1

CANADIAN THESES ON MICROFICHE

I.S.B.N.

THESES CANADIENNES SUR MICROFICHE

National Library of Canada
Collections Development Branch

Canadian Theses on
Microfiche Service

Ottawa, Canada
K1A 0N4

Bibliothèque nationale du Canada
Direction du développement des collections

Service des thèses canadiennes
sur microfiche

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us a poor photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de mauvaise qualité.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

National Library
of CanadaBibliothèque nationale
du Canada

Canadian Theses Division

Division des thèses canadiennes

Ottawa, Canada
K1A 0N4

59997

PERMISSION TO MICROFILM — AUTORISATION DE MICROFILMER

• Please print or type — Écrire en lettres moulées ou dactylographier

Full Name of Author — Nom complet de l'auteur

CARL Michael Anderson

Date of Birth — Date de naissance

JAN. 2 1956

Country of Birth — Lieu de naissance

CANADA

Permanent Address — Résidence fixe

243 GLEBE AVE
OTTAWA, ONTARIO
K1S 2C8
CANADA

Title of Thesis — Titre de la thèse

"A SLICED SONAR SIGNAL PROCESSOR"

University — Université

CARLETON University

Degree for which thesis was presented — Grade pour lequel cette thèse fut présentée

Master's Engineering

Year this degree conferred — Année d'obtention de ce grade

1983

Name of Supervisor — Nom du directeur de thèse

Prof. M. A. Copeland

Permission is hereby granted to the NATIONAL LIBRARY OF
CANADA to microfilm this thesis and to lend or sell copies of
the film.The author reserves other publication rights, and neither the
thesis nor extensive extracts from it may be printed or other-
wise reproduced without the author's written permission.L'autorisation est, par la présente, accordée à la BIBLIOTHÈ-
QUE NATIONALE DU CANADA de microfilmer cette thèse et de
prêter ou de vendre des exemplaires du film.L'auteur se réserve les autres droits de publication; ni la thèse
ni de longs extraits de celle-ci ne doivent être imprimés ou
autrement reproduits sans l'autorisation écrite de l'auteur.

Date

Sept. 15/83

Signature

Carl M. Anderson

$$\tau_d = \frac{s}{v} = \frac{d \sin \theta}{v}$$

where τ_d = time delay between inphase wavefronts
 $s = d \sin \theta$ from geometric considerations
 v = velocity of sound in water

With this formula we can add inphase wavefronts and increase the signal gain. The wavefront which is now touching H4, hydrophone 4, also passed H3 some τ_d time previous and H2 some $2\tau_d$ previous etc. If we add the sample from H1 taken at $3\tau_d$ time intervals ago to that from H2 at $2\tau_d$ intervals previous to that from H3 at $1\tau_d$ delay to the current sample from H4 we will have four times the signal strength since all samples were in phase. Basically this is how "phasing" an array, by choosing particular time samples to sum, increases the gain for a particular look direction (θ). Different look directions have different τ_d values which means different sensor sample sums are needed. In the method just illustrated full sample histories are kept and samples from selected previous time intervals are chosen to phase the array correctly. Another way to do this, without a full sensor history requirement is to use partial beam sums. The partial beam technique uses a number of partial beams continuously being updated as each array of sensor values comes in. After all array sensors are processed one beam sum will be complete ready for further processing.

The undersigned recommend to the Faculty of Graduate
Studies and Research acceptance of the thesis

"A Sliced SONAR Signal Processor"

submitted by Carl M. Anderson, B.Sc.

in partial fulfilment of the requirements for
the degree of Master of Engineering.

Prof. M.A. Copeland
Thesis Supervisor

Prof. A.R. Boothroyd
Chairman
Department of Electronics

Carleton University
September 19, 1983

ABSTRACT

This thesis describes a new signal processor architecture for passive SONAR systems. Using this architecture a set of microprocessor driven modules replace the CPU more normally used for signal processing of the SONAR data. Each module is a "slice" of the signal processor handling all the spectrum analysis for a particular look direction of the total SONAR system. The novel part of this approach is not the idea of distributed processing, but rather the idea of distributed processing by TASK rather than the more common FUNCTION oriented approach. FUNCTION oriented distribution commonly involves inline processing where a distributed processor is dependent on the results of previously processed data. Distributed processing by TASK means each distributed processor is independent and non-interactive, a very desirable characteristic from a system design viewpoint.

The viability of this concept is demonstrated for an Acoustic Data Processor in a SONAR system processing a line array of hydrophones. In this case a task of the problem is beam formation (look direction) in the system. The system is "sliced" by beam and demonstrated using simulated SONAR data produced by a custom test waveform generator.

ACKNOWLEDGEMENTS

I would like to express my gratitude to the following people and organizations for their assistance throughout the preparation of this thesis: to my advisor Professor Miles Copeland for guidance and moral support; to Computing Devices for supplying the topic; and to Gary Carruthers, Dave Guertin, Ian Hogg, Terry Kitzul and Zyg Wojciechowski without whose efforts this thesis would not have been possible.

Finally I would like to thank Christine Hoffmann for her moral and mental support throughout this project.

TABLE OF CONTENTS

	PAGE
CHAPTER 1 SOUND IN THE SEA	1
1.0 Introduction	1
1.1 Background	3
1.2 Scope	7
CHAPTER 2 A TYPICAL PASSIVE SONAR SYSTEM	8
2.0 Introduction	8
2.1 The SONAR system	8
2.2 The Hydrophone Array	8
2.3 The Acoustic Data Processor	10
2.3.1 Beamforming	11
2.3.2 Bandshifting	14
2.4 The Display and Control Console	18
2.5 Central Acoustic Data Processor Limitations	21
2.6 The "Slicing" Concept	23
2.7 Summary	26
CHAPTER 3 A "SLICED" ACOUSTIC DATA PROCESSOR	28
3.0 Introduction	28
3.1 System Integration	28
3.2 Slice Architecture	29
3.2.1 Overview	29
3.2.2 The Front End Processor (FEP)	31
3.2.2.1 Input FIFO buffer	32
3.2.2.2 Interpolation Filtering	35
3.2.2.3 Beamforming	35
3.2.2.4 Bandshifting	36
3.2.2.5 Decimation Filtering	39
3.2.3 The Master and Peripheral Processor	43
3.3 Slice Firmware	43
3.3.1 Control Structure	43
3.3.2 Standard Running Mode	44
3.4 Design Aspects of Interest	47
3.4.1 Three-Ported, Prioritized, Dynamic RAM Memory	47
3.4.1.1 Design Considerations	47
3.4.1.2 Architecture	50
3.4.1.3 Priority Encoder	50
3.4.1.4 Memory Ports	52

3.4.1.5	Dynamic RAM	54
3.4.2	Slave Microprocessor as a Spectrum Analyzer	54
3.5	Summary	55
CHAPTER 4	SPECIAL TEST EQUIPMENT	56
4.0	Introduction	56
4.1	Design Objective	56
4.2	Intelligent Interface	56
4.3	Test Waveform Generator (TWG)	59
4.3.1	Performance Description	59
4.3.2	Architecture	60
4.3.3	Noise Generation	64
4.4	Summary	66
CHAPTER 5	DEMONSTRATED OPERATION	68
5.0	Introduction	68
5.1	The Special Test Equipment (STE)	68
5.1.1	Monotone Output	70
5.1.2	Multiple Tone Summation	70
5.1.3	Relative Phase Delay	75
5.1.4	Noise	75
5.1.5	Signal Amplitude Variation	75
5.2	Sample Slice Output	80
5.3	Summary	80
CHAPTER 6	CONCLUSIONS	85
6.0	Summary	85
6.1	Conclusions	86
6.2	Suggestions For Further Work	87
6.3	Specific Contributions	87
REFERENCES		89

LIST OF ILLUSTRATIONS

	PAGE
Figure 1.1: Attenuation Coefficient () in Seawater as a Function of Frequency and Temperature	2
Figure 1.2: Power Spectrum of Broadband Noise	2
Figure 1.3: Power Spectrum of Tonal Noise	5
Figure 1.4: Power Spectrum of Noise Radiated by a Vessel	5
Figure 2.1: Noise Measurement Range at the Atlantic Undersea Test and Evaluation Centre (AUTEC)	9
Figure 2.2: Illustration of Planar Wavefront Impinging on Hydrophone Array at Angle	9
Figure 2.3: Illustration of Frequency AZimuth (FRAZ) Output Display	19
Figure 2.4: Illustration of Bearing Intensity (BI) Output Display	19
Figure 2.5: Sound Spectrum (LOFARgram) of a Surface Ship Travelling at 11 Knots	20
Figure 2.6a&b: Distributed Processing Applied to a Signal Processor Handling 8 Beams, a) by Function and b) by Task (sliced)	24
Figure 3.1: Major Subsystems of a Signal Processing Module Slice	30
Figure 3.2: Block Diagram of the Front End Processor on a Slice	33
Figure 3.2.1: Input FIFO buffer for Array Data Capture	34
Figure 3.2.2: Data Paths for Interpolation Filtering	34
Figure 3.2.3: Data Paths for Beamforming Operation	37
Figure 3.2.4: Data Paths for Bandshifting Operation	37

