SLAC-232
UC-37
(I/A)

ADAPTIVE SIGNAL PROCESSOR*

Helmut Volker Walz

Stanford Linear Accelerator Center
Stanford University
Stanford, California 94305

July 1980

Prepared for the Department of Energy

under contract number DE-AC03-76SF00515

Printed in the United States of America. Available from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Road, Springfield, VA 22161. Price: Printed Copy A04; Microfiche A01.

---

* Engineer Thesis.

ABSTRACT

An experimental, general purpose adaptive signal processor system has been developed, utilizing a quantized (clipped) version of the Widrow-Hoff least-mean-square adaptive algorithm developed by Moschner. The system accommodates 64 adaptive weight channels with 8-bit resolution for each weight. Internal weight update arithmetic is performed with 16-bit resolution, and the system error signal is measured with 12-bit resolution. An adapt cycle of adjusting all 64 weight channels is accomplished in 8 μsec. Hardware of the signal processor utilizes primarily Schottky-TTL type integrated circuits.

A prototype system with 24 weight channels has been constructed and tested. This report presents details of the system design and describes basic experiments performed with the prototype signal processor. Finally some system configurations and applications for this adaptive signal processor are discussed.

ACKNOWLEDGEMENTS

The patient support of Dr. B. Widrow of the School of Electrical Engineering at Stanford University is gratefully acknowledged. His encouragement and guidance were essential factors during the course of this work. The design, development, and prototyping work for the Adaptive Signal Processor system was carried out at the Stanford Linear Accelerator Center. I wish to thank Dr. J. L. Brown, Dr. B. Richter, and R. S. Larsen for contributing the required assistance and resources from their respective departments. Specifically the hardware fabrication and testing work by V. Smith and J. Olsen deserve my special recognition.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Interest in adaptive systems arose from experimental work carried out with an adaptive signal processor system at the Information Systems Laboratory of the School of Electrical Engineering at Stanford University. This system was implemented with peripheral devices connected to a general purpose minicomputer (Hewlett-Packard 2116B).[1]

The ability to handle signal processing problems, which are dynamic in nature, or to realize nearly optimal filter performance under unknown conditions in the field, was thought to be highly interesting for numerous instrumentation and control system problems in high energy physics research and in particle accelerator systems.

To explore some of these applications, a self-contained, portable adaptive processor has been developed, with improved performance as compared to the laboratory-based system. The processor system represents an attempt to optimize cost, speed, and accuracy parameters, while retaining flexibility to implement many different adaptive system configurations for bench and application experiments.

After a brief review of adaptive system concepts, the processor architecture and design are described in detail. Operation and performance of a prototype signal processor are reported and results of basic adaptive experiments are presented.

CHAPTER II

REVIEW OF ADAPTIVE SIGNAL PROCESSING

The following brief review summarizes ideas and concepts in adaptive signal processing, which are fundamental to the work described in this report. For a complete treatise of all topics, the reader is referred to the extensive reference literature.[2-5]

A. Adaptive Linear Combiner

The adaptive linear combiner (ALC), shown in Fig. 1, forms the heart of most adaptive systems. The output y_j is the sum of the weighted components of the signal input vector X_j, where j is an index of discrete time intervals. This is expressed as the inner product of input vector X_j and weight vector W:

$$y_j = X_j^T W = W^T X_j$$

where input and weight vectors are defined as:

$$X_j \triangleq \begin{Bmatrix} x_{0j} \\ x_{1j} \\ \vdots \\ x_{n_j} \end{Bmatrix} \qquad W \triangleq \begin{Bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{Bmatrix} .$$

We now define the error signal ε_j as the difference between the desired response and the combiner sum output y_j:

$$\varepsilon_j = d_j - y_j = d_j - X_j^T W = d_j - W^T X_j .$$

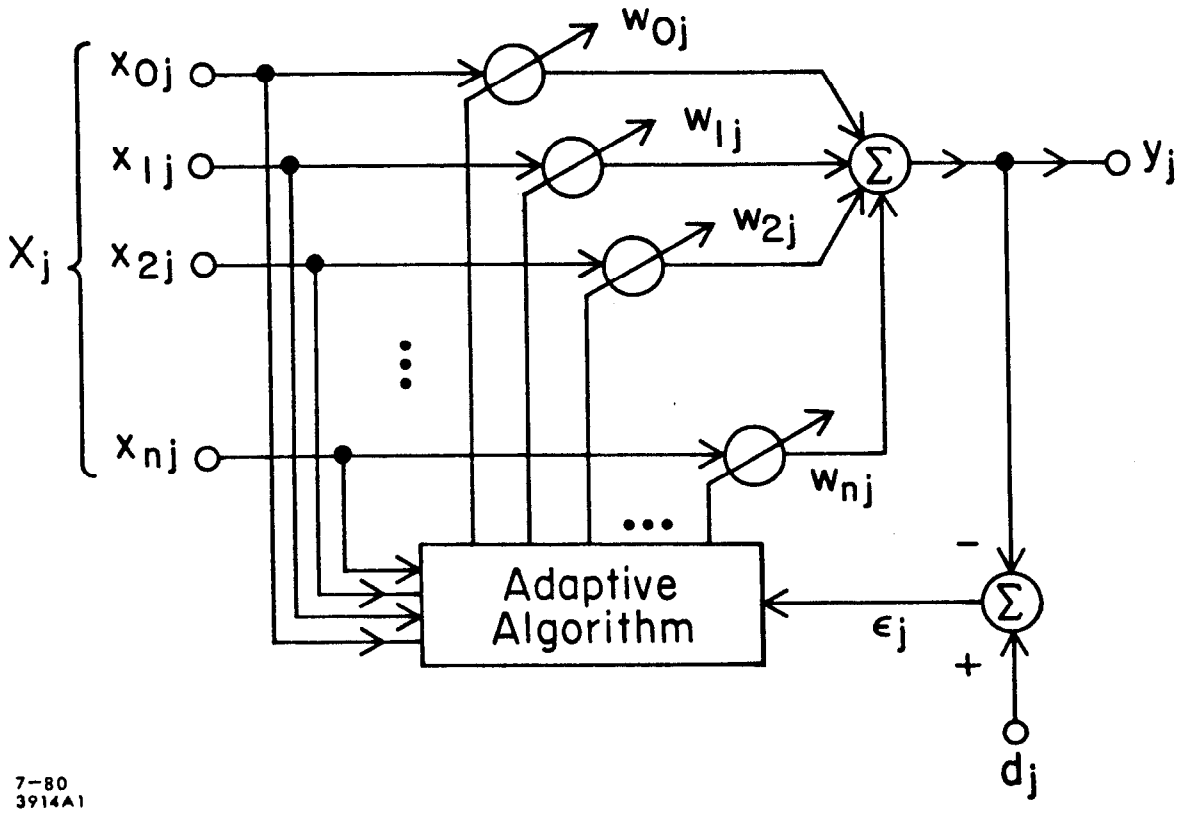


Fig. 1. Adaptive Linear Combiner.

The desired response is an externally supplied input. For a specific application, a signal is chosen, that is as similar as possible to the desired output of the combiner. The adaptive algorithm represents some mechanism which controls the weight vector W as a function of the error ε_j.

B. Adaptive Algorithms

The algorithm used as part of the ALC in Fig. 1 is the means of adjusting the weights to minimize the error in a mean-square sense. The processor described in this report utilizes an adaptive algorithm, which is based on the Least-Mean-Square (LMS) adaptive algorithm by Widrow and Hoff.[2] The weight values at time (j+1) are determined from:

$$W_{j+1} = W_j + 2\mu\varepsilon_j X_j \quad .$$

The LMS algorithm calculates the weight vector W based on the method of steepest descent, whereby the next value of a weight w_i is equal its present value plus a correction, $2\mu\varepsilon_j x_{ij}$, which is proportional to an estimate of the negative gradient. The LMS algorithm has been shown[4] to be stable and converge to produce a minimum mean-square error for:

$$\frac{1}{\lambda_{max}} > \mu > 0$$

where μ is the parameter controlling the rate of convergence and λ_{max} is the largest eigenvalue of the input correlation matrix. Also for input vectors, which are uncorrelated over time, the weight vector converges to the optimum Wiener weight vector.

We note that the execution of the LMS algorithm requires, among other operations, for each update of the weight vector W, composed of n weights, a measurement of each component of the input signal vector and the corresponding multiplication of $\epsilon_j x_{ij}$. Performing these n measurements and n multiplications may be very costly in a real-time adaptive processor system. A trade-off of performance and execution speed leads to the clipped LMS algorithm by Moschner.[5] Here adaptation is based only on the polarity of each component of the input vector, resulting in a significant saving of computation effort required. The clipped LMS algorithm is given by:

$$W_{j+1} = W_j + 2\mu\epsilon_j\tilde{X}_j$$

where $\tilde{X}_j$ is a vector with components defined as:

$$\tilde{x}_{ij} = \text{Sgn}[x_{ij}] \triangleq \begin{cases} +1 & \text{if } x_{ij} \geq 0 \\ -1 & \text{if } x_{ij} < 0 \end{cases} .$$

The adaptive signal processor described here has been tailored to the execution of the clipped algorithm, taking full advantage of the possible increase in execution speed and savings in hardware cost. A detailed comparison of performance between the conventional LMS and the clipped LMS algorithms is found in Ref. 5.

C. Basic Adaptive System Configurations

A basic system configuration is shown in Fig. 2. The system consists of the ALC circuit and a fixed preprocessor, which transforms the input B to the X input vector of the weights in the ALC. The details of the preprocessor are application-dependent. Typical

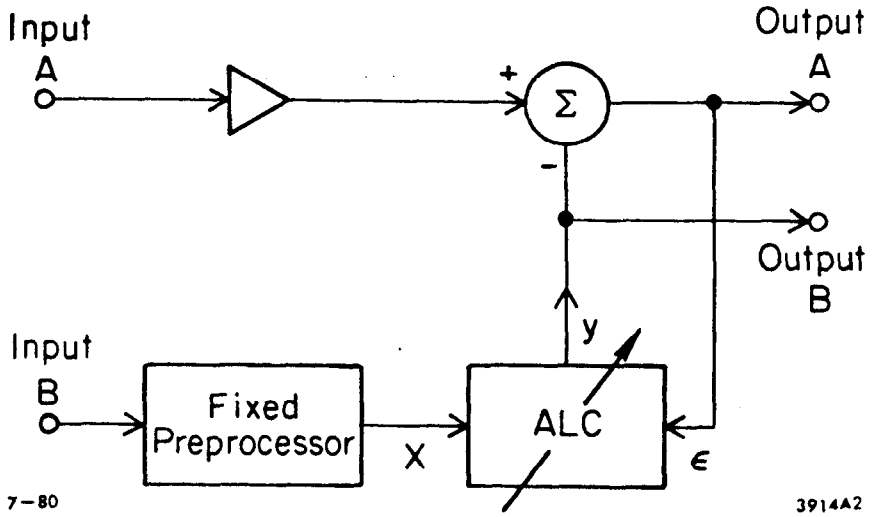


Fig. 2. Basic Adaptive System Configuration.

preprocessors are transversal filters (tapped delay lines), arrays of fixed filters (such as sets of simple RC filters connected in series or parallel), reference oscillators, harmonic generators, phase shifters, to mention a few possibilities.

The adaptive system shown may be utilized in two ways:

a) as an adaptive filter;

b) as an adaptive noise canceller.

When used as an adaptive filter, input A is referred to as desired signal input, input B is connected to the signal to be processed, and output B (the ALC output y_j) is taken as the system output. An example for this configuration is a multichannel filter (utilizing several B inputs), connected to an array of sensors.[6] By means of a training signal connected to the desired input (A), the filter is "taught" to perform a spatial separation of signals with overlapping frequency spectra.

The adaptive noise canceller designates input A as primary input and input B as reference input. The difference signal at output A is used as system output. Now the ALC produces an output y, which is an optimal estimate of a noise component in the primary input, and which is derived from correlated noise at the reference input. This minimizes noise at the system output and optimizes the ratio of signal to noise. For some signal processing problems a configuration with only one input may be used successfully (Fig. 3). The separation of broadband from periodic signal components is possible by use of a delay. This delay is required to be of sufficient length to cause the broadband signal components in the reference input to become decorrelated from those in

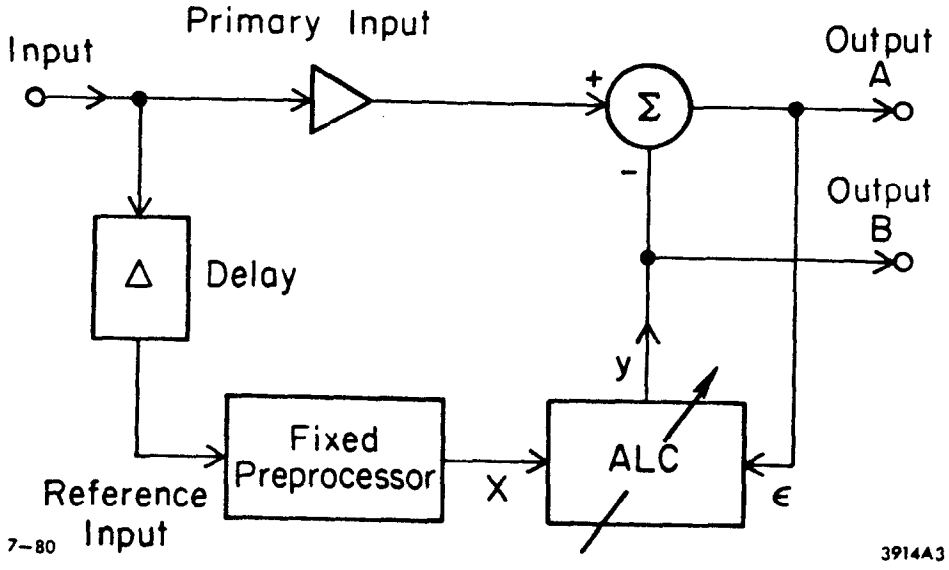


Fig. 3. Single Input Adaptive System.

the primary input. On the other hand, periodic components will remain correlated with each other. After adaptation, broadband signals remain in output A, whereas output B will contain an optimum estimate of the periodic signal components.

In Chapter IV, we will show results from real-time signal processing experiments, based on some of those configurations described here.

D. Basic Implementations of Adaptive Signal Processors

Realizations of adaptive signal processors may be categorized into 4 types of systems:

(1) Analog hardware processors.

(2) Digital hardware processors.

(3) Computer simulated processors.

(4) Hybrid hardware processors.

Any of these different types of systems is a natural choice for a certain class of application. For this project, we decided to implement a hybrid processor system. Before describing this system in more detail, we want to mention disadvantages associated with the other implementations, which led to the choice of a hybrid processor. Analog systems suffer in general from lack of stability, accuracy, and ease of programmability. Digital hardware and computer simulated processors utilize analog-to-digital conversions for all input signals and a digital-to-analog conversion to produce an output signal. This imposes a basic signal bandwidth limitation, directly related to the rate of adaptation of the processor system. According to the Nyquist criterion,

the processor has to update the output conversion at least at twice the frequency of the highest frequency component of the input signals. A hybrid system eliminates these disadvantages and combines analog and digital hardware for optimum performance.

