- BSQSM for square sum

If a radix-2 FFT is desired, instead of a mixed-radix FFT in the above mentioned example, then the five butterfly passes can be replaced by nine radix-2 butterfly passes (corresponding to a radix-2 FFT flowgraph).

In the above mentioned examples, a pass means the transferring of an N-point data array from one memory to another (or to the same memory in a single memory system) through the DASP. It is possible to implement the set of those passes on a recursive dual memory system, a recursive single memory system, a cascaded memory system (where passes are distributed over multiple stages) or intermediate hybrid systems to achieve the desired performance. If the memory can be operated at a cycle time of C, then the pass will be of the duration of about NC for a dual-memory system and about 2NC for a single-memory system. Note that some types of passes, such as FFTNN, will not be possible on a single-memory system due to the non-in-place nature of the pass.

Formulas for determining processing times for any size FFT on various system architectures are presented in Table 7. In these formulas, N is the data array size, M is $\log_4 N$ or $\log_2 N$ and C is the memory cycle time. K is the number of optional passes for performing windowing or magnitude-squaring if desired. Note that 24 memory cycles have been added to account for the DASP data path latency and the PAC instruction switching overhead. The formulas assume a radix-4 FFT when N is a power of 4. When N is an odd power 2, a mixed-radix FFT is assumed. The FFTNN function is utilized when the data is real. Some benchmarks, based on the above mentioned formulas, for specific size FFT's are shown in Table 2, assuming a machine cycle time of 80 nS (C = 20 nS).

When the transform size is small, the additional 24 memory cycles per data pass added due to the pipeline effect, may become significant for some applications. In this case, the user has an option of using a dedicated

TABLE 7: FORMULAS FOR COMPUTING FFT PERFORMANCE

T-49-12-05



| OPERATION | CASCADED SYSTEM | DUAL MEMORY SYSTEM (ONE STAGE) | ONE MEMORY SYSTEM (ONE STAGE) |
|---|---|---|---|
| N Complex-Point FFT/IFFT If $N = 4^M$ | (24+N)C (M+K Stages) | (M+K)(N+24)C | 2(M+K)(N+24)C |
| N Complex-Point FFT/IFFT If $N = 2(4^M)$ | (N+24)C (M+K+1 Stages) | (M+K+1)(N+24)C | 2(M+K+1)(N+24)C |
| N Real Point FFT/IFFT If $N = 4^M$ | $\frac{(N+24)C}{2}$ (M+K+1 Stages) | $\frac{(M+K+1)(N+24)C}{2}$ | 2(M+K+1)(N+24)C |
| N Real Point FFT/IFFT If $N = 2(4^M)$ | $\frac{(N+24)C}{2}$ (M+K+2 Stages) | $\frac{(M+K+2)(N+24)C}{2}$ | 2(M+K+2)(N+24)C |

controller instead of the PAC which can eliminate the pipeline effect. Such a controller can be built by using a counter and a few PROM's.

### 4.6 Fast Convolution and Digital Filtering

For large sequences, a time domain convolution may be more efficiently performed by first transforming the two sequences to the frequency domain via the FFT, multiplying the spectra together and performing the inverse FFT transform on the result [Reference 2 (p.198), Reference 3 (p.633)]. This process is known as fast convolution. Often times, the linear convolution of an infinitely long sequence with a finite length, constant sequence is desired (as in digital filtering). One method for accomplishing this convolution is to break the long sequence into data arrays of length N. This data is convolved by using the above mentioned method. Because of the finite array length, the next data array must be overlapped with the previous array by an amount at least equal to the constant sequence length, K (filter length) as shown in Figure 15 [Reference 3 (p.679)]. When data is output, the first K values must be discarded, as shown in Figure 15. This I/O scheme, which is known as the "overlap/discard" method, could become very complex for LSI/MSI implementation. The PAC accommodates this overlap discard method by providing special addressing sequences, control signals, and a special control register to set the overlap amount. A data flow diagram, illustrating the process of digital filtering in the frequency domain, is shown in Figure 16. A recursive dual memory system implementation for such a system is shown in Figure 17. Basically, two additional K-word memories (OVA and OVB) which hold the overlapped points, have been added to the recursive system architecture discussed earlier. The memories on the ADRX address bus hold the trigonometric coefficient table for the FFT, window coefficients and the desired frequency response of the digital filter. A cascaded system architecture can also be built for digital filter applications by following the schemes discussed in section 4.4.