Figure 3.2.5:	Decimation Filtering Data Paths	41
Figure 3.3:	Software Control Hierarchy for Slice	41
Figure 3.4:	Example Frequency Spectrum Showing Broadband Noise and Tone	46
Figure 3.5:	Noise Normalized Form of Spectrum in Figure 3.4	46
Figure 3.6:	Example of Threshold Setting for Bit Reduction	48
Figure 3.7:	Bit Reduced Form of Spectrum Shown in Figure 3.6	48
Figure 3.8:	Elements of the Slice Dynamic Memory Store	51
Figure 3.9:	Priority Encoder for Slice Dynamic Memory Store	51
Figure 3.10:	Typical Porting Style of Dynamic Memory Store	53
Figure 3.11:	Radix-2 Butterflies for Sandy-Tukey and Bruune FFT Algorithms	53
Figure 4.1:	Submodules of the Special Test Equipment	57
Figure 4.2:	Block Diagram of Special Test Equipment: Intelligent Interface	57
Figure 4.3:	Block Diagram of Special Test Equipment: Test Waveform Generator	62
Figure 4.4:	Serial to Parallel Clocking Option for Pseudo-Random Noise Generator	65
Figure 4.5:	Block Diagram of the Noise Generator for the Test Waveform Generator	67
Figure 5.1:	Measurement Set-up Used to Measure Test Waveform Generator Output	69
Figure 5.2:	12 Bit DAC Wired for 8 Bit Operation	71
Figure 5.3:	Photo of 100 Hz Tone Output	71

Figure 5.4:	Frequency Spectrum of 100 Hz Tone Showing Fundamental and Harmonics	72
Figure 5.5:	Frequency Spectrum of 100 Hz Tone at Minimum Amplitude Showing Increased Harmonic Content (input now square wave)	73
Figure 5.6:	Photo of Square Wave Obtained by Loading Seven Tones in Test Waveform Generator	74
Figure 5.7:	Frequency Spectrum of Tones Summed to get Time Domain Square Wave Shown in Figure 5.6	76
Figure 5.8:	Photo of 32 Square Waves i.e. Each Sensor's Tones Delayed Relative to the Following Sensor (shows phasing ability)	77
Figure 5.9:	Photo of Noise Signal Generated	77
Figure 5.10:	Frequency Spectrum of Generated Noise	78
Figure 5.11:	Maximum Tone Level and Minimum Tone Level	79
Figure 5.12:	Frequency Spectrum of Test Signal sets for Slice Operation Check	81
Figure 5.13:	Slice LOFAR Output With Beam Steered to 0 Degrees, inphase input	82
Figure 5.14:	Slice LOFAR Output With Beam Steered to 0 Degrees, input out of phase	83

LIST OF SYMBOLS AND ABBREVIATIONS

AUTEC	Atlantic Undersea Test and Evaluation Centre
BI	Bearing Intensity
CAS	Column Address Strobe
CPU	Central Processing Unit
CRT	Cathode Ray Tube
DAC	Digital to Analogue Converter
DEMON	DEModulated Noise
EPROM	Eraseable Programmable Read Only Memory
FEP	Front End Processor
FFT	Fast Fourier Transform
FIFO	First In First Out
FIR	Finite Impulse Response
FRAZ	FRrequency AZimuth
ISR	Input Sample Rate
LOFAR	Low Frequency Audio Record
MAC	Multiplier ACcumulator
MHz	MegaHertz
MP	Master Processor
MRQ	Memory Request
PP	Peripheral Processor
RAM	Random Access Memory
RABIN	Row Address Strobe IN
RFSH	ReFreSH in
SCC	Serial Communication Controller

SNR	Signal to Noise Ratio
SONAR	SOund Navigation and Ranging
SPM	Signal Processing Module
STE	Special Test Equipment
TWG	Test Waveform Generator
VLSI	Very Large Scale Integration

CHAPTER 1

SOUND IN THE SEA

1.0 Introduction

Using seawater as a transmission medium is a difficult proposition. Communication transmissions at commonly used radio frequencies are subject to severe attenuation. The attenuation problem only becomes worse at light frequencies. From the published data of Figure 1.1 it can be seen that attenuation in seawater is a non-linear function of frequency. Standard communication systems are clearly not useful when a 100 kilohertz transmission is attenuated 25 dB in 1000 yards, even a 20 kilohertz transmission is attenuated to half power (3 dB) in 1000 yards. Seawater attenuation is much higher than that of pure water and has been ascribed to three processes:

1. Shear viscosity - the transmitted frequency shears adjacent planes of water molecules;
2. Volume viscosity - this source of absorption is caused by the time lag for required water molecules to fill "holes" in the adjacent material; and
3. Ionic relaxation of $MgSO_4$ molecules into the seawater.

Only frequencies in the lower end of the sound spectrum have low enough attenuation coefficients to penetrate the

Figure 1.1: Attenuation Coefficient (α) in Seawater as a Function of Frequency and Temperature ⁽¹⁾

Figure 1.2: Power spectrum of Broadband Noise

seawater medium for a reasonable distance. For example a one kilohertz signal is only attenuated 3 dB in 50,000 yards. It is not surprising then that sound has found an important place as a communication and exploration medium in seawater. The science of sound transmission in seawater, dating back at least as far as Leonardo da Vinci, is now known as SONAR, an acronym for SOUNd NAVigation and Ranging.

1.1 Background

There are two major types of SONAR systems implimented:

1) Active SONAR - a specific frequency burst (a "ping") is projected out from a transponder. Any reflected signal, at that frequency, is compared to the original signal amplitude and phase. The amplitude variation gives an indication of the range of the object reflecting the signal and the phase change indicates the speed. The bearing of the object is obtained from the phase angle for which the receiver is tuned.

2) Passive SONAR - only unidirectional transmission is used in these systems. No signal is generated, so, instead of listening for a specific returned tone the noise radiated by an operating vessel is listened for. The generated noise is characteristic of a vessel class and can be used to identify it. As with the active system the bearing of the vessel is obtained from the phase angle the receiver is

tuned for.

Our main interest is in passive SONAR systems. These systems are designed to pick-up and display the radiated noise of an operating vessel. Ships and submarines produce readily detectable sources of noise due to the rotating machinery (such as compressors and engines) carried on board and from the propeller rotation in the water. This self-generated noise can be buried in more than 30 dB of random noise from the seawater environment. The signal processing done by the system is designed to distinguish these noise sources from the background of ambient noise in the observed area. Waves, wind and many other sources combine to generate the ambient noise of a particular local.

The radiated noise spectrum of a vessel is composed of two different spectra:

- 1) Broadband noise with a power level which is a continuous function of frequency (Figure 1.2); and
- 2) Tonal noise which is present only at certain frequencies in the spectrum (Figure 1.3).

Typical radiated noise from a vessel consists of a mixture of these two types of noise over a large range as illustrated in Figure 1.4.

The broadband and tonal noise spectrums are generated by different components in a ship. Machinery such as

Figure 1.3: Power Spectrum of Tonal Noise

Figure 1.4: Power Spectrum of Noise Radiated by a Vessel

rotating out-of-round shafts, motor armatures, turbine blades or reciprocating piston engines all produce a tonally dominant spectrum. Cavitation and turbulence in fluid carrying lines or pumps and mechanical friction in bearings or journals produce a continuous spectrum output unless a structural member resonates, in which case tonal components also occur. The combination of these sources of noise is a composite spectra as indicated in Figure 1.4.

There is another major source of noise from a marine vessel. This is propeller noise, which is different from machinery generated noise. This is due to propeller noise being generated outside the hull whereas machinery noise reaches the water through the hull. Cavitation is the principal source of surface vessel propeller noise, whereas submarines can decrease cavitation noise by diving to a lower level. The low-pressure area located along the tip of the rotating propeller physically ruptures the water and minute cavities (bubbles) form. The cavitation produced bubbles collapse emitting a small burst of sound. The generated noise spectrum is continuous as at any given moment a large number of random bursts are caused by the collapsing bubbles. Each rotation of a propeller blade creates another low pressure area and hence more cavitation bubbles. This causes the continuous spectrum of the noise

to be amplitude modulated at a frequency equal to the shaft rotation frequency multiplied by the number of blades on the propeller.

After processing the received signals the passive SONAR system displays the results of its spectrum analysis to a human operator for comparison against known spectra of various classes of vessels. The processing involved with obtaining the above spectra can be followed in chapter two which deals with a typical SONAR system.