The hybrid system approach taken is shown in Fig. 4. The system utilizes an analog signal path from the inputs through preprocessor, adjustable weight channels, summing and difference amplifiers, to the system output. The adaptive algorithm is executed by a special purpose, digital hardware processor with limited programming flexibility. Error and weight channel input signals are digitized and supplied to the digital processor.

In the following chapter we will present a detailed description of the system architecture, hardware and software implementation, and general performance of our adaptive signal processor.

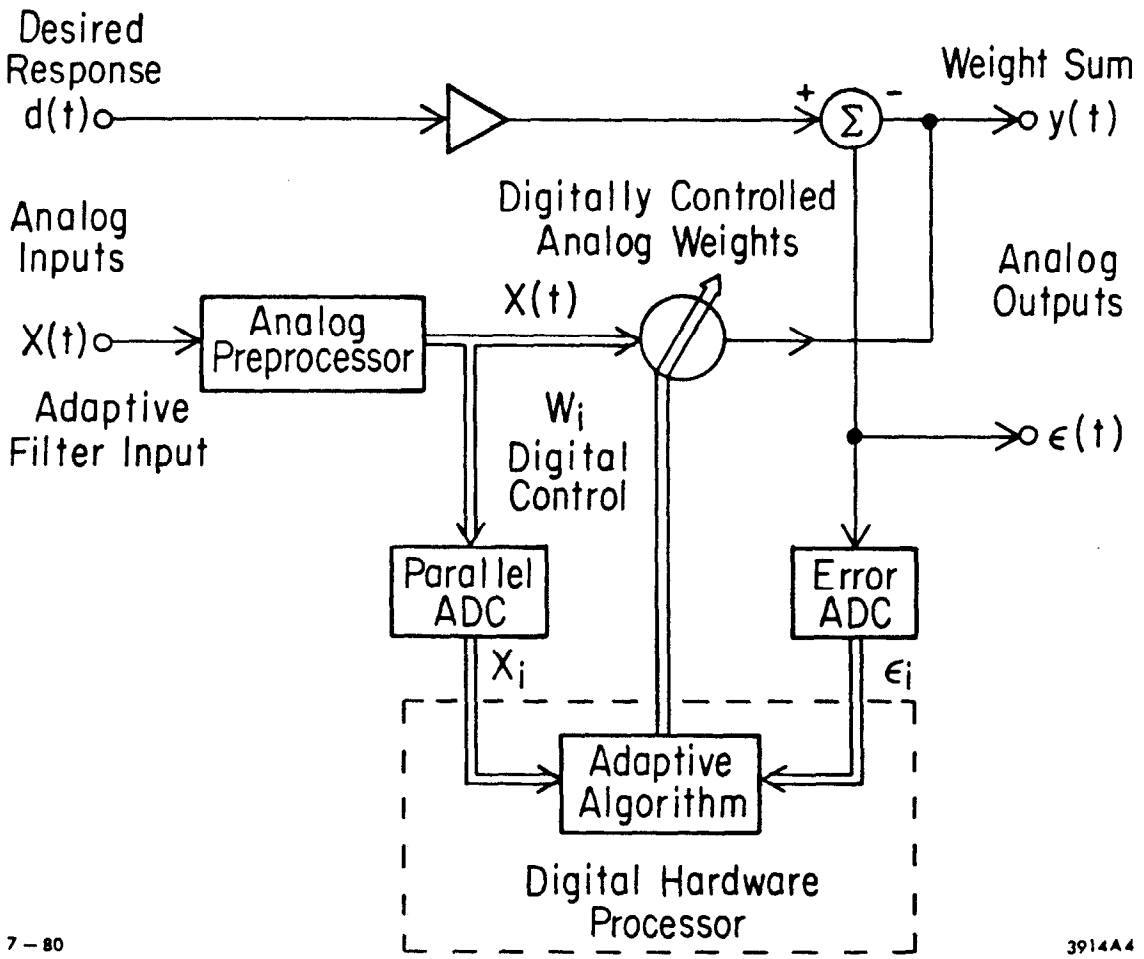


Fig. 4. Hybrid Adaptive Processor.

CHAPTER III

GENERAL SYSTEM DESCRIPTION

The design of this adaptive signal processor, ASP, is based on the hybrid processor configuration. The system has been optimized for the execution of the clipped LMS algorithm. A system block diagram is shown in Fig. 5. Four different types of modules and a system dataway are utilized to implement a system. A processor with 64 weight channels consists of 4 filter array modules (analog preprocessors), 8 weight processor modules, one error digitizer module, and one test controller module. Except for the filter array modules, all other modules connect to the system dataway for exchange of digital data and control information.

The system dataway consists of a 16-bit bidirectional data bus, a 4-bit operation code bus, a 5-bit address bus, a 4-bit clock phase bus, and several miscellaneous control lines. System supervision is handled by the test controller module which downloads initialization data and microprograms to all modules in the system and generates basic clock signals. A microcoded adapt sequencer controls the synchronous execution of the adaptive algorithm.

Each weight processor module contains 8 hybrid weight channels (with digitally controlled analog weights), followed by an analog summing amplifier. A self-contained, high-speed, digital processor executes the adaptive algorithm and adjusts sequentially each one of the 8 weight channels. This architecture of distributed satellite processors maintains a typical adapt cycle time of 8 μsec with the

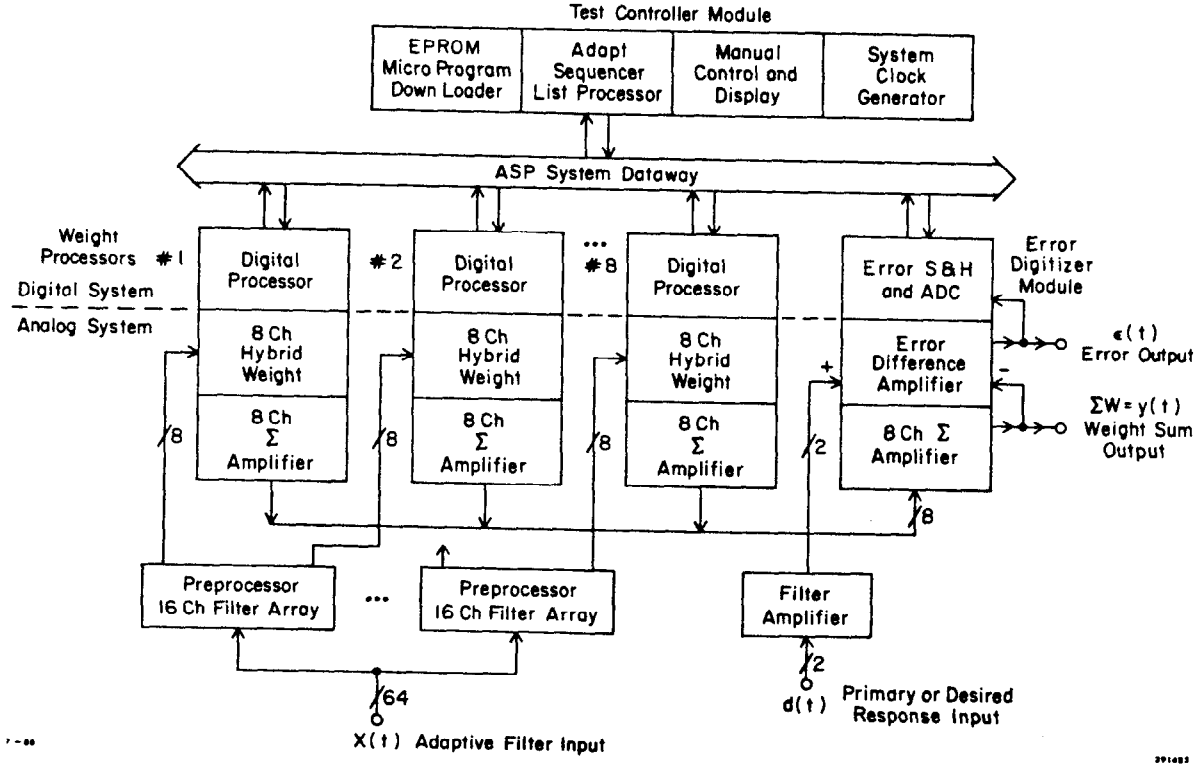


Fig. 5. ASP System Block Diagram.

clipped LMS algorithm, independent of the number of processors used in the system. Hence for a system with 8 weight processor modules and 64 weight channels, one achieves an update time of 125 nsec per weight. Each satellite processor has a 64-word by 16-bit program memory, a 16-word by 16-bit data memory, and a 16-bit wide fast ALU.

The error digitizer module completes the analog signal path with the required analog summing and difference amplifiers and a high-speed analog-to-digital converter (ADC) to measure the error signal. Figure 6 shows the basic analog signal path through the entire ASP system. Programming of the weight processor modules, the error digitizer, and the adapt sequencer is based on a machine instruction set, specifically defined for this signal processor. All programs are stored as firmware in field programmable read-only memories in the test controller module. Microcode used in the weight processors and initialization data for the error digitizer are copied from the test controller PROMS and transferred to the modules in the system by a special program downloading cycle. Instruction set and programs for the ASP system are found in Appendix B. In the following sections we describe each of the 4 modules of the signal processor in detail.

A. Weight Processor Module

The weight processor (WP) module is a self-contained, high speed, microprogrammed processor executing an adaptive algorithm. It consists of a microprogram control section, a 16-bit arithmetic-logic unit, and an 8-channel hybrid weight section (Fig. 7). The hardware implementation utilizes mostly high speed Schottky TTL MSI logic devices for optimum cost, speed, and functional complexity trade-off.

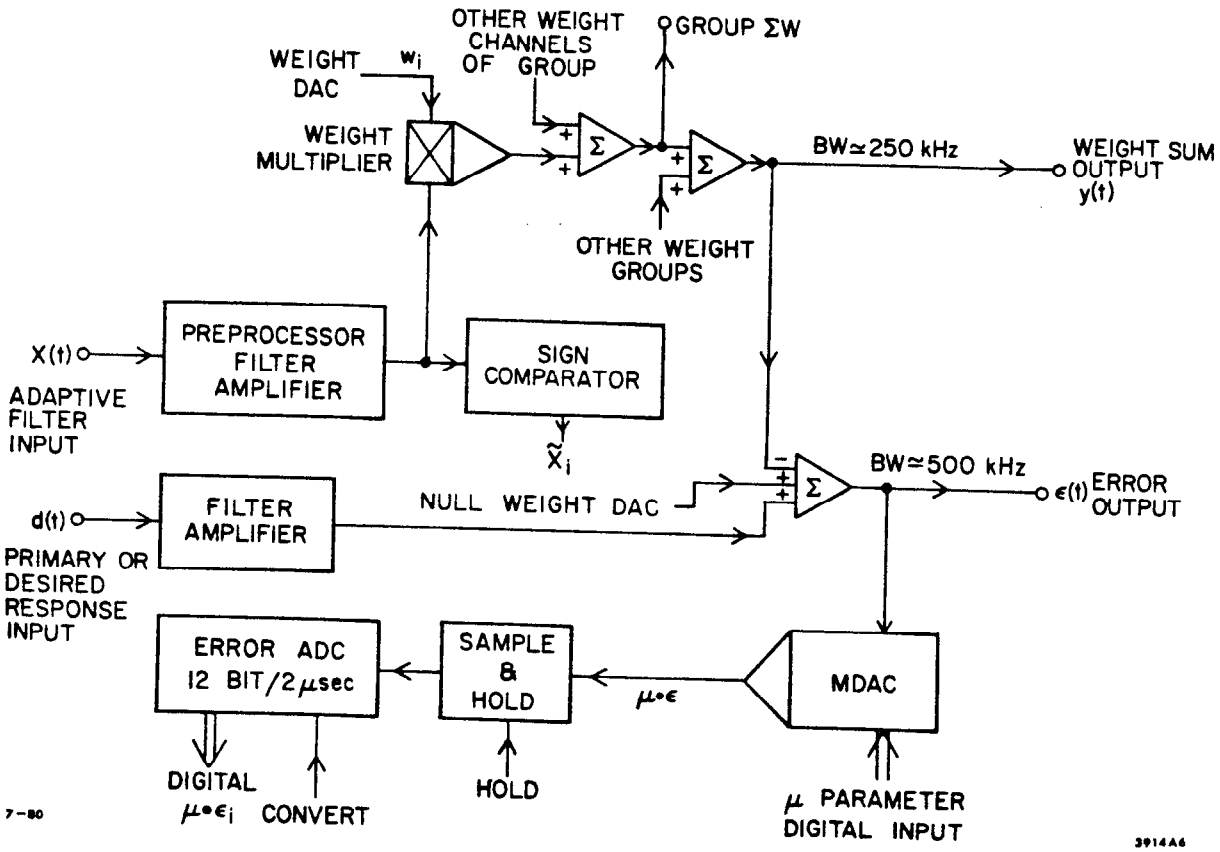


Fig. 6. ASP Basic Analog Signal Path.

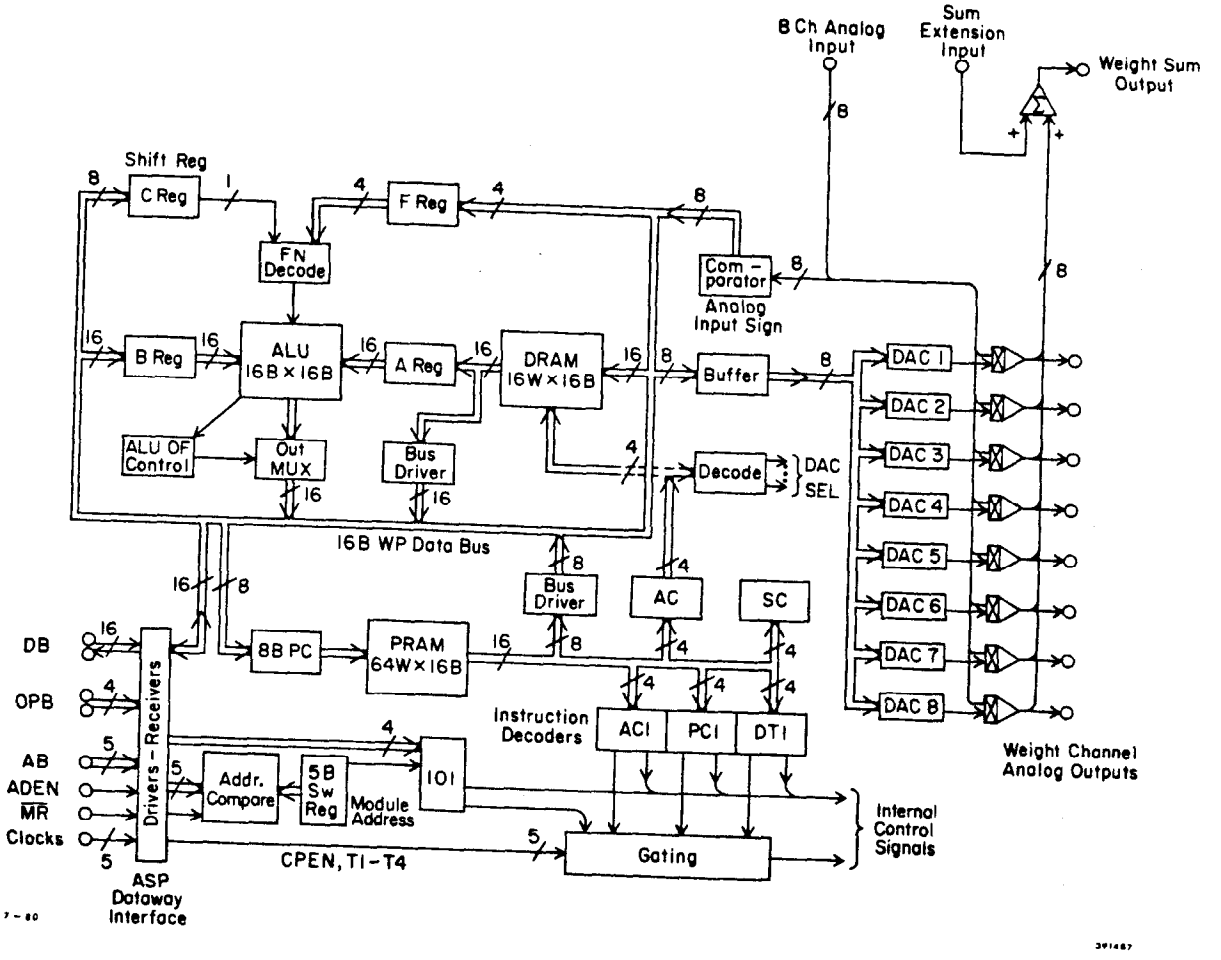


Fig. 7. Weight Processor Block Diagram.