### 4.7 Digital Filter System

The digital filter algorithms can be implemented on a desired system architecture by simply programming a few passes in the PAC. Let us assume that a digital filter needs to be built with 512 poles and 50% overlapping data frames. Therefore, the data frames will consist of 1024 complex points. The filter algorithm will consist of the following 11 passes (PAC instructions).

T-49-12-05



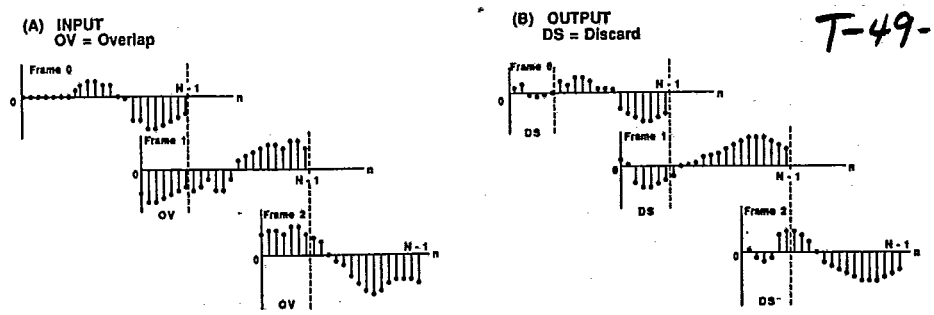


FIGURE 15: OVERLAP/DISCARD FAST CONVOLUTION

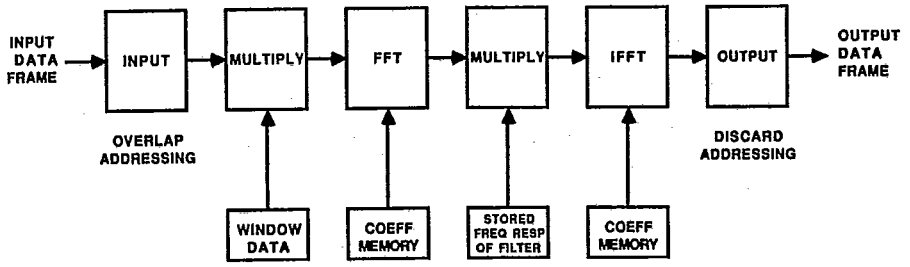


FIGURE 16: DIGITAL FILTERING IN FREQUENCY DOMAIN



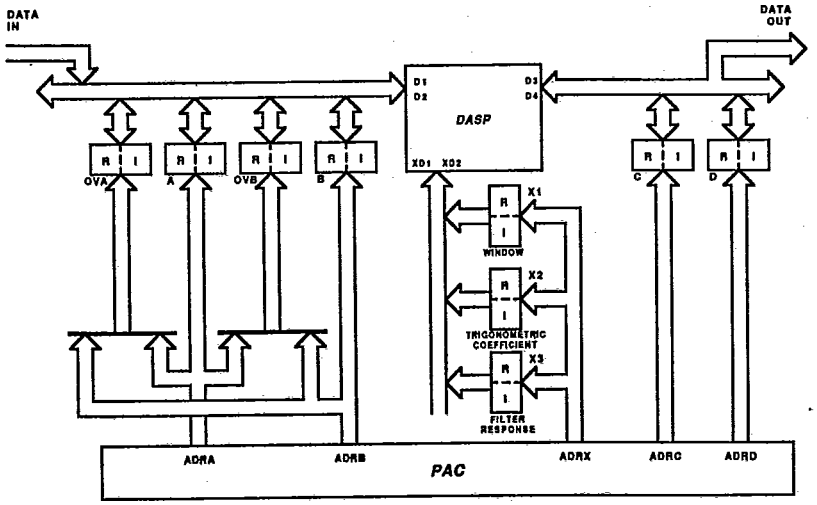


FIGURE 17: OVERLAP/DISCARD RECURSIVE SYSTEM WITH I/0 BUFFERS

- BMUL for windowing
- BLFY4 for FFT Column 0
- BLFY4 for FFT Column 1
- BLFY4 for FFT Column 2
- BLFY4 for FFT Column 3
- BLFY4 for FFT Column 4
- BMUL for multiplying with filter response
- BFLY4 for IFFT Column 0
- BFLY4 for IFFT Column 1
- BFLY4 for IFFT Column 2
- BFLY4 for IFFT Column 3
- BFLY4 for IFFT Column 4

