Proceedings of the Institutue of Acoustics

A VERSATILE ADAPTIVE DIGITAL SONAR BEAMFORMER

M TRIVEDI (1), D C ATMORE (2) & C H STARKIE (1)

(1) Marconi Underwater Systems Ltd., Signal Processing Group, Blackmoor Lane, Croxley Mill, Watford, Hertfordshire, U.K.
(2) Marconi Underwater Systems Ltd., Sonar Research Techniques Group, Blackmoor Lane, Croxley Mill, Watford, Hertfordshire, U.K.

1. INTRODUCTION

The capability of operating effectively in the presence of directional broadband interference sources is an obvious and a desirable feature in modern sonar design. The ability of adaptive processing techniques to suppress interfering sources on a sonar array is well known [1]. This is achieved by attenuating the array's response in the direction of the noise sources in an adaptive manner. The consequence is improved bearing resolution and probability of signal detection [2]. Over the past few years a number of algorithms and architectures have been proposed for adaptive digital beamforming [3,4,5,6]. The architectures investigated have mainly been simple multi-channel (ie. one per array element) suited to systolic VLSI type of implementation [5], or those requiring a hardware intensive and a dedicated solution [6].

The recent availability of fast, floating-point single-chip DSP processors has made compact implementation of some of these adaptive array processing techniques possible for sonar applications. In particular, higher rates of weight update for multi-element, multi-tap transversal filter structures can now be realised accurately using compact and economical hardware.

The work presented here outlines such an implementation and forms part of an on-going research programme into adaptive beamforming techniques at Marconi Underwater Systems Ltd at Croxley, Watford.

A VME bus controlled single card beam processor has been developed to allow real-time assessment of a variety of adaptive broadband beamforming algorithms to be made. The hardware comprises a fast multi-element, multi-tap transversal filter structure the weights of which are continually updated by a floating-point DSP chip – the TMS320C30.

Here we will consider the implementation of three well known adaptive beamforming techniques. The first two techniques are the Exact Least Square (ELS) method and its simplified but faster form known as the Frost LMS gradient descent approach [3]. Both schemes employ a zero order derivative constraint. The third technique known as the Generalised Sidelobe Canceller (GSC) approach employing higher order derivative constraints [4] will also be considered for implementation through its simpler and faster LMS form.

The ELS approach involves the computation of the inverse signal covariance matrix. Even with a zero order derivative constraint the ELS scheme is computationally quite intensive and therefore, takes longer to process. A faster but less optimal implementation of the ELS method is via the Frost technique where a constrained LMS power minimisation algorithm is employed.

A beamformer employing a simple zero order constraint can point a very narrow beam in the 'look-direction'. The incorporation of higher order derivative constraints is desirable since it increases the degree of tolerance to signal misalignment errors in the look-direction. However, this can make the ELS approach slower to implement. An alternative approach for implementing higher order derivative constraints is through the GSC technique to which an unconstrained LMS power minimisation algorithm can be applied.

Some performance benchmarks together with hardware simulations are used to illustrate the processing capability of the implemented beamformer.