1.2 Scope

Chapter two describes a typical passive SONAR system instituted with a central CPU as the acoustic data processor. An example outlines the problems with this design through a calculation of the signal processing rates required. The concept of a "sliced" signal processing unit is introduced. In chapter three the hardware and software architecture of the "slice" is presented. Chapter four introduces the Special Test Equipment which generates the simulated SONAR data through its Test Waveform Generator (TWG). Demonstrated performance of both the TWG and the slice are presented in chapter five.

CHAPTER 2

A TYPICAL PASSIVE SONAR SYSTEM

2.0 Introduction

In this chapter a typical passive SONAR system is described. The signal processing carried out on the recieved signals is discussed, as are limitations on a single central processor system. Finally a new distributed processing scheme , "slicing", is introduced.

2.1 The Sonar System

Figure 2.1 shows a passive SONAR system used for tracking and for radiated noise measurement. Systems of this type can be blocked into three functional subsystems:

- 1) The hydrophone array; 2) the acoustic data processor; and
- 3) the display and control console.

2.2 The Hydrophone Array

An array of hydrophones is used to provide increased detected signal gain through multiple sensor inputs and directionality through phased array tuning techniques. Sound travels as a pressure wave in an incompressible medium. Each hydrophone is a transducer able to convert

Figure 2.1: Noise Measurement Range at the Atlantic Undersea Test and Evaluation Centre (AUTEC) (1)

Figure 2.2: Illustration of Planar Wavefront Impinging on Hydrophone Array at Angle

pressure into a proportional voltage. The analogue output of the transducer is low-pass filtered and then digitized. The low-pass filtering prevents frequency aliasing (interpreting a higher frequency effect as that of the frequency of interest) when the waveform is sampled for the A to D conversion. The A to D conversion rate is chosen, as per the Nyquist theorem, at greater than two times the highest input frequency. The number of bits of quantization used depends on the dynamic range requirements of the system. The quantization and digital filtering operations have an inherent noise associated with them and have to be specified at levels below the minimum detectable signal level required or they will mask the signals they were designed to show. As a final operation the digitized samples from each sensor are time multiplexed onto a data bus to the signal processing unit.

2.3 The Acoustic Data Processor

The acoustic data processor is typically a single CPU with a large, high bandwidth memory attached. It performs all the beamforming, bandshifting and subsampling operations before doing a Fast Fourier Transform (FFT) on the results to produce frequency spectrums of the hydrophone signals.

To get a better idea of the amount of processing required each operation is explained below.

2.3.1 Beamforming

A typical system divides the area of interest into a number of regions of "x" degrees angular coverage. The location of a vessel's signature in one of these regions (a beam) is the method used to determine the direction of the vessel. The beamforming operation forms these "directional" beams from sums of array sensor samples. They are directional in that a particular beam will have its largest signal gain for input signals in phase with it. The process is known as time delay beamforming since sensor samples from previous time intervals are combined with a current sample. The choice of the correct time delayed sensor values "phases" the array since any signals in phase with the delay are cumulative and have a higher gain than those out of phase with the array. In Figure 2.2 a plane wave disturbance (a sound wave from a source infinitely far away) is shown impinging on an array of hydrophones. The time delay between inphase wavefronts is determined from a simple geometric formula

$$\tau_d = \frac{s}{v} = \frac{d \sin \theta}{v}$$

where τ_d = time delay between inphase wavefronts
 $s = d \sin \theta$ from geometric considerations
 v = velocity of sound in water

With this formula we can add inphase wavefronts and increase the signal gain. The wavefront which is now touching H4, hydrophone 4, also passed H3 some τ_d time previous and H2 some $2\tau_d$ previous etc. If we add the sample from H1 taken at $3\tau_d$ time intervals ago to that from H2 at $2\tau_d$ intervals previous to that from H3 at $1\tau_d$ delay to the current sample from H4 we will have four times the signal strength since all samples were in phase. Basically this is how "phasing" an array, by choosing particular time samples to sum, increases the gain for a particular look direction (θ). Different look directions have different τ_d values which means different sensor sample sums are needed. In the method just illustrated full sample histories are kept and samples from selected previous time intervals are chosen to phase the array correctly. Another way to do this, without a full sensor history requirement is to use partial beam sums. The partial beam technique uses a number of partial beams continuously being updated as each array of sensor values comes in. After all array sensors are processed one beam sum will be complete ready for further processing.

Using this technique a current sensor sample is added to a partial beam sum that will not be complete until N time samples later, i.e. to this beam sum you have effectively added the N th time delayed value of the sensor. The formation of beams occurs at the input sensor sample rate and hence the large history of sensor samples is not required (instead you have a "history" of beams).

In practise the exact delay needed to phase the array is usually not an integral number of sample time intervals, i.e. τ_d is not equal to a discrete number of sample intervals. Data is only obtained at time intervals equal to the digitization sample rate so some method of estimating the intermediate sensor values must be used. Thus, if a sensor value at 2.5 sensor sample times is needed a value must be calculated somehow. A simple linear average may be used, but for greater precision an interpolation filter is used. The interpolation filter used is a Finite Impulse Response (FIR) type digital filter. The size of the filter (the number of coefficients) depends on the number of intermediate time values (the degree of interpolation) one is trying to resolve.

The interpolation operation can be thought of as padding zeros between input samples, to increase the apparent sample rate, and then low pass filtering to retrieve a sample value

at some intermediate time. Choosing different sets of coefficients for the FIR filter yields different sensor values for the different intermediate time values. To perform the FIR filtering a short history of each hydrophone's samples must be stored. For an eight point interpolation filter the last seven samples from each hydrophone must be kept.

To recap then, the actual first operation is an interpolation to estimate the sensor value at the required intermediate sample time. With this sensor value, time delay beam forming by partial sums occurs. Each sensor sample is added to the correct partial beam sum in a continuous cycle until the last sensor in the array. When this sensor sample has been added to a partial beam sum the sum is complete and one beam is ready for the bandshifting operation.

2.3.2 Bandshifting

The bandshifting operation translates a chosen center frequency (ω_0) to baseband by quadrature techniques. The shifted frequency band of interest is now located at a lower frequency range and hence can use a lower sample rate to represent it. A lower representative sample rate means some samples of the input time stream can be discarded

(subsampling) and a lower processing bandwidth is needed. Bandshifting by analog techniques involves mixing a complex vector, rotating at frequency ω_0 , with the input signal. Spectrums result at $(\omega_0 + \omega)$ and $(\omega_0 - \omega)$. The $(\omega_0 + \omega)$ component is discarded by low pass filtering the resulting signal. In a digital implementation the real and imaginary bandshifted spectra are formed by multiplying the input digital samples with synthesized cosine and sine waveforms rotating at a frequency ω_0 . The sine and cosine vectors can be synthesized by indexing through a look-up table of values. If the indexing is via an accumulator which is incremented at the input sample rate a simple formula determines the generated frequency.

$$\text{Frequency generated} = \text{ISR} \frac{\Delta\phi}{2\pi}$$

where ISR = input sample rate in hertz

$\Delta\phi$ = accumulator increment (in radians)

Used in this manner the accumulator increment is a phase increment to the generated sine or cosine vector.

To perform the bandshifting operation the completed beam sample is multiplied by one value of the synthesized sine vector to form a bandshifted imaginary time series sample. Similarly the real time series sample is formed by multiplying the beam sample by one value of the synthesized

cosine vector. Before the next beam sample is bandshifted the accumulator addressing the look-up tables is incremented by the chosen phase increment. In this way the look-up table is indexed through at a rate determined by ISR and generates the requisite rotating vectors.

After the bandshifting operation there are two time series (Real and Imaginary) to be processed. As in the analog case, the shifting operation must be followed by low pass filtering to eliminate unwanted high frequencies. The low frequency component of the shifted signal is over-sampled for the baseband frequency region and subsampling (decimation) is instituted to discard unneeded time samples and lower the effective sample rate for following processes. The final low pass filter may also be a FIR type provided the number of coefficients increases proportionately to the decimation occurring. This correlation comes about since the higher the decimation level the lower the bandwidth output and hence more aliasing (frequency foldback) is occurring. To avoid corrupting the signal band of interest the low pass filter stopband stopping level must be lowered as the passband decreases. Following all of the above operations, samples from one look direction at one frequency band of interest are ready for spectrum analysis. Spectrum analysis is the second source

of gain in the system, where the first was the gain from an array of sensors versus one sensor. The processed time series are used as input time samples for a Fast Fourier Transform (FFT) which is typically performed using a Cooley-Tukey radix-2 algorithm. The signal gain is relative to the input noise levels as the noise, which is assumed white, is evenly distributed across the frequency spectrum, while the signal remains at its input level in a particular frequency cell of the FFT result. In this manner an N cell FFT returns tonal components enhanced N times relative to the input noise level (since the noise has now $1/N$ its value in all the FFT cells). Since the signal is consistently present over many analysis intervals some further gain is obtained by integrating several FFT spectra before storing the spectrum. To recover the modulated frequency of the propeller there is some preprocessing necessary before an FFT is performed. The processing closely follows AM radio demodulation since the signal (propeller revolution rate) is impressed upon a carrier of broadband noise (due to cavitation). The demodulation requires input signal rectification before low pass filtering to recover the modulating frequency. This rectified low pass filtered time series is input to a FFT and the signal frequencies displayed on the operator

console.