These passes can be implemented on a recursive DUAL memory architecture (Figure 17) or a cascaded system. Benchmarks for 32, 128, 512 and 2048 pole filters, implemented with fast convolution, are stated in Table 3, assuming a machine cycle time of 80 nS.

## 5.0 NUMERIC PERFORMANCE

As mentioned earlier, the DASP's internal arithmetic is a mixture of 16 and 20-bit two's complement formats. Hence, the error induced by its internal quantization and rounding is significantly less than that induced by straight 16-bit arithmetic. Additionally, because of DASP's block floating point capability, the dynamic range of the input data that can be processed is increased dramatically. Figures 18 and 19 illustrate the signal-to-noise ratio as well as the dynamic range of data transformed on the DASP. The input data is 16-bit full scale random noise for the signal-to-noise ratio analysis and 12-bit (input SNR 72 dB) for the dynamic range analysis.

## 6.0 TEST FEATURES AND RELIABILITY

Great emphasis on test and reliability has been placed in the architecture definition, design, production and testing of the DASP and the PAC devices. The architectures of the DASP and the PAC are highly accessible from the pins, allowing an acceptable fault coverage with a moderate number of test vectors. Furthermore, all registers of the DASP and the PAC are on a serial scan path, providing an easy access to all the hardware elements. In addition to the production testing of devices, the serial scan path is operated through a separate interface, called the Non-Functional-Test (NFT) interface, which is independent of the normal functional pins.

## 7.0 SPECIFICATIONS T-49-12-05

### 7.1 DASP (HDSP66110)

The DASP is packaged in a ceramic, 269 pin, Pin-Grid-Array (PGA) Package (a surface mount version will be available in the future). The key parameters of the HDSP66110 are listed in Table 8. The package is illustrated in Figure 20 and the pin description is shown in Table 9. The device is qualified for both commercial and military applications.

### 7.2 PAC (HDSP66210)

The PAC is packaged in a ceramic,180 pin, PGA package (a surface mount version will be available in the future). The package is illustrated in Figure 21 and the pin description is shown in Table 10. The key parameters of the HDSP66210 are listed in Table 11.

## 8.0 SYSTEM DEVELOPMENT SUPPORT TOOLS

The primary emphasis in the architectures of the DASP/PAC devices is on high performance, ease of use and providing total system solutions. Therefore, a DSP system can be easily built around DASP/PAC devices with minimum of effort and a great hardware efficiency. The system development task is further eased by several support tools.

- Printed User's Guides for the DASP and the PAC
- Software Simulators for the DASP/PAC Systems (VAX VMS and IBM PC)
- Hardware Evaluation Module (EVM) for the DASP/PAC System

Figure 22 snows a typical DSP system development cycle using DASP/PAC support tools.

### 8.1 User's Guides and Application Notes

The DASP and PAC User's Guides contain detailed architectural descriptions, interface information, electrical/mechanical specifications and system application notes.

