Design and Implementaion of a Real-time Digital Signal Processing Algorithm Instruction Developer (DSP-AID)

Ke-Kang Chin, Patrick J. Gries and Jafar Saniie

Deptartment of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago, IL 60616.

This paper, tutorial in nature, presents the design and hardware implementation of a DSP algorithm instruction developer (DSP-AID) which integrates two popular numeric processors, namely the MC68000 and the 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. Real-time sample applications with experimental results are presented.

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). 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 demands a more thorough analysis of the design prior to the start of hardware development. 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.

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 ADC is far from being infinite. Also, the registers for storing the coefficients and the resulting products are eventually of finite length, resulting in a truncation of the values. All of these potential error issues must be analyzed to verify that the system remains stable and operates within an acceptable range. 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 frontend analog-to-digital converter and a backend digital-to-analog converter and can be interfaced to any personal computer using an RS-232 port for

host control. 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 1.). The MC68000 facilitates set-up and algorithm development and provides the mechanism for user interaction [1]. The TMS32020 performs realtime processing tasks efficiently [2]. 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.

Design Philosophy and Procedural Operation

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.

In general, DSP circuits are integrated into a system using one of three primary configurations: (1) stand-alone, (2) slave or (3) multiprocessor. Usually, the most economical configuration is the stand-alone, however, the software effort involved with developing an interactive user interface can be very time consuming.

