An Educational Digital Signal Processing Algorithm Instruction Developer (DSP-AID)

by

Patrick J. Gries, member IEEE, **Ke-Kang Chin**, member IEEE

and **Jafar Saniie**†, member IEEE

†Phone: (312) 567-3400

Department of Electrical and Computer Engineering

Illinois Institute of Technology

Chicago, IL 60616

Abstract

This paper describes an educational digital signal processing algorithm instruction developer (DSP-AID) which is designed to be interactive and is a marriage of a general purpose microprocessor (MC68000) with a DSP processor (TMS32020). The design architecture allows for the MC68000 to control the user interface and the TMS32020 to optimally perform the signal processing tasks. The system is equipped with a frontend analog-to-digital converter and a backend digital-to-analog converter for real-time digital signal processing. This system is economical and has the potential for being a valuable educational tool for students learning microprocessors and signal processing algorithm development at the machine level. Targeted users are design engineers, senior level undergraduates and first-year graduate students. This paper presents both system design description and several real-time application examples with results.

I. Introduction

In recent years, the trend in signal processing has been to complement or replace analog systems with digital system implementations. Engineers are often tasked with analyzing a system to find a means for improvement and are always looking for the best approach to solve a given problem. From a system level perspective, signal processing has gained another tool for problem solving in the form of *economical* digital signal processing (DSP). DSP techniques have been around for several decades, but with the advent of DSP chips the implementation of DSP techniques has been greatly simplified. The development of DSP system components into integrated circuits has simplified the design and allowed for the practical and real-time development of a DSP system for such applications as filtering and spectral analysis. This paper, tutorial in nature, presents the design and hardware implementation of a DSP Algorithm Instruction Developer (DSP-AID) system which integrates two popular microprocessors, namely the MC68000 and the TMS32020. In addition, sample applications with actual results are presented.

In general, the input to a signal processing system is either analyzed for its information content or is modified in some prescribed manner. In an analog signal processing system (Figure 1a) the incoming analog signal is processed directly via electronic circuitry. In a DSP system (Figure 1b) the incoming analog signal is sampled, often uniformly, to generate a sequence of discrete values, x(n). The sampled signal is processed numerically, as in the case of digital filters, which results in an output sequence, y(n). If desired, the output sequence, y(n), can be converted to an analog signal using an analog-to-digital converter. The advantages of using a digital processing approach are long term stability (i.e. no component

value drift due to aging), ease of modifying the processing characteristics and ability to realize more elaborate processing structures.

Any analog signal processing system is a potential candidate for digital signal processing, although a digital signal processing approach is not the answer to every signal processing problem. A thorough understanding of the realistic expectations of a DSP system is needed. Hardware costs are decreasing and DSP processor performances are increasing. DSP requires software development whereas analog signal processing does not. The software requirements of DSP demand a more thorough analysis of the design prior to the start of hardware development. Design flaws will thus be found earlier and remedied. Once the hardware is in place, DSP applications become a software issue. That is, a different processing structure will require only a different set of processing coefficients or a different software program to be run. A DSP system can be used to implement many different algorithms such as digital filtering, fast Fourier transform (FFT) and waveform generation.

Many of the common processing algorithms such as filtering can represented in the form of a difference equation [1-4].

$$y(n) = \sum_{i=1}^{M} a_i\, y(n-i) + \sum_{i=0}^{N} b_i\, x(n-i)$$

In the above equation, the *y(n)* represents the output sequence and *x(n)* represents the input sequence. The terms a_i and b_i are constants governing the characteristics of the processing algorithm. If the coefficients a_i are zero, then the above equation results in a finite impulse response (FIR) or nonrecursive filter. On the other hand, if any of the a_i coefficients are nonzero, then the equation is recursive which results in an infinite impulse response (IIR) filter.

Note that the above equation is nothing more than a sequence of

multiplication and addition operations. DSP chips are particularly designed to perform efficient multiplications and additions which is needed in many real-time digital signal processing systems. Further discussion of digital filters and other processing such as waveform generation and spectral analysis are presented in the applications section.

II. Hardware Implementation

A DSP system adds its own set of design issues. For instance, an ideal DSP system assumes infinite precision/resolution with respect to the sampled signal, the filter coefficients, and the multiplication operations. Analog-to-digital converters are available with increasing bits of resolution but a 16-bit or greater ADC is far from being infinite. Also, the registers for storing the coefficients and the resulting products are of finite length, resulting in a truncation of the values. All of these potential error issues must be analyzed to verify that the system (i.e. filter) remains stable and operates within an acceptable range. Stability means that the output will not oscillate and/or yield a saturated result. For example, the FIR filter is always stable due to the lack of a feedback component, however, the IIR filter may become unstable due to potential errors. The incurred expense for stability of an FIR filter is greater processing time resulting from a longer filter structure (i.e. more delay taps).

Any microprocessor is capable of implementing DSP algorithms. In general, digital signal processing is computationally intensive. However, a general purpose microprocessor is not very efficient in multiplication and accumulation (MAC) operations which are at the heart of many DSP applications. The architectures of many DSP chips are optimized for single cycle multiplications. In filter applications, efficient data movement is also necessary to achieve a high

throughput. The unit presented here is a complete DSP system, with a front-end analog-to-digital converter and a back-end digital-to-analog converter and can be interfaced to any personal computer using an RS-232 port for host control. An experimental set-up is shown in Figure 2. The DSP-AID is interactive and is a marriage of a general purpose microprocessor (MC68000) with a general purpose digital signal processor (TMS32020) to achieve an economical algorithm development system (see Figure 3). The MC68000 facilitates set-up and algorithm development and provides the mechanism for user interaction [5]. The TMS32020 performs real-time processing tasks efficiently [6].

This system will aid in the determination of the feasibility of a digital approach over an analog approach. A side benefit of this design is the comparison of processing powers of the two processors utilized. This would be useful in industry when a choice of processors must be made. This unit would serve educationally in an advanced undergraduate or graduate lab course in digital signal processing. Targeted applications for this unit are adaptive signal processing, biomedical signal processing, speech processing, image processing, servo control, automotive control, robotics, etc.

The functional blocks of the DSP-AID system are: (1) analog input, (2) analog-to-digital converter, (3) numeric processor, (4) digital-to-analog converter and (5) analog output.

1. Analog Input

The analog input section consists of an 8:1 analog multiplexer and an anti-aliasing filter. Any one of 8 analog inputs can be switched into the DSP-AID for subsequent processing. An analog multiplexer is an economical way to increase

signal processing capabilities. The other alternative approach would be to add multiple A/D converters which are more costly than analog multiplexers. One of the inputs to the analog multiplexer could be from a +2.5V precision reference which is very useful for calibrating the analog signal path from system input to system output. A second dedicated analog input could be the output from a function generator integrated circuit. Several manufacturers offer IC's which generate sine, triangle and rectangular waveforms (e.g. XR2206 manufactured by Exar Corporation).

