eal-Time Adaptive Sidelobe Canceller Architectures nd Implem[illegible]

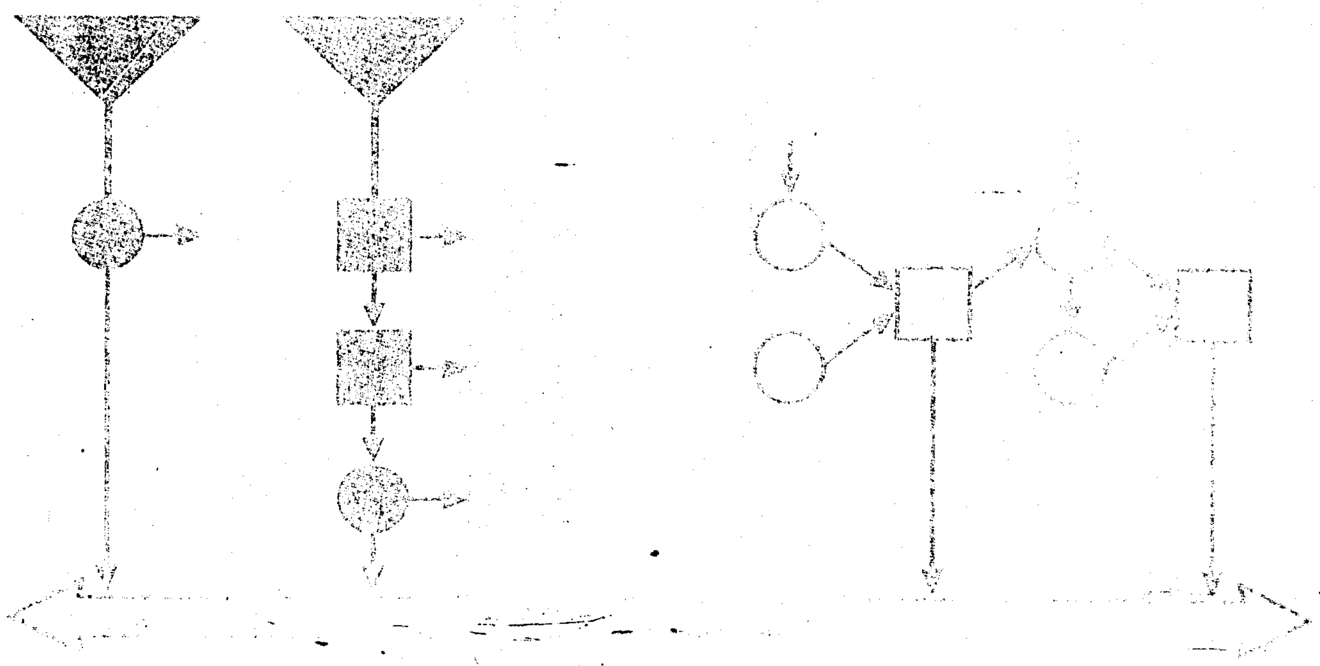
(1)

Terry L. Rorabaugh, John J. Vaccaro, Kevin H. Grace, Eric K. Pauer

Richard A. Games, Willard L. Eastman, Michael J. Sousa

July 1992

M91-60



MITRE

Bedford, Massachusetts

Real-Time Adaptive Sidelobe Canceller Architectures and Implementations

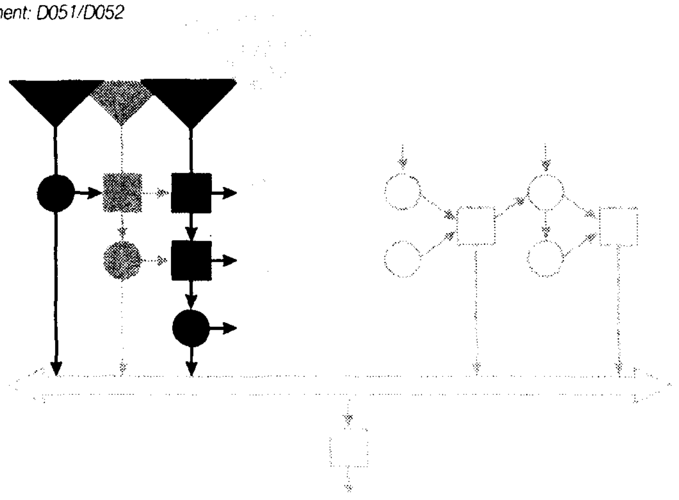
Terry L. Rorabaugh, John J. Vaccaro, Kevin H. Grace, Eric K. Pauer
Richard A. Games, Willard L. Eastman, Michael J. Sousa

July 1992

M91-60

A Publication in
Adaptive Signal Processing
by MITRE's Signal Processing Specialty Group

Contract Sponsor: Air Force Sponsored MOIE
Contract Number: F19628-89-C-0001
Project Number: 7030/7070
Department: D051/D052



Cover Illustration: Unconstrained sidelobe canceller implemented using the two-dimensional Gram-Schmidt architecture (shaded cells). Also shown (clear cells) is MITRE's add-on architecture for efficiently implementing constraints, used to prevent signal cancellation. Both architectures can be mapped to MITRE's modular digital signal processing array to provide a flexible real-time sidelobe cancellation experimental capability.

MITRE

Bedford, Massachusetts

PREFACE

This document contains two papers presented at the 1991 International Symposium on Optical Applied Science and Engineering held in San Diego, CA, on 21-26 July 1991. The symposium was sponsored by SPIE–The International Society for Optical Engineering. The papers were presented in Technical Conference 1566: Advanced Signal Processing Algorithms, Architectures, and Implementations II, and appear in the conference proceedings on pages 312-322 and 323-328, respectively.

The papers document results obtained under MITRE's wide-bandwidth high-frequency adaptive array processing research program. The first paper, entitled "A DSP Array for Real-time Adaptive Sidelobe Cancellation," describes MITRE's programmable, reconfigurable processing array, which implements a real-time adaptive sidelobe canceller using the Gram-Schmidt orthogonalization algorithm. The resulting unconstrained sidelobe canceller implementation is scalable , making it applicable to large antenna arrays that are used to form many simultaneous main beams. This processor also provides a near-term experimental capability and forms a core system for incorporating more advanced sidelobe cancellation techniques. The second paper, entitled "Fast Algorithm and Architecture for Constrained Adaptive Sidelobe Cancellation," describes an architecture to implement one such advanced technique – the inclusion of constraints to prevent cancellation of the desired signal. This paper describes an efficient algorithm and architecture for performing constrained processing using a separate main beam processor that is simply added to the existing core system.