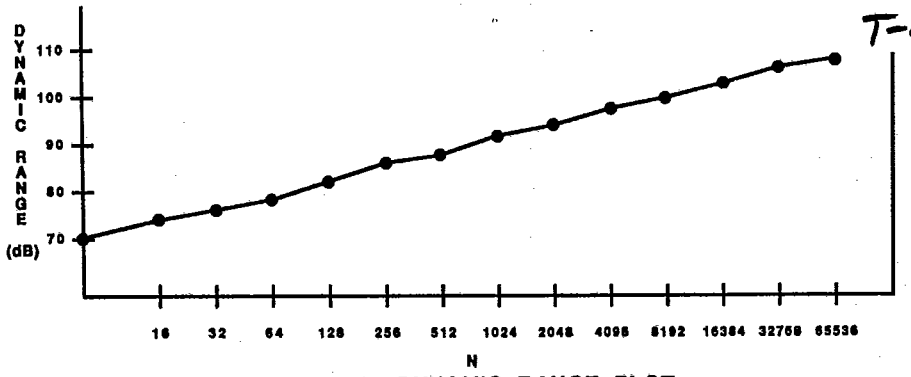


FIGURE 18: DYNAMIC RANGE PLOT

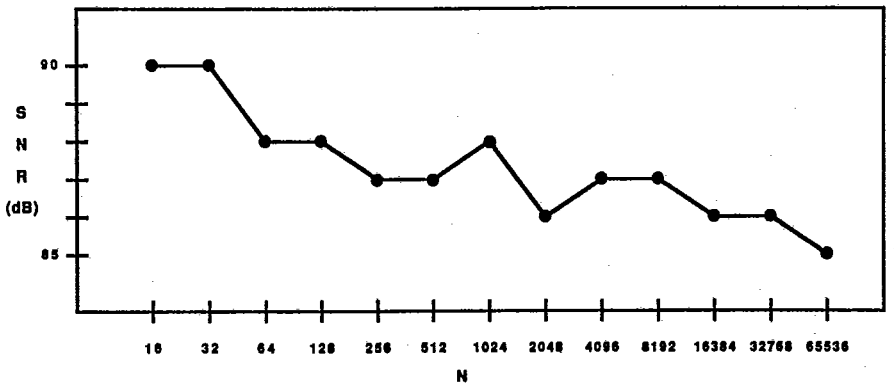


FIGURE 19: SIGNAL-TO-NOISE RATIO PLOT

6

TABLE 8: DASP KEY PARAMETERS

- Typical Supply Voltage: 5 Volts
- Typical Power Dissipation: 1.2 Watts
- TTL Compatible Interfaces
- Minimum Machine Cycle Time: 80 nS (Tm)
- Maximum Input Master Clock (CLKIN)
  - — 25 MHz (2/Tm) in Quad Mode
  - — 50 MHz (4/Tm) in Dual Mode