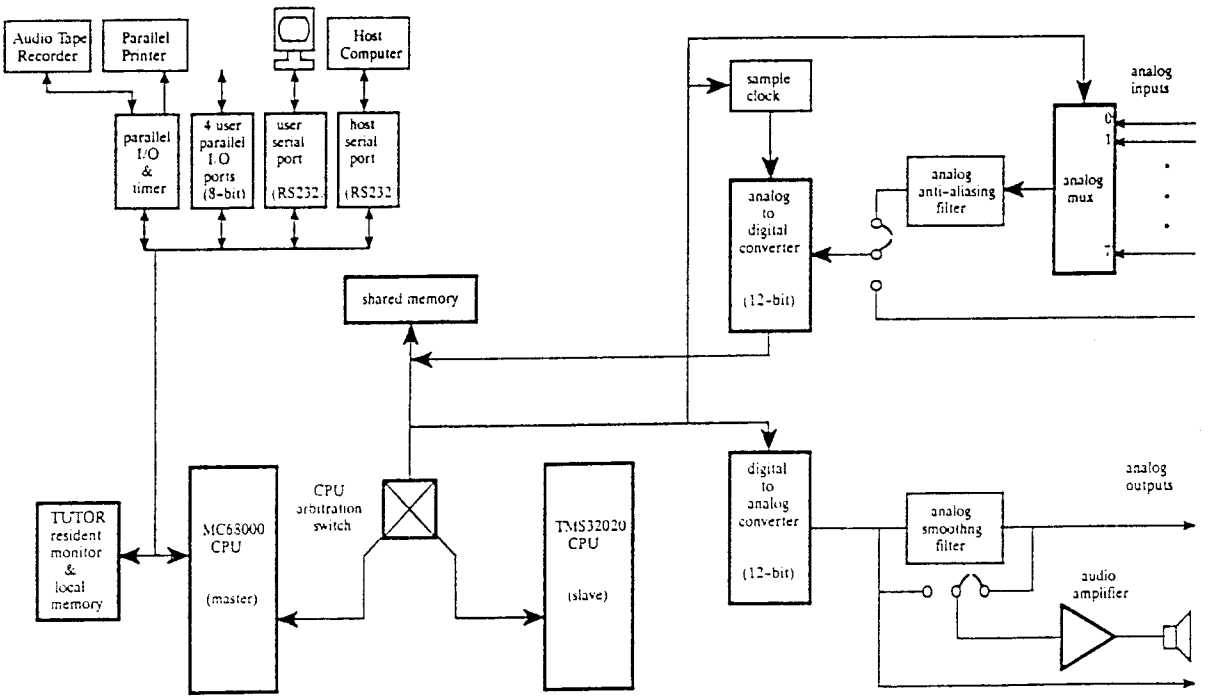

Figure 1. Functional block diagram of the DSP-AID. The DSP application code is downloaded from the host by the MC68000 to the shared memory via the CPU arbitration switch. The MC68000 commands the CPU arbitration switch to allow the TMS32020 access to the DSP application code. The TMS32020 proceeds as follows: (a) the analog input is sampled by the ADC, (b) the data is processed internally by the TMS32020 according to the algorithm and (c) the processed data is output to the DAC for subsequent conversion to analog form.

This problem can be overcome using a slave configuration which utilizes a general purpose microprocessor for user interaction. A multi-processor DSP configuration is important for high performance parallel processing, although, this configuration may not simplify the procedures of algorithm development. The DSP-AID system presented here uses a slave configuration with an MC68000 as the host processor and a TMS32020 as the slave. This system is designed such that both the MC68000 and the TMS32020 have access to the DSP resources. A debugger monitor from Motorola called TUTOR is a unique feature of this system which facilitates user interaction. Functions such as assembly/disassembly (MC68000), program and data entry, print, tape storage and interfacing to host computer are provided by TUTOR.

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 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 superceded by the MC68000.

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 TUTOR debugger monitor. Upon completion of the download process the TMS32020 is given a reset so that it will start executing the new code from the beginning. The MC68000 then releases the TMS32020 into the active mode. The TMS32020 will then be executing the new application algorithm. Only one processor at a time may access the DSP resources. Although this design is intended to take advantage of each processors strengths, 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 *setting* the memory mapped arbitration switch which puts the TMS32020 into an inactive standby mode. Upon acknowledging that the TMS32020 has released control of the DSP resources the arbitration switch allows the MC68000 access to the DSP resources. The MC68000 returns control of the DSP resources to the TMS32020 by *clearing* the arbitration switch.

The DSP-AID system (Figure 1.) consists of the following functional units, collectively referred to as DSP resources: (1) analog input buffer and multiplexer coupled with an anti-aliasing filter, (2) analog-to-digital converter, (3) digital-to-analog converter and (4) analog output filter and driver.

Analog Input Circuitry. The analog input section consists of an 8:1 analog multiplexer and an anti-aliasing filter. An analog multiplexer is an economical way to increase signal processing capabilities. An alternaltive approach would be to add multiple analog-to-digital converters which are more costly than analog multiplexers. Anti-aliasing filters are generally recommended and provide signal buffering. The frontend of any DSP system requires a relatively simple analog filter to remove the higher unwanted components of the signal. Usually, a 4th order active low pass filter will suffice. A fundamental requirement of discrete time-sampled systems 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 10 KHz, the maximum allowed frequency which can be theoretically reconstructed is 5 KHz. A 6 KHz frequency component will alias itself back and appear as a 4 KHz component. 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 insignificant phase distortion and has a modest rolloff.

Analog-to-Digital Converter. The analog-to-digital converter (ADC) in the design of the DSP-AID has a conversion time of 5μs which provides a theoretical sampling rate up to 200 KHz. For real-time

applications the sample rate limit is actually determined by the processing algorithm since the data must be numerically processed between each sample time. The TMS32020 has a 200ns instruction cycle time. With the ADC sampling at full speed the TMS32020 would only be allowed to execute 25 instructions which is enough to implement only a relatively low order filter. On the other hand, it is possible to perform quasi real-time applications wherein the ADC samples the input data in a burst mode and then processes the entire block of data. This is essentially how a digital spectrum analyzer is implemented. An accurate conversion requires a stable and non-changing input. This is accomplished using a sample-and-hold (S/H) 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. Two important issues to consider 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 and 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.

A software controlled 16-bit counter provides a variable rate clock 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.