Anti-aliasing filters are generally recommended and provide signal buffering. The front-end of any DSP system requires a relatively simple analog filter to remove the higher unwanted components of the signal. Usually, a 2nd or 3rd order active low-pass filter will suffice. DSP requires adherence to the theory of discrete time-sampled systems. A fundamental requirement is the Nyquist criteria which states that no signal component can be greater than one-half the sampling frequency. A frequency above the Nyquist frequency will "alias" itself back into a lower frequency. For example, in a system with a sampling frequency of 1200 Hz, the maximum allowed frequency which can be theoretically reconstructed is 600 Hz. A 1000 Hz frequency component will alias itself back and appear as a 200 Hz component (see Figure 4). In a filter application, the anti-aliasing filter may be thought of as "coarse processing" and the resulting numeric calculations as "fine processing." The DSP-AID uses a 4th order Butterworth filter which is maximally flat in the passband, has an insignificant phase distortion, and has a modest rolloff.

2. Analog/Digital Converter

The analog-to-digital converter (ADC) has a conversion time of 5μs which provides a theoretical sampling rate up to 200 KHz. However, achieving an accurate conversion requires a stable, non-changing input. This is because when using a successive approximation converter (SAC) a conversion is tried and the digital result is fed to a digital-to-analog converter and the resulting analog signal is compared to the original analog signal. The process continues until a match is within the ADC's resolution. This type of ADC requires a holding circuit that is synchronized to the ADC's sampling rate. The needed component is a Sample-and-Hold (S/H) circuit which is typically available as a single integrated circuit. The resolution of the ADC used in this design is 12 bits. For an input range of 5 Volts, this unit yields a conversion resolution of 1.22 mVolts/bit. Noise may be produced as a result of quantification. The ADC should be buffered from the noisy data bus as unwanted frequency components may be coupled into the ADC during a conversion. The signal to noise ratio (SNR) for the conversion process is [2]:

$$\text{SNR} = 6\text{N} - 1.24\text{dB},$$

where N is the number of bits of the ADC. For the 12-bit ADC, the SNR is about 70dB.

Two important issues which should be considered when using ADC's are: (1) system power supply and (2) format of digital output from the converter. First, switching power supplies typically have switching spikes which must be filtered. Switchers generally operate in the range from 20 KHz to 200 KHz and the switching spikes may be 100 mVpp or higher. Second, since the digital data is to be used in numerical calculations, a conversion should be made to a two's complement value.

This is necessary so that multiplication and addition carries & overflows are handled appropriately. The DSP-AID uses an ADC with a straight binary output and converts it into two's complement format via software.

The sampling clock allows for discrete sampling rates from 38Hz to 2.5MHz. Having software control of the sampling rate results in a useful and powerful feature. Different signal inputs require different sampling rates. A side benefit is the ability to shift a filter's cutoff frequencies since the coefficients of a filter are calculated using normalized cutoff frequencies. Adaptive filtering is thus far easier to accomplish with digital techniques.

3. Numeric Processor

At the heart of a DSP system is the component which performs the number crunching. The numeric processor must be able to multiply two numbers very fast while maintaining a high degree of precision before truncating the result. The numeric processor must also be an efficient mover of data since many multiplications and additions are required before producing each output value. The processors used in the DSP-AID are the MC68000 and the TMS32020. The MC68000 is a general purpose microprocessor with an open internal architecture freeing it from being dedicated to any particular class of applications. The TMS32020 on the other hand is a digital signal processor with an internal architecture optimized for DSP applications. The two processors are tightly coupled by sharing a common memory. In the DSP-AID the user interface is through the MC68000. A one-line assembler/disassembler is resident for MC68000 code development. The MC68000 is the master processor and has its own local memory. Memory for the TMS32020 is shared with the MC68000. It is through the shared memory that code for the TMS32020 is entered. The TMS32020 uses

the DSP resources until superseded by the MC68000.

4. Digital/Analog Converter

In the DSP-AID the choice of a suitable digital-to-analog converter (DAC) is dependent upon the accuracy of the numeric processor and usually has the same resolution as the ADC. In our system, the DAC has 12 bits of resolution. The settling time of the DAC is of primary concern. It is recommended that the slew rate of the DAC output be greater than 10 times the input signal bandwidth. Since a finite amount of time is spent processing the input signal a constant time delay is present in the output. A constant time delay can be thought of as a linear phase delay component to be added to the theoretically calculated phase delay component of the processing algorithm. This processing delay may be altered by waiting until the next input sample is taken and then output the signal. This will result is a "unit-sample delay" system wherein the delay is constant, regardless of the processing time. Lastly, the computed binary result must be converted from two's complement to straight binary prior to conversion to analog form. This design converts the 2's complement format into straight binary via software.

5. Analog Output

In the DSP-AID the analog output may come from one of three sources: (1) directly from the DAC, (2) from the smoothing filter or (3) from the audio amplifier with or without smoothing. The basic motive for the smoothing filter is to remove the "staircase" effect associated with a DAC. A 4th order low pass filter was chosen for the smoothing filter and is similar to the anti-aliasing filter.

An audio amplifier is available for applications involving digital signal processing of speech signals. Also, it can be used to produce sounds created by a

DSP tone generator. A DSP tone generator is a difference equation for a sinusoidal wave. A DSP tone generator can be designed using the sample rate clock set within the audio range of 20Hz to 20KHz and applying sampled data system concepts.

III. Procedural Operations of the DSP-AID

To execute an algorithm the compiled object code for the TMS32020 is resident on a host computer. The TMS32020 is put into an inactive standby state. The MC68000 accomplishes this by asserting the TMS32020 HOLD input. The MC68000 downloads the code from the host computer (such as a personal computer) to the shared memory using the resident monitor [5]. Upon completion of the download process the TMS32020 is given a reset so that it will start executing the new code from the beginning [6]. The MC68000 then releases the TMS32020 into the active mode. The TMS32020 will then be executing the new application algorithm.

A DSP resource manager allocates control of the DSP resources to the processors. Only one processor at a time may access the DSP resources. Although this design is intended to evaluate processing speeds, it is possible to allow both processors to be executing independent algorithms simultaneously.

The arbitration switch directs address, data and control lines from the selected processor to the DSP resources. The MC68000 is the system master and may request resource control by accessing the memory mapped arbitration switch. The TMS32020 is put into an inactive standby mode by writing an appropriate value to the switch. Upon acknowledging that the TMS32020 has released control of the DSP resources the arbitration switch allow the MC68000 access to the DSP resources. The MC68000 returns control of the DSP resources to the TMS32020 by

writing the valid "release resource" data value to the switch.