DSP

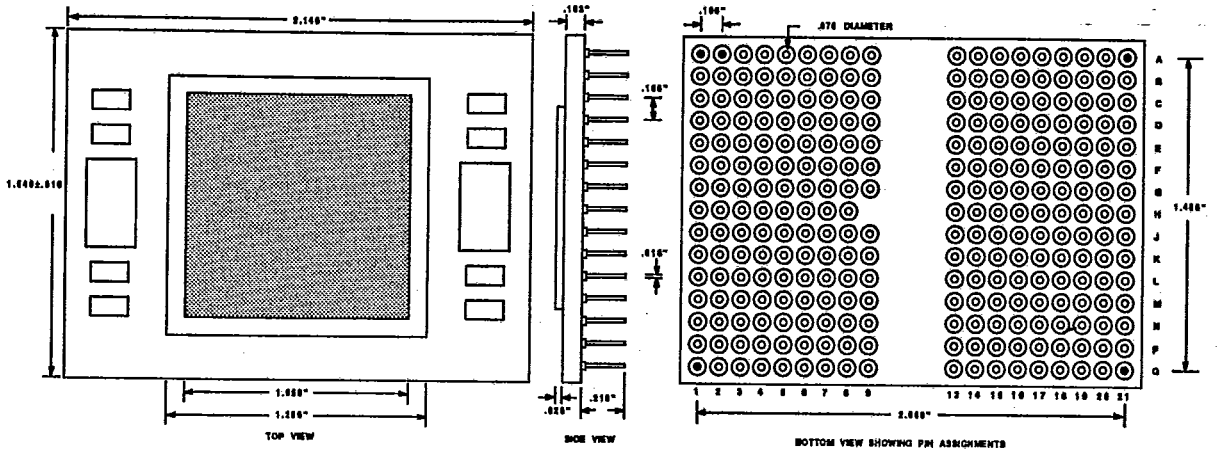


FIGURE 20: DASP PACKAGE

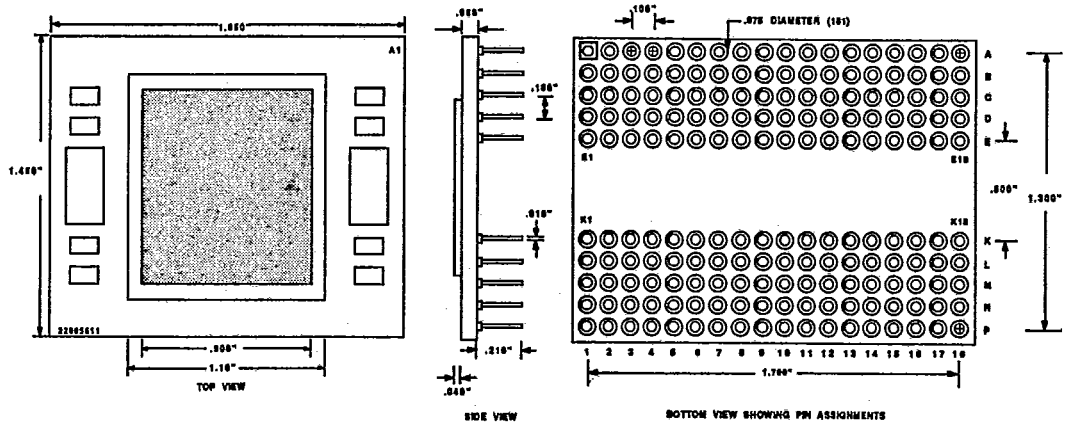


FIGURE 21: PAC PACKAGE

TABLE 9: DASP PIN DESCRIPTION

T-49-12-05

| PIN NAME | I/O TYPE | DESCRIPTION | NUMBER OF PINS USED | |
|---|---|---|---|---|
| | | | QUAD MODE | DUAL MODE |
| **DATA/AUX-DATA BUSES** | | | | |
| D1 (15:0) | I/O | I/O Data Bus | 16 | 16 |
| D2 (15:0) | I/O | I/O Data Bus | 16 | 16 |
| D3 (15:0) | I/O | I/O Data Bus | 16 | 16 |
| D4 (15:0) | I/O | I/O Data Bus | 16 | 16 |
| D5 (15:0) | I | Input Data Bus | 16 | None |
| D6 (15:0) | I | Input Data Bus | 16 | None |
| D7 (15:0) | O | Output Data Bus | 16 | None |
| D8 (15:0) | O | Output Data Bus | 16 | None |
| XD1 (15:0) | I | Input Aux Data Bus | 16 | 16 |
| XD2 (15:0) | I | Input Aux Data Bus | 16 | 16 |
| XD3 (15:0) | I | Input Aux Data Bus | 16 | None |
| XD4 (15:0) | I | Input Aux Data Bus | 16 | None |
| **CONFIGURATION CONTROLS** | | | | |
| QUAD | I | Quad Bus Mode | 1 | 1 |
| MUXRW | I | Multiplexed Read Write (Single Memory System) | 1 | 1 |
| RVDIR | I | Reverses the Directionality of I/O Data Buses | 1 | 1 |
| OE' | I | Output Enable: Tristates All Data Buses if High | 1 | 1 |
| **DATA CONTROLS** | | | | |
| FC (5:0) | I | Function - Code to Define Function | 6 | 6 |
| SFI (2:0) | I | Shift Input: Controls the Right Shift on Incoming Data | 3 | 3 |
| XSFI | I | Auxiliary Shift Input: Controls the Shift on Incoming Auxiliary Data | 1 | 1 |
| COMP (3:0) | I | Complement Control for the Input Data Buses | 4 | 2 |
| OVF/OVFP/ OVFA | O | Overflow Outputs from Internal Processors | 3 | 3 |
| **SHIFT FACTOR GENERATOR FOR BLOCK FLOATING POINT** | | | | |
| SFO (2:0) | O | Scale Factor (SF) Output: Determines Scaling for the Next Pass | 3 | 3 |
| ASFI (3:0) | I | Accumulated SF Input | 4 | 4 |
| ASFO (3:0) | O | Accumulated SF Output | 4 | 4 |

TABLE 9: DASP PIN DESCRIPTION (Cont.)