2.4 The Display And Control Console

The function of a passive sonar system is to allow the operator to identify a vessel in the receiving field of the hydrophone array and give its relative bearing to the receiving system. The identification requires the processed information be displayed for the operator. Various displays of the frequency spectra are used to aid the operator. Three formats used are:

- 1) FRAZ (Frequency, Azimuth), see Figure 2.3;
- 2) BI (Bearing, Intensity), see Figure 2.4;
- 3) LOFAR (Frequency, Time), see Figure 2.5.

The FRAZ and BI displays are commonly displayed on a CRT. The Lofargram maybe either shown on a CRT for 20-30 minutes or (more commonly) produced as a continuous hardcopy output. The operator can tilt the hardcopy output slightly off of horizontal and the integration along the line of sight increases the operators ability to detect tonal tracks present in the output. It is interesting to note that trained observers easily pick out more tracks than the most sophisticated target algorithms can detect.

The display console also serves as the input mechanism

Figure 2.3: Illustration of Frequency Azimuth (FRAZ) Output Display

Figure 2.4: Illustration of Bearing Intensity (BI) Output Display

Figure 2.5: Sound Spectrum (LDFARgram) of a Surface Ship Travelling at 11 Knots

for the operator requests. The operator can request that new beams (different look directions) be displayed or that the frequency ranges which are being spectrum analyzed be changed. In this manner the operator can choose and display frequency bands which aid in the vessel classification. As mentioned above the relative bearing of a vessel is obtained from noting which displayed beam the noise signature is present in.

2.5 Central Acoustic Data Processor Limitations

The Central Processor concept suffers from two severe requirements: 1) High processor bandwidth; and 2) Large storage memory with a compatible high bandwidth to allow processor accesses. To illustrate these problems consider a hypothetical SONAR system which has an array of forty hydrophones, each sampled at two kilohertz. The system needs twenty beams with four different frequency bands analyzed in each beam.

The array data rate the CPU sees is 40 sensors times 2 kilohertz per sensor or an 80 kilohertz apparent data rate. To perform beamforming it must perform the eight interpolation multiplies for all twenty beams in the time interval between sensor samples i.e. it must be multiplying

at 12.8 megahertz. The bandshifting operation requires an additional two multiplies per band and the interpolation filter would require a minimum of eight multiplies for each of the real and imaginary time series per band per beam. The full multiply rate then is some 18 megahertz; not too easy to achieve. Of course, this is not allowing for any overhead on addressing operations or memory accesses which can easily take up 30 percent of the available time. The memory has to be fast enough to service the 18 megahertz multiply rate and also store up to 30 minutes of LOFAR data for each analysis band in each beam. This data store can be quite large. In our hypothetical case suppose each of the spectra are updated in ten second intervals and have two hundred cells of FFT data stored in memory. Even with packed, bit reduced (2 bits per frequency cell of the FFT) data this system would require nearly a megabyte of fast storage.

It should be apparent that the bottleneck in the system is the single CPU doing all signal processing operations for all beams. This processing bottleneck can be removed by changing to distributed processing. The distributed processing removes the load from the central CPU by spreading the signal processing tasks among various specific function modules. Applied to a signal processor for a line

array of hydrophones this distribution may involve a beamforming module followed by several bandshifting/subsampling modules. In a similar manner the spectrum analysis is split amongst a number of FFT processor modules before a final integration module collects the FFT results for bulk storage prior to display. This system of distributed processing can and does work¹. However: 1) the system design concept is unique to each configuration; 2) the system is awkward to implement; and 3) the system is subject to a disastrous hardware failure mode (a single module failure can cause system failure).

2.6 The "Slicing" Concept

The slicing concept is a different way of partitioning a signal processing task for distributed processing, i.e. for processing through submodules. Distributed processing is not a new concept, the method of distributing the processing is the new concept. A typical distributed process is distributed by FUNCTION. Referring to Figure 2.6a, a typical distribution for an eight beam signal processor, shows the processing split by function, first beamforming then filtering, analyzing, detecting and finally

1 For instance the ESE (Toronto) Teamed Architecture Signal Processor

Figure 2.6a&b: Distributed Processing Applied to a Signal Processor Handling 8 Beams, a) by Function and b) by Task (sliced)

storing all eight beams. At any stage in this process the next stage is dependant on the previous stage of processing.

The slicing concept involves distributing a process by TASK (not function). Thus each submodule is a complete signal processor for part of the required total task and is independent of other modules. Figure 2.6b shows how the slicing concept would be applied to the task of an eight beam signal processor mentioned above. The system is divided into eight identical, self-contained, non-interactive end to end entities which are referred to as slices. Each slice does all the processing required for a single beam, hence eight identical slices complete the system. The slices are downloaded with the required configuration software which allows each slice individual processing parameters. The slice approach has the following advantages:

- 1) Graceful degradation - If one slice of an N slice system fails the system has lost only $1/N$ of its processing power, and if required any other slice can be reprogrammed to take the place of the failed slice's place;
- 2) Software Simplicity - Software is developed for one slice and then just copied for all other slices in the system;
- 3) Simplified System Reconfiguration - New system

designs need not be unique. A system that requires more processing power merely adds more slices. A minimal configuration is one slice;

4) Low Manufacturing Cost - The slices in a system are all identical so the standard design/manufacturing costs associated with different boards in a system are avoided.

As shown in Figure 2.6b, the slicing concept can be applied to a system whose task is separable into beams such as an Acoustic Data Processor for a line array of hydrophones; i.e. the example discussed in the previous section. Using the slice partitioning scheme a single slice performs all the signal processing operations for one beam of the system. The benefits of this distributed processing approach are immediately apparent. The system would consist of twenty slices, one for each beam. The individual slice (processing only one beam) would need only one twentieth the multiply rate (one megahertz) and one twentieth the memory store (fifty kilobytes), both requirements easily met with current technology. A demonstration slice of a signal processor for a line array of hydrophones has been built; its description follows in chapter 3.

2.7 Summary

The three subsystems of a passive SONAR system: the hydrophone array, the Acoustic Data Processor and the display and control console, were discussed. The operation of the Acoustic Data Processor was covered in some detail.

To get a feel for the processing load on such a system an example was discussed. Limitations of a single central processor architecture used for the example were used as an introduction to "slicing". Slicing was presented as a new way of distributing the process load, by TASK versus the more common by function approach.

CHAPTER 3

A "SLICED" ACOUSTIC DATA PROCESSOR

3.0 Introduction

In this chapter the demonstration slice of a Signal Processing Module (composed of multiple slices) is described. The slice architecture is discussed and its component modules are explained. Some of the interesting aspects of the design are presented.

3.1 System Integration

From an operators point of view, a sliced acoustic data processor would appear no different than a single CPU type. As in a central CPU implementation data is still displayed from a number of beams. The only difference is that data for each beam is collected from a different slice instead of a different location in a single large store. Operator requests for beam redirection or a change in the frequency band being spectrum analyzed still require a software reconfiguration, though in the sliced case only a single slice need be interrupted during this operation. The sliced signal processor in such a system will be referred to as a Signal Processing Module. All slices are connected by a common bus to form the Signal Processing Module (SPM).

Data transactions on the SPM bus are controlled by a display processor interfacing to the bus. A typical slice of the SPM is discussed in the following section.

3.2) Slice Architecture

3.2.1 Overview

Each slice of a SPM is composed of three processors and a common memory bank (shared by two of the processors). These processors are:

- 1) The Front End Processor (FEP) - a specially designed, hardwired processor for beamforming, bandshifting and subsampling the input hydrophone time series data;
- 2) The Master Processor (MP) - a segmented 16 bit microprocessor (the Z8001) used for slice control and minimal data handling; and
- 3) The Peripheral Processor (PP) - a nonsegmented 16 bit microprocessor (the Z8002) used as a spectrum analyzer.

The major subsystems of the slice and their associated data paths are shown in Figure 3.1. Digitized hydrophone sensor samples are captured from the array data bus, which is common to all slices. They are beamformed, bandshifted and subsampled in the FEP. Output beam samples from the FEP are transferred to the time series store in main memory by way

Figure 3.1: Major Subsystems of a Signal Processing Module Slice

of the MP. These time series stores are used by the PP as input samples for the FFT algorithm. The results of the FFT, now in the frequency domain, are stored back in main memory in a temporary storage buffer. Results in the storage buffer are removed by the MP and stored in a display buffer area. Data from the display store is passed to an external display processor for display on the operator's console.