ADAPTIVE SONAR BEAMFORMER

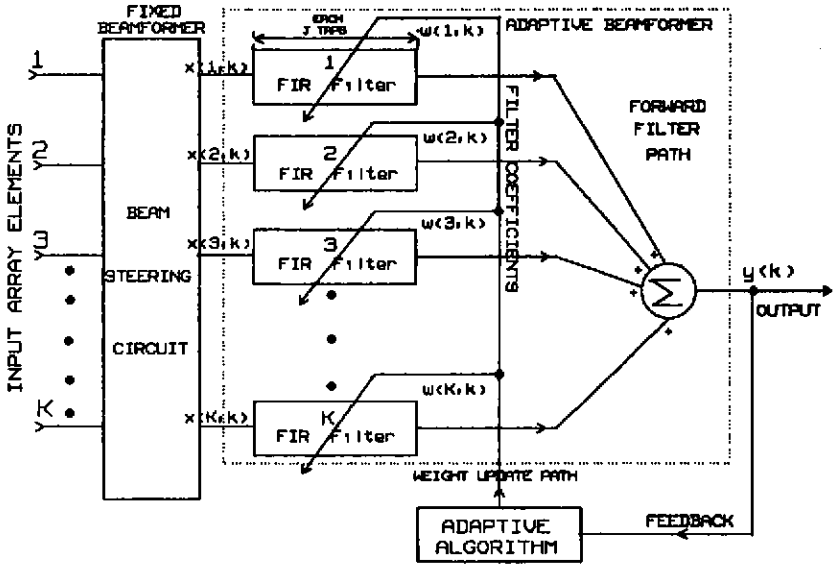

Fig. 1 BLOCK DIAGRAM OF THE ADAPTIVE BEAMFORMER

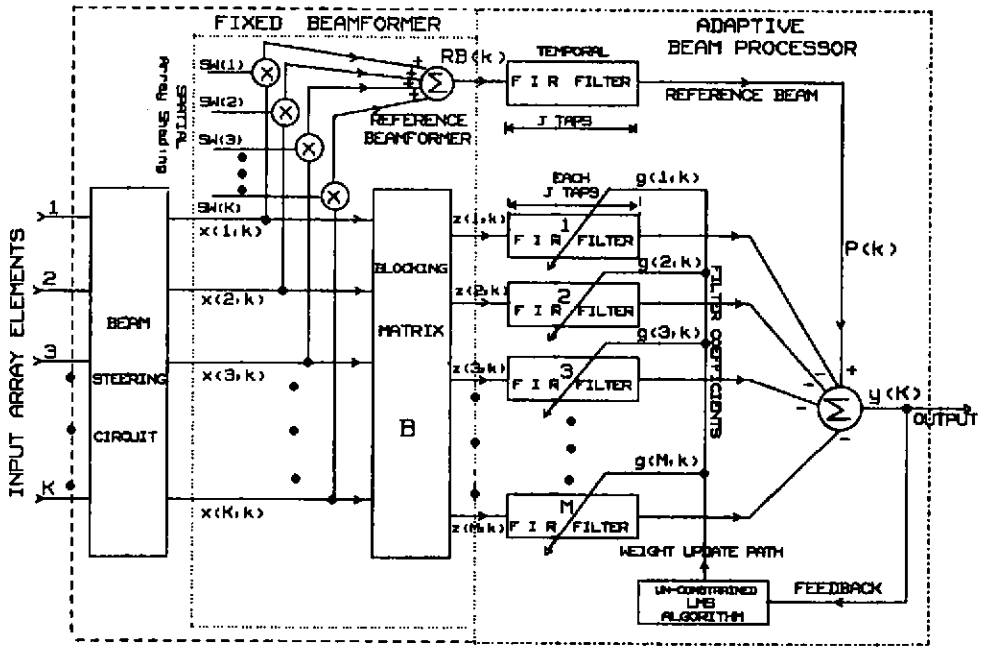

Fig. 2 BLOCK DIAGRAM OF THE GENERALISED SIDELOBE CANCELLER

ADAPTIVE SONAR BEAMFORMER

2. ADAPTIVE ARRAY PROCESSING

A block diagram of the adaptive beam processing under consideration is given in Fig 1. A beam steering circuit for forming the beam in the look-direction is shown preceding the adaptive beamforming section. Beam steering is achieved by inserting inter-element broadband time delays appropriate to the array geometry and the desired look-direction. The adaptive section comprises a forward filter path and a weight update generator implementing a Least Square type of algorithm.

The forward filter path comprises a row of K transversal filters operating in parallel to realise a specified finite impulse response (FIR). Each filter (one per input array element) has J taps. The forward path output is a summation of individual row filter outputs. Weight generation strategies for the three forementioned adaptive schemes are considered next.

2.1 Weight Generation. The weight update path generates weights in order to minimise the output signal power. The weight generation schemes considered are: the ELS, the gradient descent Frost LMS and the GSC LMS methods. Overall, the GSC approach is dissimilar to the other two methods in its implementation as shown in Fig 2. It differs from the basic arrangement of Fig 1 in that the output is formed by subtracting the non-look-direction array response from the look-direction reference beam P(k). This beam is formed by conventional spatial weighting (SW(k)), summation and temporal filtering. The function of the blocking matrix operator B is to only pass undesired components outside the look-beam for spatial filtering and subsequent subtraction from the reference beam. The matrix B in effect, applies higher order derivative constraints. This is in addition to the zero order (look-direction constant gain) constraint assumed for the other schemes. Details of the ELS and the Frost schemes can be found in Ref 3 while those of the GSC approach can be found in Ref 4. Expressions for weight generation for the three schemes are considered next.

2.1.1 ELS Scheme. In case of the Exact Least Square (ELS) approach the inverse of the signal covariance is required for formulating the filter weights at regular intervals. The optimum set of filter weights, W(opt), is given by (see Eqn 16, Ref 3):

$$W(opt) = R^{-1} \cdot C \cdot (C^T \cdot R^{-1} \cdot C)^{-1} \cdot FV \quad \ldots\ldots(1)$$

where, R^{-1} is the (KJxKJ) inverse signal covariance matrix. In practice however, only an estimate of the true signal covariance matrix R can be formed by taking a large number of array samples. C is the (KJxJ) directional gain constraint matrix with K unity entries per column shifted down by K row positions for successive columns. C^T denotes the transpose of C. FV is the (Jx1) column constraint vector which specifies an overall desired frequency response for the forward filter path. W(opt) has dimensions KxJ.

2.1.2 Frost LMS Scheme. The Frost constrained LMS gradient descent approach (a simplification of the ELS approach) requires snap-shots of the filter data matrix and the corresponding forward path output values in the evaluation of the filter weights. Here, the forward path filter weights, W(k+1), are generated iteratively from W(k) as (see Eqn 22, Ref 3):

$$W(k+1) = P \cdot (W(k) - mu \cdot y(k) \cdot X(k)) + FF \quad \ldots\ldots(2)$$

where, W(0) = FF and $FF = C \cdot (C^T \cdot C)^{-1} \cdot FV$ with FV as the overall frequency response constraint vector met earlier in Eqn 1. The matrix $P = I - C.(C^T.C)^{-1}.C^T$ is referred to as the projection matrix and y(k) is k^{th} filtered output. C is the constraint matrix (of Eqn 1) and C^T is its transpose. X(k) is the transversal filter data matrix whose elements are weighted by W(k). 'mu' is the convergence factor which controls the trade off between the rate of convergence and the misadjustment noise [3]. The weight matrix W has dimensions KxJ.

2.1.3 GSC LMS Scheme. A gradient descent GSC approach is considered. The expression for weight generation is iterative and similar to the Frost LMS method (Eqn 2). Additionally, the use of an unconstrained LMS process makes the GSC weight updating expression even simpler and is given by (see Eqn 14, Ref 4):

$$G(k+1) = G(k) + mu \,.\, y(k) \,.\, Z(k) \quad \ldots\ldots(3)$$

where, G(k+1) is the new GSC filter weight set derived from G(k). 'mu' is the convergence factor met earlier in Eqn 2 and y(k) is the k^{th} filtered output. Z(k) is the modified filter data matrix obtained by operating on the initial data matrix X(k) with the blocking matrix B. Incorporation of higher order derivative constraints causes matrices Z and G to have fewer number of rows (M) than the input data matrix X which has K rows (see Fig 2). This depends on the number and the order of constraints applied. Consequently, matrix Z and G have dimensions smaller than matrix X (MxJ instead of KxJ). Matrix B has dimensions MxK. From Eqn 3 it is evident that the GSC LMS weight generation process is simpler and faster than the Frost LMS approach. Overall however, additional computations necessary for generating the reference beam and applying the blocking matrix B to input data elements need to be carried out, see Fig 2.