Digital-to-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 in 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.

Analog Output Circuitry. 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.

Real-time Applications

The DSP-AID is capable of performing real-time or quasi real-time applications. Applications presented here are digital filtering and audio frequency spectral analysis.

Digital Filtering. With the DSP-AID system, the filtering process can be easily implemented digitally. Next, two common methods of designing digital filter will be discussed. These methods are Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) digital filter design.

The FIR filter is designed using a truncated Fourier series expansion approximation of the ideal frequency response of the filter. To compensate for the effect of the truncation of the Fourier series, sometimes, a "window" function is applied, which overcome the Gibbs phenomenon. The example presented

here is a length-80 FIR lowpass filter with a 10 KHz sampling frequency and a 3 KHz bandwidth. Coefficients of this filter are generated by using a Fourier series expansion and then modified by a Hanning window function. Figure 2 shows the real-time input and output signals of this FIR filter. The input signal (upper trace) is a sinusoidal chirp signal with a frequency sweeping from 1 KHz to 5 KHz. The output signal (lower trace) is cut off at 3 KHz. Note that the slight attenuation in the passband is due to the ouput lowpass reconstruction filter. Better performance can be obtained by increasing the length of the filter, however, the corresponding high computation time would limit the sampling rate. In fact, this would lower the frequency limit on the applicable input signal.

An IIR filter is another method of realizing real-time digital filters. IIR filters are computationally more efficient than the FIR counterparts. However, greater care must be exercised upon implementation to maintain stability due to the fact that the output is fed back into the processing structure. In practice, an IIR filter is designed using an analog filter as the desired prototype and then mapping the parameters from the analog domain to the digital domain. The IIR example here is a two-stage cascaded fourth order Chebyshev lowpass filter with a 10 KHz sampling frequency and a 3 KHz bandwidth. Figure 3 shows the input and output signals. The input signal (upper trace) is also a sinusoidal chirp which has the same characteristics as the input used in the FIR filter example. The output signal (lower trace) has a cutoff frequency at about 3 KHz. Note that the minor variations in the amplitude of the output signal are due to the ripple existing in the passband of the Chebyshev IIR filter.

Spectrum Analyzer. 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, so the frequency resolution is 156 Hz. Note that. the frequency resolution is determined by two factors, sampling frequency and number of samples. Since amplitude spectrum is always symmetrical for real function, we only need to show half of the spectrum, in this case, 32 frequency components.

The following steps are taken to implement the spectrum analyzer: (1) set up a lookup table of precalculated cosine & sine values, (2) capture 64 points input samples using the analog-to-digital converter, (3) apply DFT algorithm to calculate the first 32-point power spectrum and store them in memory, (4) display the 32-point power spectrum, and (5) repeat from step 2. The result shown in Figure 4 confirms the fact that the Fourier transform of a sine function is an impulse function. Furthermore, the Fourier transform of the square wave shown in Figure 5 only has odd harmonics (first, third and fifth) as expected.

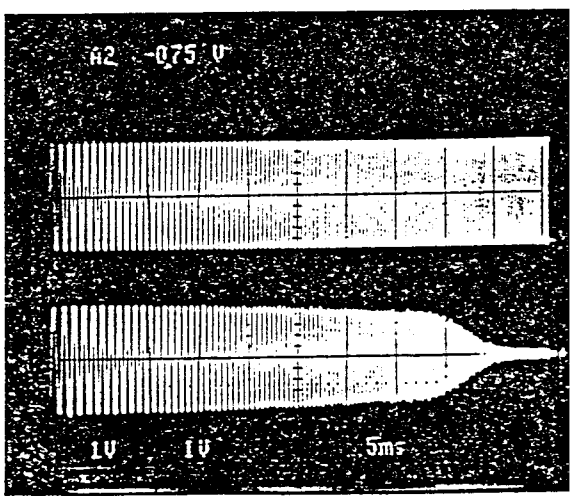

Figure 2. Real-time input and output signals of a length-80 lowpass digital FIR filter.