3.2.2 The Front End Processor (FEP)

The FEP is a hardwired, microsequenced processor designed to perform the bulk of the high speed, time critical, processing of incoming array data samples. It implements an eight point FIR filter for data interpolation and beamforms by partial beam sums. The completed beam samples are bandshifted to baseband (using sine and cosine vectors generated by the MP), and subsampled (decimated) during a final low-pass filter operation implemented as a variable coefficient FIR filter. The decimation rate chosen determines the number of coefficients used, as mentioned in the signal processing discussion for the typical sonar system. The FEP is reconfigurable by reloading one of three parameters:

- 1) The bandshift center frequency ω_0 , (a parameter used

by the MP to generate the sine and cosine values;

2) The decimation rate - a code used to determine the number of coefficients in the final lowpass filter; or

3) The phase delay RAM - a RAM used as a pointer for the correct time delays for beam forming.

By reloading these parameters the slice's displayed frequency bands, frequency bandwidth and look direction (beam displayed) are alterable.

A block diagram of the FEP is shown in Figure 3.2. The processor operations revolve around the Multiplier/ACcumulator (MAC) shown in the center of the figure. At first glance this diagram is rather complicated. Rather than try to explain the data flows from Figure 3.2, relevant portions of the figure will be reproduced when implemented FEP operations are being discussed.

3.2.2.1 Input FIFO Buffer

The input FIFO buffer shown in Figure 3.2.1 allows the FEP to be data driven, i.e. only when a full array worth of samples has been stored is the FEP required to process it. This allows the available time interval of one array's worth of data to be split into four operations: 1) Interpolation; 2) Partial Sum Beamforming; 3) Bandshifting; and 4)

Figure 3.2: Block Diagram of the Front End Processor on a Slice

Figure 3.2.1: Input FIFO for Array Data Capture

Figure 3.2.2: Data Paths for Interpolation Filtering

Decimation Filtering.

3.2.2.2 Interpolation Filtering

The interpolation filtering involves the data paths shown in Figure 3.2.2. The interpolation process, as was stated earlier, involves a partial history of sensor samples. The FEP uses an eight point FIR filter and hence requires the previous eight samples of a particular sensor. A counter, showing the current sensor number being interpolated, is used as an offset address to the eight deep sample history store. The sensor number in this manner indexes to the correct area in the RAM and a simple three bit counter summed with this offset allows the correct sensor values to be multiplied with the coefficients. The coefficients required are a function of the delay required. Hence some of the output from the phase delay ram is used to index the correct area of the coefficient store. After eight multiplications and accumulations, the interpolated data value is ready to be added to a partial beam sum.

3.2.2.3 Beamforming

The phase delay RAM, addressed by the sensor number (S#), holds more than the offset address for the

interpolation coefficient address control. The interpolation represents the fractional sample time interval required, so the additional information is the integral number of delays that a particular sensor value requires. This additional information is used as an address pointer to the partial beam sum, to which the current sensor value must be added so that the completed beam will be correctly phased. The beamforming operation shown in Figure 3.2.3, involves obtaining the current partial beam sum value, preloading it into the accumulator portion of the MAC and adding the just calculated interpolated sensor value to it. The updated sum is then returned to the same location in memory. This process continues sensor by sensor until the last sensor in the array is being processed. The last sensor is handled differently since the partial beam sum it is added to is complete and ready for further processing. The only difference in the as described beamforming operation is that the partial beam sum (now complete) is not returned to memory. Instead a zero value, ready for the next partial sum, is returned and the completed beam sample is returned to the input of the MAC ready for the bandshifting operation.

3.2.2.4 Bandshifting

Figure 3.2.3: Data Paths for Beamforming Operation

Figure 3.2.4: Data Paths for Bandshifting Operation

In the section describing a typical signal processor a look-up table of sine/cosine values was used to generate sine and cosine vectors rotating at the bandshift center frequency ω_0 . A register used to address the table was incremented, by a phase increment, to generate the next phase of the vectors. In the FEP a PROM is used to hold the table of sine values, while cosine values are obtained by addressing with a 90 degree phase shift. The sine PROM is addressed by phase values which are generated by the MP i.e. the MP does the phase accumulations for the FEP. The FIFO buffer shown in Figure 3.2.4 allows the MP to stack up to 32 sine and cosine incremental address values ready for the bandshifting operation. The FIFO buffer interrupts the MP when it is empty. The bandshifting operation involves multiplying the beam samples by a complex vector rotating at the frequency of interest. This is accomplished by multiplying a complete beam sample by one sine value to obtain the imaginary time series sample and then by a cosine value to obtain the real time series sample. The real time series sample and the imaginary time series sample, obtained after bandshifting to baseband, are the required complex time series for the FFT algorithm. Before they can be used, a final low-pass filtering operation is required to

eliminate the high frequency bandshifted spectrum ($\omega_0 + \omega$) formed at the same time as the baseband spectrum ($\omega_0 - \omega$).

3.2.2.5 Decimation Filtering

During the final low-pass filtering operation, the complex time series is decimated (subsampled) to reduce the sample rate consistent with Nyquist requirements for the now lower bandwidth signal. The sample rate reduction allows microprocessors to handle further slice signal processing but also requires a special FIR filter be implemented by the FEP. The number of coefficients used in the FIR filter is increased according to the degree of subsampling taking place. As discussed earlier the narrower the passband the more stringent is the stopband requirement and hence the more coefficients needed by the FIR filter. The FIR filter is also implemented using a partial sum method. To understand why a partial sum method was used a slight digression is necessary.

One possible way of subsampling a filtered time series is to only take the samples that are needed, say every Nth sample, and ignore the other filtered output values. This method is wasteful of computing resources since the CPU has spent the time to filter all output points and only used

every Nth output. For a small FIR filter this may be acceptable but certainly not when using 512 or more coefficients such as the FEP decimation FIR filter might require. Examination of the FIR algorithm suggests another method of obtaining the required outputs and only those outputs. If the filter output is examined it is possible to determine which coefficients were used for each input sample which contributed to the required Nth output sample. The partial sum method of filtering involves multiplying each input sample only with those coefficients it would have contributed to the required Nth output sample and storing them as partial filter sums. The sums accumulate in much the same way that partial beam sums do in that each value multiplied by a particular coefficient will form one part of a partial sum. When the last product is added to a partial sum the sum is complete and represents the Nth output sample of a non subsampled filtering operation.

Returning to the FEP, Figure 3.2.5 indicates the data paths used by the decimation filtering operation. Both real and imaginary components are treated identically. The bandshifted sample is returned to the multiplier input. It is multiplied by a decimation filter coefficient and the product is accumulated with one of the partial sums which was previously preloaded into the MAC accumulator register.

Figure 3.2.5: Decimation Filtering Data Paths

Figure 3.3: Software Control Hierarchy for Slice

The partial sum is returned to the memory and another partial sum removed and preloaded into the MAC accumulator while the input sample is being multiplied by the next coefficient. This process continues until all products involving the current input sample are summed with their relevant filter partial sums. The number of partial products handled on each pass is a function of the number of partial sums kept for the filter. This implementation uses eight partial sum locations and hence eight partial products are formed on each pass through the filter. In this manner the multiplier load for the decimation filtering operation is always a constant eight multiplies one each for real and imaginary time samples, another advantage of the partial sum filtering technique.

The decimation filter does not necessarily produce an output every processing cycle. The rate at which outputs occur is a function of the degree of subsampling since outputs only occur for those points which would have been chosen in a non subsampled output time series. When a partial filter sum is complete, i.e., the last partial product has been added. The sum represents a complete filtered time series sample, and is placed in the output FIFO buffer, the partial sum location in memory is then zeroed in preparation for a new partial sum of products.

The output FIFO buffer allows data samples to accumulate should the MP be engaged in another operation, such as filling the bandshift FIFO buffer. After the real and imaginary samples have undergone the decimation filtering operation, the FEP sequence is complete. A new sequence starts when the input FIFO buffer indicates another complete array of data has been collected.

3.2.3 The Master and Peripheral Processors

Following the signal processing done by the FEP the incoming beam data rate is reduced, due to the final decimation filtering operation, and VLSI microprocessors can be utilized for the remaining data processing. The Master Processor (MP) is a Z8001A, 16 bit, segmented microprocessor capable of addressing all the dynamic RAM (128 Kbytes) located on the slice. The Peripheral Processor (PP) is a Z8002A, 16 bit, nonsegmented microprocessor capable of addressing only half the dynamic memory store. Both processors are operated off a six megahertz clock. The functions of these processors are described in the following section on slice software.