T-49-12-05



| PIN NAME | I/O TYPE | DESCRIPTION | NUMBER OF PINS USED | |
|---|---|---|---|---|
| | | | QUAD MODE | DUAL MODE |
| **SHIFT FACTOR GENERATOR FOR BLOCK FLOATING POINT (CONT.)** | | | | |
| INITPR | I | Initializes Scale Factor Processor | 1 | 1 |
| RDX16 | I | Configures the Scale Factor Generator for Radix-16 Mode | 1 | 1 |
| **TIMING CONTROLS** | | | | |
| CLKIN | I | Clock Input: At 4/Tm Frequency in Dual Bus Mode and 2/Tm Frequency in Quad Bus Mode | 1 | 1 |
| SYNC | I | System Clock Synchronization Signal | 1 | 1 |
| MCLKOUT | O | An Output Clock Signal at Machine Cycle Rate (Frequency 1/Tm) | 1 | 1 |
| CLKOUT | O | A Delayed Version of CLKIN, Compensated for Clock to Output Data Delay. Useful for Memory Writes. | 1 | 1 |
| BOP | I | Beginning of Pass Signal | 1 | 1 |
| EOP | I | End of Pass Signal | 1 | 1 |
| **SERIAL SCAN FOR NON-FUNCTIONAL-TEST (NFT)** | | | | |
| NFT | I | NFT in Progress | 1 | 1 |
| SIN | I | Serial Scan Input | 1 | 1 |
| SOUT | O | Serial Scan Output | 1 | 1 |
| NFTR | I | Reset Registers for NFT | 1 | 1 |
| NFTS | I | NFT Shift in Progress | 1 | 1 |
| EN | I | Enable Registers for Normal Operation | 1 | 1 |
| **SUPPLY** | | | | |
| VCC | I | Voltage Supply | 3 | 3 |
| GND | I | Ground | 3 | 3 |
| **TOTAL PINS USED ON THE PACKAGE** | | | **244** | **146** |

**TABLE 10: PAC PIN DESCRIPTION**

T-49-12-05

| PIN NAME | I/O TYPE | DESCRIPTION | NUMBER OF PINS USED |
|---|---|---|---|
| **ADDRESS BUSES** | | | |
| ADRA (15:0) | I/O | (Output) Address Bus for Memory A/(Input) Host Address Bus | 16 |
| ADRB (15:0) | I/O | (Output) Address Bus for Memory B/(Input) Host Data Bus | 16 |
| ADRC (15:0) | O | Address Bus for Memory C | 16 |
| ADRD (15:0) | O | Address Bus for Memocy D | 16 |
| ADRX (15:0) | O | Address Bus for Auxiliary Memory X | 16 |
| OE' | I | Output Enable: Tristates Address Buses if High | 1 |
| **MEMORY CONTROLS** | | | |
| AWE' | O | Memory A Write Strobe | 1 |
| BWE' | O | Memory B Write Strobe | 1 |
| CWE' | O | Memory C Write Strobe | 1 |
| DWE' | O | Memory D Write Strobe | 1 |
| OVAWE' | O | Filter Overlap Memory A Write Strobe | 1 |
| OVBWE' | O | Filter Overlap Memory B Write Strobe | 1 |
| OVACS' | O | Filter Overlap Memory A Chip Select | 1 |
| OVBCS' | O | Filter Overlap Memory B Chip Select | 1 |
| **PASS EXECUTION CONTROLS** | | | |
| BOP | O | Beginning of Pass Signal | 1 |
| EOP | O | End of Pass Signal | 1 |
| EOPR | O | End of Process Signal | 1 |
| IBUSY | O | Input Data Collection In Progress | 1 |
| OBUSY | O | Output Data Dump In Progress | 1 |
| IFULL | O | Input Memory Full | 1 |
| OEMPTY | O | Output Memory Empty | 1 |
| **HOST INTERFACE/INITIALIZATION CONTROLS** | | | |
| AUTOBOOT | I | Autoboot from a Memory | 1 |
| BOOTDONE | O | PAC Autoboot Complete | 1 |
| CS' | I/O | (Input) PAC Chip Select from Host/(Output) Autoboot Memory Select | 1 |
| WR' | I | Write Strobe from Host | 1 |
| RST' | I | Reset Input | 1 |
| INITPR | O | Initializes External Processor | 1 |
| GO | I | Start Instruction Execution | 1 |
| | | NOTE: Host Data/Address Buses Listed in "Address Buses" Above. | |

TABLE 10: PAC PIN DESCRIPTION (Cont.)