A DSP Array for Real-time Adaptive Sidelobe Cancellation

Terry L. Rorabaugh, John J. Vaccaro, Kevin H. Grace, Eric K. Pauer

ABSTRACT

A programmable, reconfigurable digital signal processing (DSP) array has been designed in response to the need for a real-time adaptive sidelobe canceller to support wide-bandwidth high-frequency (HF) radar concepts. These concepts incorporate multiple (up to 128) main beams and many degrees of freedom by using many auxiliary antenna elements, and employ frequency sub-banding to partition a 1 MHz instantaneous bandwidth.

The real-time sidelobe canceller is based on the Gram-Schmidt orthogonalization procedure and uses concurrent block adaptation to derive an optimal solution for the available data. The sidelobe canceller implementation configures the modular DSP array into a two-dimensional Gram-Schmidt processor. A proof-of-concept sidelobe canceller implementation will be able to perform sidelobe cancellation on two simultaneous main beams using eight auxiliary channels. The DSP array sidelobe canceller can be expanded to use over 40 auxiliary channels to support over 128 main beams.

The array consists of TMS320C30-based processing nodes that provide four independent data ports (two inputs and two outputs) connected via high-speed serial data links that support 2-megaword-per-second sustained throughput rates (8-megaword-per-second burst rates). These manually configured data connections are separated from the control structure, allowing a variety of interconnection strategies (one-dimensional, two-dimensional, ring, etc.). A host processor is used to download application code and control the system. This programmable, reconfigurable array processor can also be used for a variety of other applications, including the singular value decomposition, matrix-matrix multipliers, and frequency transforms.

1. BACKGROUND

An experimental adaptive sidelobe canceller has been developed as part of MITRE's wide-bandwidth HF radar and communication research program. The supported surveillance radars will have many simultaneous main beams and will employ frequency sub-banding to partition a 1 MHz instantaneous bandwidth. The real-time sidelobe canceller uses concurrent block adaptation to derive an optimal solution for the available data.

An initial unconstrained sidelobe canceller will serve as a core for expanded capabilities, including linear null constraint processing[1] and reduced rank processing through the use of singular value decomposition[2] or iterated least squares.[3] To support these many research objectives, the processor is expandable in size (main beams and auxiliaries) and capability (algorithms supported) and is easily reconfigured.

The programmable, reconfigurable DSP array used to implement the adaptive sidelobe canceller is a cost-effective real-time processor that is ideal for the rapid prototyping requirements of a research and experimentation program. This rapid prototyping tool can be used to provide experimental test beds, off-line hardware accelerators, or actual implementations of commercial and military signal processors, reducing the hardware costs and development time for providing real-time processing capabilities. The DSP array is particularly suited for input/output (I/O) intensive applications that employ block-oriented processing (matrix-matrix operations, singular value decompositions, frequency transforms, etc.). Due to the wide applicability of the DSP array, this paper will focus on the attributes and motivation of the design and will present the adaptive sidelobe canceller implementation only as an illustrative example.

2. DSP ARRAY ARCHITECTURE

The DSP array's system architecture was motivated by the need to support many parallel input channels (over 30) of high-throughput complex data (64-bit words at 2 megawords per second). This intensive I/O requirement and the block-oriented structure of our signal processing applications led to the development of a cellular multiprocessor array architecture.

The cellular array architecture is designed for coarse grain processing, where task-oriented processing cells receive, process, and pass data vectors upon receiving a block synchronization signal. The blocks of data are processed synchronously at the intercellular (system) level but are processed asynchronously at the intracellular level. This cellular approach has advantages for both hardware and software. Hardware advantages include eliminating system-wide clock distribution requirements and the associated timing-skew problems across a large array. Additionally, the distributed (local) memory architecture afforded by the coarse grain processing eliminates the space and data bus bandwidth requirements of memory arbitration schemes. Software advantages include a simplified design from concentrating on task-oriented implementations and modular testing.

The DSP array architecture is partitioned into three subsystems: processing nodes, interface boards, and a host test/control processor, as shown in figure 1. The processing nodes are compact, concentrating their available board space on data processing. The nodes use high-performance serial data interconnection (two input and two output ports) to support high-throughput applications. These high-speed data links can be interconnected in a variety of configurations (one dimensional, two-dimensinal, ring, etc.), based on the application. The interface boards are processing nodes with added capabilities for interfacing with external systems (that use up to 32-bit parallel data) and with the test/control processor that serves as the user interface to the system. These subsystems are described in subsequent sections.

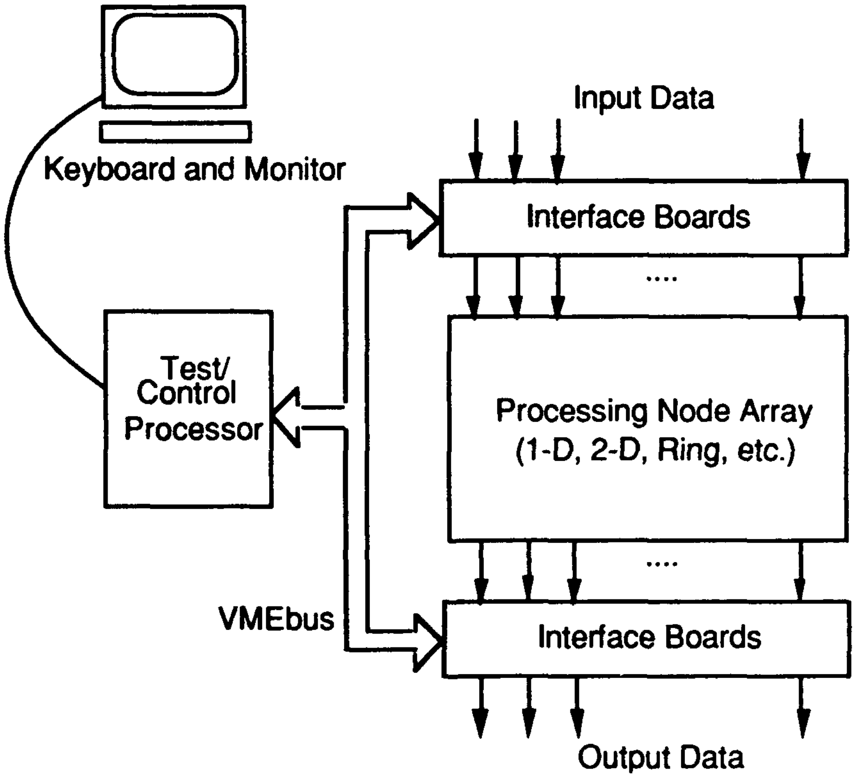


Figure 1. DSP Array Architecture

2.1. Processing Nodes

A single, small (4 by 9 inch) custom printed circuit board containing a single processing node was implemented to provide a modular building block approach to system expansion. The modular design also provides for small, line replaceable units, which allows fast, low-cost repairs. Our "simple," low-risk design philosophy led to the use of standard off-the-shelf parts.