3. HARDWARE IMPLEMENTATION

The single card processor implements the composite FIR filter structure of the forward path the weights of which are computed in accordance with the three criteria discussed above (ie. via Eqns 1, 2 and 3). A block diagram of the hardware which implements the basic adaptive processing (in Fig 1) is shown in Fig 3. The initial look-direction beam steering is not shown. Real data samples and real filter weights are assumed in this paper.

3.1 Forward Filter Path. Referring to Fig 3, successive input data column vectors are clocked into a row of K parallel programmable shift registers. Each register pipeline is J taps deep and 16 bits wide. At any given instant this register array holds the KxJ data element matrix X. After each clocking operation a new data column enters this matrix displacing all the previous column elements by one column. The oldest data column is lost. The matrix elements are weighted by the corresponding elements of the KxJ weight matrix (W) and these are simultaneously summed by the 16x16 bit multiplier/accumulator (MAC) chip, see Fig 3. The output y(k) is generated after KxJ such MAC operations defining the forward path cycle. This cycle is repeated every time a new data column is clocked into the data pipeline register array. The forward filter path operates at 20 MHz per tap, which for a KxJ=36 tap filter implies a minimum cycle period of 1.8 μsec.

The filter weights are held in a dual random-access memory (RAM) operating in a 'ping-pong' mode. While the current weight set W(k) is applied to the continuously operating forward path from one RAM a newly updated set, W(k+1), can be stored in the other. On completion of a weight update the roles of these RAMs are interchanged when the next forward path cycle starts. The new set W(k+1) is then applied until the next weight update.

3.2 Weight Update Path. The above filter weights are computed by the 33 Mflops TMS320C30 DSP chip to a single-precision floating-point accuracy. However, all data external to the DSP chip is in a 16 bit two's complement format. Since the forward filter path is free running, the DSP chip cannot access snapshots of the input data matrix without interfering with the forward path operation. In order to overcome this difficulty, the block labelled 'Data Shadow Ram' (DSR) in Fig 3 takes a copy of the data matrix elements every time these are accessed from the data pipeline registers for the forward path MAC operations. When the DSP chip is ready to start a new weight update, it write-disables the DSR and proceeds to read its contents at its own pace (60 nsec read cycle). The DSR is write-enabled soon afterwards. The DSP chip can access 32Kx32 bit of on board zero-wait-state RAM.

3.3 Processor Communications & Control. Data communications are performed over three separate parallel ports. The data column elements, which have been steered in the look-direction, enter the processor over a fast 16 bit input bus. The spatially and temporally filtered values are output over a second 16 bit bus. A third port supports a 16/32 bit VME interface and allows the card to be slave driven by a VME controller, if desired. The forward filter path operates under micro-code control and runs asynchronously to the weight generator DSP chip. Overall functionality testing of the forward path is under direct TMS320C30 control.