(1) WP Microprogram Control Section

A 64-word by 16-bit program memory (PRAM) stores the microprogram code. The instruction address is provided by an 8-bit program counter (PC). The most significant bit (A7) is not used; A6 enables program down loading by disabling instruction decoding; $A_5 - A_0$ select the current instruction memory location. Each 16-bit microcode word represents a composite instruction. The normal format contains 3 simple instruction fields and a data field, each 4-bits wide.

(MSB) PD0 PD3	PD4 PD7	PD8 PD11	PD12 PD15 (LSB)
4-Bit Data	ACI Arithmetic Control Instruction	PCI Program Control Instruction	DTI Data Transfer Instruction

A special program jump instruction format has an 8-bit data field, which contains the jump address to be loaded into the program counter, and 2 simple instruction fields

(MSB) PD0 PD7	PD8 PD11	PD12 PD15 (LSB)
8-Bit Reference Address	PCI	DTI

Instruction decoding is accomplished with 3 PROM decoders. The 4-bit data field of each normal instruction word can be loaded into a device address counter (AC) or a step counter (SC). The AC selects data reference locations in the data memory (DRAM).

The SC allows for multiple execution of instruction loops or a general step or word count. SC terminal count (overflow) generates a program address advance (IPC) and inhibits a program jump (LPC). Where required for proper hardware timing, the instruction decoder outputs are gated with clock phase signals. Each system clock period is divided into 4 time slots.

An interface to the ASP system dataway is provided. The 16-bit data bus is used for microprogram down loading into the PRAM, loading of the PC, and read and write operations to the DRAM. Interface control uses a 5-bit address bus comparator and a 4-bit OP code decoder. The OP code specifies input-output instructions (IOI).

(2) 16-Bit High Speed ALU Section

The 16-bit Schottky ALU with external carry look-ahead generation executes arithmetic and logical operations on two 16-bit operands typically in 30 nsec. The function is specified by the 4-bit F register and the output from the C shift register. The C shift register is 8 bits wide and holds a sign word during execution of the clipped LMS algorithm. The F and C registers select a subset of all possible ALU functions through a PROM look-up table (ALU function control PROM). ALU operands are stored in registers A and B. Register A is loaded from the DRAM scratch pad memory. Register B is loaded from the data bus. The ALU output is gated onto the data bus through a multiplexer. The ALU overflow or underflow is detected and the output multiplexer is used to substitute the minimum or maximum ALU output value, and the ± OF flag is generated to the ASP system controller. The 16-word by 16-bit DRAM scratch pad stores all current data values.

(3) 8 Channel Hybrid Weight Section

This section contains 8 weight channels and a summing amplifier circuit. Each channel has a voltage discriminator circuit to measure the sign of the analog input. The hybrid weight is implemented with an 8-bit DAC and a 4 quadrant analog multiplier. The current weight value is loaded from the data bus into the internal DAC holding register. All 8 weight outputs are summed with a high speed summing amplifier circuit. The 8 weight outputs and the sum signal are available as WP module outputs.

B. Error Digitizer Module

The error digitizer (ED) module combines an analog signal processing section with a high speed ADC and a digital interface section to the ASP system bus (Fig. 8). Summing, difference, and filter amplifier circuits are provided to develop the analog error signal in the adaptive process. The error signal is sampled and measured and transferred to the WP modules for the next adapt cycle.

(1) Analog Signal Processing

A 9-input amplifier produces a final sum of the partial sum outputs from all weight processor modules. Two reference input filter amplifiers are available with adjustable output attenuation for desired signal inputs. The reference inputs are switchable to ground under program control to allow system offset measurements and corrections. Sum, reference signals, and the output from a null weight DAC are subtracted with the error difference amplifier.

The null weight utilizes a 12-bit resolution DAC with output attenuation adjustment. It may be used under program control to compensate DC offsets in the analog signal path.

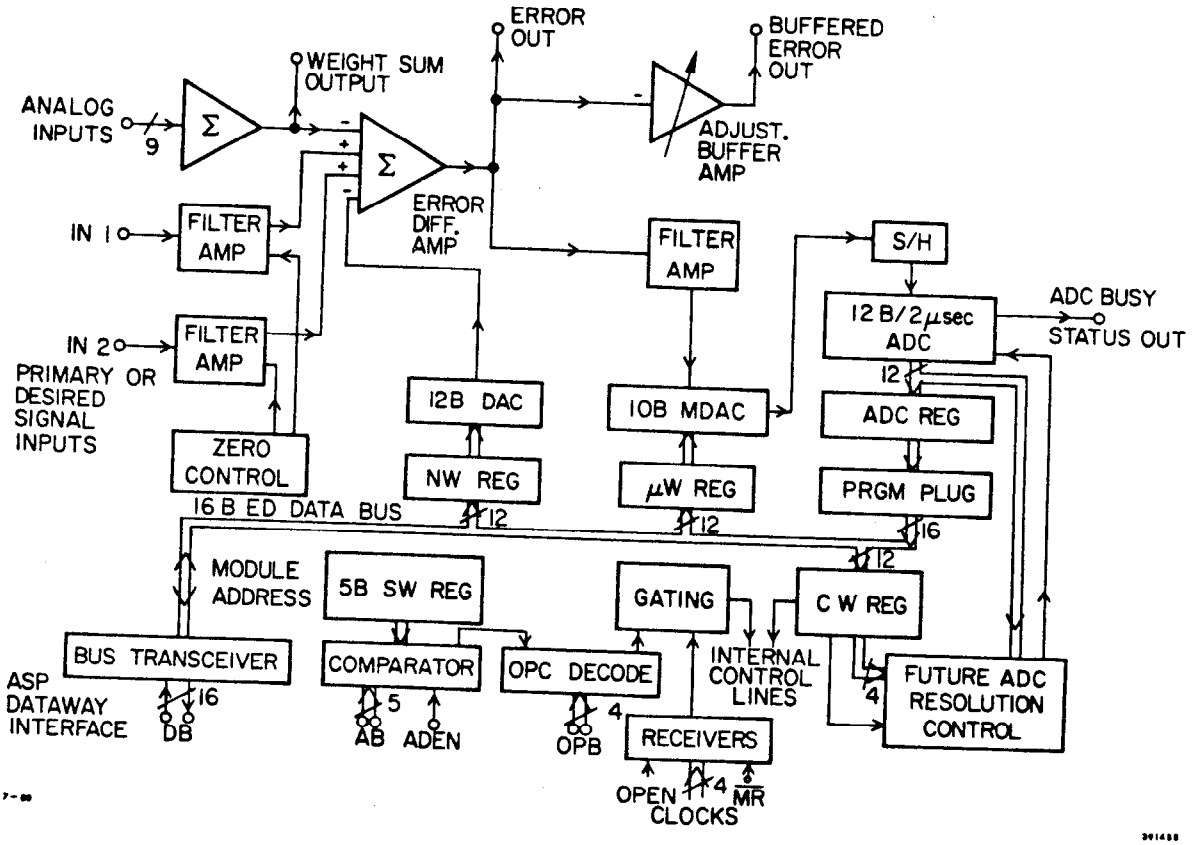


Fig. 8. Error Digitizer Block Diagram.

The error signal from the difference amplifier is further processed by a low pass filter amplifier. A buffered output of the error signal with gain and offset adjustments is also available for external use. The filtered error is then scaled with the μ weight. The μ weight has a 10-bit resolution DAC performing a 2-quadrant multiplication of the filtered error with the programmable adapt constant μ. This allows control of the rate of convergence of the adaptive process. The output signal, μ•error, is then fed to the sample-and-hold and ADC.

(2) Analog-To-Digital Converter

The output from the analog section, μ•error, is sampled and digitized with a high speed, bipolar ADC with 12-bit resolution. The ADC is of the successive approximation type. An IO instruction, Start Convert ADC, samples the analog signal and initiates a conversion cycle. For the duration of each conversion cycle, the status output ADC Busy is returned to the ASP system. The ADC output is automatically transferred to a holding register.

(3) Digital System Interface

The error digitizer module has interface hardware for the ASP system dataway. Connections for the data, address, IO OP code, and clock phase buses are included. This allows the system controller to load all writable registers of the module, control the ADC operation, and read the ADC output holding register. A 12-bit control word register contains control bits for different modes of operation of the module.

C. Test Controller Module

The test controller (TC) module handles ASP system control and provides a manual test facility. The block diagram (Fig. 9) shows the major functions in this module: a microprogram loader, an adapt sequencer, a manual switch input register, a LED output register, and a clock generator. The module can occupy any position on the ASP system bus. Its primary purpose is to provide test and diagnostic capability; however it is also used to handle system control for the basic ASP prototype system.

(1) Microprogram Loader

The microprograms used in the WP modules, and initializing data for WP and ED modules are stored in the loader memory.

In addition the memory holds OP codes, bus address codes, and values for bus control lines, required when using the ASP bus for program downloading. The loader memory is a 1K word by 29-bit EPROM. It is divided into 4 pages of 256 words each, selected by 2 manual switches. Program and data contents of a page are transferred to the ASP modules via the system bus under control of an 8-bit address counter. Page downloading is accomplished by using 2 manual switches to enable the loader mode and to start a loader sequence. Each loader page can store a different set of microprograms.

(2) Adapt Sequencer

To handle control of the ASP system bus during the synchronous execution of WP microprograms and the convert cycle of the ED module, a firmware programmed sequencer circuit is used. A 4-bit address counter, incremented by the ASP system clock, fetches sequence instructions from a high speed PROM memory, consisting of 2 pages of 16 words by 16 bits.

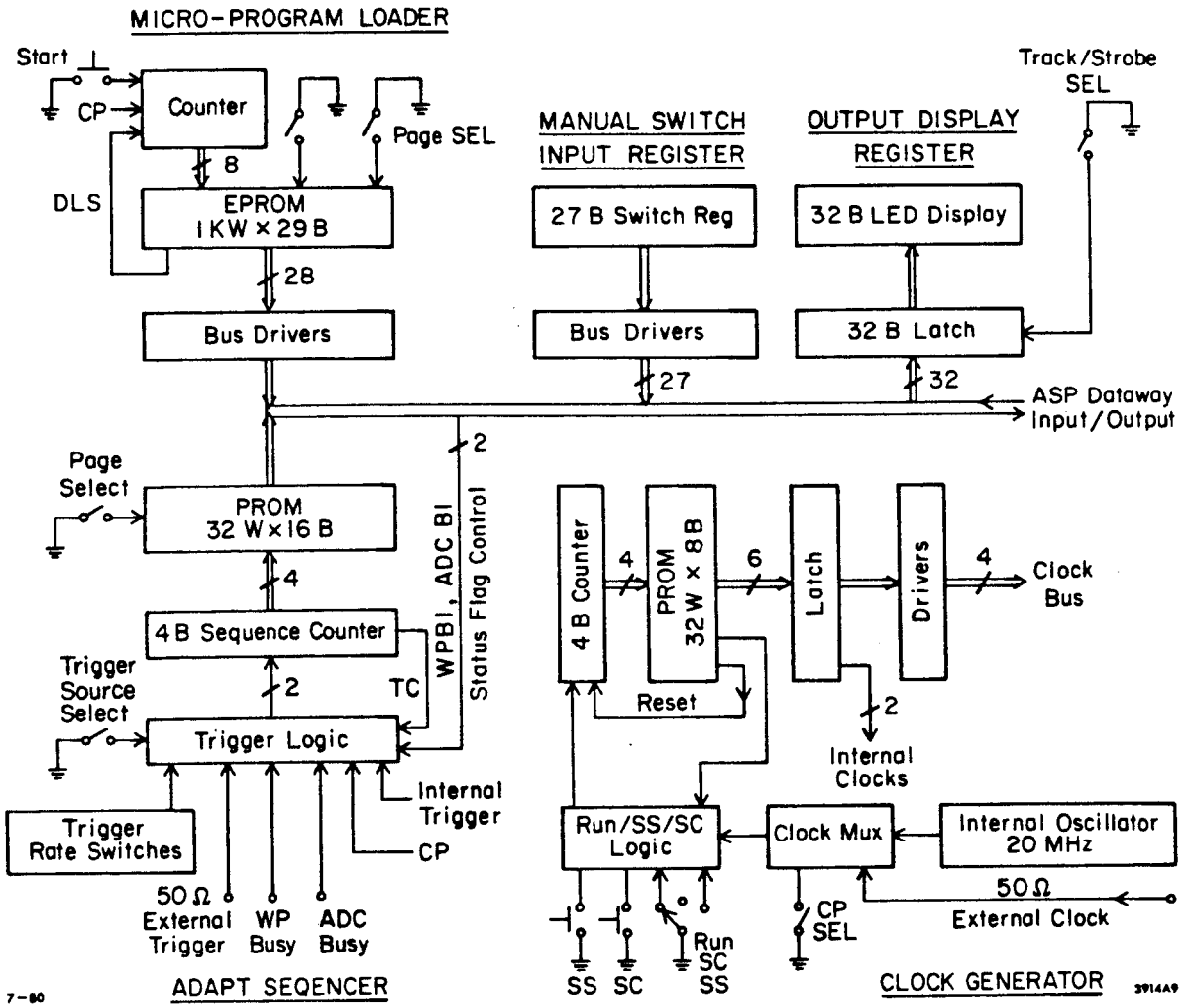


Fig. 9. Test Controller Block Diagram.

A manual page select switch allows use of 2 sequencer programs. Sequence instructions contain control bits to stop execution in the WP modules, and to enable BUSY status flags from ASP modules, to force the sequencer execution into a wait state. These program controls are used for system synchronization during execution of adapt program cycles.