T-49-12-05



| PIN NAME | I/O TYPE | DESCRIPTION | NUMBER OF PINS USED |
|---|---|---|---|
| **GENERAL COMMUNICATION SIGNALS** | | | |
| PO (7:0) | I/O | (Output) Programmable Outputs from Internal Instruction/(Input) External Instruction Fields XSHF (1:0), MIXMD, NODE (4:0) | 8 |
| IP (4:0) | I/O | (Output) Internal Instruction Memory Pointer/ (Input) External PAC Instruction Field BSC (3:0) | 5 |
| BSC (3:0) | O | Bus Switch Code: Controls Internal Bus Multiplexers | 4 |
| MUXRW | O | Multiplexed Read Write (Single Memory System) | 1 |
| **QUAD-MODE CONTROLS** | | | |
| QAS0 | O | Quad-Mode Address Strobe 0 | 1 |
| QAS1 | O | Quad-Mode Address Strobe 1 | 1 |
| QAS2 | O | Quad-Mode Address Strobe 2 | 1 |
| QAS3 | O | Quad-Mode Address Strobe 3 | 1 |
| **CLOCK SIGNALS** | | | |
| CLKIN | I | System Clock Input (Frequency 4/Tm) | 1 |
| ICLK | I | Input Data Collection Clock | 1 |
| OCLK | I | Output Data Dump Clock | 1 |
| HCLK | I | Host Interface Clock | 1 |
| SYNC | I | System Clock Synchronization Signal | 1 |
| CLKOUT | O | Delayed System Clock Output (Frequency 4/Tm) | 1 |
| MCLKOUT | O | Machine-Cycle Clock Output (Frequency 1/Tm) | 1 |
| **SERIAL SCAN FOR NON-FUNCTIONAL-TEST (NFT)** | | | |
| NFT | I | NFT Testing in Progress | 1 |
| NFTS | I | NFT Scan in Progress | 1 |
| SIN | I | NFT Serial Scan Input | 1 |
| SOUT | O | NFT Serial Scan Output | 1 |
| **SUPPLY** | | | |
| VCC | I | Voltage Supply | 2 |
| GND | I | Ground | 2 |
| **TOTAL PACKAGE PINS USED** | | | 156 |

**TABLE 11: PAC KEY PARAMETERS**

T-49-12-05

- **Typical Supply Voltage: 5 Volts**
- **Typical Power Dissipation: 2 Watts**
- **TTL Compatible Interfaces**
- **Minimum Machine Cycle Time: 80 nS (Tm)**
- **Minimum Micro Cycle Time: 20 nS (Tm/4)**
- **Maximum Input Master Clock (CLKIN): 50 MHz (4/Tm)**