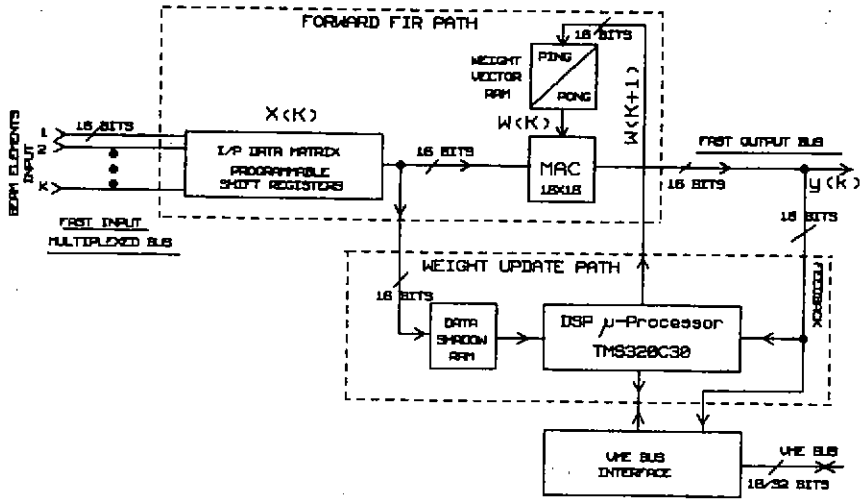

Fig. 3 BLOCK DIAGRAM OF THE ADAPTIVE BEAMFORMER HARDWARE

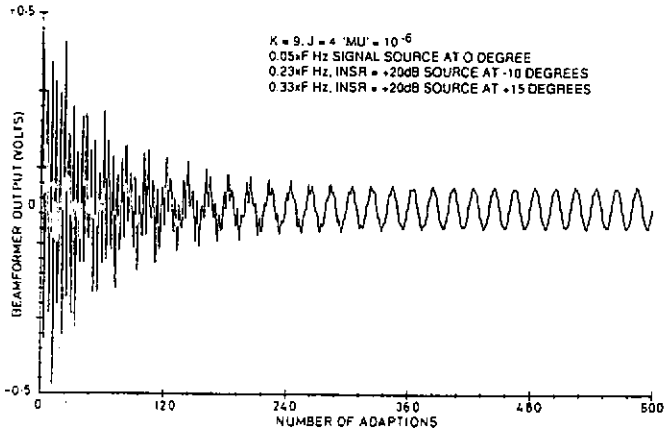

Fig. 4 FROST NARROWBAND RESPONSE

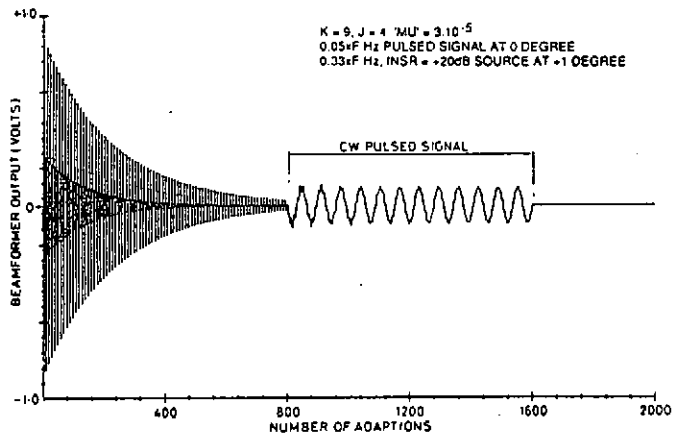

Fig.5 FROST NARROWBAND RESPONSE TO A PULSED SIGNAL

2.1.3 GSC LMS Scheme. A gradient descent GSC approach is considered. The expression for weight generation is iterative and similar to the Frost LMS method (Eqn 2). Additionally, the use of an unconstrained LMS process makes the GSC weight updating expression even simpler and is given by (see Eqn 14, Ref 4):

$$G(k+1) = G(k) + mu \,.\, y(k) \,.\, Z(k) \quad \ldots\ldots(3)$$

where, G(k+1) is the new GSC filter weight set derived from G(k). 'mu' is the convergence factor met earlier in Eqn 2 and y(k) is the k^{th} filtered output. Z(k) is the modified filter data matrix obtained by operating on the initial data matrix X(k) with the blocking matrix B. Incorporation of higher order derivative constraints causes matrices Z and G to have fewer number of rows (M) than the input data matrix X which has K rows (see Fig 2). This depends on the number and the order of constraints applied. Consequently, matrix Z and G have dimensions smaller than matrix X (MxJ instead of KxJ). Matrix B has dimensions MxK. From Eqn 3 it is evident that the GSC LMS weight generation process is simpler and faster than the Frost LMS approach. Overall however, additional computations necessary for generating the reference beam and applying the blocking matrix B to input data elements need to be carried out, see Fig 2.

3. HARDWARE IMPLEMENTATION

The single card processor implements the composite FIR filter structure of the forward path the weights of which are computed in accordance with the three criteria discussed above (ie. via Eqns 1, 2 and 3). A block diagram of the hardware which implements the basic adaptive processing (in Fig 1) is shown in Fig 3. The initial look-direction beam steering is not shown. Real data samples and real filter weights are assumed in this paper.

3.1 Forward Filter Path. Referring to Fig 3, successive input data column vectors are clocked into a row of K parallel programmable shift registers. Each register pipeline is J taps deep and 16 bits wide. At any given instant this register array holds the KxJ data element matrix X. After each clocking operation a new data column enters this matrix displacing all the previous column elements by one column. The oldest data column is lost. The matrix elements are weighted by the corresponding elements of the KxJ weight matrix (W) and these are simultaneously summed by the 16x16 bit multiplier/accumulator (MAC) chip, see Fig 3. The output y(k) is generated after KxJ such MAC operations defining the forward path cycle. This cycle is repeated every time a new data column is clocked into the data pipeline register array. The forward filter path operates at 20 MHz per tap, which for a KxJ=36 tap filter implies a minimum cycle period of 1.8 μsec.

The filter weights are held in a dual random-access memory (RAM) operating in a 'ping-pong' mode. While the current weight set W(k) is applied to the continuously operating forward path from one RAM a newly updated set, W(k+1), can be stored in the other. On completion of a weight update the roles of these RAMs are interchanged when the next forward path cycle starts. The new set W(k+1) is then applied until the next weight update.

3.2 Weight Update Path. The above filter weights are computed by the 33 Mflops TMS320C30 DSP chip to a single-precision floating-point accuracy. However, all data external to the DSP chip is in a 16 bit two's complement format. Since the forward filter path is free running, the DSP chip cannot access snapshots of the input data matrix without interfering with the forward path operation. In order to overcome this difficulty, the block labelled 'Data Shadow Ram' (DSR) in Fig 3 takes a copy of the data matrix elements every time these are accessed from the data pipeline registers for the forward path MAC operations. When the DSP chip is ready to start a new weight update, it write-disables the DSR and proceeds to read its contents at its own pace (60 nsec read cycle). The DSR is write-enabled soon afterwards. The DSP chip can access 32Kx32 bit of on board zero-wait-state RAM.

3.3 Processor Communications & Control. Data communications are performed over three separate parallel ports. The data column elements, which have been steered in the look-direction, enter the processor over a fast 16 bit input bus. The spatially and temporally filtered values are output over a second 16 bit bus. A third port supports a 16/32 bit VME interface and allows the card to be slave driven by a VME controller, if desired. The forward filter path operates under micro-code control and runs asynchronously to the weight generator DSP chip. Overall functionality testing of the forward path is under direct TMS320C30 control.