Each 16-word instruction sequence is started by a trigger input. A manual switch selects from an external or an internal trigger source. Programming switches in the module make 9 different trigger repetition rates available.

(3) Manual Switch Input Register

A manual switch register, 27 bits wide, can be enabled onto the ASP system bus and be used to manually execute individual bus instructions. The data portion of the register may also be used in conjunction with the adapt sequencer. This allows input of a variable 16-bit data word under program control during execution of an adapt sequence.

(4) Output Display Register

An output register driving a 32-bit LED display is provided to read out the status of all ASP system bus lines and adapt sequencer instruction bits. Two display modes are available with a select switch. In track mode, contents of all signal lines is continuously displayed. In strobe mode, the display is updated once every instruction cycle. Timing of the strobe mode is programmable with 4 switches in the module. Any combination of clock phase signals T1 to T4 may be used to update the LED register.

(5) Clock Generator

The test controller module generates all clock and timing signals used in the ASP system. A manual switch selects an external input or an

internal crystal oscillator as the system clock source. The system clock drives a firmware programmed timing generator, which produces the 4 clock phase signals (T1, T2, T3, T4) used in the ASP system, and clocks used to increment the adapt sequencer and the program loader inside the TC module. Logic controls clock generation in 3 modes: RUN, SINGLE CYCLE, SINGLE STEP. RUN mode is used for continuous program execution. SINGLE CYCLE mode produces with a push button a single clock phase cycle consisting of T1 to T4. Each cycle executes a composite instruction in the WP modules on the ASP system bus, and increments the adapt sequencer or the program loader. SINGLE STEP mode advances the timing generator by single master clock pulses allowing a step-by-step generation of the clock phase signals Tx and execution of simple instructions.

D. Filter Array Module

The filter array module contains 16 programmable filter channels and 2 wide band buffer amplifiers (Fig. 10). Input and output connections to all circuits are made through 2 patch plugs. Selection of inputs and outputs and filter array configurations is thereby easily tailored to requirements of ASP experiments. Typical array configurations are parallel and series connections of all filter channels.

Each filter channel consists of an operational amplifier and a filter network, which form a basic active filter. A typical filter channel is shown in Fig. 11.

All discrete components of the filter network are contained on one plug-in header. For a given adaptive experiment a set of headers is prepared, with the required network components to implement the filter

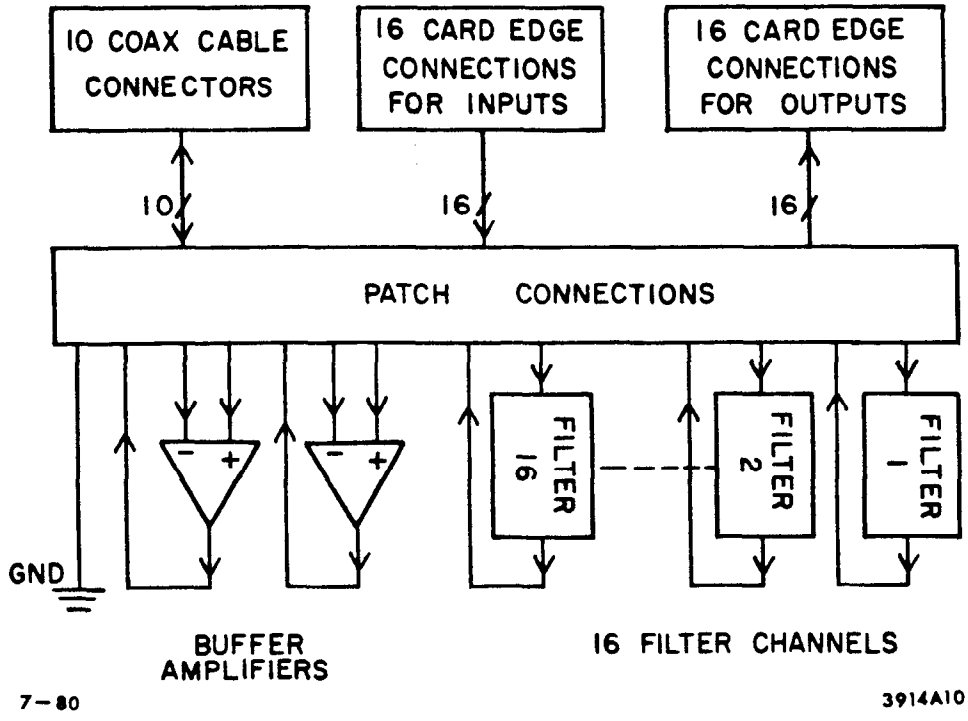


Fig. 10. Filter Array Block Diagram.

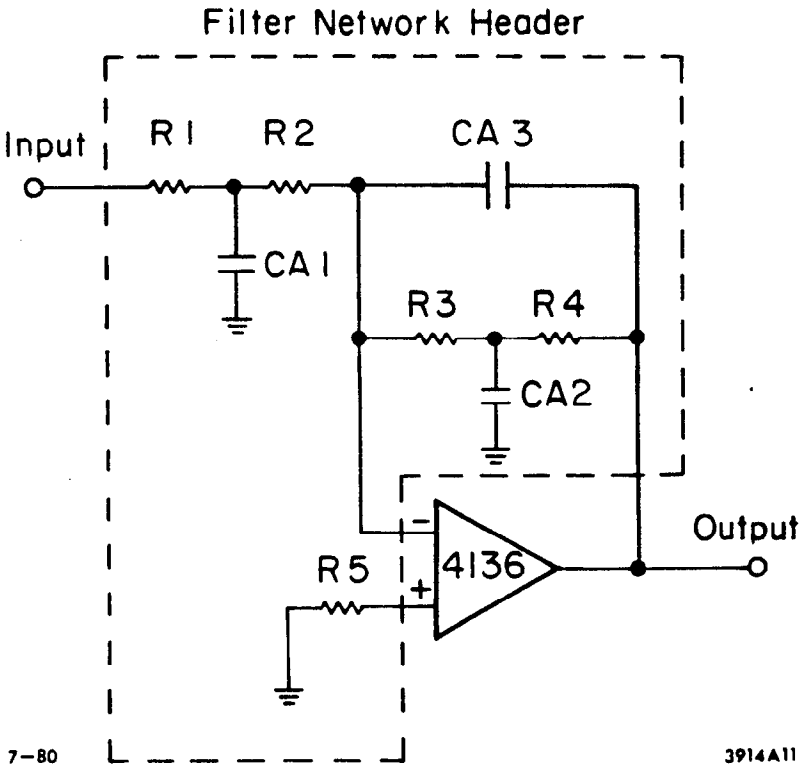


Fig. 11. Filter Array Typical Filter Channel.

characteristic for each channel, and plugged into the filter array module.

The module has 2 general purpose buffer amplifiers. Both circuits have differential input connections and are unity gain amplifiers.

CHAPTER IV

PROTOTYPE SYSTEM AND TEST EXPERIMENTS

Prototype hardware has been developed and utilized for several basic adaptive signal processing experiments. The hardware, shown in Fig. 12, consists of a chassis, plug-in printed circuit card modules, and a set of power supplies.

The chassis is equipped and wired for an ASP system with 64 weight channels. Printed circuit card file backplanes, with distributed voltage regulation, handle all power supply requirements. DC power to the chassis is derived from a set of bulk power supplies. Three weight processor modules, two error digitizer modules, one test controller module, and two filter array modules are available for experiments with up to 24 weight channels. Prototypes of these modules are shown in Fig. 13.

The system executes dataway operations and weight processor composite instructions in 250 nsec. The adapt cycle time for the clipped LMS algorithm is 8 μsec. This results in a bandwidth of approximately 60 KHz for adaptive processing. However, maximum analog signal path bandwidth through preprocessor and weight channels to adaptive filter sum output is typically 250 KHz, and from desired input through error difference amplifier to error output typically 500 KHz. Hardware specifications and performance data are summarized in Appendix A. Next we describe several bench experiments performed with the ASP prototype system. The purpose of these experiments was to test and verify proper operation of the signal processor, to compare performance of a real-time adaptive processor with results from computer simulations reported in the reference literature,

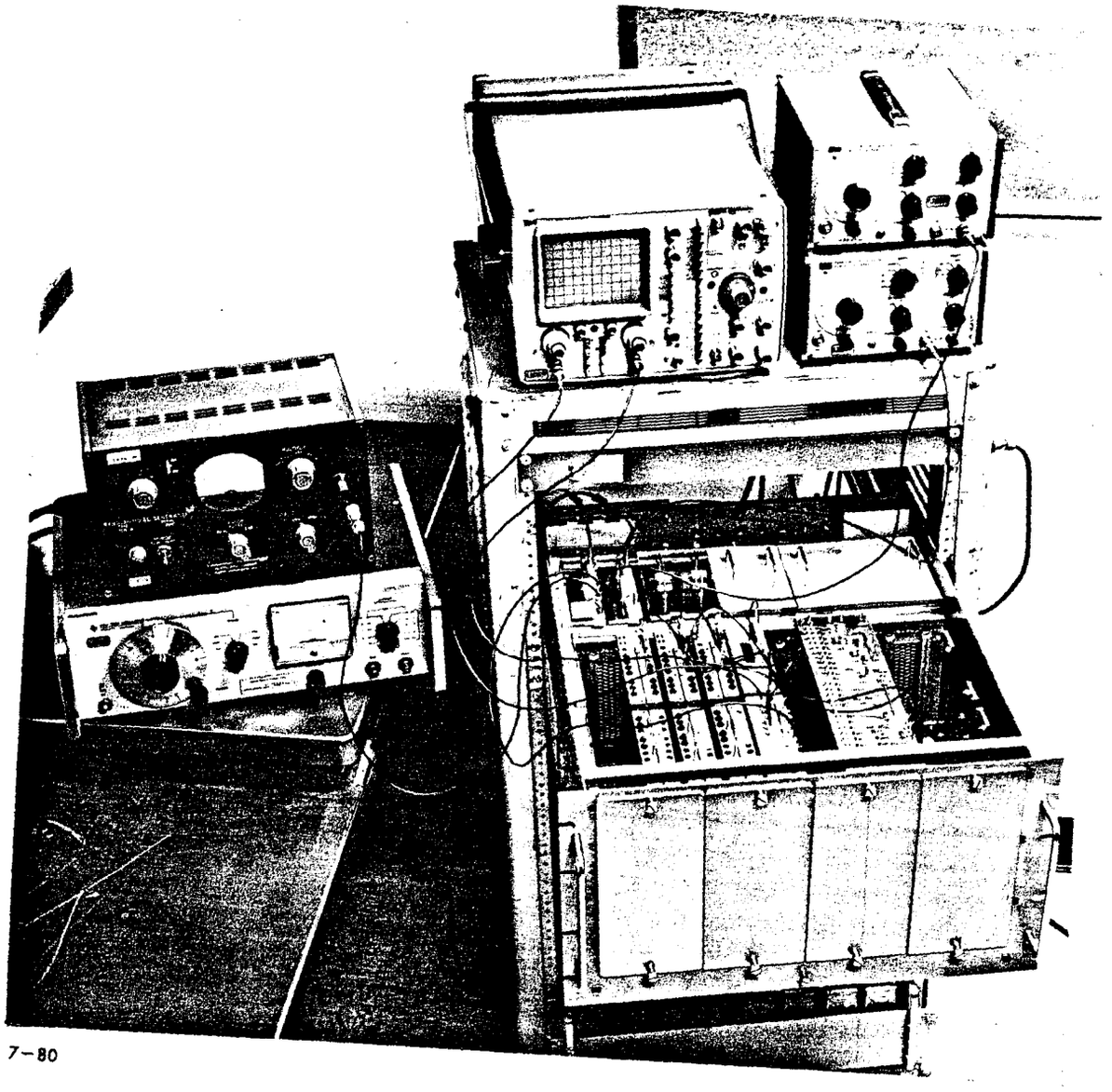
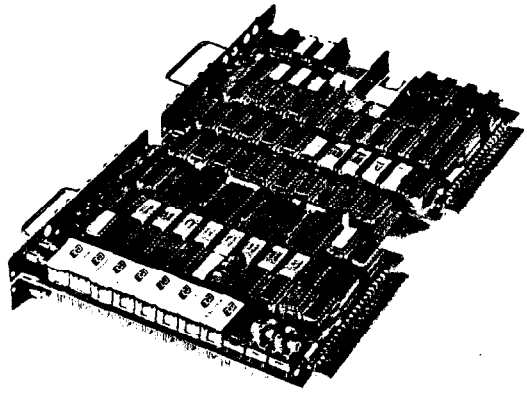
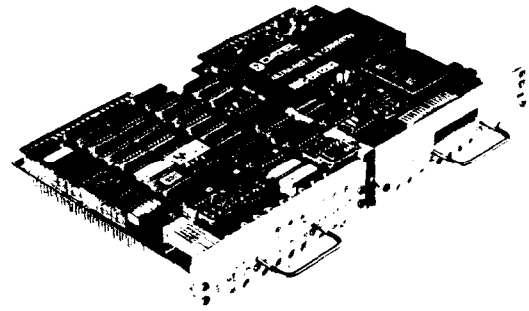


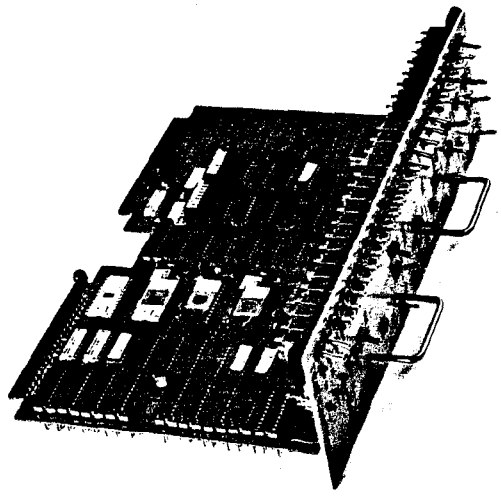
Fig. 12. ASP Prototype Chassis with Modules.



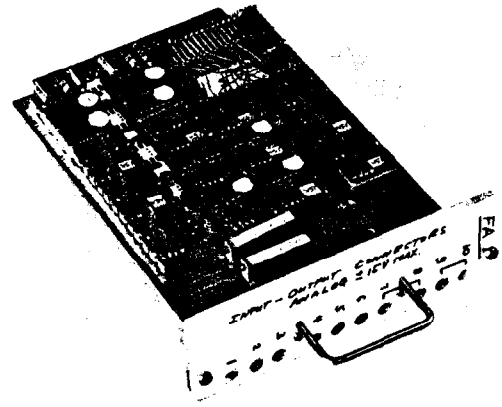
Weight Processor



Error Digitizer



Test Controller



Filter Array

3914A13

7-80

Fig. 13. ASP Prototype Modules.

and to gain experience with different system configurations and application possibilities.