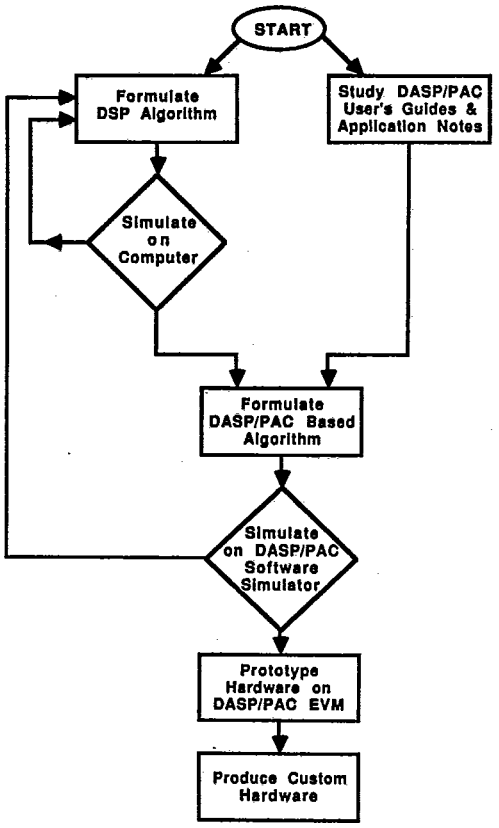




**FIGURE 22: DASP/PAC SYSTEM DEVELOPMENT CYCLE**

## 8.2 Software Simulators

Software simulators for the DASP and the PAC run on the IBM PC and VAX VMS systems. The key features of these simulators are shown in Tables 12 and 13 respectively. The IBM PC based software simulator is a menu-driven, turn-key program which integrates a and graphical display modules. The user interface is compatible with that of the EVM board, which is discussed later.

The VAX VMS based simulator can be used as a stand alone simulation tool or integrated into a larger system simulation by calling the simulator module as a subroutine. Both the DASP arithmetic system and a floating point version are supported. This enables the system designer to easily execute an algorithm on both arithmetic systems and compare the results for signal-to-noise ratio and dynamic range analysis.

**TABLE 12: IBM-PC BASED SOFTWARE SIMULATOR**

- PURPOSE — To become acquainted with DASP/PAC architectures, verify user's algorithm on DASP/PAC, use intermediate arrays for debugging hardware.
- Coded in Assembly language and Z-Basic.
- Data array generation capability (sinusoids).
- DSP algorithm defined in the PAC instruction memory and control registers.
- Algorithm executed on the DASP model.
- Array display/print options.
- Data/program file I/O.
- Graphical displays and menu driven user interface (compatible with EVM board).

**TABLE 13: VAX-VMS BASED SOFTWARE SIMULATOR**

- PURPOSE — Verify algorithm on DASP arithmetic and compare results with floating point arithmetic. Higher level system simulation by calling simulator as a subroutine.
- Coded in "C" language.
- DASP and PAC models are integrated to execute the desired algorithm.
- Algorithm defined by composing the PAC passes in an external file.
- Corresponding floating point computations available for error analysis.
- Simulator module may be called as a "C" subroutine.

### 8.3 Hardware Evaluation Module (EVM)

The EVM, which integrates the DASP/PAC and associated system hardware, allows algorithm verification on the hardware. It is capable of exercising the majority of features of the DASP/PAC devices, therefore providing great freedom to the user for system design. The key features of the EVM board are listed in Table 14. The EVM board provides an in-depth understanding of the DASP/PAC based hardware in addition to algorithm verification. A user specific system, which normally would be a subset of the EVM hardware, can be easily built after working with the EVM.



TABLE 14: EVM KEY FEATURES

- **Circuit board contains one DASP, one PAC and memories to support the processing of data arrays up to 8K size.**
- **Supports overlap/discard memories for digital filtering.**
- **Supports recursive dual memory systems.**
- **Double buffered auxiliary (coefficient) memory for FFT based adaptive systems.**
- **IBM PC interface for uploading/downloading the memories and initializing the PAC.**
- **VME bus compatible (uses PC to VME adapter board).**
- **Euro-Card 9U standard card size.**
- **Utilizes J3 connector for private I/O bus to communicate with other modules such as an A/D, D/A or another EVM.**
- **Multiple EVMs can be cascaded to emulate a cascaded DASP/PAC system.**



## REFERENCES

Reference 1: Digital Signal Processing, A.V. Oppenheim and R.W. Schafer, Prentice-Hall, 1975

Reference 2: The Fast Fourier Transform, E.O. Brigham, Prentice-Hall, 1974

Reference 3: Handbook of Digital Signal Processing, D.F. Elliott, Academic Press, 1987

Reference 4: A Unified Approach to Time-and Frequency-domain Realization of FIR Adaptive Filters, G.A. Clark, S.R. Parker and S.K. Mitra, IEEE Trans. Acoust. Speech Signal Proc., ASSP-31, 1983

**For Ordering Information See Section 1.*