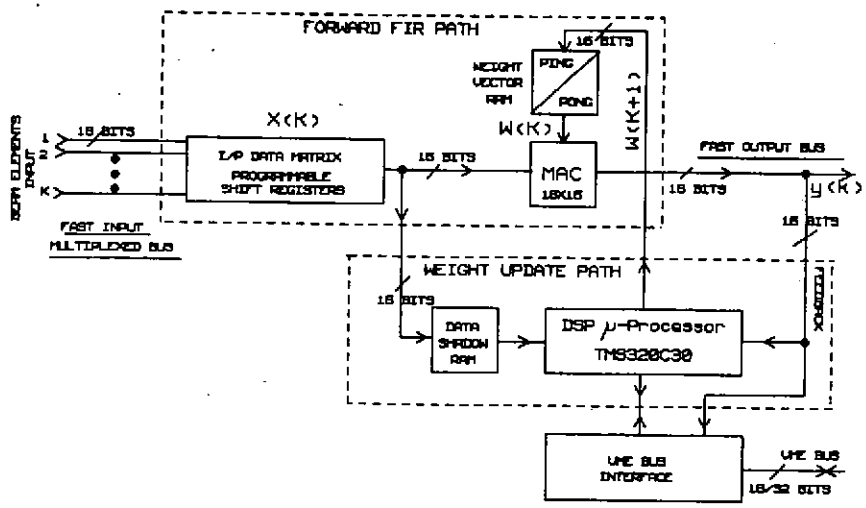

Fig. 3 BLOCK DIAGRAM OF THE ADAPTIVE BEAMFORMER HARDWARE

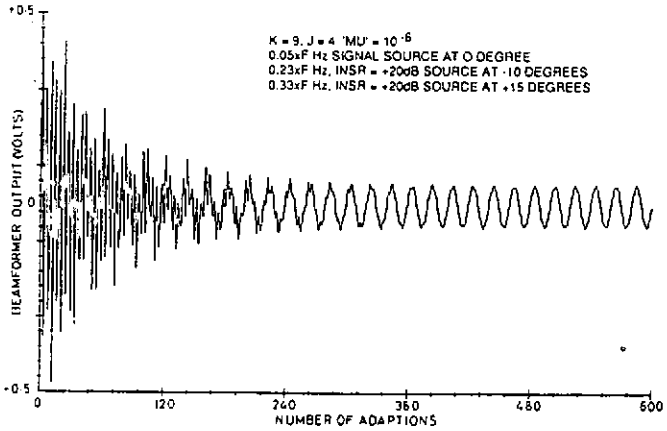

Fig. 4 FROST NARROWBAND RESPONSE

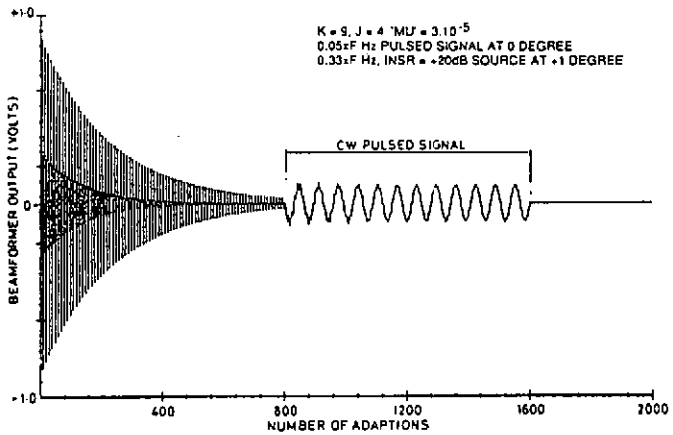

Fig.5 FROST NARROWBAND RESPONSE TO A PULSED SIGNAL

4. ALGORITHM MAPPING

When computing filter weights according to the Frost LMS (Eqn 2) or the GSC LMS (Eqn 3) criterion, the Data Shadow RAM (DSR) in Fig 3 stores the most recent snapshot of the X or the Z data matrix, respectively. In the ELS case (Eqn 1) however, a specified number of consecutive snapshots of data matrix X are stored by the DSR. This then allows the DSP chip to compute a long term estimate of the inverse of the signal covariance matrix R. Unlike the first two methods the (16 bit) filter output is not required. The inverse of matrix R can be computed via the Householder triangularisation method [7] or via the matrix inversion lemma identity. In either case, matrix R need not be computed explicitly.

The hardware implements the forward filter path for all the three methods directly. In the GSC case, the FIR filter weights defining the temporal response of the reference beam (Fig 2) remain fixed while the filter weights for the remaining M filters are computed according to Eqn 3. The spatial weights (SW(k)) for the reference beamformer and the coefficients for the blocking matrix B, once computed for a given look-direction, remain fixed. Hence, these operations can be implemented as part of the preceding fixed beamformer structure, see Fig 2.

5. WEIGHT UPDATE BENCHMARKS

The Frost LMS and the GSC LMS weight generation methods are somewhat similar in their implementation. For the generation of Frost weights using a composite, 36 tap FIR filter (K=9, J=4) the TMS320C30 chip takes about 8.5 μsec. A straight line TMS320C30 assembly-level coding is used. The fetching of the data snapshot matrix, the y(k) value and writing of the updated weights back into the 'ping-pong' RAM yields an overall update period of about 20 μsec. For the GSC case (with a zero and a first order constraint) a similar weight generation takes only about 4 μsec with an overall update period of about 13 μsec. This assumes that only the adaptive part in Fig 2 is implemented by the single-card processor. The reference beamformer and the blocking matrix operations are implemented using extra hardware as part of the preceding fixed beamformer section.