A. Adaptive Response to Step Input

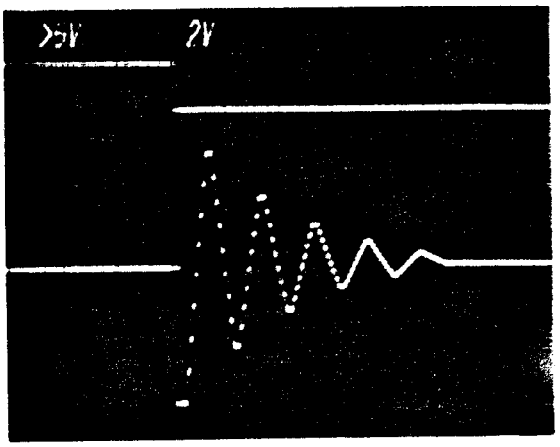
This basic test demonstrates the response of the adaptive signal processor to a step input signal. The inputs to 8 weight channels are held at a fixed DC voltage level and a ±5V step is applied to the desired input. Figure 14 shows the ASP error output for different μ values. In Figs. 14a and 14b the adapt period has been slowed to approximately 500 μsec to show the adjustment of each individual weight channel. The smaller μ value in B results in faster convergence to a minimum output error. Figure 14c shows the response at normal adapt speed with 8 μsec period.

B. Waveform Synthesis

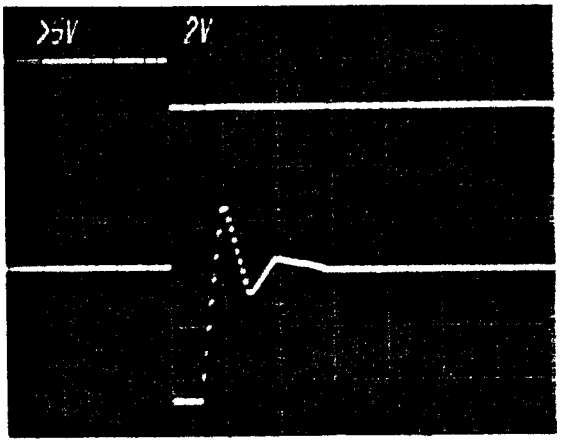
Inputs to 8 weight channels are connected to a fixed DC voltage level. A reference waveform is applied to the desired input. By continually adjusting all weight channels, the ASP synthesizes an optimum replica of the desired waveform at the weight sum output. Figure 15 shows results for ramp and sinusoidal waveforms at 1 KHz and 4 KHz. In addition, spectra for the tests with sine waveforms are included. These show harmonic distortions introduced by the signal processor.

C. Adaptive Filtering of Square Wave to Desired Response

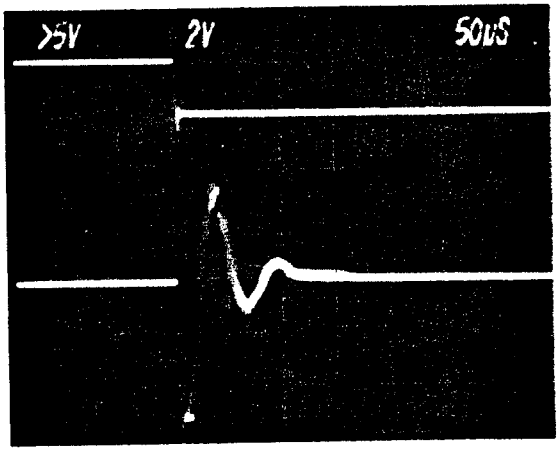
This experiment utilizes the 16-channel filter array module with parallel low pass filters covering a range of cutoff frequencies from 2 to 20 KHz. The sum of the 16 weight channels is taken as the system



(a) Adapt Period = 500μsec
μ = 0.327
Horizontal Scale: 1 msec/cm



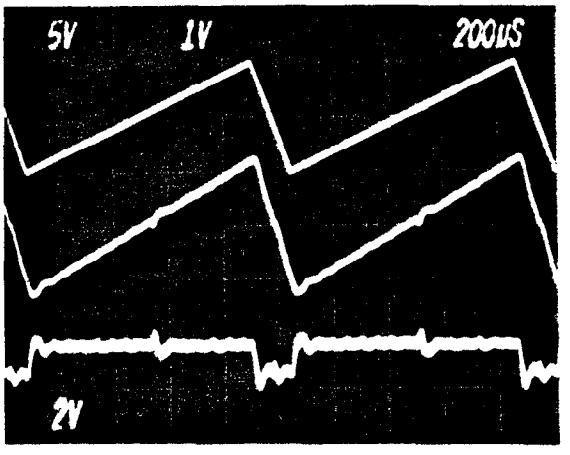
(b) Adapt Period = 500 μsec
μ = 0.265
Horizontal Scale: 1 msec/cm



(c) Adapt Period = 8 μsec
μ = 0.175
Horizontal Scale 50 μsec/cm

7-80
3914A14

Fig. 14. Adaptive Response to Step Input.

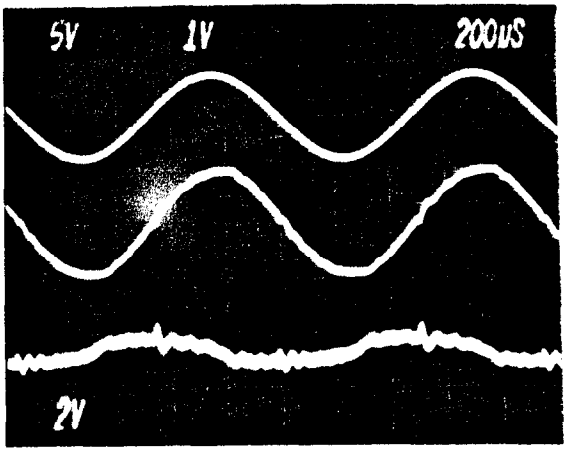


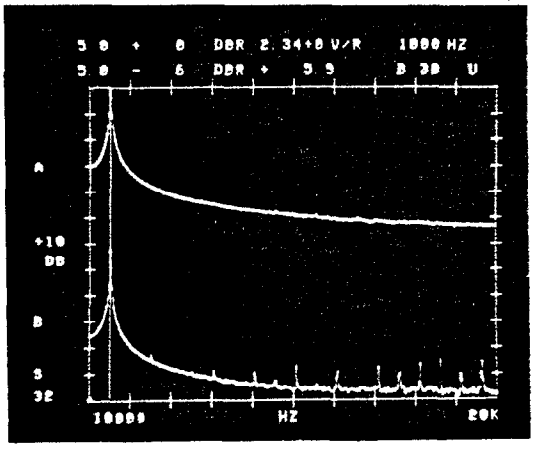
(a) Ramp Waveform 1 kHz

Top: Desired Input

Center: Weight Sum Output

Bottom: Error Output
$\mu = 0.374$





(b) Sinusoidal Waveform and Spectra at 1 kHz

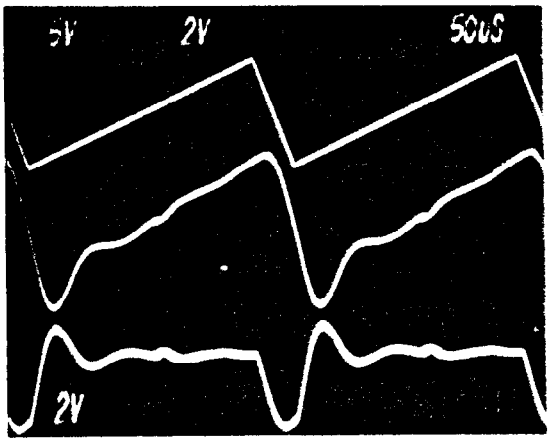
Top: Desired Input
Center: Weight Sum Output
Bottom: Error Output

Top: Desired Input
Bottom: Weight Sum Output
$\mu = 0.374$

7-80

3914A15

Fig. 15. Waveform Synthesis.

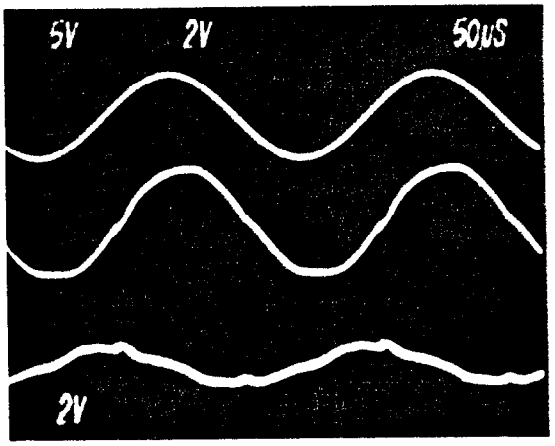


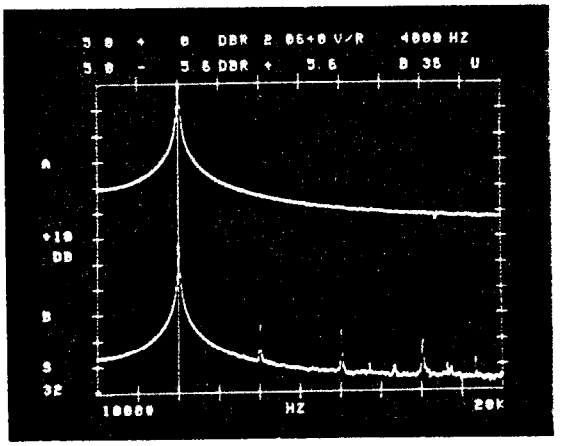
(c) Ramp Waveform 4 kHz

Top: Desired Input

Center: Weight Sum Output

Bottom: Error Output
$\mu = 0.374$





(d) Sinusoidal Waveform and Spectra at 4 kHz

Top: Desired Input
Center: Weight Sum Output
Bottom: Error Output

Top: Desired Input
Bottom: Weight Sum Output
$\mu = 0.374$

7-80

3914A21

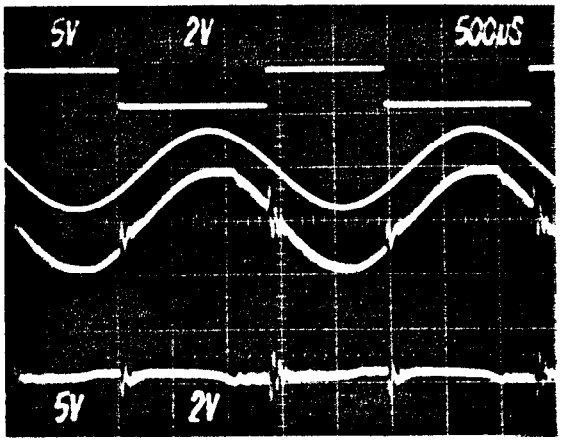
Fig. 15. Waveform Synthesis.

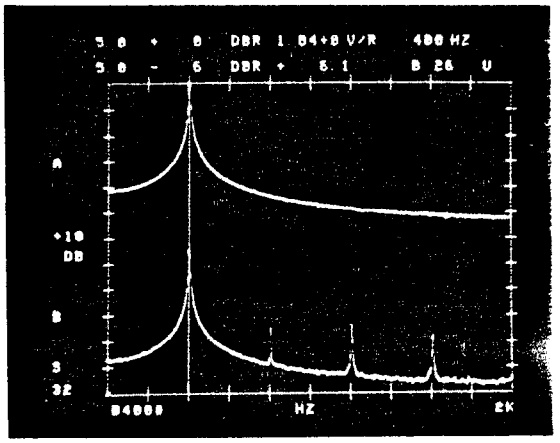
output. A square or pulsed waveform is applied to the filter input and adpatively filtered to match a sinusoidal or triangular waveform at the desired signal input. Both input signals are synchronized but not in phase with each other. Figure 16 presents time waveforms and frequency spectra for tests at 400 Hz and 4 KHz. These results may be compared to results from a similar experiment described in Ref. 1.

D. Narrow Band Adaptive Experiment

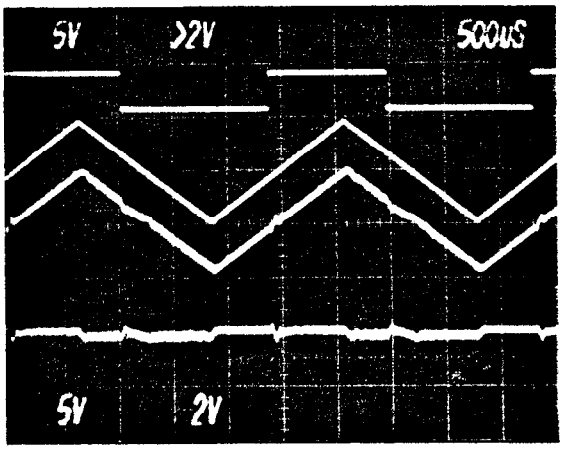
Separation of a broad-band signal from a sinusoidal signal is achieved with a bandpass or band-reject (notch) filter. The ASP configuration shown in Fig. 17a realizes a dual bandpass adaptive filter. Two pure sinusoids are supplied to the preprocessor as band center reference frequencies. The preprocessor consists of two pairs of buffer amplifiers and 90^{o} phase shifters. The primary system input is the filter input, bandpass and band-reject outputs are taken from the weight sum and error outputs respectively. Two weight channels are required to adjust filter gain and phase for each sinusoid to be filtered out. One extra weight channel is used as a bias weight, to eliminate constant offset and slowly varying drift from the primary input.