IV. Applications

In this section, we present several classical examples of DSP applications such as digital filtering, waveform generation and spectral analysis.

IIR Low-pass Filter

In many applications, filters may be needed to suppress unwanted signals. With DSP-AID system, the filtering process can be easily implemented digitally. These methods are IIR and FIR digital filter designs as were described briefly in the introduction. IIR filter design is based on choosing an analog filter as a desired prototype. Then, we apply the bilinear transformation technique to this analog transfer function in order to achieve the digital counterpart [1-4]. The example presented here is a two-stage cascaded fourth order Chebyshev low-pass filter with a 10 KHz sampling frequency and a 3 KHz bandwidth. The transfer function of the digital IIR filter for each stage is given by

$$H_i(z) = \frac{\sum_{k=0}^{2} b_{ki} z^{-k}}{1 - \sum_{k=1}^{2} a_{ki} z^{-k}} \qquad i=1,2$$

where a_{ki} and b_{ki} are filter coefficients. Then, the cascaded transfer function becomes

$$H(z) = H_1(z) H_2(z).$$

The block diagram of this filter is shown in Figure 5, and, the coefficients of this filter are presented in Table 1. Note that these coefficients are generated using the bilinear transformation method. The magnitude and phase frequency response

writing the valid "release resource" data value to the switch.

IV. Applications

In this section, we present several classical examples of DSP applications such as digital filtering, waveform generation and spectral analysis.

IIR Low-pass Filter

In many applications, filters may be needed to suppress unwanted signals. With DSP-AID system, the filtering process can be easily implemented digitally. These methods are IIR and FIR digital filter designs as were described briefly in the introduction. IIR filter design is based on choosing an analog filter as a desired prototype. Then, we apply the bilinear transformation technique to this analog transfer function in order to achieve the digital counterpart [1-4]. The example presented here is a two-stage cascaded fourth order Chebyshev low-pass filter with a 10 KHz sampling frequency and a 3 KHz bandwidth. The transfer function of the digital IIR filter for each stage is given by

$$H_i(z) = \frac{\sum_{k=0}^{2} b_{ki} z^{-k}}{1 - \sum_{k=1}^{2} a_{ki} z^{-k}} \qquad i=1,2$$

where a_{ki} and b_{ki} are filter coefficients. Then, the cascaded transfer function becomes

$$H(z) = H_1(z) H_2(z).$$

The block diagram of this filter is shown in Figure 5, and, the coefficients of this filter are presented in Table 1. Note that these coefficients are generated using the bilinear transformation method. The magnitude and phase frequency response

of this IIR low-pass filter are shown in Figure 6. This figure shows 0.5 dB ripple in the passband and nonlinear phase response which can cause some distortion in the output signal.

In general, when designing an IIR filter, we can specify exact ripple level for the passband, stopband and the attenuation rate. Better performance can always be obtained using higher order IIR filters, however, there would be greater phase distortion associated with the high order filters. One potential problem with IIR filter is that errors resulting from truncating coefficients and calculated values due to the finite resolution of the processor may drive the filter into an unstable state.

The above filter has been implemented on the DSP-AID system and the real-time performance is shown in Figure 7. The input signal is a chirp sinusoidal function with a frequency sweeping from 1 KHz to 5 KHz as shown in the upper trace. The output signal, shown in the lower trace, has a cutoff frequency at about 3 KHz as expected. Notice that the minor variations of the amplitude in the output signal is due to the ripple existing in the passband of Chebyshev IIR filter.

FIR Low-pass Filter

FIR filter is designed using a truncated Fourier series approximation of the ideal frequency response of the filter [1-4]. To compensate for the effect of the truncation of the Fourier series, sometimes, a window function is applied, which overcomes the Gibbs phenomenon. The example presented here is an 80-tap low-pass digital FIR filter with 10 KHz sampling frequency and 3 KHz bandwidth. The transfer function is given by

$$H\ (\ z\ ) = \sum_{i=0}^{2M} a_i z^{-i}.$$

where 2M is the number of delay taps and the terms a_i are the filter coefficients. The block diagram of this filter is shown in Figure 8. It is important to point out that due to lack of a feedback path in the diagram more coefficients are needed to achieve a satisfactory attenuation rate in the transition band. Note that the FIR filter is always stable and can be designed to have a linear phase response in the frequency domain, which is equivalent to a constant delay in the time domain.

Coefficients in this low-pass filter are generated by using a Fourier series method and then modified by a Hanning window function. The coefficients are calculated using the following equation:

$$C_m = \frac{1}{2}\left[1 + \cos(m\frac{\pi}{40})\right]\frac{\sin 0.6m\pi}{m\pi} \qquad -40 \leq m \leq 40 \quad m \text{ is an integer}$$

$$a_i = C_{i-40} \qquad \text{for} \qquad i = 0,1,2,...,80$$

Since coefficients are symmetrical, only half of them are presented in Table 2. Interested readers can find more information about this subject in many digital signal processing books [e.g., see 1-4]. The magnitude and phase frequency response of this FIR filter are shown in Figure 9. Contrary to the IIR filter, the FIR filter shows linear phase response which implies no phase distortion in the output signal.

The real-time input and output signals of this FIR filter are shown in Figure 10. The input is a chirp sinusoidal signal shown in the upper trace ranging from 1 KHz to 5 KHz. In the lower trace, the output signal is cut off at 3 KHz. Note that the gradual attenuation of the output amplitude in the passband is due to the characteristics of the output low-pass reconstruction filter. In general, better performance can be obtained by increasing the number of delay taps of the filter, however, this will result in higher computation time which, in turn, limits the sampling rate. In other words, this would lower the frequency limit on the

applicable input signal.

Waveform Generation

In many applications, certain special waveforms are required that cannot be generated easily using available instrumentation. Furthermore, many conventional function generators can only provide several common waveforms such as sine, cosine, triangle and square waves. Therefore, if system flexibility and versatility are the concerns of the user, the DSP-AID may be the alternative choice serving as a waveform generator. Waveform generation uses the same processing algorithm as the IIR filter. Hence, one can generate different waveforms using different coefficients. The Z-transform of the desired waveform can be used to obtain a difference equation in which its impulse response is the waveform of interest.

The example presented here is a sine function generator. The difference equation is given by

$$y(n) = a_1\ y(n-1)+a_2\ y(n-2)+b_1\ x(n-1)$$

$$a_1 = 2\cos(2\pi f/f_s), \qquad a_2 = -1, \qquad b_1 = \sin(2\pi f/f_s)$$

where f is the frequency of the sine function and f_s represents the sampling rate.