The processing node (see figure 2) is based on a programmable DSP device (TMS320C30) selected primarily for its floating-point capability (33 million floating-point operations per second), dual external bus architecture, relatively low cost, and extensive development support. The processing and I/O throughputs are evenly matched through the use of four independent high-speed (500 megabit-per-second) serial data ports (labeled V1 and V2 in figure 2). These serial data paths are transmitted via differential pair to maintain electrical signal integrity. A photograph of the board is shown in figure 3.

The single-node board design provides testability and observability features. A loopback capability supports stand-alone functional testing and provides a power-up pass/fail diagnostic capability. An 80-pin edge connector presents all on-board data and control signals to the edge of the board for logic analyzer access during full system testing. These test points eliminate the need for extender cards and ease in-the-field testing. Memory mapped light emitting diodes (LEDs) provide user-definable status indicators for each node. Types of status currently reported include power-up pass/fail, serial-link connectivity, and (limited) error detection (see section 5).

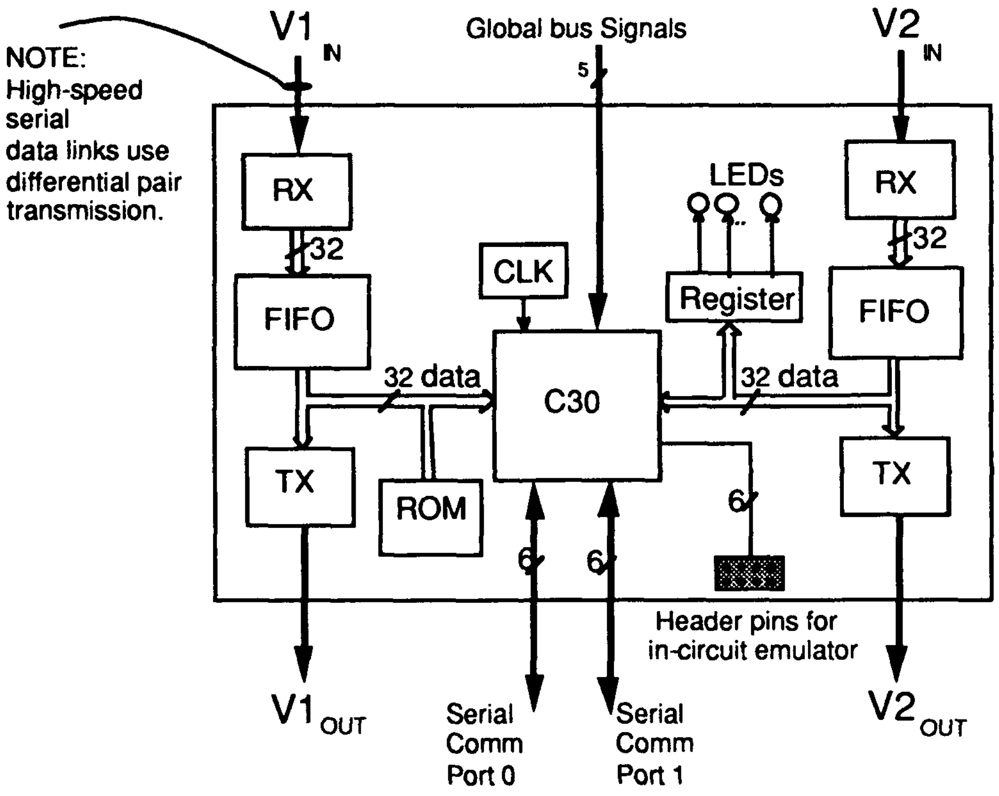


Figure 2. Processing Node Functional Block Diagram

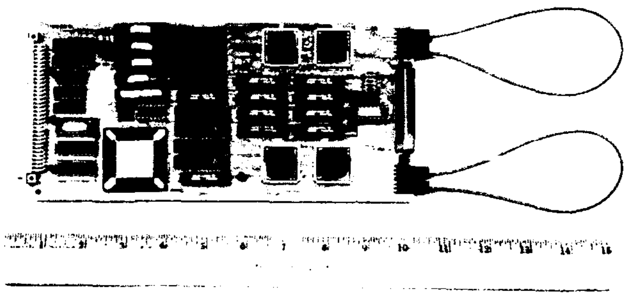


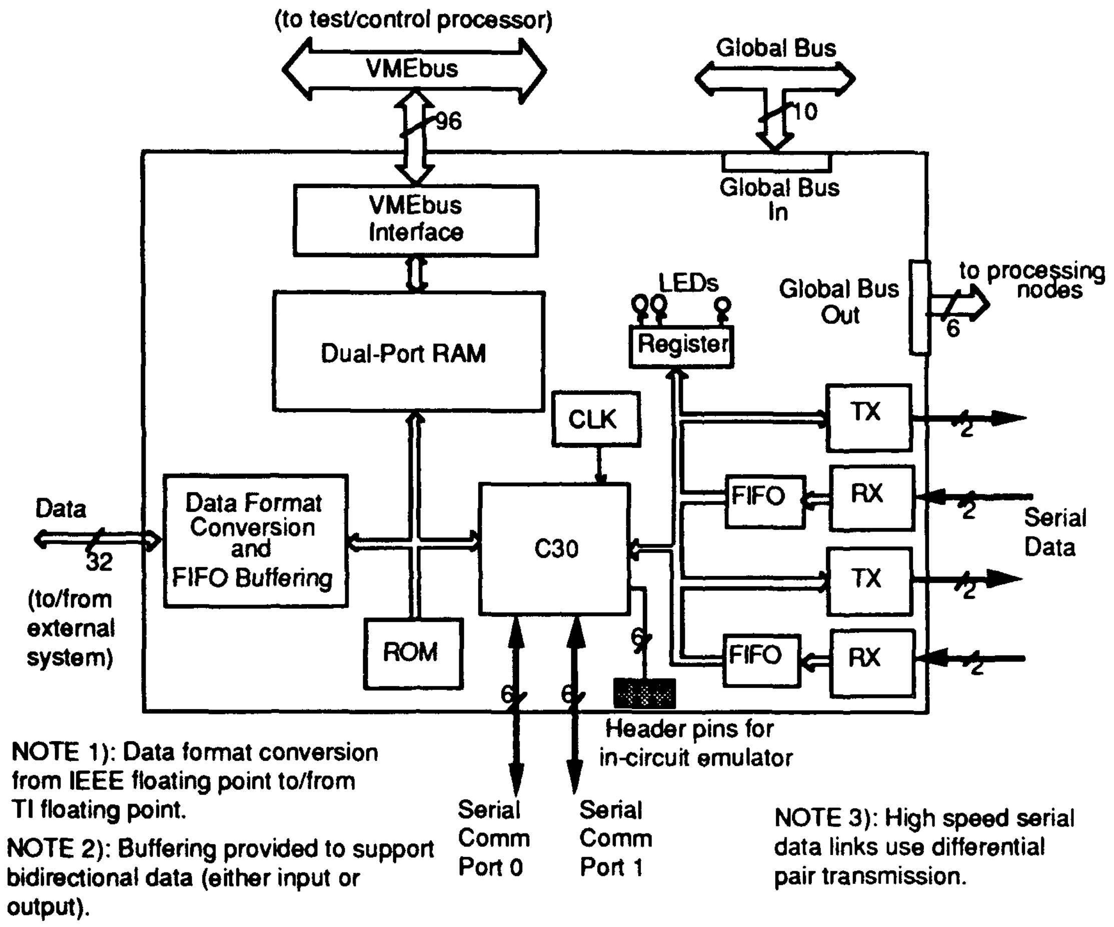
Figure 3. Processing Node Printed Circuit Board (4 by 9 inches)