It is possible to map the entire GSC structure in Fig 2 (excluding look-beam steering) onto the single card processor which has the architecture shown in Fig 1. In this case the weights are still calculated via Eqn 3. However, the reference beam formation and the blocking matrix operations are now implemented in software as part of a more elaborate weight generation process. This involves expressing the MxJ weight matrix G (in Eqn 3) back in terms of a bigger KxJ Frost weight matrix W (of Eqn 2). The above equivalence between the GSC and the Frost type of adaptive structure is detailed in Ref 4 (see Eqn 20). This feature allows the extra hardware complexity of the GSC approach to be traded off against an increase in software complexity. For example, mapping the GSC LMS approach onto the simpler Frost LMS structure increases the previously mentioned weight update interval of 13 μsec to 60 μsec. Here, K=9, J=4 and M=K-L=6 with L=3. The parameter L defines the loss in the degrees of freedom due to the inclusion of zero and first order derivative constraints. Now, the 6x4 GSC weight matrix G has to be expressed in terms of a 9x4 Frost weight matrix W. Other terms such as the (Jx1) temporal weights for the reference beam, the (Kx1) spatial weights SW and the coefficients of the (MxK) blocking matrix B are also involved in the generation of the weight matrix W.

In case of an ELS implementation (Eqn 1) the majority of the weight update time is spent in computing the inverse R matrix, ie R^{-1}. For KxJ=36 and using a 36x36 data matrix to compute the square root covariance matrix (via the Householder technique) the generation of R^{-1} can take up to 4 msec. An overall weight update period about 5 msec is then acheived. For K=9, J=7 and using a 63x63 data matrix a weight update period of around 20 msec results.

5.1 Benchmark Comparisons. A lack of readily available weight update benchmarks for the Frost and the GSC LMS methods makes comparison with our benchmarks difficult. For the ELS approach however, rough comparisons can be made with the performance results obtained by Wells for the 'FLAP' processor [6]. The FLAP is a multi-beam, multi-element processor which splits each look-direction beam into a number of spectral bands. It then generates weights for each of these sub-band beams. The weights are computed by evaluating the signal covariance matrix and adding to it the constraint covariance matrix appropriate to a given look-direction beam. For a time-domain implementation of the FLAP using a 28x28 covariance matrix the computation of a 28 element weight vector for one beam takes 1.3 msec, see Ref 6.

Our single-beam, time-domain approach, using a 36x36 square root covariance matrix, generates a 36 element weight vector update in 5 msec. Bearing in mind the degree of parallelism the FLAP architecture employs, our approach, using a single TMS320C30 processor chip, gives a reasonable update rate. It is difficult to make a proper comparison with the FLAP system since little detail of the hardware employed is given in Ref 6.

6. HARDWARE SIMULATION RESULTS

Initial results indicate a close agreement between the hardware simulations and the actual performance. Presented here are some hardware simulation results in response to narrowband and broadband scenarios. In the following illustrations an equispaced, 9 element linear array geometry has been employed. An inter-element spacing of C/F is used where, C is velocity of sound in water and F is array sampling frequency. All signal frequencies are expressed as fractions of F. An all-pass forward filter path response has been employed. The values given to the convergence factor 'mu' are relative to 16 bit data and feedback y(k) values. A filter weight update rate equal to F is assumed. A practical maximum value of F is 500 KHz. In the following illustrations the normal to the linear line array is defined as the look-direction and assigned a bearing of zero degree. Source positions in the first quadrant are assigned values from 0 to 90 degrees while those in the second quadrant from 0 to -90 degrees. All sources other than at zero degree are treated as interferers while a source in the look-direction is treated as the desired signal. The interference-noise-to-signal ratios (INSR) in the following section refer to source power levels at the array elements. However, these source power levels can be attenuated by the array's directional response to give modified INSRs at the input of the adaptive beamformer section. These modified input INSRs are given in brackets.

6.1 Narrowband Scenario. A Frost response (with zero order constraint) to a narrowband scenario is illustrated in Fig 4. Here, a weak look-direction sinusoidal signal (ie. at 0 degrees) with a frequency 0.05xF Hz is being interfered with by two other sinusoidal sources at -10 and +15 degrees. The 0.23xF Hz source at -10 degrees has an INSR of +20 dB (+18.1 dB) relative to the signal while the other 0.33xF Hz source has an INSR of +20 dB (+8.9 dB). A filter configuration employing 4 taps per element has been used with a 'mu' value of 10^{-6}.

The filtered output recovers the weak 0.05xF Hz signal in the look-direction after some 300 adaptions. Decreasing the 'mu' value increases the convergence time. Increasing the weight update interval also has a similar effect. Increasing the number of filter taps/element leads to a faster convergence due to a more rapid spatial de-correlation of the non-look-direction signals. The presence of some misadjustment noise in Fig 4 is also evident. These effects are described by the LMS steady-state performance analysis given in Ref 3.

The response to a pulsed sinusoidal signal is illustrated in Fig 5. Here a 0.33xF Hz sinusoidal source positioned at 1 degree and having an INSR of +20 dB (+19.9 dB) is interfering with a 0.05xF Hz look-direction pulsed signal. For rapid convergence 'mu' is set to $3x10^{-5}$. The source at 1 degree is attenuated severely while the pulsed source is allowed to pass unaltered due to the constant gain constraint in the look-direction. The intolerance of the Frost beamformer to signals outside the look-direction can also be illustrated using Fig 5. For instance, if the signal itself had been misaligned to the look-direction by one degree, it would be eliminated just as readily. It is this narrow look-direction response of the Frost process which makes its practical use very susceptible to signal direction misalignment errors. Incorporation of higher order derivative constraints broadens the beam shape about the look- direction, thereby preventing unwanted signal cancellation due to small misalignment errors.