In this example, we chose f = 1 KHz and f_s = 40000 samples/second. The real-time output of this sine function generator (i.e., the impulse response of the above difference equation) is shown in Figure 11. This output signal is a sine function with 1 KHz frequency as designed. Since an audio amplifier is implemented in the DSP-AID system, if this output signal is used to feed in this audio amplifier, a constant tone can be generated for other applications.

Spectrum Analyzer

Often engineers are confronted with evaluating the frequency contents of signals. The DSP-AID system presented here has the capability of being programmed as a spectrum analyzer. This spectrum analyzer is implemented by using 64-point Discrete Fourier Transform (DFT) of the sampled input signals. The sampling frequency used here is 10 KHz, resulting in the frequency resolution of 156.25 Hz. Note that, the frequency resolution is determined by the ratio of the sampling frequency over the number of samples. Since the amplitude spectrum is always symmetrical for real function, we need only to show half of the spectrum, in this case, 32 frequency components. The DFT of a sequence x(n) of N samples is given by

$$X(k) = \sum_{n=0}^{N-1} x(n) \left[\cos(2\pi kn/N) - j\sin(2\pi kn/N) \right], \quad 0 \le k \le N-1, \; N=64.$$

The following steps are taken to implement the spectrum analyzer based on the above equation:

Step 1 - Set up a lookup table of precalculated cosine & sine values.

Step 2 - Capture 64 points input samples using Analog/Digital converter.

Step 3 - Apply DFT algorithm to calculate the first 32-point power spectrum, $|X(k)|^2$, and store them in memory.

Step 4 - Display the 32-point power spectrum.

Step 5 - Repeat from Step 2.

The spectrum analyzer was evaluated using a sine and a square wave

function with the same frequency of 937.5 Hz, and the results are presented in Figure 12 and Figure 13. The results shown in Figure 12 confirm that the Fourier transform of this sine function is an impulse function at a frequency of 937.5 Hz. Further, the Fourier transform of the square wave shown in Figure 13 has only odd harmonics. The fundamental, third and fifth harmonics are displayed at 937.5 Hz, 2812.5 Hz and 4687.5 Hz. The power of the third harmonic is 1/9 of the power of the fundamental frequency, and this can be verified from Figure 13. The power of the fifth harmonic is 1/25 of the power of the fundamental frequency, which also can be verified from the same figure.

V. Conclusion

In this paper we have presented an economical system (DSP-AID) for developing DSP algorithms, which is also suitable for use as an educational tool. This interactive and versatile system can be interfaced to any personal computer and is designed for implementing and learning digital signal processing system at the machine level. The utilization of two popular processors, namely the MC68000 and the TMS32020, yields a system rich in flexibility and processing power. In this paper several applications have been presented along with real-time processing results. The ease of use of the DSP-AID for educational purposes allows the user to test and evaluate several algorithms within a 3-4 hour lab session. Targeted application areas for this unit are adaptive signal processing, biomedical signal processing, speech processing, image processing, servo control, automotive control and robotics.

VI. References

[1] John G. Proakis, Dimitris G. Manolakis, *Introduction to Digital Signal Processing*, Macmillan Publishing Company, New York 10022, 1988

[2] Oppenheim, A.V., and Schafer, R.W., *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1975.

[3] Stanley, W.D., Dougherty, G.R., and Dougherty, R., *Digital Signal Processing*, 2nd Edition, Reston Publishing Company, Inc., Reston, VA, 1984.

[4] Andreas Antoniou *Digital Filters: Analysis and Design*, McGraw-Hill Book Company, Inc., 1979.

[5] *MC68000 Educational Computer Board User's Manual, MEX68KECB/D2*, Motorola, Phoenix, Arizona 85036, 1982.

[6] *Digital Signal Processing Applications with the TMS320 Family*, Texas Instruments, Inc., 1986.

LIST OF FIGURES

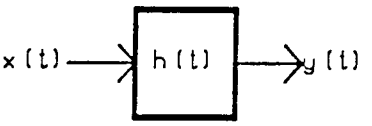
LIST OF TABLES

TABLE 1. Coefficients of a cascaded fourth order Chebyshev digital IIR filter

b_{01}	0.57606268	b_{02}	0.23769910
b_{11}	1.15212537	b_{12}	0.47539820
b_{21}	0.57606268	b_{22}	0.23769910
a_{11}	-0.58028252	a_{12}	0.22869057
a_{21}	-0.72396821	a_{22}	-0.17948698

TABLE 2. Coefficients of an 80 delay taps low-pass FIR filter

a_0	0.0000000000	a_{20}	0.0000000000
a_1	-0.0000119643	a_{21}	-0.0085916491
a_2	0.0000303090	a_{22}	0.0060101780
a_3	0.0000698585	a_{23}	0.0067874995
a_4	-0.0002057874	a_{24}	-0.0123837381
a_5	0.0000000000	a_{25}	0.0000000000
a_6	0.0004852304	a_{26}	0.0157202696
a_7	-0.0004177380	a_{27}	-0.0109560083
a_8	-0.0005583204	a_{28}	-0.0123779674
a_9	0.0011698785	a_{29}	0.0226972067
a_{10}	0.0000000000	a_{30}	0.0000000000
a_{11}	-0.0018297040	a_{31}	0.3026640825
a_{12}	0.0013772231	a_{32}	0.0211539500
a_{13}	0.0016544351	a_{33}	0.0247589289
a_{14}	-0.0031787275	a_{34}	-0.0477054760
a_{15}	0.0000000000	a_{35}	0.0000000000
a_{16}	0.0043579534	a_{36}	0.0738305860
a_{17}	-0.0031178419	a_{37}	-0.0615043640
a_{18}	-0.0035870296	a_{38}	-0.0929730571
a_{19}	0.0066423503	a_{39}	0.3022640825
		a_{40}	0.6000000000



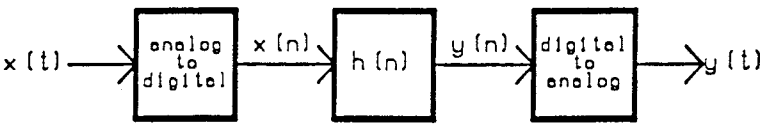


Figure 1. Real-time signal processing block diagrams: (a) analog (b) digital.