2.2 Interface Boards

To meet our expansion requirements (potentially hundreds of nodes), the processing node design minimized board space requirements by concentrating on high-throughput processing. Interface boards were developed to provide capabilities for interfacing with external systems (see figure 4). The interface boards are processing nodes augmented with the following interface capabilities: (1) a standard commercial bus interface (VMEbus) to the host test/control processor, (2) a 32-bit parallel external data interface to serialize data sent to the processing node array, and (3) a data-format conversion capability supporting fixed-point or IEEE floating-point inputs.

The interface boards also use TMS320C30 processors and are configured by the test/control processor as either input or output boards, processing up to 32-bit data from/to an external system. The interface boards reside on the VMEbus and provide direct communication with the VMEbus-based test/control processor via the dual-port random-access memory (RAM). Bidirectional serial communication ports (Serial Port 0 and Port 1) are used to communicate with the processing nodes.

To facilitate expansion, the interface boards also provide a fan-out mechanism for global bus signals (six in all), reducing the electrical loading requirements for these signals across large array configurations.



(to test/control processor)
VMEbus
96
Global Bus
10
Global Bus In
VMEbus Interface
Dual-Port RAM
LEDs
Register
Global Bus Out
6
to processing nodes
CLK
TX
2
FIFO
RX
2
Serial Data
Data
32
(to/from external system)
Data Format Conversion and FIFO Buffering
C30
TX
2
FIFO
RX
2
ROM
6
6
6
Header pins for in-circuit emulator
NOTE 1): Data format conversion from IEEE floating point to/from TI floating point.
NOTE 2): Buffering provided to support bidirectional data (either input or output).
Serial Comm Port 0
Serial Comm Port 1
NOTE 3): High speed serial data links use differential pair transmission.

Figure 4. Interface Board Functional Block Diagram

2.3. Test/Control Processor

The test/control processor is an off-the-shelf, 80386SX (16 MHz) MS DOS-based computer that resides on the VMEbus. An 80486 (33 MHz) upgrade is planned for our real-time experimental work. This processor serves as a software development platform and runs the system software (see section 4.1), which provides a variety of DSP array system functions, including off-line array configuration, downloading of application code to all processing cells, and uploading of computational results for graphical/statistical displays.

The test/control processor adds to the flexibility of the DSP array by supporting a variety of system applications. As a hardware accelerator, the DSP array uses the test/control processor to input data and collect outputs via the VMEbus interface connection with the interface boards. As a real-time experimental system, the DSP array uses the test/control processor to capture live data inputs and corresponding computed outputs at a near real-time rate for data recording or display purposes. The test/control processor can also be used to test the DSP array system, supplying all data and control signals for stand-alone operation.

3. HARDWARE CONFIGURATION

The DSP array hardware configuration separates the high-speed serial data connections from the system control structure. This approach maximizes data throughput rates by offering full use of the available bandwidth for data connections, since communication and control signals use separate busses. The separate data and control configuration also adds flexibility in physical packaging by allowing for arbitrary physical data connections, since the physical location of the nodes are independent of their logical connection. The logical connections are taken into account during the off-line array configuration

3.1. Data Connections

The DSP array data connections are shown in figure 5. Input data vectors enter the array through the interface (input) boards. To provide flexibility for supporting a variety of applications, the input data can be obtained either from external systems or from the test/control processor via a download process. To support experimental work, computational results collected by the interface (output) boards can be uploaded to the test/control processor for analysis/display.

The processing nodes are interconnected via high-speed serial data links. A node's two input ports and two output ports support a variety of interconnection strategies. These manually reconfigured connections eliminate the expansion problems associated with typically used interconnection schemes of other parallel processing systems, such as crossbar switches, common busses, or dual-port RAMs. Miniature twin-axial cable is used for these data connections, eliminating the need for bulky, unreliable 32-bit parallel data connectors.

3.2. Control Connections

The control structure consists of daisy-chained serial communication links and global control signal busses (see figure 6). The test/control processor uses the VMEbus to pass messages (i.e., application code, status information) or global control signals to the interface boards. The messages are passed from the interface boards to the processing nodes via the serial communication links, which are daisy-chained across a physical subrack of nodes. Interface boards serve as a serial communication master controller for a particular subrack. Global control signals are distributed to the processing nodes via separate control busses that are also contained within a physical subrack. These control signals (six in all) are primarily processor interrupt signals used to synchronize the task-oriented processing.