ADAPTIVE SONAR BEAMFORMER

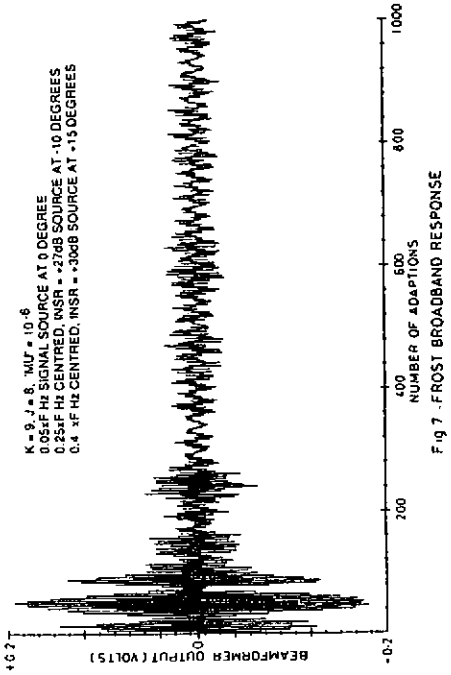

Fig 7 .FROST BROADBAND RESPONSE

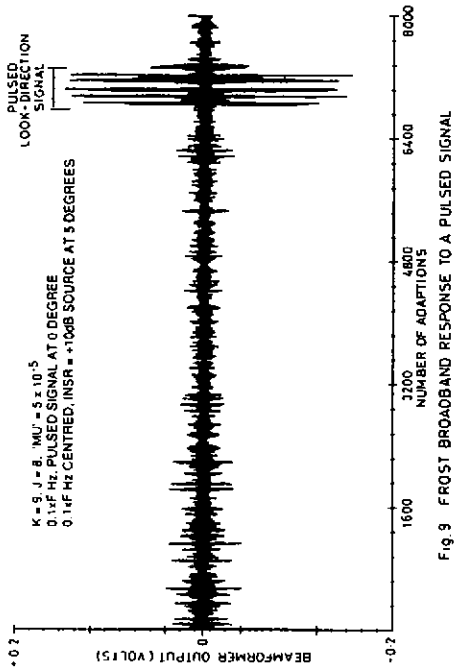

Fig.9 FROST BROADBAND RESPONSE TO A PULSED SIGNAL

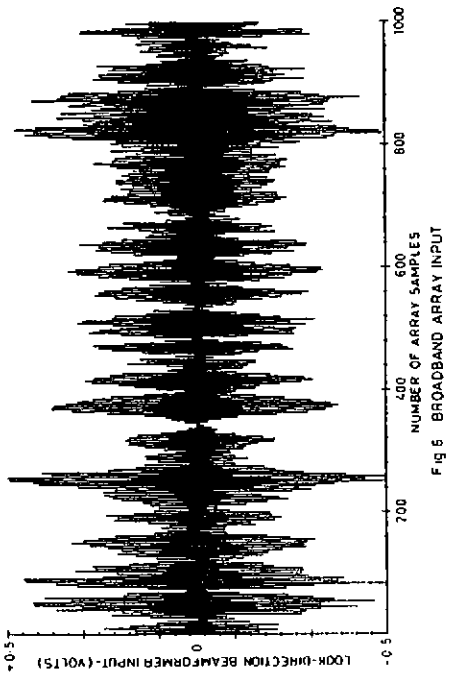

Fig 6 BROADBAND ARRAY INPUT

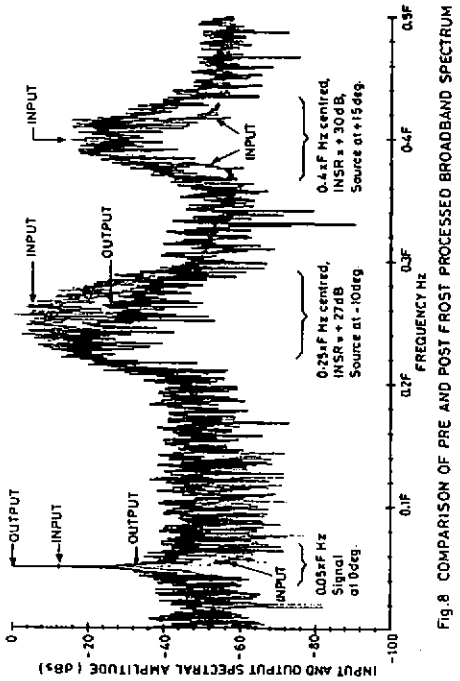

Fig.8 COMPARISON OF PRE AND POST FROST PROCESSED BROADBAND SPECTRUM

6.2 Broadband Scenario. The Frost response to a broadband input is shown in Fig 7. The input is illustrated in Fig 6. A couple of broadband noise sources centred at 0.25xF Hz and 0.4xF Hz are positioned at -10 and +15 degrees respectively, to the array's normal. The former source has an INSR of +27 dB (+24.7 dB) relative to the look-direction sinusoidal signal while the latter source has an INSR of +30 dB (+7.6 dB). The bandwidth of the first noise source is 0.04xF Hz and is double that of the other noise source. The 0.05xF Hz look direction signal is recovered after some 700 adaptions, see Fig 7. Unlike the narrowband case (Fig 4), the filtered output now exhibits noticeable interference breakthrough. This is to be expected since now the input has non-deterministic (noise source) components and the filter weights can only change to minimise the output error power by monitoring deviations within it. The resulting improvement in signal strength can be seen by comparing the signal spectrum at the output of the adaptive beamformer to that at its input, see Fig 8. The 0.05xF Hz signal has improved by about 15 dB relative to the 0.25xF Hz centred noise source and by about 13 dB relative to the other noise source. The 4096 sample spectra were taken after the output had converged. A 'mu' factor of 10^{-6} was used with J=8 taps/element. A reduction in J to 4 leads to similar results but with somewhat reduced suppression of the noise sources.