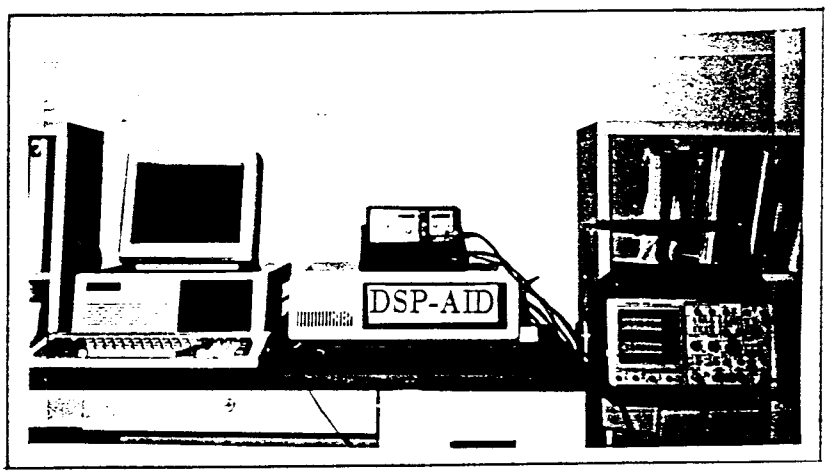


Figure 2. Experimental lab set-up showing the DSP-AID, host computer, function generator and oscilloscope.

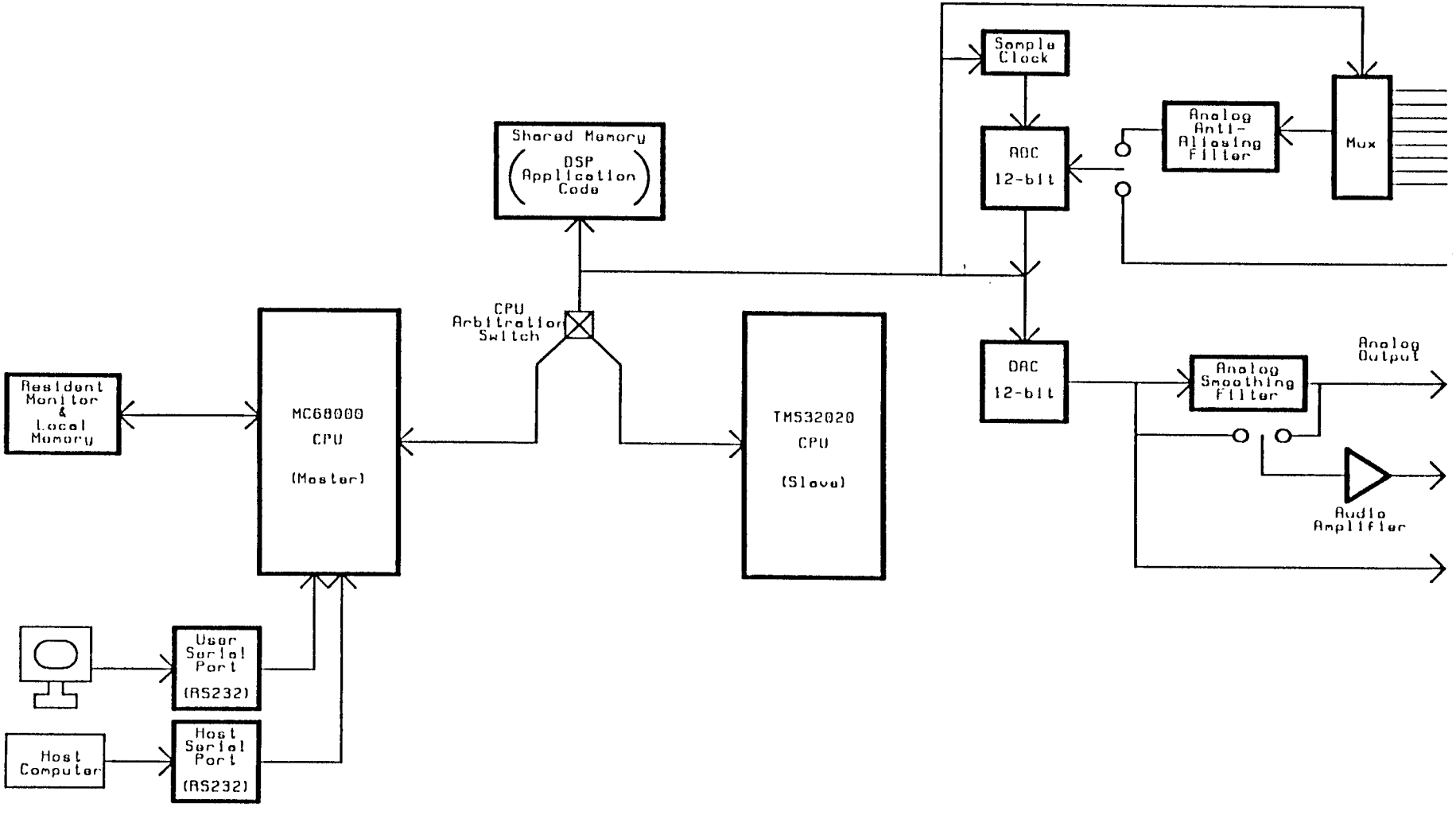


Figure 3. Functional block diagram of the DSP-AID.

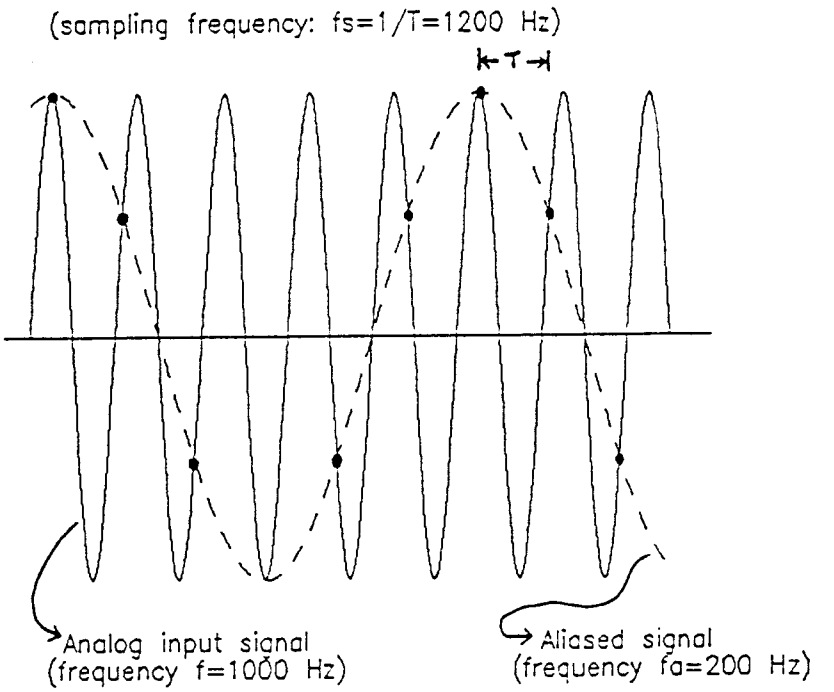


Figure 4. Frequency aliasing due to improper sampling.