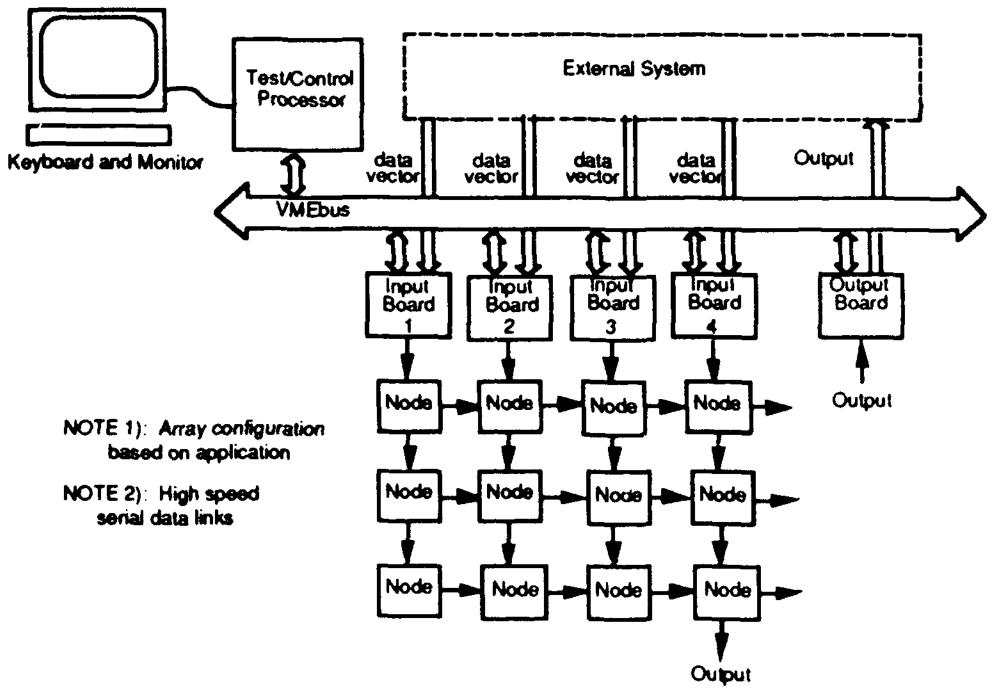


Figure 5. DSP Array Data Connection

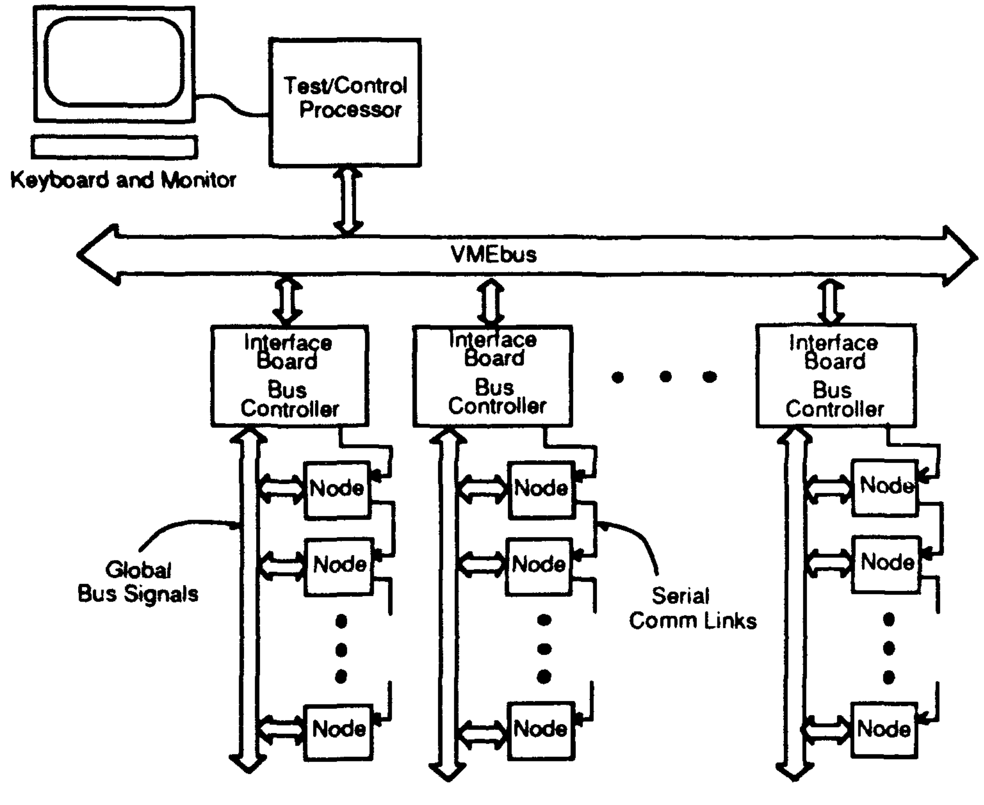

Keyboard and Monitor
Test/Control Processor
VMEbus
Interface Board Bus Controller
Interface Board Bus Controller
Interface Board Bus Controller
Node
Global Bus Signals
Serial Comm Links

Figure 6. DSP Array Control Connection

4. SOFTWARE DESIGN

Application algorithms being implemented with the DSP array must first be decomposed into a task-oriented structure amenable to block processing. Once the tasks and data flow have been established, the tasks can be mapped onto a processing node configuration. The software design was simplified by performing this algorithmic mapping off-line. Once the node configuration is established and the tasks are assigned, the system can operate on-line and proceed with array configuration (node ID assignments) and application code download procedures.

The software design falls into two major areas: system software and application software.

4.1. System Software

System software was developed to provide a user interface and an operating system for the DSP array. The operating system is used to download application code to the individual processing nodes, to control the operation of the array, and to periodically monitor its performance. The operating system, which is executed on the host processor, also uses utilities resident on the interface boards to perform various functions.

The system software was designed for modularity and flexibility, using current industry standards as much as possible. These goals were met by using an object oriented programming approach, which encourages the development of modular code that is reusable and relatively simple to expand. The system software was written using a popular C++ development package. The included class library served as a baseline from which we developed our own classes to implement the system software. These new classes were used as templates to create "objects" that represent the various types of hardware in the DSP array. The resulting operating system consists mainly of message passing among these system objects.

To provide flexibility for system users, a graphical user interface (GUI) was developed from standard class libraries. The GUI currently supports experimental data display and system status windows, and provides the flexibility for future (user-defined) additions and enhancements.

The remainder of the system software is executed by the interface boards and consists of utilities that provide communication between the host processor and the processing nodes. These utilities, written in TMS320C30 assembly language, are read-only-memory (ROM) based to maximize the RAM available for application purposes.

4.2. Application Software

The application-dependent software is developed off-line using TMS320C30 assembly language, a DSP array macro (assembly code) library, or the C language if the overhead introduced does not limit the desired real-time performance. There are two basic considerations for application software: it must be task oriented (block processing), and it must be able to meet the real-time throughput requirements of the application. This task-oriented software can be tested off-line with the TMS320C30 simulator and on-line with the emulator.

5. ADAPTIVE SIDELOBE CANCELLER IMPLEMENTATION

MITRE's real-time adaptive sidelobe canceller uses concurrent block adaptation to derive an optimal solution for the available data, which is collected into *m*-vectors from each of the *k* main beams and *n* auxiliary channels. The sidelobe cancellation problem can be stated as follows: for each main beam *m*-vector $\mathbf{y}_i$ where $i = 1, 2, \ldots, k$, compute the minimum power residual vector output, that is, solve the least squares problem

$$\min_{\mathbf{w}} \| \mathbf{z}_i \|^2 = \| \mathbf{y}_i - \mathbf{A}\mathbf{w}_i \|^2,$$

where $\| \mathbf{x} \|^2 = \mathbf{x}^H \mathbf{x}$ and H denotes the (hermitian) complex conjugate transpose operator. The *m*-by-*n* auxiliary data matrix **A** is composed of the *m*-vectors from each of the *n* auxiliary (antennae) channels, and $\mathbf{w}_i$ is the *n*-vector of optimal weights corresponding to the ith main beam. The adapted output (residual) *m*-vector for each main beam is

$$\mathbf{z}_i = \mathbf{y}_i - \mathbf{A}\mathbf{w}_i .$$

The residual can be directly computed by the QR decomposition of the matrix formed by appending the main beam vector $\mathbf{y}_i$ to the auxiliary data matrix **A**. The QR decomposition of this augmented matrix yields

$$[\mathbf{A} \quad \mathbf{y}_i] = [\mathbf{Q} \quad \mathbf{q}_{n+1}] \begin{bmatrix} \mathbf{R} & \mathbf{Q}^H \mathbf{y}_i \\ \mathbf{O} & \alpha \end{bmatrix}$$

where the residual is given by

$$\mathbf{z}_i = \mathbf{y}_i - \mathbf{Q}\mathbf{Q}^H\mathbf{y}_i = \alpha^{(i)}\mathbf{q}^{(i)}_{n+1}.$$