Adjusting the μ parameter controls bandwidth (or notch width) of the filter. Small values of μ produce small values of bandwidth, and bandwidth is not proportional to band center frequency. To measure the frequency characteristic of the filter, we used 2.5 KHz and 4.0 KHz reference frequencies and applied a 20 KHz random noise signal to the primary input. The spectra of the random noise signal and the filter pass characteristic are shown in Fig. 17b. Figure 17c compares filter properties for 2 different values of μ. The interpolated bandpass width





(a) Sinusoidal Waveform and Spectra at 400 Hz
$\mu = 0.281$



(b) Triangle Waveform at 400 Hz
$\mu = 0.281$

For Fig. 16 a,b,c,d
Oscilloscope Traces from Top:

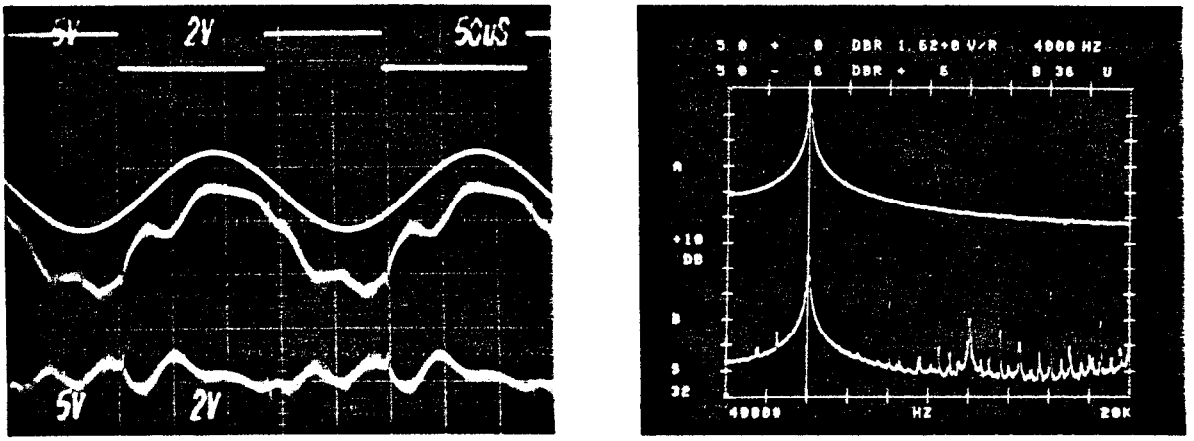
Filter Input at Preprocessor
Desired Input
Weight Sum Output
Error Output

Spectra from Top:

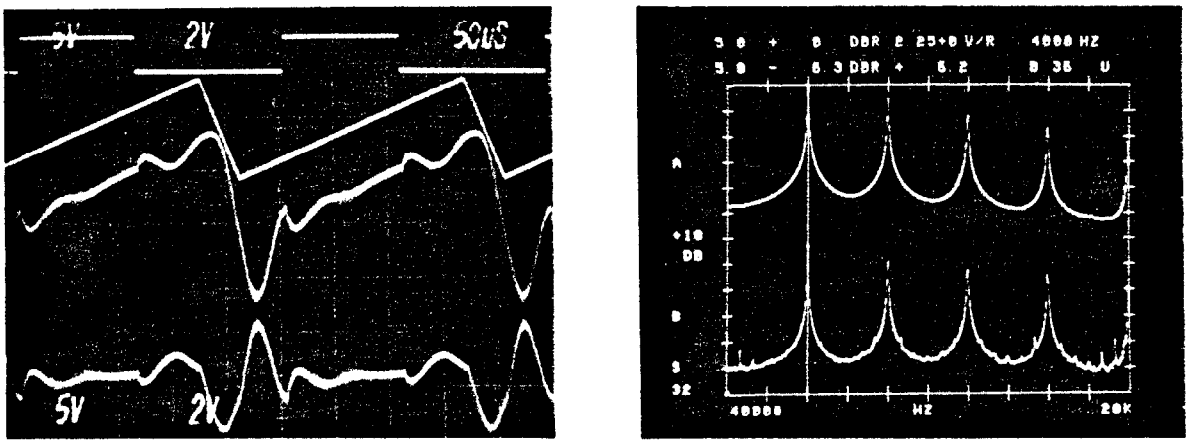
A. Desired Input Spectrum
B. Weight Sum Output Spectrum

7-80
3914A16

Fig. 16. Adaptive Filtering of Square Wave to Desired Response.



(c) Sinusoidal Waveform and Spectra at 4 kHz
$\mu = 0.320$

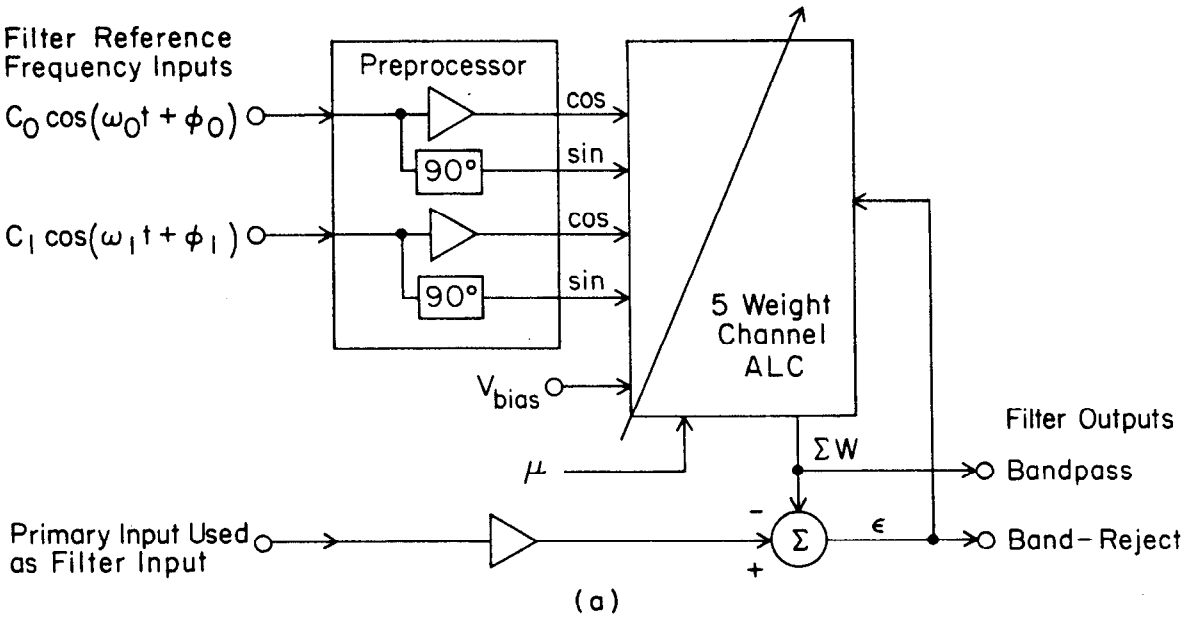


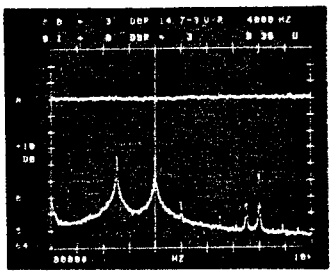
(d) Ramp Waveform and Spectra at 4 kHz
$\mu = 0.281$

7-80

3914A22

Fig. 16. Adaptive Filtering of Square Wave to Desired Response.





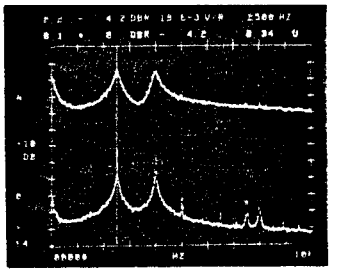
(b)

Spectrum A:
20 kHz BW Random
Noise Input

Spectrum B:
Filter Passbands at
2.5 kHz and 4.0 kHz
for $\mu = 0.015$

Interpolated (−3 dB)
Bandwidths from Spectra:

	at 2.5 kHz	at 4.0 kHz
$\mu = 0.140$	138 Hz	150 Hz
$\mu = 0.015$	11.2 Hz	9.6 Hz



(c)

Filter Passbands at
2.5 and 4.0 kHz

Spectrum A: $\mu = 0.140$
Spectrum B: $\mu = 0.015$

7-80
3914817

Fig. 17. Dual Bandpass Adaptive Filter.

(-3 db attenuation) ranges from 9.6 Hz (0.24% fc) to 138 Hz (5.5% fc) for both values of μ and band center frequency fc.

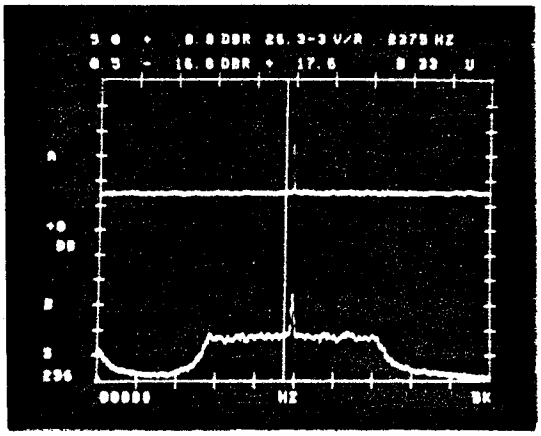
Next we performed a narrow band filtering experiment with ASP as a single bandpass filter.

We used a 2.5 KHz reference frequency and applied a sinusoid (1 Vpp) buried in broadband random noise (10 Vpp and 20 KHz BW) to the primary input. Figure 18a shows spectra of mixed input signal and detected sinusoidal output. Figures 18b and 18c are time waveforms of detected output at 2.5 KHz and stop band output at 3 KHz. From the output spectrum we see that all background frequencies are typically attenuated by -17 db below the 2.5 KHz output peak.

E. Wide Band Adaptive Experiment

The final experiment demonstrates filtering of a wide band triangular signal. ASP as a braodband filter is shown in Fig. 19. All prototype hardware is used to form a 24-weight channel system. The preprocessor consists of 2 filter array modules with parallel connected low pass filters, covering a range of cutoff frequencies from 50 Hz to 20 KHz. A reference triangular signal is supplied to the preprocessor input. Based on this reference signal the 24 channel adaptive filter forms pass bands to match the first 12 spectral lines of the reference signal. A mixed signal, consisting of a 5 Vpp triangle buried into 10 Vpp, 20 KHz BW random noise, is fed to the primary input. The bandpass output is again taken from the weight sum output of ASP.

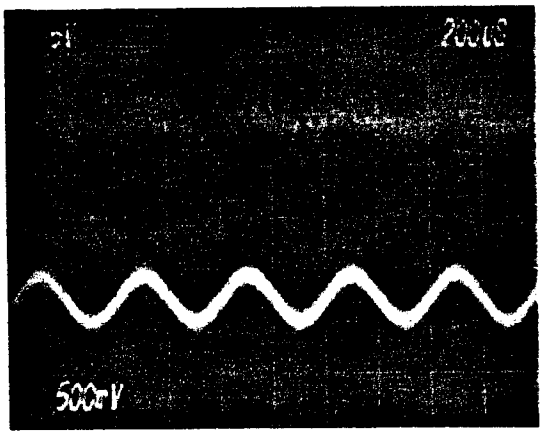
With a 750 Hz triangular signal as a frequency reference, the filter characteristic in Fig. 20c is formed. The base frequency passband width is approximately 6.8 Hz with a μ of 0.015. Figure 20b shows the spectrum



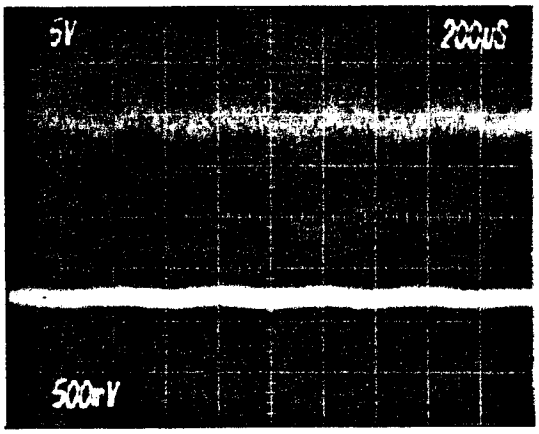
(a) Filter Input - Output Spectra

Top: Filter Input Spectrum of 20 kHz BW Random Noise Mixed with 2.5 kHz Sinusoid ($S/N \simeq 0.1$)

Bottom: Filter Output with 2.5 kHz Sinusoid $\mu = 0.062$



(b) Filter Passband Output at 2.5 kHz
Top: Mixed Input Signal
Bottom: Detected 2.5 kHz Output Signal



(c) Filter Stopband Output
Top: Mixed Input Signal
Bottom: Filter Output at 3.0 kHz

7-80

3914A18

Fig. 18. Narrow Band Filtering Experiment.

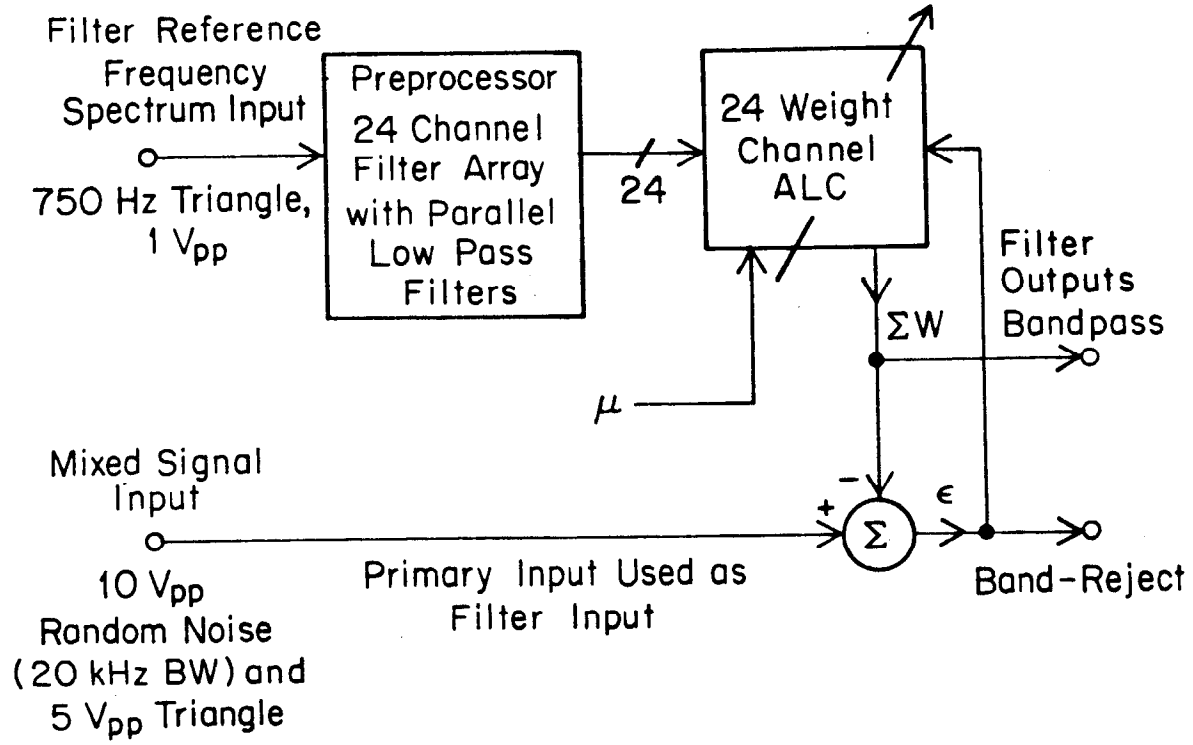
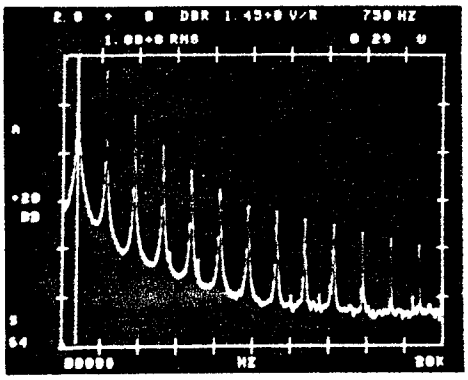
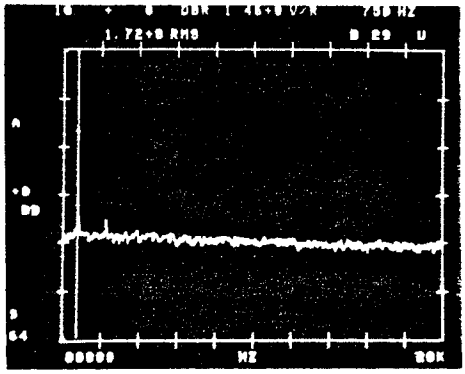


Fig. 19. Wide Band Adaptive Filter with 24 Channels.