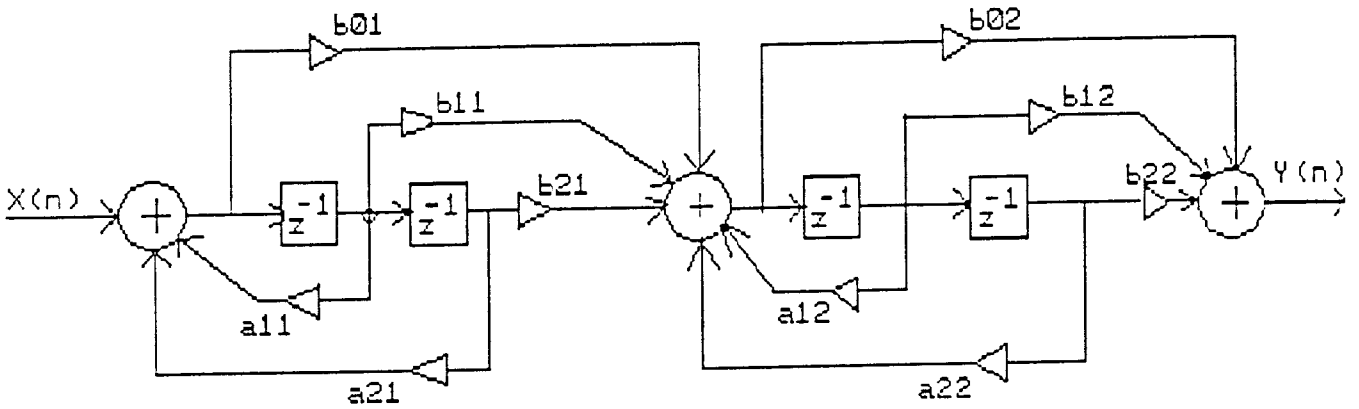


Figure 5. Block diagram of a two-stage cascaded fourth order IIR filter.

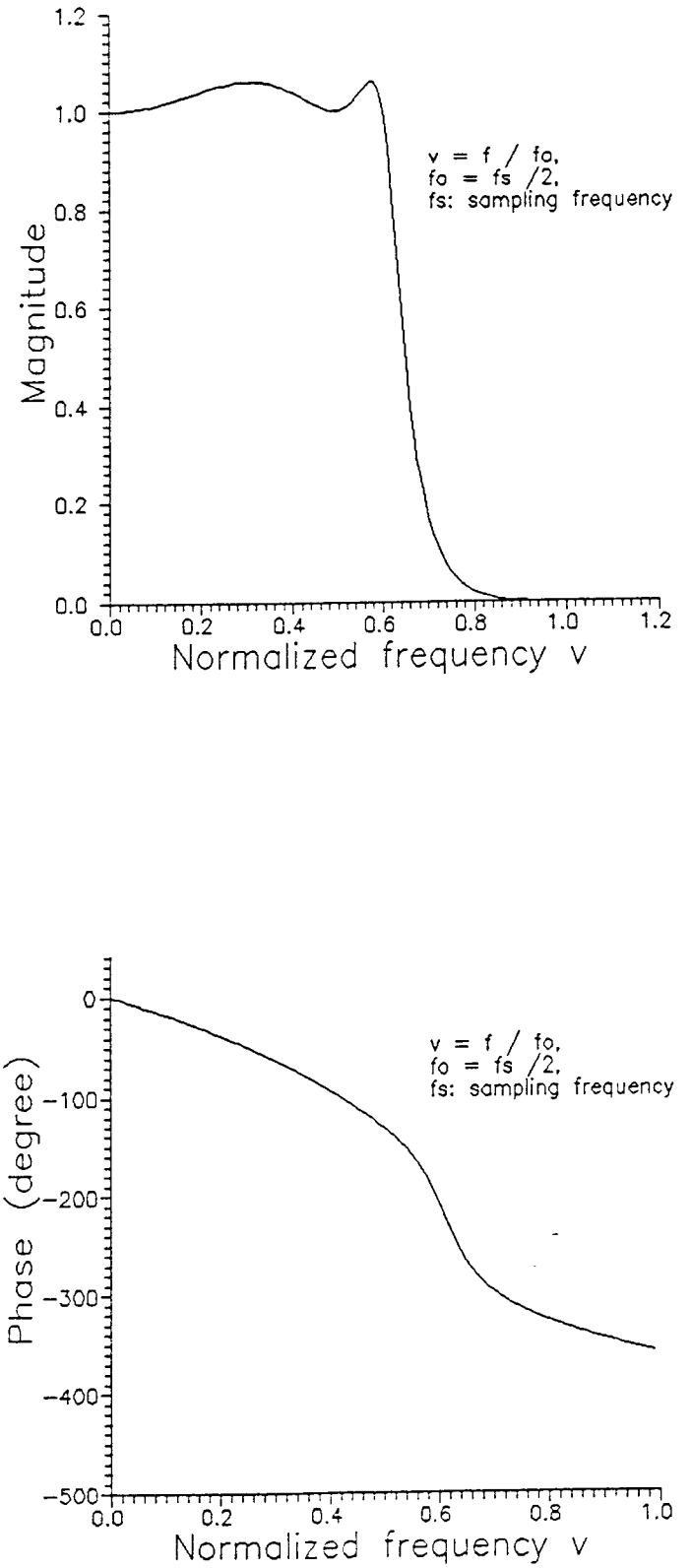


Figure 6. Frequency response of a two-stage cascaded fourth order low-pass IIR filter: (a) magnitude response and (b) phase response.

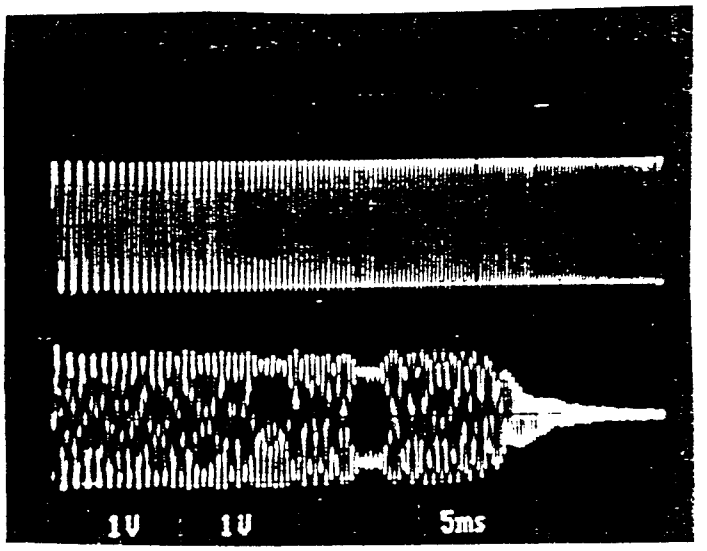


Figure 7. Real-time input and output signals of a two-stage cascaded fourth order low-pass IIR filter.