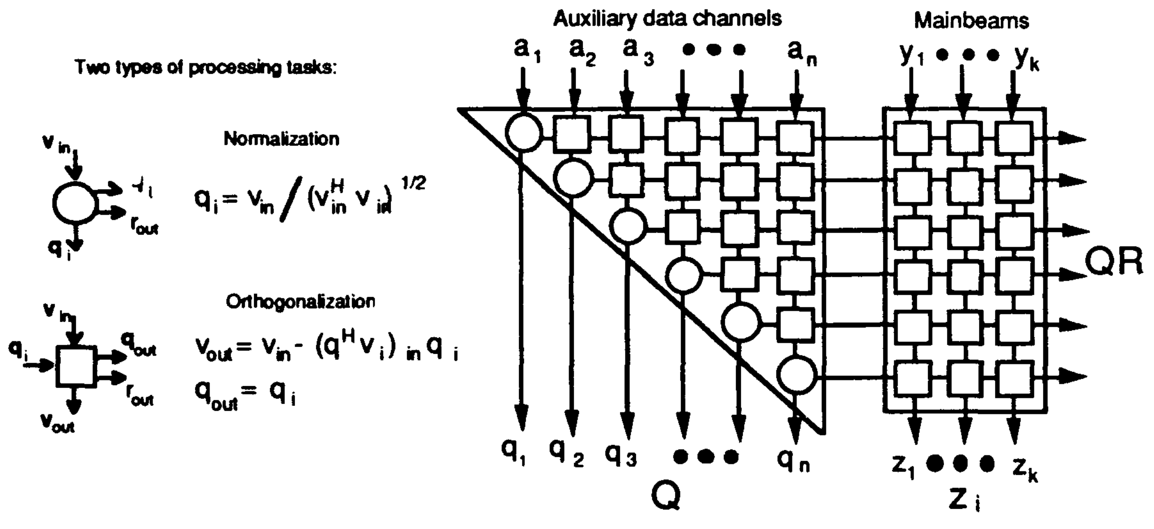
A Gram-Schmidt orthogonalization processor is used to perform the QR decomposition. The multiple main beam, augmented Gram-Schmidt orthogonalization algorithm is:

```
$\mathbf{Q} \longleftarrow [\mathbf{A} \,\vdots\, \mathbf{y}_1 \cdots \mathbf{y}_k]$
For j = 1 to n
    $r_{jj} \longleftarrow \sqrt{\mathbf{q}_j^H \mathbf{q}_j}$
    $b \longleftarrow r_{jj}^{-1}$
    $\mathbf{q}_j \longleftarrow b\mathbf{q}_j$
    For i = j+1 to n
        $r_{ji} \longleftarrow \mathbf{q}_j^H \mathbf{q}_i$
        $\mathbf{q}_i \longleftarrow \mathbf{q}_i - r_{ji}\mathbf{q}_j$
    endfor
endfor
$\mathbf{z}_i = \mathbf{y}_i - \mathbf{Q}^H\mathbf{Q}\,\mathbf{y}_i = \alpha^{(i)}_{n+1}$
```

Each residual is a transformation of the corresponding main beam (less normalization) via the same transformation that maps the ith auxiliary into $\mathbf{q}_i$. The square root operation is performed in order to obtain orthonormal $\mathbf{q}_i$'s for enhancements (e. g. constraint processing).

5.1. Two-Dimensional Sidelobe Canceller Architecture

To facilitate system expansion (additional main beams and/or auxiliaries) and distribute the processing tasks, a two-dimensional architecture for performing this augmented Gram-Schmidt orthogonalization was used for implementation (see figure 7). The architecture is very flexible, being completely scalable in problem size and having the timing requirements per processing task depend only on the input data rate and not on the number of input channels. Each processing task in the two-dimensional orthogonalization architecture is mapped onto a single DSP array processing node, which implements one of the two simple functions indicated in figure 7. The processing node's configuration is matched to the triangular structure of the orthogonalization architecture. Due to the DSP array's separation of data structure from control, the physical system is free to optimize volume, unrestricted by the algorithm's triangular data flow structure.



Two types of processing tasks:
Normalization
$q_i = v_{in} / (v_{in}^H v_{in})^{1/2}$
Orthogonalization
$v_{out} = v_{in} - (q^H v_i)_{in} q_i$
$q_{out} = q_i$
v_{in}
q_i
r_{out}
q_{out}
v_{out}
Auxiliary data channels
a_1 a_2 a_3 ••• a_n
Mainbeams
y_1 ••• y_k
QR
q_1 q_2 q_3 ••• q_n
Q
z_1 ••• z_k
z_i

Figure 7. Two-Dimensional Gram-Schmidt Orthogonalization Architecture

The architecture is pipelined in both dimensions, with the time available for real-time block processing equal to *m* times the input sample rate (time to collect an input vector). A strobe signal is used to discriminate blocks (input vectors) of data. This strobe signal is used to implement the block synchronization mechanism described earlier.

Our initial unconstrained, two-auxiliary-channel sidelobe canceller implementation is shown in figure 8. The architecture's two-dimensional pipeline requires the interface (input) boards to perform data alignment on vectors that enter the first row (refer to figure 7). Interface (output) boards are used to output each residual and to provide tagged data to the host/controller for data archiving or interactive display. Additional interface boards are used to collect intermediate results (q and r vectors), allowing display of computed weights or nulled beam patterns.

MITRE's wide-bandwidth experimental HF system requires a 2.048 MHz sample rate with a length of 128 complex data vectors. Despite the simple algorithmic requirements, a single processing element cannot meet the required 62.5 microsecond throughput rate. The interface boards have two output ports for multiplexing between two separate two-dimensional arrays (doubling the time available for real-time processing) thereby satisfying the required throughput rate.