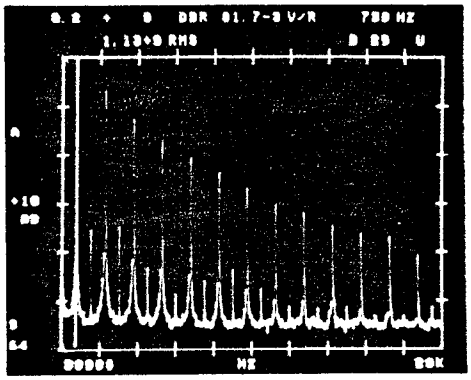
of the mixed input signal. Only base frequency and the 2250 Hz spectral line are distinguishable. The spectrum of the buried triangle signal is fully shown in Fig. 20a. Its base frequency width is 10.7 Hz. The spectrum of the detected triangle signal at the weight sum output is shown in Fig. 20d. The width of the 750 Hz peak is 7.2 Hz. Figures 20e and 20f are time waveforms of the filter output at 750 Hz and all other frequencies. A close inspection of the frequency spectra reveals that filter and detected signal spectra are sharper than the spectrum of the input triangle (base frequency band widths of 6.8 Hz and 7.2 Hz compared to 10.7 Hz). This is the result of the adaptive processor maintaining the pass band centers exactly at the spectral line frequencies of the reference signal and a bandwidth determined by μ.



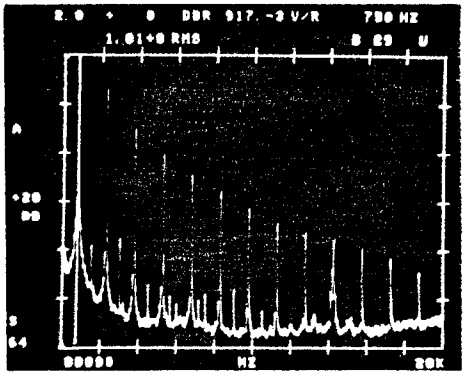
(a) Input Triangle Signal at 750 Hz



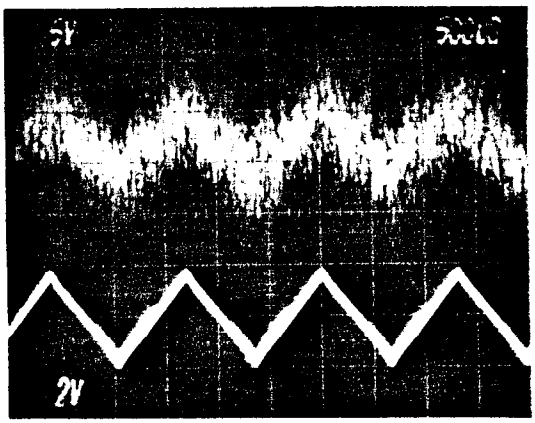
(b) Spectrum of Mixed Input Signal with 750 Hz Triangle and 20 kHz BW Random Noise ($S/N \simeq 0.5$)



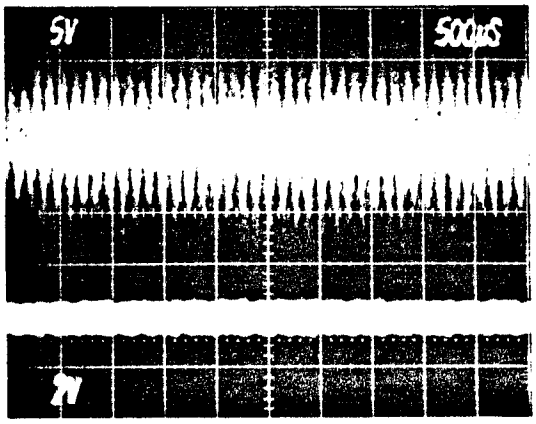
(c) Broadband Filter with Adaptively Formed Passbands Reference Spectrum: 750 Hz Triangle $\mu = 0.015$



(d) Detected Output Signal Spectrum: 750 Hz Triangular



(e) Detected Output at 750 Hz



(f) Filter Stopband Output

7-80
3914A20

Fig. 20. Wide Band Adaptive Filter Experiment.

CHAPTER V

CONCLUSION, FUTURE PLANS, AND APPLICATIONS

We have described the design of an experimental adaptive signal processor system and presented results from tests with a prototype. Programmed for the clipped LMS adaptive algorithm, the processor adjusts all weights in typically 8 μsec. The basic processor design accommodates up to 64 weight channels. The test experiments have demonstrated the ability to realize high performance, dynamic, time domain filters with programmable parameters.

Future refinements of the processor software and some tune-up of the hardware are expected to achieve a minimum adapt cycle period of 4 μsec. This will correspond to a weight update time of 62.5 nsec and an adaptive processing bandwidth of approximately 125 KHz. The design and modular implementation of ASP provides flexibility to experiment with many different signal processor configurations, and to test the applicability of adaptive processing to many instrumentation problems. Possible ASP configurations include the following (Fig. 21):

(a) Processor with up to 64 weight channels and a single signal input vector X with 64 input components; configurations with many weight channels are required for wideband signal processing.

(b) Multi-input processor for sensor arrays; this allows spatial and time domain filtering of signals from an antenna array.[6]

(c) Master-slave processor to accommodate the use of a pilot or training signal; the weights of the slave processor are adjusted identical to weights of the master; however the sum output of the slave processor

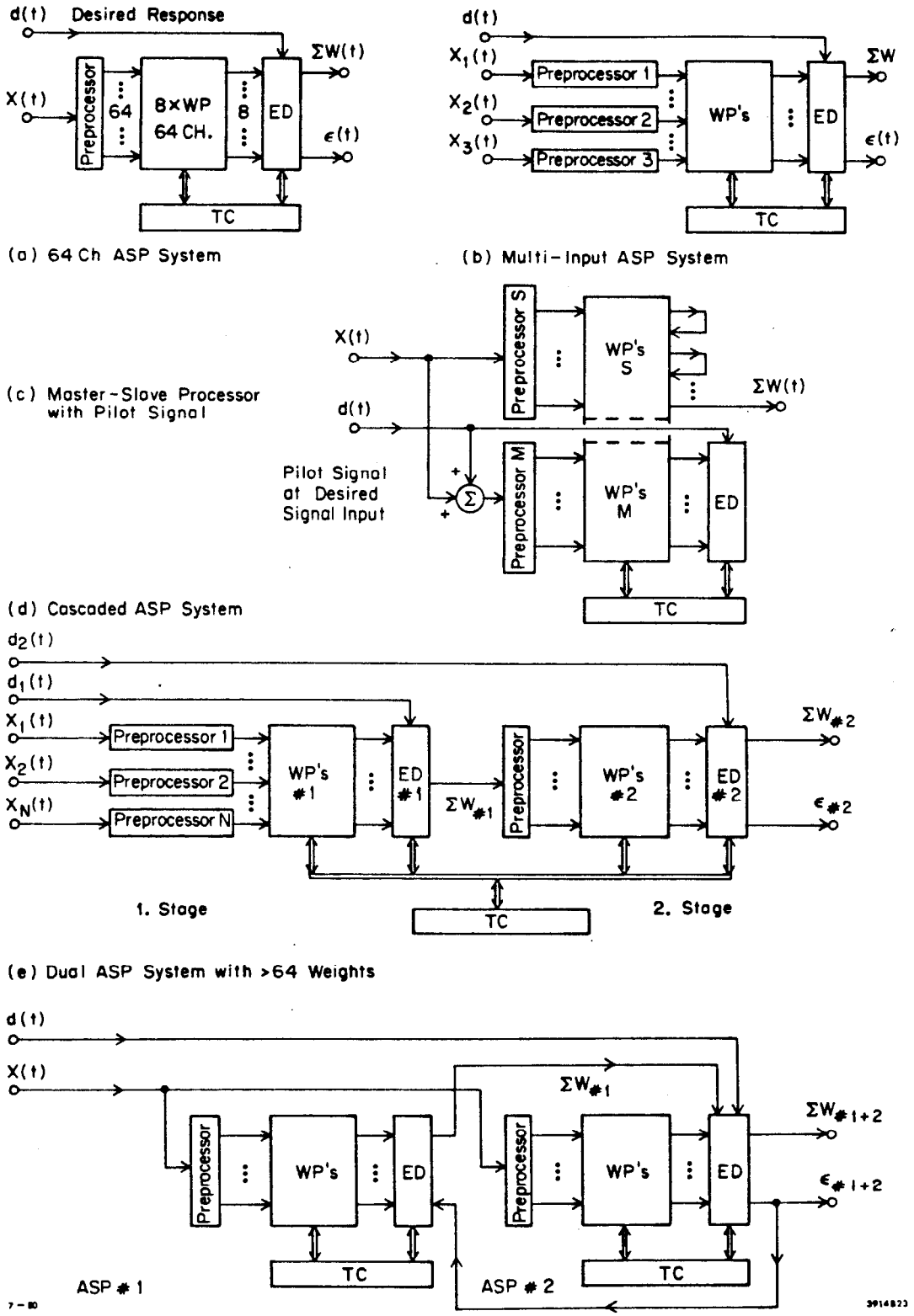


Fig. 21. Experimental ASP Configurations.

weights does not contain the injected training signal.[6]

(d) Multiple or cascaded adaptive processor system; for example a wide-band sensor array processor for spatial filtering, followed by a narrow-band, tracking notch filter for extracting a specific signal component.

(e) Processor with more than 64 weight channels; this may be implemented for example by interconnecting two ASP chassis; weight sum and error signals are generated for the combined system.

Finally, an application experiment with the prototype ASP system is being planned. SLAC is utilizing a precision toroidal charge monitor system[7] to measure beam charge per beam pulse delivered to experiments in the End Station A beam line. The charge monitor utilizes a ferrite toroid operated in a resonant mode at approximately 5 KHz. The dynamic range and sensitivity of beam charge measurements is limited by signal corruption due to the electrically noisy research area environment. With the adaptive processor we expect to demonstrate an improved signal-to-noise ratio and thereby increased measurement range and sensitivity.

REFERENCES

1. J. Kaunitz, "General Purpose Hybrid Adaptive Signal Processor," SEL-71-023 (TR No. 6793-2), Stanford Electronics Laboratories, Stanford, California, April 1971.

2. B. Widrow and M. Hoff, Jr., "Adaptive Switching Circuits," in IRE WESCON Conv. Rec., pt. 4, pp. 96-104, 1960.

3. B. Widrow, "Adaptive Filters I: Fundamentals," Rept. SEL-66-126 (TR No. 6764-6), Stanford Electronics Laboratories, Stanford, California, 1966.

4. B. Widrow, "Adaptive Noise Cancelling Principles and Applications," Proc. IEEE, Vol. 63, No. 12, pp. 1692-1719, December 1975.

5. J. L. Moschner, "Adaptive Filtering with Clipped Input Data," SEL-70-053 (TR No. 6796-1), Stanford Electronics Laboratories, Stanford, California, June 1970.

6. B. Widrow et al., "Adaptive Antenna Systems," Proc. IEEE, Vol. 55, No. 12, p. 2152, December 1967.

7. R. S. Larsen and D. Horelick, "A Precision Toroidal Charge Monitor for SLAC," SLAC-PUB-398, Stanford Linear Accelerator Center, Stanford, California, April 1968.

APPENDIX A

ASP PROTOTYPE SYSTEM SPECIFICATIONS

(1) System Chassis Specifications

Chassis Capacity:

9 double size modules (WPs and EDs)

1 system controller module

4 regular modules (FAs)

4 regular module locations for future microprocessor system controller peripherals

2 regular module locations for spares

Chassis Properties:

Printed card file backplanes

Power distribution buses for +6 V, digital GND, ±15 V, analog GND, miscellaneous GND

Regulators for +5 V Vcc at each module location

Fan-forced air cooling

Physical Size:

System chassis:	10.5 H × 19 W × 23 D {in} 26.7 H × 48.3 W × 58.4 D {cm}
Double card module:	10.25 × 6.5 {in} 26 × 16.5 {cm}
Single card module:	4.5 × 6.5 {in} 11.4 × 16.5 {cm}

(2) ASP System Specifications

Basic system has 64 weight channels maximum

Analog signal amplitude ±10 V maximum

Primary inputs, error signal bandwidth 500 KHz minimum

Weight channel bandwidth 250 KHz minimum	
Weight linearity ± 0.7% F.S.	
Clock phase duration (T1, T2, T3, T4)	50 nsec
Composite instruction period	250 nsec
Adapt cycle time	8 µsec
Adapt time per weight (64 weights)	125 nsec
Word width for data, instructions, weight arithmetic	16 Bit
Error measurement resolution	12 Bit
Weight channel resolution	8 Bit

(3) ASP Module Specifications

Weight Processor Module (WP):

PRAM - Program Memory 64 Word × 16 Bit

DRAM - Data Memory 16 Word × 16 Bit

ALU - 16 Bit × 16 Bit with data input registers (A,B), function registers (C,F), overflow detection and correction

Weights - 8 channels with 8 Bit MDACs, sign voltage comparators, and summing amplifier

Error Digitizer Module (ED):

ADC - 12 Bit resolution, 2 µsec maximum conversion time, ±5 V maximum input

µ Weight Register - 10 Bit

Null Weight Register - 12 Bit

Control Word Register - 12 Bit

Test Controller Module (TC):

Microprogram Loader Memory - EPROM 1K Word × 29 Bit
organized as 4 pages of 256 words

Adapt Sequencer Memory - PROM 32 Word × 16 Bit
organized as 2 pages of 16 words

Sequencer retrigger wait time selectable as External, 0, 2^4 to 2^{11} instruction execution periods

Filter Array Module (FA):

Filter channel amplifiers	-	16 ea
Buffer amplifiers	-	2 ea
Filter network header components	-	8 ea
Filter amplifier bandwidth	-	250 KHz minimum
Buffer amplifier bandwidth	-	500 KHz minimum

Note: Where applicable, all values are typical unless otherwise specified.