In cases such as the one illustrated above, additional noise suppression can be achieved if the signal and noise sources exhibit non-overlapping spectra. This can be achieved by imposing a bandpass (instead of an allpass) filter response centred around the signal frequency band. The desired filter response is specified through the frequency constraint vector FV (via vector FF in Eqn 2).

The Frost response to a pulsed, 0.1xF Hz sinusoidal signal in the presence of a broadband noise source is shown in Fig 9. Here, an interfering source with an INSR of +10 dB (+9.9 dB) is positioned at 5 degrees relative to the look-direction pulsed signal. The noise source has bandwidth of 0.02xF Hz which is also centred at 0.1xF Hz. The adaptively filtered output clearly registers the occurrence of the pulsed signal although both the sources have overlapping spectra. The non-deterministic input has a time-varying envelope (for example, see Fig 6). The pulsed signal would not have been detected so readily if the noise source had exhibited a constant amplitude envelope instead. The peaks in the detected pulse correspond to the nulls in the noise source envelope. It is in these 'interference-free' regions that the look-direction signal gets through without being cancelled as a result of spectral coherence. For fast convergence 'mu' was set to 5×10^{-5} with J=8.

7. ARCHITECTURAL ENHANCEMENTS

The throughput of the forward filter path decreases as the size of the input data matrix increases. This is a consequence of using a single MAC chip serially. This bottleneck can be reduced by employing MACs in parallel or by employing parallel, multi-tap, transversal architectures such as the non-canonical structure of the IMS A-100 filter chip from Inmos. Data throughput rate of several MHz can be realised with such VLSI chips. However, the rate of adaptation, stability and transient response of non-canonical structures are different from those of their canonical counterparts [8]. This behavioural difference stems from the inherent latency of the structure to flush intermediate product terms in response to filter weight updates.

The rate of convergence depends on the weight update period. Of the three adaptive weight generation techniques considered earlier, the more optimum ELS technique yields relatively longer weight update periods. Quicker updates may be achieved by employing special chips for implementing dedicated functions such as matrix inversion. These chips could then act as co-processors to the more general purpose, weight generation DSP chip. The development of fast matrix updating techniques such as the Hyperbolic Householder transformations [9] for multi-tap systems would also be beneficial in this respect.

8. CONCLUSIONS

The design and hardware implementation of an adaptive beamformer for sonar applications has been described. The single card digital beamformer has a simple but versatile transversal filter architecture which allows a variety of broadband adaptive beamforming techniques to be mapped. This multi-element, multi-tap filter architecture operates in conjunction with the latest floating-point DSP chip, the TMS320C30, which generates the filter weights adaptively. Initial results show the hardware performance of the forward filter path to be close to the initial 16 bit design.

Array processing algorithms requiring a diversity of computing complexity from matrix inversion to solving simple unconstrained LMS equations have been implemented. Data throughput rates of up to 500 KHz can be accommodated for a 36 tap, FIR filter configuration. This configuration comprises 9 transversal filters operating in parallel, each 4 taps long. The filter weights are updated recursively every 20 μsec using the Frost LMS algorithm. Mapping the more complex GSC algorithm onto the same hardware yields a weight update period of 60 μsec. Use of additional hardware to implement its non-adaptive sections can reduce this interval to 13 μsec. For an ELS mapping, a weight update period of 5 msec is achieved.

Suggestions for enhancing the performance of the constructed beam-processor card have also been made.

9. ACKNOWLEDGEMENTS

This work was funded under the 'Sonar Techniques' Private Venture Research Programme by Marconi Underwater Systems Ltd, Croxley, Watford. The authors thank the Technical Directorate of the Marconi Company for permission to publish this paper. Thanks are also due to Dr K C Sharman (currently of Glasgow University) for his advice during the course of this work. This work has been carried out with the support of the Procurement Executive, U.K. Ministry of Defence.

10. REFERENCES

[1] A A WINDER, "Underwater Sound - A Review: II Sonar System Technology", IEEE Trans Sonics and Ultrason, Vol SU 22(5), p291 (Sept 1975)

[2] A C SMITH, G C L SEARLE, "Empirical Observation of a Sonar Adaptive Array", IEE Proc F, Commun, Radar & Signal Processing, Vol 132(7), p595 (1985)

[3] O L FROST, III, "An Algorithm for Linearly Constrained Adaptive Array Processing", IEEE Proc, vol 60(8), p926 (Aug 1972)

[4] K M BUCKLEY and L J GRIFFITHS, "An Adaptive Generalised Sidelobe Canceller with Derivative Constraints", IEEE Trans Antennas and Propagation, Vol AP-34(3), p311 (March 1986)

[5] C R WARD, P J HARGRAVE and J G McWHIRTER, "A Novel Algorithm and Architecture for Adaptive Digital Beamforming", IEEE Trans Antennas and Propagation, Vol AP-34(3), p338 (March 1986)

[6] M C WELLS, "Floating Point Adaptive Processor for a Sonar Array Applications", UDT Conf Proc, session 7B, Sonar VI - Signal Processing, Undersea Defence Technology Conf, London, p395 (October 1988)

[7] G H GOLUB and C F VAN LOAN, "Matrix Computations" Baltimore MD: John Hopkins University Press, (1983)

[8] J J SORAGHAN, R W STEWART and T S DURRANI, "Implementation of the LMS Algorithm on Non-Canonical Transversal Filters", IEE colloqium (Groups E10 & E16) on "Digital Signal Processing for VLSI", Savoy Place, London, Digest No 1988/137 (December 1988)

[9] C M RADER, "Hyperbolic Householder Transformations", IEEE Trans, Acoustic, Speech and Signal Processing, Vol ASSP-34(6), p1589(December 1986)