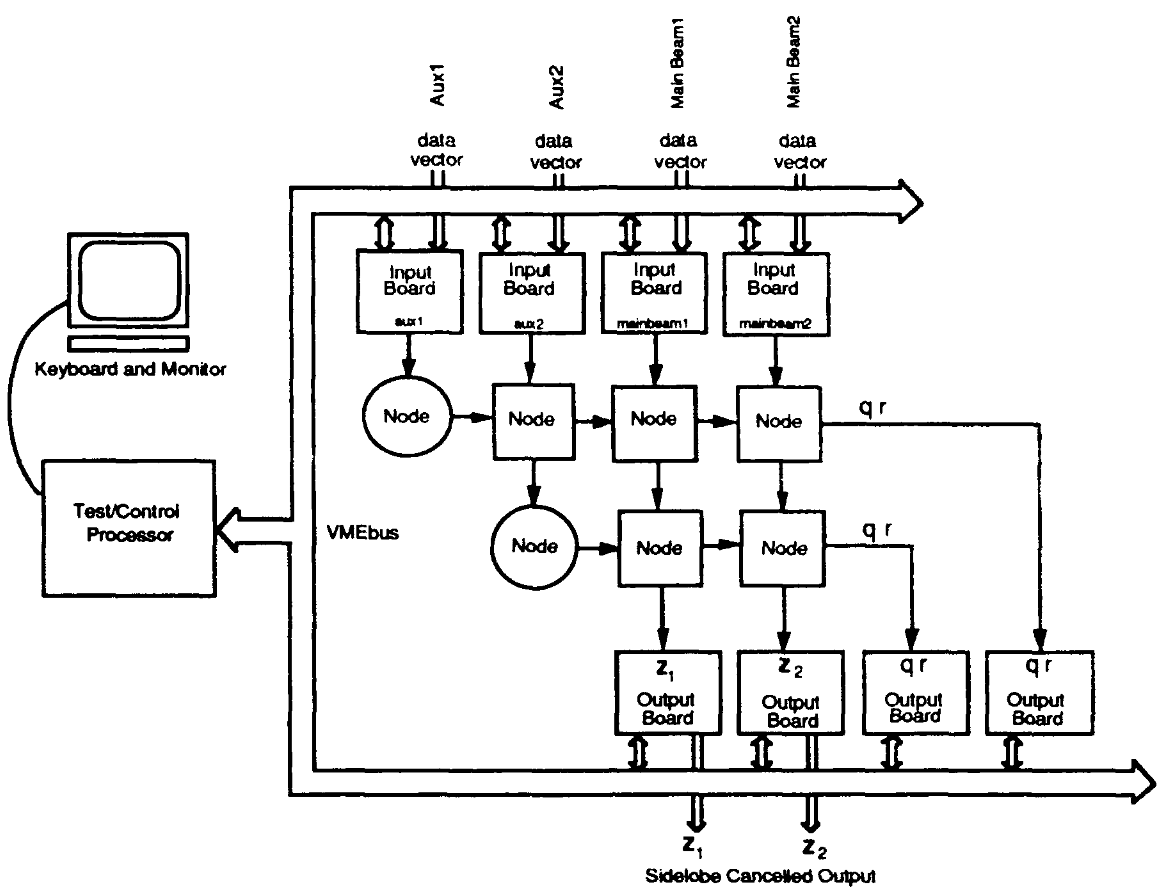


Figure 8. Two-Dimensional Sidelobe Canceller Configuration

On-line system testing is performed by appending small (eight-sample) test vectors onto the live input data vectors. Sufficient time exists to perform the orthogonalization on the appended vectors, compare the computed results to expected results, and report errors. This serves as a limited concurrent error detection strategy. The DSP array architecture inherently supports the use of algebraic checksum matrices,[4] where addition of a single column to figure 8 provides concurrent error detection for the whole system. The precomputed approach was selected for the initial sidelobe canceller application because of the resource time available, the elimination of redundant hardware, and the statistical nature of the checksum matrix strategies due to finite precision effects. However, other applications using the DSP array may take advantage of algebraic checksums.

5.2. One-Dimensional Sidelobe Canceller Architecture

The DSP array has the flexibility to support alternative sidelobe canceller configurations. For lower throughput rates, a one-dimensional Gram-Schmidt orthogonalization architecture can be implemented (see figure 9). The processing tasks for each node in the one-dimensional configuration encompass an entire column of processing tasks from the two-dimensional architecture in figure 7. Intermediate results can be obtained by including interface (output) boards connected to each processing node. These results can be used to evaluate the improvement in cancellation performance for each additional auxiliary channel. Additional flexibility is provided by using the interface boards to provide finite impulse response (FIR) filtering (or discrete Fourier transforms (DFTs)) to isolate frequency bands of interest from the (lower throughput rate) input data.

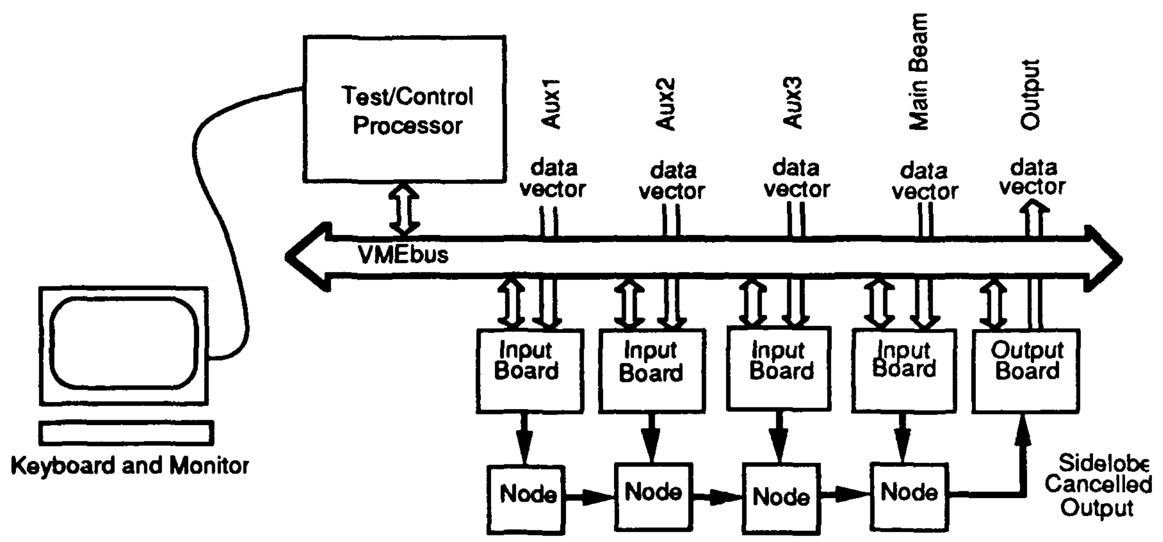


Figure 9. One-Dimensional Sidelobe Canceller Configuration

6. SUMMARY

The programmable, reconfigurable processing array developed at MITRE combines the attractive features of both fine and coarse grain architectures, providing powerful (33 million floating-point operations per second) processing nodes that can be combined in a variety of parallel-pipelined configurations to form large, powerful systems (billions of floating-point operations per second) utilizing high-bandwidth interconnections (64 megawords per second). The resulting system fills a gap in existing commercially available processors, combining powerful processors with high I/O bandwidth and simple reconfiguration at a relatively low cost.