APPENDIX B

ASP MACHINE INSTRUCTIONS, FORMATS,

MNEMONIC SYSTEM PROGRAMS

(1) ASP Machine Instructions

Weight Processor Instruction Set

ACI - Arithmetic Control Instructions		PCI - Program Control Instructions	
NOP	No Operation	NOP	No Operation
LAR	Load A Register	LPC	Load Program Counter
PAR	Present A Register	IPC	Increment Program Counter
LBR	Load B Register	LAC-IPC	Load Address Counter and Increment Program Counter
CBR	Clear B Register	IAC	Increment Address Counter
LCR	Load C Register	LSC-IPC	Load Step Counter and Increment Program Counter
SCR	Shift C Register	ISC	Increment Step Counter
LFR	Load F Register	IPC-IAC	Increment Program and Address Counters
CFR	Clear F Register	ISC-LPC	Increment Step Counter and Load Program Counter

DTI - Data Transfer Instructions		IOI - Input Output Instructions (OP Codes)	
NOP	No Operation	NOP	No Operation
SMD	Store Memory Data	MDI	Move Data In
MMD	Move Memory Data	MDO	Move Data Out
MAD	Move ALU Data	MDI-SPD-IPC	Load Program Data and Increment Program Counter
MSR	Move Sign Register	LPC	Load Program Counter
MPD	Move Program Data	IPC	Increment Program Counter
$\overline{\text{BUSY}}$	WP Not Busy	MDI-LPC	Load Program Counter from System Bus
MAD-SMD	Move ALU Data and Store Memory Data	RAR-MDI	Read ADC Register from Error Digitizer
MSR-SMD	Move Sign Register and Store Memory Data		

Error Digitizer Instruction Set

OP Code Instructions

NOP	No Operation
LNW	Load Null Weight
LμW	Load μ Weight
LCW	Load Control Word
RAR	Read ADC Register
SCA	Start Convert ADC
RAR-MDI	Read and Transfer ADC Register to Weight Processor

Test Controller Instruction Set

Internal Control Instructions

Program Downloader

ADEN	Address Direct Enable
CPEN	Clock Enable
DLS	Down Load Stop

Sequencer Control

ADEN	Address Direct Enable
CPEN	Clock Enable
ABUSY	Adapt Busy Status
WPBI	Weight Processor Busy Inhibit
ADCBI	ADC Busy Inhibit
DATIN	Enable System Data Bus Input
DATOUT	Enable Controller Data Output

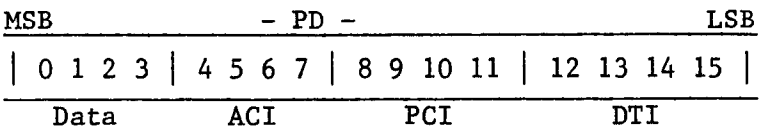
Instruction Code Summary Table

Code Dec.	Weight Processor				Error Digitizer OP Code	
	ACI	PCI	DTI	IOI		
0	NOP	NOP	NOP	NOP	NOP	IOI-OPC
1	LAR	LPC	SMD	MDI	LNW	Used
2	PAR	IPC	MMD	MDO	LμW	With
3	LBR	LAC-IPC	MAD	MDI-SPD-IPC	LCW	ADEN = 0
4	CBR	IAC	MSR	LPC	RAR	
5	LCR	LSC-IPC	MPD	IPC	SCA	
6	SCR	ISC	$\overline{\text{BUSY}}$	MDI-LPC	NOP	
7	LFR	IPC-IAC	MAD-SMD	NOP	NOP	
8	CFR	ISC-LPC	MSR-SMD	NOP	NOP	IOI-OPC
9	NOP	NOP	NOP	NOP	NOP	Used
10	|	|	|	RAR-MDI	RAR-MDI	With
11	|	|	|	MDI-SPD-IPC	NOP	ADEN = 1
12	|	|	|	IPC	|	
13	|	|	|	MDI-LPC	|	
14	|	|	|	MDI	|	
15	NOP	NOP	NOP	NOP	NOP	

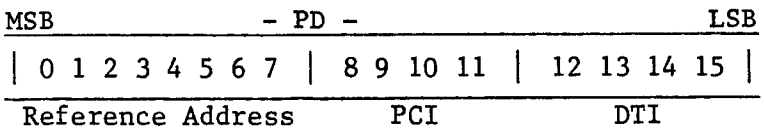
(2) ASP Program and Data Word Formats

Weight Processor Formats

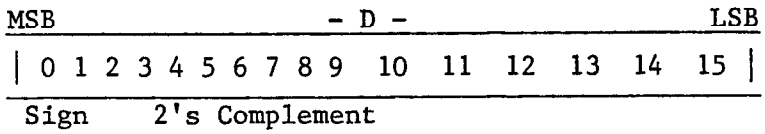
Program Word



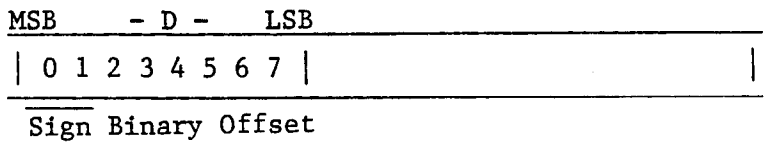
Program Word With Jump



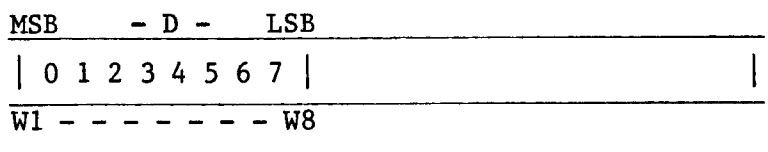
ALU Data



Weight Registers

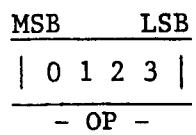


Sign Register



(Weight X input = +) → (WX = 1)

IOI-OP Code



Error Digitizer Formats

ADC Register

MSB						- D -								LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

ADC

0	0	0	0	0	1	2	3	4	5	6	7	8	9	10	11

12 Bit ADC, LSB Justified
2's Complement Data
For some experiments an MSB justified or a nonlinearly weighted connection of the 4 most significant ADC output bits is also used.

μ Weight Register

MSB						- D -				LSB					
		2	3	4	5	6	7	8	9	10	11				

Unipolar Binary

Null Weight Register

MSB						- D -				LSB					
0	1	2	3	4	5	6	7	8	9	10	11				

Offset Binary

Control Word Register

MSB						- D -				LSB					
0	1	2	3	4	5	6	7	8	9	10	11				

Future ADC Resolution Control
Not Used
Resolution Control On
Zero Reference 2 Input
Zero Reference 1 Input

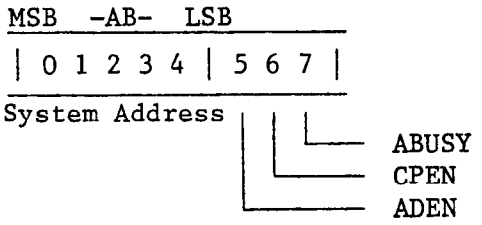
OP Codes

MSB			LSB
0	1	2	3

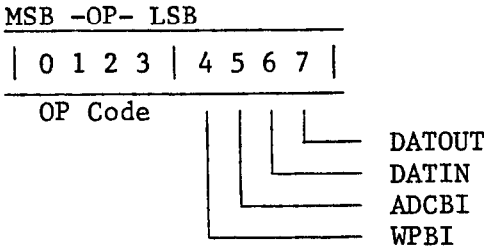
- OP -

Test Controller Formats

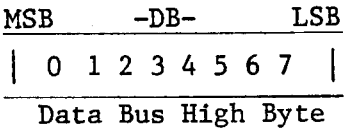
Sequencer Word 1



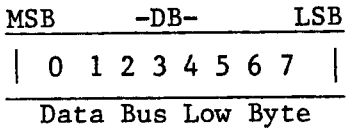
Sequencer Word 2



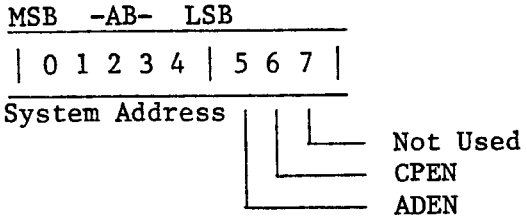
Program Downloader Word 1



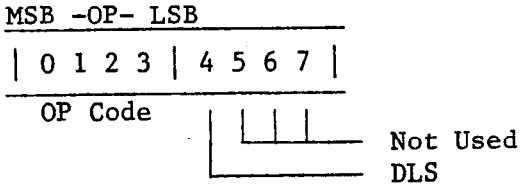
Program Downloader Word 2



Program Downloader Word 3



Program Downloader Word 4



(3) SIGNUM PROGRAM

SEQUENCER			WEIGHT PROCESSOR			REMARKS
Line	Addr.	Instruction	Line	Addr.	Instruction	
			1	0	PAR-LAC(0)-IPC	Initialize DRAM and weights; data comes from downloader addresses 2A-33; Executed during downloading
			2	1	CBR-SMD-IAC-IPC	
			3	2	CFR-SMD-IAC-IPC	
			4	3	SMD-IAC-IPC	
			5	4	SMD-IAC-IPC	
			6	5	SMD-IAC-IPC	
			7	6	SMD-IAC-IPC	
			8	7	SMD-IAC-IPC	
			9	8	SMD-IAC-IPC	
			10	9	SMD-IAC-IPC	
			11	A	SMD-IAC-IPC	
1	0	$\overline{\text{ADEN}}$(1)-CPEN-$\overline{\text{WPBI}}$-$\overline{\text{ADCBI}}$-LμW	12	B	$\overline{\text{BUSY}}$-LAC(0)-IPC	Start Signum Program
2	1	$\overline{\text{ADEN}}$(1)-CPEN-WPBI-ADCBI-SCA	13	C	LCR-MSR-SMD-IAC-IPC	
3	2	$\overline{\text{ADEN}}$(0)-$\overline{\text{CPEN}}$-WPBI-$\overline{\text{ADCBI}}$	14	D	LBR-SMD-IAC-IPC	Wait for EOC
4	3	ADEN-CPEN-WPBI-ADCBI-RAR	|	D	LBR-SMD-IAC-IPC	Get and Store Error
5	4	$\overline{\text{ADEN}}$(0)-CPEN-$\overline{\text{WPBI}}$-ADCBI	15	E	LAR-IAC-IPC	
|	|	|	16	F	MAD-SMD-IAC-IPC	Update 1st Weight
|	|	|	17	10	SCR-IPC	
|	|	|	18	11	LAR-IPC	
|	|	|	19	12	MAD-SMD-IAC-IPC	Update 2nd Weight
|	|	|	20	13	SCR-IPC	

5	4	$\overline{ADEN}$(0)-CPEN-$\overline{WPBI}$-ADCBI	21	14	LAR-IPC	
			22	15	MAD-SMD-IAC-IPC	Update 3rd Weight
			23	16	SCR-IPC	
			24	17	LAR-IPC	
			25	18	MAD-SMD-IAC-IPC	Update 4th Weight
			26	19	SCR-IPC	
			27	1A	LAR-IPC	
			28	1B	MAD-SMD-IAC-IPC	Update 5th Weight
			29	1C	SCR-IPC	
			30	1D	LAR-IPC	
			31	1E	MAD-SMD-IAC-IPC	Update 6th Weight
			32	1F	SCR-IPC	
			33	20	LAR-IPC	
			34	21	MAD-SMD-IAC-IPC	Update 7th Weight
			35	22	SCR-IPC	
			36	23	LAR-IPC	
			37	24	MAD-SMD-IAC-IPC	Update 8th Weight
			38	25	$\overline{BUSY}$	Advance Sequencer
6	5	$\overline{ADEN}$(0)-CPEN-$\overline{WPBI}$-ADCBI	39	26	LPC-MPD (4B)	Jump to Start
			40	B	$\overline{BUSY}$-LAC(0)-IPC	Start of 2nd Pass
7	6	$\overline{ADEN}$(1)-CPEN-WPBI-ADCBI-SCA	41	C	LCR-MSR-SMD-IAC-IPC	
8	7	$\overline{ADEN}$(0)-$\overline{CPEN}$-WPBI-$\overline{ADCBI}$	42	D	LBR-SMD-IAC-IPC	Wait for EOC
9	8	ADEN-CPEN-WPBI-ADCBI-RAR		D	LBR-SMD-IAC-IPC	Get and Store Error
10	9	$\overline{ADEN}$(0)-CPEN-$\overline{WPBI}$-ADCBI	43	E		Update Weights 1 - 8
			65	24		

10	9	$\overline{\text{ADEN}}$(0)-CPEN-$\overline{\text{WPBI}}$-ADCBI	66	25	$\overline{\text{BUSY}}$	
11	A	$\overline{\text{ADEN}}$(0)-CPEN-$\overline{\text{WPBI}}$-ADCBI	67	26	LPC-MPD (4B)	
			68	B	$\overline{\text{BUSY}}$-LAC(0)-IPC	Start of 3rd Pass
12	B	$\overline{\text{ADEN}}$(1)-CPEN-WPBI-ADCBI-SCA	69	C	LCR-MSR-SMD-IAC-IPC	
13	C	$\overline{\text{ADEN}}$(0)-$\overline{\text{CPEN}}$-WPBI-$\overline{\text{ADCBI}}$	70	D	LBR-SMD-IAC-IPC	Wait for EOC
14	D	ADEN-CPEN-WPBI-ADCBI-RAR		D	LBR-SMD-IAC-IPC	Get and Stroe Error
15	E	$\overline{\text{ADEN}}$(0)-CPEN-$\overline{\text{WPBI}}$-ADCBI	71	E		Update Weights 1 - 8
			93	24		
			94	25	$\overline{\text{BUSY}}$	
16	F	$\overline{\text{ADEN}}$(0)-$\overline{\text{CPEN}}$-WPBI-ADCBI	95	26	LPC-MPD (4B)	Stop and Wait for Sequencer to Restart
	0	$\overline{\text{ADEN}}$(1)-CPEN-$\overline{\text{WPBI}}$-$\overline{\text{ADCBI}}$-LμW		26	LPC-MPD (4B)	
				B	$\overline{\text{BUSY}}$	Start of New Sequence
	1	$\overline{\text{ADEN}}$(1)-CPEN-WPBI-ADCBI-SCA		C	LCR-MSR-SMD-IAC-IPC	
	2	$\overline{\text{ADEN}}$(0)-$\overline{\text{CPEN}}$-WPBI-$\overline{\text{ADCBI}}$		D	LBR-SMD-IAC-IPC	Wait for EOC
	3	ADEN-CPEN-WPBI-ADCBI-RAR		D	LBR-SMD-IAC-IPC	Get and Store Error
	4	$\overline{\text{ADEN}}$(0)-CPEN-$\overline{\text{WPBI}}$-ADCBI		E		Update Weights 1 - 8
				24		
				25	$\overline{\text{BUSY}}$	
	5	$\overline{\text{ADEN}}$(0)-CPEN-$\overline{\text{WPBI}}$-ADCBI		26	LPC-MPD (4B)	
				B	$\overline{\text{BUSY}}$-LAC(0)-IPC	
	6	ADEN(1)-CPEN-WPBI-ADCBI-SCA		C	LCR-MSR-SMD-IAC-IPC	Get and Store Error

SIGNUM PROGRAM DOWNLOADER

LINE	ADDRESS	INSTRUCTION	COMMENTS
1	0	MDI-LPC(00)-ADEN-CPEN	Set program counter to zero.
2	1	MDI-SPD-IPC-ADEN-CPEN	Down load signum program.
40	27		
41	28	MDI-LPC(40)-ADEN-CPEN	Set program counter to (H40); start execution.
42	29	NOP-ADEN-CPEN	
43	2A	MDI-ADEN-CPEN	Sign - Reg
44	2B		Error - Reg
45	2C		Weight 1
46	2D		Weight 2
47	2E		Weight 3 — Initial data
48	2F		Weight 4
49	30		Weight 5
50	31		Weight 6
51	32		Weight 7
52	33		Weight 8
53	34	MDI-LPC(00)	Stop weight processor.
54	35	LNW-$\overline{\text{ADEN}}$(1)-CPEN	Initialize error digitizer.
55	36	LμW-$\overline{\text{ADEN}}$(1)-CPEN	
56	37	LCW-$\overline{\text{ADEN}}$(1)-CPEN	
57	38	MDI-LPC(4B)-ADEN-CPEN	Reset weight processor at (H4B).
58	39	DLS-$\overline{\text{ADEN}}$(0)-$\overline{\text{CPEN}}$	Stop downloader