3.3 Slice Firmware

3.3.1 Control Structure

Figure 3.3 shows a control hierarchy for slice software operations. Control parameter changes, requested by the operator at the display-control console, are passed to the slice through the display-control processor. The parameters are stored in a section of the slice main memory called, appropriately enough, the control buffer. On the slice the MP periodically checks for a status bit change indicating that new slice parameters are requested. When changes are requested for the PP, the MP changes the control buffer used by the PP. Changes to the FEP are affected by the MP overwriting the current FEP RAM (such as changing the phase delay RAM values) to the new requirement. Slice status information is also passed to the operator through the common memory link.

3.3.2 Standard Running Mode

When the slice is operating the MP and PP are executing a number of data processing/handling routines. The MP removes completed beam samples from the FEP. If the samples are to be processed for DEMON signals the sign bit is stripped to produce a rectified time series. Following rectification the samples are treated as any other time series output from the FEP and stored in memory using a circular buffer technique.

Using this technique the data storage size is static but the location of the latest sample stored changes as the previous data is gradually overwritten in a "circular" manner. The MP also sets status flags for the PP to tell it when a new time series is available for the FFT and when the FFT integration time is up. The frequency spectrum returned by the PP is moved to the display buffer area (also a "circular" buffer) by the MP. The MP has one function which is interrupt driven. The FEP FIFO for bandshift vectors interrupts the MP when it is empty. The MP calculates the next thirty-two phase accumulation values for the sine and cosine vectors and loads the sixty-four values into the FIFO before exiting the interrupt handler.

The PP does the spectrum analysis on the time series using a Bruune FFT algorithm. Several FFT spectra are integrated before a noise normalization algorithm is executed. This algorithm compares each FFT cell with an average noise power value obtained from adjacent cells of the FFT spectrum. The comparison is in the form of a ratio and the result stored as the actual finished FFT. The noise normalization is required to prevent broadband noise in one part of the FFT spectrum from masking (through display medium saturation) a valid tone in another part of the spectrum. The effects of noise normalization are shown in

Figure 3.4: Example Frequency Spectrum Showing Broadband Noise and Tones

Figure 3.5: Noise Normalized Form of Spectrum in Figure 3.4

Figures 3.4 and 3.5. Following noise normalization the spectrum is bit-reduced to two bits/FFT cell and packed for storage. The bit reduction in data values is required due to the large time buffers kept (approximately thirty minutes) for the FFT data. The bit reduction, as shown in Figure 3.6, is affected by setting three quantization levels for the FFT noise normalized spectra. The quantization algorithm is then quite straight forward. Below the first level is 0 (no signal), below the second is 1, and so on to above the third being 3 (a valid signal). Figure 3.7 shows the resultant spectrum. The noise normalized, bit reduced FFT spectrum is finally returned to the memory ready for the MP to store away and the display-controller to display.

3.4 Design Aspects of Interest

3.4.1 Three-ported, Prioritized, Dynamic RAM Memory

3.4.1.1 Design Considerations

The dynamic memory store design is critical to the slice concept. Unless the store has sufficient bandwidth to handle the required processor accesses the scheme is useless. To avoid problems with bus arbitration and/or the extreme bandwidth requirements of a single bus architecture a multiport memory was designed. A multiport memory represents one of many schemes⁽²⁾ currently being investigated

Figure 3.6: Example of Threshold Setting for Bit Reduction

Figure 3.7: Bit Reduced Form of Spectrum Shown in Figure 3.6

for multi-microprocessor system design. In implementing a multiport memory it is important that processors are not starved (locked out of memory by other processors) but it is equally as important that time critical processor functions are relatively unaffected by any port arbitration scheme. Processors hung-up by slow memory access suffer an equivalent apparent increase in processing load i.e. it takes longer to complete the operation.

The MP and PP in each slice perform time critical tasks which involve memory accesses whereas the display-control processor access (every few seconds for the latest data) is not critical at all. For this reason a full prioritizing scheme was instituted instead of a "round robin" equal access time sharing scheme. In addition, the prioritization circuit locks out the last successful processor when strobing for a new priority. This produces little or no loading on the locked out processor since it cannot make another valid memory access for at least two clock cycles and a memory access is only three clock cycles long.

The memory is 128 Kbytes of RAM configured from (64k by one) dynamic RAM chips. The prioritizing circuit enables three processors and an external refresh (from the Z8001) to access the memory. The highest priority access supports a 6Mhz Z8000 memory access without wait states.

3.4.1.2 Architecture

The design was dependent on: 1) Two processors, the MP and PP, in synchronous operation with the priority encoder; and 2) the non-synchronous processor being wait-stated as soon as its memory request line is asserted. The priority encoder provides automatic wait-stating for the synchronous processors.

As shown in Figure 3.8, there are three distinct sections to the memory: i) Priority encoder; ii) Memory ports from processors; and iii) the dynamic RAM memory.

3.4.1.3 Priority Encoder

The priority encoder is really the interesting feature of this memory. It provides automatic wait-stating for the synchronous processors, generates $\overline{\text{RASIN}}$ for the National Semiconductor 8408 dynamic RAM controller and locks out all requests except that with the highest priority. Figure 3.9 shows the simple circuit which provides the priority encoding, $\overline{\text{RASIN}}$ generation and auto wait stating.

When the MP or PP asserts a memory request ($\overline{\text{MRQ}}$) the normally tri-stated buffer on the wait line is enabled. If the processor does not get access to the memory or if another processor is already accessing the memory the wait line is pulled low and the processor is automatically

Figure 3.8: Elements of the Slice Dynamic Memory Store

Figure 3.9: Priority Encoder for Slice Dynamic Memory Store

wait-stated. If the processor is granted memory access the priority output is low and the wait line is returned to a high state so the processor can continue.

The priority encoder is a 74147, an eight bit to Binary Coded Decimal encoder with inputs wired to produce one of four exclusive outputs. Memory request inputs are registered to prevent $\overline{MREQ}$ transitions during the priority encoder set-up time (40 nanoseconds). The 12 MHz clock generated synchronously with the 6MHz clock on the K1160BB oscillator chip presented a convenient strobe for the entire system when NORed with the generated $\overline{RASIN}$ pulse going to the dynamic RAM controller.

3.4.1.4 Memory Ports

The memory ports are very simple. The address output from the processor is latched, the data input to the memory is buffered and the valid data output from the memory is latched. Figure 3.10 shows the scheme employed.

All inputs to the RAM are tri-stated until the priority encoder allows access and the inputs are enabled. Input data goes directly to the RAM while the address goes to the 8408 Dynamic RAM Controller to be split into row and column addresses. Data out from the memory is latched by the rising edge of $\overline{CAS}$ ($\overline{CAS}$ is generated by the 8408) ready for the processor to read.

Figure 3.10: Typical Porting Style of Dynamic Memory Store

Figure 3.11: Radix-2 Butterflies for Cooley-Tukey and Bruune FFT Algorithms

3.4.1.5 The Dynamic RAM

The dynamic RAM store itself consists of sixteen Inmos 64K by one bit, 120 nanosecond access, dynamic ram chips and one National Semiconductor 8408 Dynamic RAM Controller. The refresh input, $\overline{\text{RFSH}}$ switches the 8408 from an automatic access mode to an auto refresh mode.

3.4.2 Slave Microprocessor as a Spectrum Analyzer

Due to the sample rate reduction occurring through decimation filtering by the FEP and the use of a Bruune⁽³⁾ FFT algorithm it is possible to use a single Z8002 with no external multiplier as a spectrum analyzer. Sample rate reduction through decimation filtering has already been explained; however, the advantage of the Bruune over the standard Cooley-Tukey⁽⁴⁾ algorithm has not. The advantage of the Bruune algorithm is apparent when the basic FFT butterflies for both cases are examined. Figure 3.11 shows the two butterflies and the sums necessary to calculate the new points of the butterfly.

The advantage of the Bruune is apparent in the sums required to calculate the next pass values. Only two multiplications are required versus the four multiplications required for the Cooley-Tukey algorithm. The multiplication time for a microprocessor can be almost twenty times as long

as an addition time. Thus the number of multiplications is the critical factor in determining if the processor can perform the FFT in time. Using the Bruune algorithm the PP is able to do all the spectrum analysis required for the slice.

3.5 Summary

One slice of a Signal Processing Module (composed of multiple slices) was documented. The slice was shown to have four component modules: 1)The Front End Processor (FEP); 2)The Master Processor (MP); 3)The Peripheral Processor (PP); and 4)The Dynamic Memory Store. The FEP operations of interpolation, time delay beamforming, bandshifting and decimation filtering were described. Data manipulation by the MP and PP was covered before two interesting aspects of the design, the Bruune FFT and the triported memory, were discussed.

Further information on the design is available in classified report # 955886.

CHAPTER 4

SPECIAL TEST EQUIPMENT

4.0 Introduction

In this chapter a specially designed piece of test equipment is described. The interfacing and waveform generation functions of the Special Test Equipment (STE) are presented in some detail.