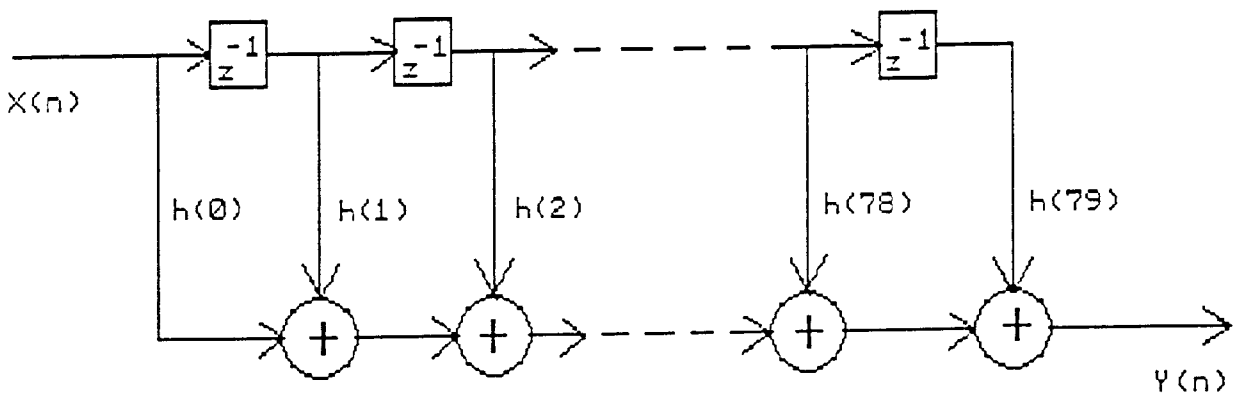


Figure 8. Block diagram of an 80 delay taps FIR filter.

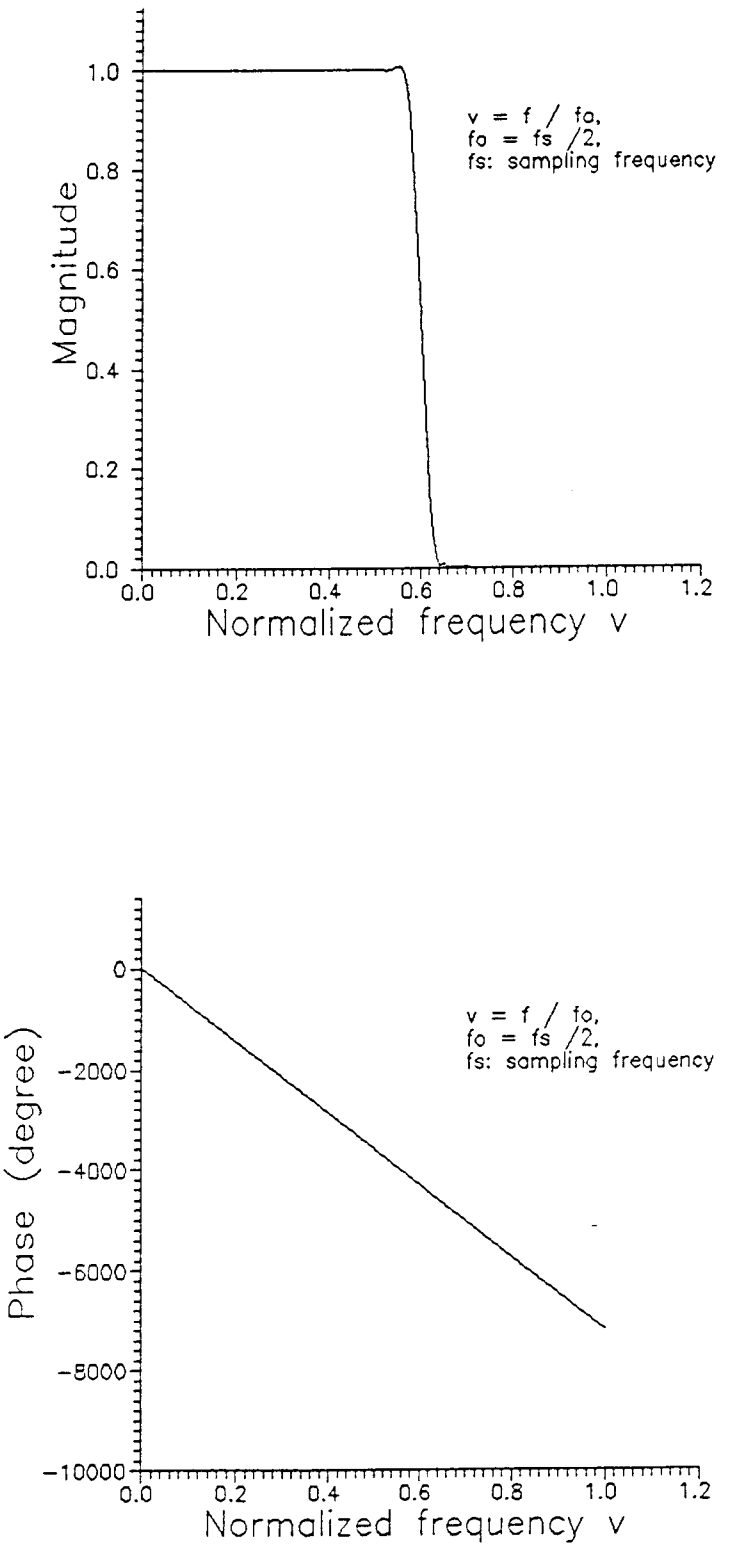


Figure 9. Frequency response of an 80 taps delay low-pass FIR filter: (a) magnitude response and (b) phase response.

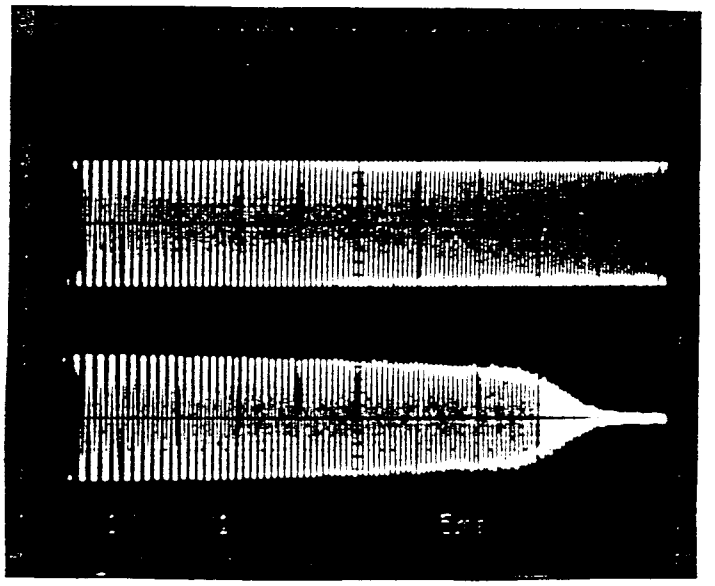


Figure 10. Real-time input and output signals of an 80 delay taps low-pass FIR filter.