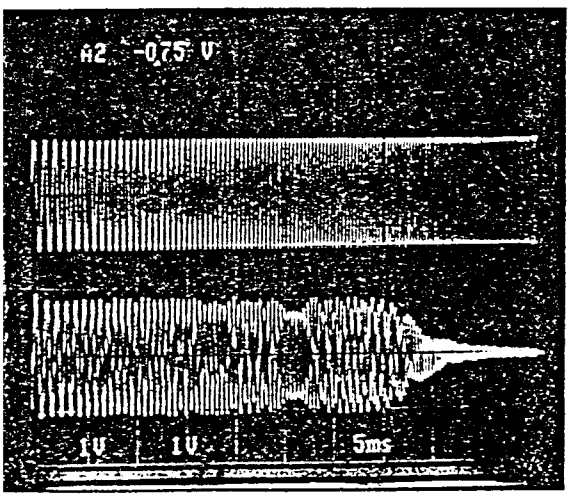

Figure 3. Real-time input and output signals of a two-stage cascaded fourth order lowpass digital IIR filter.

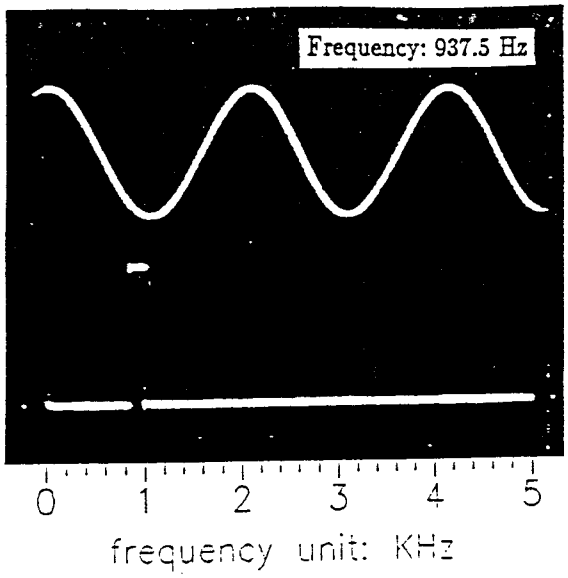

Figure 4. Power spectrum of a sine function.

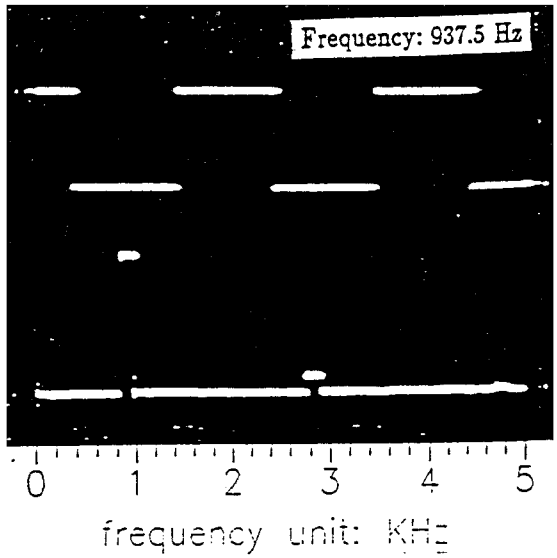

Figure 5. Power spectrum of a square function.

Summary

In this paper we presented an economical system (DSP-AID) for developing DSP algorithms, which is also suitable for use as an educational tool. This versatile and interactive system can be interfaced to any personal computer and is designed for learning and implementing digital signal processing algorithms 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 actual results. The ease of use of the DSP-AID allows the user to test and evaluate DSP algorithms efficiently. Targeted application areas for this unit are adaptive signal processing, biomedical signal processing, speech processing, servo control, and robotics.

References

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

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

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

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

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

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