4.1 Design Objective

The Special Test Equipment was designed to fulfill a two-fold purpose: 1) to perform as an intelligent interface to the SPM for the display processor, and 2) to provide characterized test SONAR data for slice performance evaluation. The two functions are independent so the STE can be blocked out as shown in Figure 4.1.

4.2 THE INTELLIGENT INTERFACE

The interface function was implemented to allow operation with few restrictions on the display processor. To provide maximum compatibility, an RS-232 serial link is provided for the display processor. Requests for data, parameter changes or initialization procedures are passed to the STE interface where a set of software modules perform the requisite functions on the SPM through the parallel

Figure 4.1: Submodules of the Special Test Equipment

Figure 4.2: Block Diagram of Special Test Equipment: Intelligent Interface

interface to the slice bus. A block diagram of the interface is shown in Figure 4.2.

The interface implementation is straightforward. A Zilog 8030 Serial Communications Controller (SCC) provides a microprocessor compatible serial link. The SCC provides interrupt features on data received as well as on various error conditions detected. The "intelligence" on the STE is a Z8002 sixteen bit microprocessor with 8K bytes of program space (EPROM) and 4K of data space (RAM). The Z8002 maps a 64 Kbyte address space "window" to address the full 128 Kbyte slice memory. Thirty-two Kbyte windows are selected through the parallel interface. The parallel interface is configured to map 32 Kbyte sections of a slice's main memory into the upper 32 Kbyte address space of the Z8002 using a block-select scheme. To access a slice window the STE interface writes a register with a two bit value which chooses one of four 32 Kbyte blocks of slice memory. After the block select has been written the interface addresses the slice as if it were the upper 32 Kbytes of its own data memory.

In addition to its interfacing functions the STE microprocessor is used to load set-up parameters for the Test Waveform Generator (TWG).

4.3 The Test Waveform Generator (TWG)

The Test Waveform Generator was designed to provide characterized test SONAR data for slice performance evaluation. The key word here is characterized, most of the SONAR data available is not characterized and the only check that can be done with such data is to see if the acoustic data processor shows the same results as the one the data was gathered for. Needless to say there is nothing to say which processor is better or which is "correct". For a proper performance evaluation known input data must be used.

The TWG was designed to bracket the performance requirements of an SPM slice. It provides data with known Signal to Noise Ratios (SNRs) and known phase relationships, i.e. a fully characterized input to test system performance.

4.3.1 Performance Description

The TWG provides eight bit hydrophone sensor data samples at array rates of greater than 300 KHz. The sensor data samples correspond to samples from a linear hydrophone array. Each sensor sample is generated independently and the final output sample may contain:

- i) up to seven tones (sinusoidal) of differing amplitudes and phases

ii) a broadside DEMON signal, generated by fractionally modulating a noise value with a tone (DEMON is broadband, amplitude modulated noise)

iii) an independent noise value

The amplitude variation allows variable SNRs since the noise value is produced with a known constant power level. The resultant signals are representative of tones in noise appropriately phased to simulate the output of a linear array of hydrophones.

4.3.2 Architecture

The TWB is a micro-sequenced machine. The actual sensor signal generation algorithm is only twenty steps and is the same for each sensor. When one sensor value is complete the sensor number counter, indicating the sensor number completed, is incremented and another data generation sequence started. The state of the sensor number counter is decoded to yield an array sync pulse on the last array sample. The Front End Processor uses this signal to sync to the array data.

Equation 1 shows the algorithm used to generate a sensor data value.

$$(1) \quad S_i = \sum_{n=0}^6 A_n T_n + A_D T_D N_D + N$$

where S_i = sensor number being generated
 A_n = Amplitude of nth tone
 T_n = nth tone
 N_D = DEMON noise value (changed once per array)
 N = Independent noise value (changed each sensor)
 $A_D T_D N_D$ = Fractionally modulated noise = DEMON

A block diagram of the circuit used to generate this algorithm is shown in Figure 4.3.

The tones are generated, as mentioned in chapter three, by addressing a look-up table of sine/cosine values stored in PROM. The address is modified by adding a phase increment ($\Delta\phi$) to the current address. The accumulated value is used as an address to obtain the next phase of the sine or cosine signal being generated. When sufficient phase increments have been accumulated the output will have traversed through 2π radians, completing a sinusoidal cycle. The frequency generated is a function of the speed at which phase increments are accumulated and their magnitude. The phase increment accumulation rate is the sample rate of the tone being generated. A separate phase accumulator is kept for each of the eight tones generated for each sensor in the array. The phase accumulators are kept in RAM and are addressed by the sensor number counter to select a bank of eight accumulators (the eight tones for that sensor.) The current state of the tone number counter (which also drives the sequence PROM) is used to access the eight phase

Figure 4.3: Block Diagram of Special Test Equipment: Test Waveform Generator

accumulators for the tone generation. Once a sensor number has been incremented the sensor datum value is generated as follows (Please refer back to equation 1):

- 1) The S_i and $T/2$ (one half of T number clock rate) values address a phase accumulator for tone 0 of sensor i . The T_0 value addresses the phase increment $\Delta\phi$ from the $\Delta\phi/\text{Amplitude}$ RAM.
- 2) The tone number increments, this changes the address to the $\Delta\phi/\text{Amplitude}$ RAM but not the address to the phase accumulator RAM since the T number there changes at half the frequency ($T \text{ number}/2$). In effect the $\Delta\phi/\text{Amplitude}$ RAM address is double clocked relative to the phase accumulator RAM address. This allows an Amplitude value (A_0) for T_0 to be retrieved. While this A_0 value is being retrieved the new phase accumulator sum (old value and $\Delta\phi$) is stored back in the phase accumulator store and addresses the sine PROM.
- 3) The Tone number increments changing both the phase accumulator RAM address and the $\Delta\phi/\text{Amplitude}$ RAM. Simultaneously the value from the sine PROM (T_0) and the amplitude value (A_0) are clocked into the multiplier. As in step one, the new addresses obtain the next phase accumulator (for T_1) and its phase increment $\Delta\phi_i$, and the cycle repeats itself eight times in all. The tones are

summed in the multiplier accumulator. The eighth tone is $A_7 T_7$ used to modulate the noise sample used for generating a DEMON signal. This tone is actually the first generated, the $A_7 T_7$ sum is returned to the input of the multiplier (via the return path shown in Figure 4.3) and multiplied with the noise sample to modulate it hence forming a DEMON signal. Following the Demon sample generation the other seven tones are generated and summed before a white noise value is added to the sum to complete the sensor sample which is then output. Any of the components (including the noise) may be omitted by using a zero amplitude value during the generation cycle. Subsequent sensor samples are generated in an identical manner.

4.3.3 Noise Generation

The noise added to each sensor had to be uncorrelated to avoid problems in estimating overall system gain. To meet this requirement a parallel implementation of a 39 stage serial psuedo-random white noise generator⁽⁵⁾ was instituted. The parallel implementation allowed a much slower clock (eight times less than a serial shift register) to be used. The conversion method from serial to parallel is shown in Figure 4.4. Basically the conversion involves increasing the number of feedback taps to correspond to the

Figure 4.4: Serial to Parallel Clocking Option for Pseudo-random Noise Generator

equired number of noncorrelated output bits required in a single clock. The corresponding feedback and input tap locations are calculated by determining where the representative bits would have been as they passed a single feedback, serial type implementation.

Figure 4.5 shows a block diagram of the psuedo-random noise generator which clocks sixteen bits of noise into two latches (each loaded separately from the eight random bits drawn from the generator). The noise value used for the DEMON signal is generated in a similar manner (with different random input bits) but the value is only changed once each array to simulate a broadside DEMON signal.

4.4 Summary

The Special Test Equipment, a dual function board designed as a communications interface and test waveform generator was described. The communication interface was microprocessor controlled to provide flexibility in interfacing between the display processor and the slices of the Signal Processing Module. The need for characterized test data was mentioned by way of introducing the Test Waveform Generator (TWG). The TWG was shown to be a microsequenced signal generator which could incorporate upto 7 tones, a DEMON signal and noise in a generated sensor datum.

Figure 4.5: Block Diagram of Noise Generator for the Test Waveform Generator

CHAPTER 5

DEMONSTRATED OPERATION

5.0 Introduction

In this chapter the performance of the Test Waveform Generator (TWG), part of the Special Test Equipment (STE), is documented to demonstrate the available range of simulated SONAR data.

Slice operation is demonstrated using simulated SONAR data generated by the TWG.

5.1 The Special Test Equipment (STE)

As mentioned earlier the STE incorporates a Test Waveform Generator (TWG) which provides simulated SONAR data. That is, the TWG output simulates digitized data from linear array of hydrophone sensors. The slice would normally get data from such an array. The TWG performance can be demonstrated by showing: a) that it can generate tones of varying amplitude; b) that multiple tone generation is feasible; c) that hydrophone data can be relatively phased; and d) that it generates white noise.