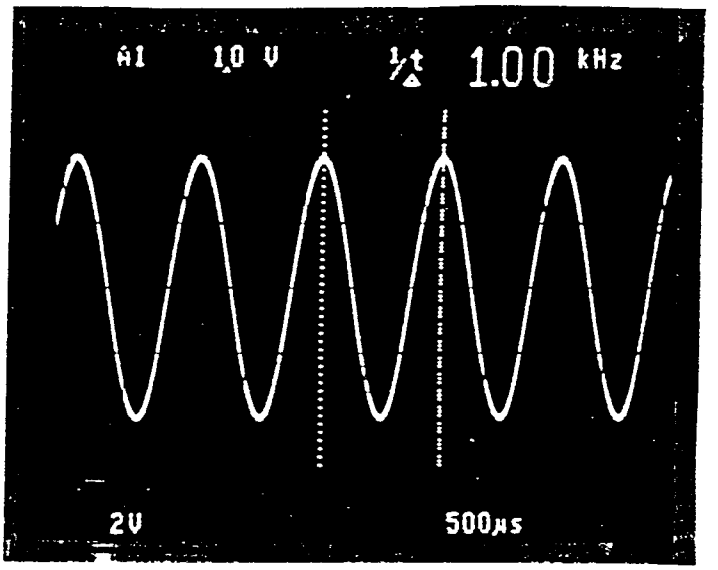


Figure 11. A sine function with 1 KHz frequency generated by DSP-AID system.

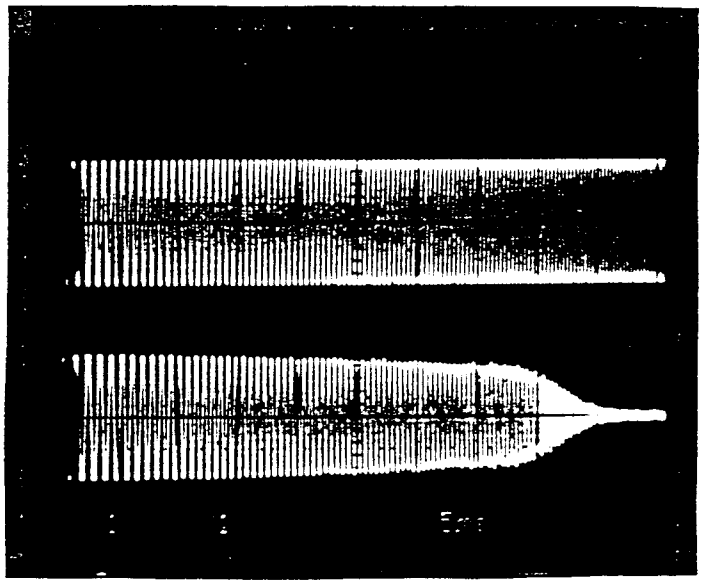


Figure 10. Real-time input and output signals of an 80 delay taps low-pass FIR filter.

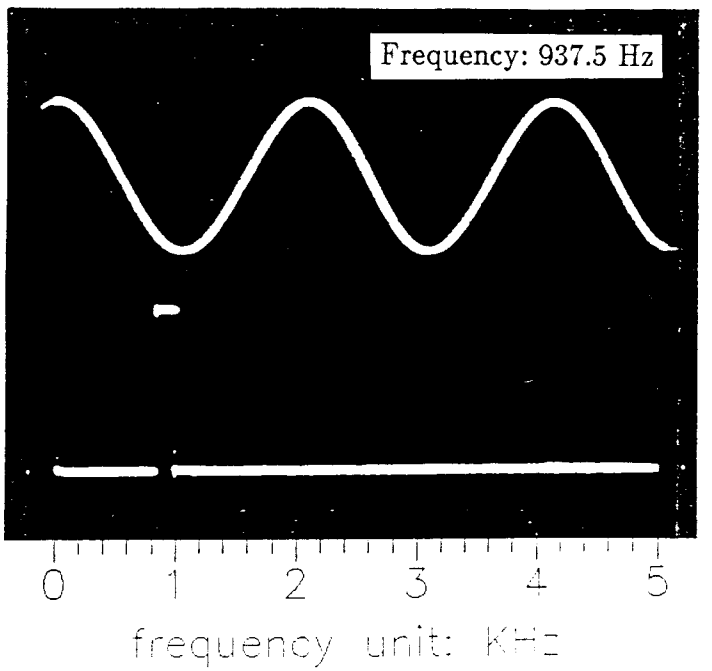


Figure 12. Power spectrum of a sine function.

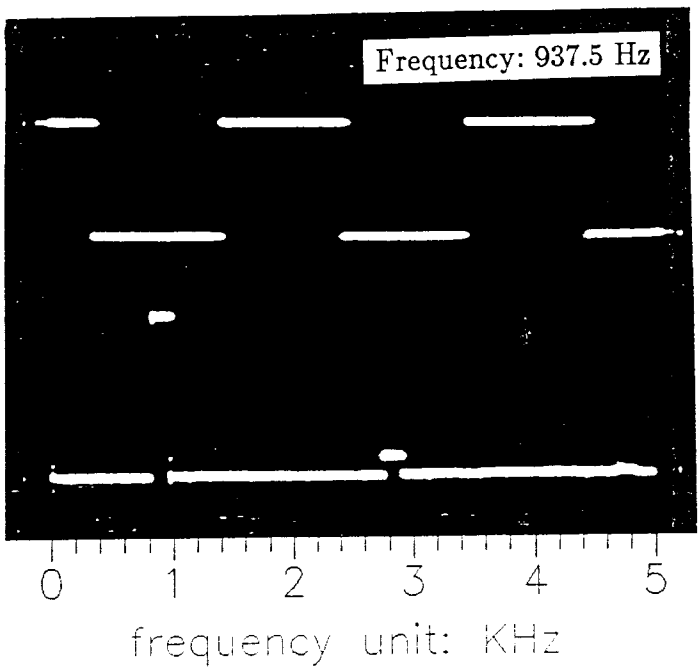


Figure 13. Power spectrum of a square function.