The TWG performance was documented using the test set-up shown in figure 5.1. The displayed analog results

Figure 5.1: Measurement Set-up Used to Measure Test Waveform Generator Output

were obtained by passing the digital output through a National Semiconductor 1200 series 12 bit DAC wired for 8 bit operation as shown in Figure 5.2. The frequency spectra with accompanying signal amplitudes were obtained with an X-Y recorder connected to the HP 3582a spectrum analyzer.

5.1.1 Monotone Output

Figure 5.3 is a photo of a 100 hertz tone generated by setting one tone of each sensor to 100 hertz. As shown in Figure 5.4 this tone was input at the maximum amplitude available. Also apparent in Figure 5.4 are the harmonics generated by the tone as well as the quantization "noise" associated with the DAC process. The "noise" associated with the latter process was accentuated by lowering the signal amplitude until it corresponded to only toggling the least significant bit of the input to the DAC. This is a square wave output and as shown in Figure 5.5 the odd harmonics are enhanced as would be expected of a square wave.

5.1.2 Multiple Tone Summation

Figure 5.6 shows a photo of a square wave obtained by loading seven tones into the TWB (the first seven sine terms of the Fourier expansion of a square wave). The frequency spectrum in Figure 5.7 shows the seven tones used and their

Figure 5.2: 12 Bit DAC Wired for 8 Bit Operation

Figure 5.3a Photo of 100 Hz Tone Output

Figure 5.4: Frequency Spectrum of 100 Hz Tone Showing Fundamental and Harmonics in the Signal.

Figure 5.5: Frequency Spectrum of 100 Hz Tone a Minimum Amplitude Showing Increased Harmonic Content (input now a square wave)

YAXIS=5 V/div
XAXIS=1 ms/div

Figure 5.6 Photo of Square Wave Obtained by
Loading Seven Tones into
Test Waveform generator

relative amplitudes. The ripple in the square wave edge is due to the low number of expansion terms used to generate the square wave.

5.1.3 Relative Phase Delay

Relative phase delays between sensors is shown in Figure 5.8. The photo shows multiple offset square waves generated by delaying the seven tones from sensor to sensor, thus each square wave generated for a sensor is offset from the previous one.

5.1.4 Noise

The white noise generated is shown as a scope trace in Figure 5.9 and as an averaged frequency spectrum in Figure 5.10. Note that the spectrum is flat across the band indicating "white" (equal power in all cells across the band) noise.

5.1.5 Signal Amplitude Variation

The multiple spectrum shown in Figure 5.11 graphically indicates the SNR variation available. maximum and minimum tones span a 50 dB range.

5.2 Sample Slice Output

YAXIS=5 V/div
XAXIS=1 ms/div

Figure 5.8: Photo of 32 Square Waves i.e. Each Sensor's Tones Delayed Relative to the Following Sensors (Shows Phasing Ability)

YAXIS=5 V/div
XAXIS=1 ms/div

Figure 5.9: Photo of Noise Signal Generated

Figure 5.10: Averaged Frequency Spectrum of Generated Noise,
Note Flatness of Spectrum.

Figure 5.11: Maximum and Minimum Tone Level

The major requirement of a slice of the Signal Processing Module is that it handle the beamforming and spectrum analysis for a particular look direction of the system.

To demonstrate slice performance two signal sets corresponding to that of a 32 sensor array, sampled at 3,152 hertz, incident at 0 degrees are input to the slice. The tonal spectrum of the signal sets is shown in Figure 5.12.

The second signal set, incident tones generated 180 degrees out of phase for alternate sensors, is used to demonstrate the slice ability to beamform and process the time series data. If the slice is performing correctly the beamforming operation will cause out-of-phase signals to cancel. The results of the spectrum analysis performed by the Slice Peripheral Processor are shown plotted as bit reduced LOFAR.

Figure 5.13 shows the spectrum present for the inphase signal set while that of Figure 5.14 shows the spectrum for the phase shifted input. The presence of the input signal set in Figure 5.13, but not in Figure 5.14, and the correct peak frequency values indicates correct slice operation.

5.3 Summary

Output waveforms and frequency spectra for various

Figure 5.12: Frequency Spectra of Test Signals for Slice Operation Check

Absolute Threshold Values

$$T_1 = 64$$

$$T_3 = 1024$$

Figure 5.13: Slice LOFAR Output With Beam Steered to 0 Degrees, input inphase.
Note that Input Signal Frequencies are Present.

Figure 5.14: Slice LOFAR Output With Beam Steered to 0 degrees, input out-of-phase.
Note Tonal Spectrum has Cancelled Out.

simulated signal sets produced by the Test Waveform Generator (TWG) were shown. These signal sets were chosen to demonstrate the TWG's ability to generate: 1) Multiple tones, 2) Tones with variable amplitude, 3) Noise and 4) Multiple sensor data.

The TWG was used to demonstrate that the slice correctly performed its beamforming and spectrum analysis operations. More extensive testing for the minimum detectable signal levels, out of band signal rejection and correct DEMON processing is currently underway. The results will be reported in classified report # 955899.

22

OF / DE

CHAPTER 6

CONCLUSIONS

6.0 Summary

Seawater as a transmission medium was discussed. The high signal loss associated with standard transmission frequencies indicated only audio frequencies (sound) could be used in this medium.

SONAR (SOund Navigation And Ranging) concepts were introduced and a brief description was given of frequency spectrums produced by various environments. A typical SONAR system was discussed in an effort to familiarize the reader with the signal processing requirements of such a system. The problems associated with a SONAR system using a Central Acoustic Data Processor were illustrated and the "sliced" solution was introduced. The sliced signal processor, representative of distributed processing by TASK rather than by FUNCTION, was discussed in some detail.

The Special Test Equipment (STE), designed to function as an interface to the sliced Signal Processing Module and as a test waveform generator, was introduced. The importance of providing characterized test data was stressed before discussing the Test Waveform Generator's (TWG) flexibility in providing such test data.

Finally the performance of the Test Waveform Generator and one slice of the Signal Processing Module were demonstrated.

6.1 Conclusions

This thesis has demonstrated a "sliced" Signal Processing Module for processing SONAR data from a linear array of hydrophones. The "sliced" Signal Processor is representative of a different distributed processing scheme which distributes processes by task rather than function.

The slice approach makes use of advances in silicon technology to execute an entire task as a single independent submodule. For the Sliced Signal Processing Module this task was processing data for a particular look direction (forming and processing data for one beam) of the system. The slicing approach lowers the system data processing speed requirement by more than an order of magnitude from that of a single central processor scheme. In addition the slice approach yields a system with graceful degradation on failure, one slice failing only reduces the system by $1/N$ where N is the number of slices in the system.

This thesis has also demonstrated that a versatile test waveform generator, for producing characterized test data, is a feasible and useful addition to a SONAR signal

processor.

6.2 Suggestions for Further Work

The sliced method of distributed processing is eminently feasible, will become simpler as gate array designs and semi-custom silicon suppliers become more prevalent, and, represents an elegant solution to the problems of system reliability, maintainability and manufacturability. With this in mind the obvious recommendation for further work is to adapt TASK oriented distribution (slicing) to other signal processing problems. The general requirement in choosing such tasks is that the task must not use data dependent upon another task i.e. slices must be non-interactive. Problems which involve interactive data transfer can still be sliced for those portions of the processing not requiring such interactive transfers. The ideal problem will contain an element of repetition so that the chosen slice will be of maximum benefit in reducing system cost and bandwidth requirements while enhancing system reliability and maintainability.

6.3 Specific Contributions

Specific Contributions to the Sliced Signal Processor project were as follows:

FEP design and implementation	-- Terry Kitzul
STE design and implementation	-- Carl M. Anderson
Slice memory design and implementation	-- Carl M. Anderson
Slice software design	-- Ian Hogg
Slice software implementation	-- Dave Buertin
Display software implementation	-- Gary Carruthers
System Integration	-- Carl M. Anderson

REFERENCES

- {1} R.J. Urick, Principles of Underwater Sound, 2nd Ed., McGraw-Hill Book Company, 1975.
- {2} G. Silverman, A. Stundel, John Lehan, Rockefeller University, "The Modular Multiprocessor-A Model for Laboratory Instrument Design", IEEE MICRO, p51-61, May 1982.
- {3} B. Bruun, "z-Transform DFT Filters and FFT's", IEEE Trans. on Acoustics, Speech, And Signal Processing, Vol ASSP-26, No. 1, p56-63 February 1978.
- {4} J.W. Cooley and J.W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," Math. Comput., Vol. 19, p297-301, 1965.
- {5} M. Damashek, "Shift Register with Feedback Generates White Noise", Electronics, p108-109, May 27, 1976.

END

0 6 H 0 9 H 8 4

FIN