The described DSP array is being used to implement a real-time adaptive sidelobe canceller for performing HF radar and communication experiments. The unconstrained sidelobe canceller performs direct residual computation by the Gram-Schmidt orthogonalization of an augmented auxiliary data matrix. The DSP array is configured to implement a two-dimensional Gram-Schmidt orthogonalization processor. The resulting unconstrained sidelobe canceller forms a core system we can expand to accommodate additional main beams and/or auxiliary channels, and upon which we can incorporate more advanced sidelobe cancellation techniques.

REFERENCES

1. Eastman, W. L., Games, R. A., Sousa, M. J.,"Fast Algorithm and Architecture for Constrained Adaptive Sidelobe Cancellation," SPIE Proceedings, July 1991.
2. Carhoun, D. O., Games, R. A., Williams, R. T., "A Principal Components Sidelobe Cancellation Algorithm," Proceedings of the 24th Annual Asilomar Conference on Signals, 5-7 November 1990.
3. Eastman, W. L., Games, R. A., Sousa, M. J., "Multiple Main Beam Sidelobe Cancellation Architecture for Constrained and/or Excess Degrees of Freedom Processing," Proceedings of ICASSP '91, 14-17 May 1991.
4. Huang, K. H., Abraham, J. A., "Algorithm-Based Fault Tolerance for Matrix Operations," IEEE Transactions on Computers, Vol. 33, No. 6, June 1984.

Fast Algorithm and Architecture for Constrained Adaptive Sidelobe Cancellation

Richard A. Games, Willard L. Eastman, and Michael J. Sousa

ABSTRACT

This paper describes an efficient implementation of auxiliary constraints for a concurrent block least squares adaptive sidelobe canceller when a single array of sensors is used to form one or more main beams. The approach is to compute a QR decomposition of the auxiliary data matrix and then send this information to main beam processors, where the constraints are applied using a blocking matrix and the individual residuals are computed. The blocking matrix can be chosen with a special structure, which is used to derive a new fast algorithm and architecture for constrained main beam processing that reduces the operation count from order n^3 to order n^2, where n is the number of auxiliary sensors.

1. INTRODUCTION

Adaptive sidelobe cancellers are used in a variety of applications to eliminate spatially coherent interference. A wideband system being developed at MITRE performs cancellation in the frequency domain in subbands determined by applying the fast Fourier transform to blocks of data. In this context it is natural to perform concurrent block processing in which the adaptive weights of the auxiliary sensors are computed from and applied to the same block of data. Signal cancellation can occur, however, if the desired signal is strong. This can be prevented by constraining the selection of auxiliary weights.[1] In this paper we efficiently implement constraints for a single array of sensors that simultaneously forms one or more main beams.

2. THE SIDELOBE CANCELLATION PROBLEM

The sidelobe cancellation system, shown in figure 1, contains two branches: (1) the primary element, corresponding to the output of a single sensor or a weighted sum of outputs from an array of sensors designed to have high relative gain in the direction of the desired signal, and (2) the auxiliary array output, a weighted sum of outputs from n sensors. If the primary element is obtained from an array of sensors, then multiple main beams pointing in different directions can be simultaneously formed. The auxiliary array weights are determined adaptively in an attempt to cancel unwanted interference in the primary element. The resulting output, or residual, is formed by subtracting the auxiliary array output from the response of the primary element.

The cancellation is accomplished by dividing the stream of data samples measured by both the primary element and the auxiliary array sensors into blocks of length m. Let $\mathbf{A}$ denote the $m \times n$, $m \geq n$, complex auxiliary data matrix, which we assume has full column rank n due to background noise; the m-vector obtained from the primary element will be denoted by $\mathbf{y}$. The adaptive weights are computed by minimizing the output power of the sidelobe canceller over each block using standard least squares methods. The adaptive weights are subsequently applied to the data block from which they were computed.

Because the desired signal is present during adaptation, it may also be cancelled. To prevent this, we apply a constraint of the form

$$\mathbf{s}_1^H \mathbf{w} = 0,$$

where $\mathbf{s}_1$ is the auxiliary array steering vector of the desired signal, $\mathbf{w}$ is the auxiliary array adaptive weight vector, and H denotes the conjugate transpose operator. This constraint insures that the adapted auxiliary antenna will have zero gain in the direction of the desired signal. Thus the signal component of the primary element sample vector will be unaffected when the residual is formed, provided the desired signal is uncorrelated with the interferers and auxiliary noise.

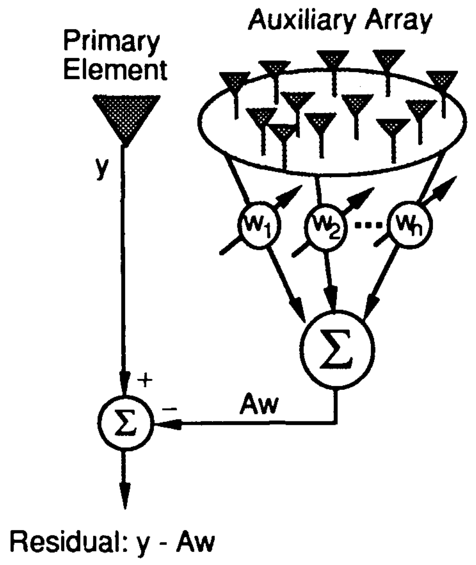


Figure 1. Sidelobe Canceller Array Configuration

Other linear null constraints of the form $\mathbf{s}_i^H \mathbf{w} = 0$ may also be applied, for example, when the direction of the desired signal is not known precisely. A set of p linear null constraints, where p is small relative to n, is collected in the $n \times p$ constraint matrix

$$\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_p].$$

3. LEAST SQUARES PROCESSING

To solve the concurrent block sidelobe cancellation problem for a single main beam, we compute the residual of the least squares problem

$$\min_{\mathbf{w}} ||\mathbf{y} - \mathbf{A}\mathbf{w}||_2^2. \tag{1}$$

It is well known that the residual of problem (1) can be computed directly from the orthogonal projection onto the column space of $\mathbf{A}$. If $\mathbf{P_A}$ denotes this orthogonal projection, then the residual is given by

$$\mathbf{z} = \mathbf{y} - \mathbf{P_A}\mathbf{y}. \tag{2}$$

The orthogonal projection $\mathbf{P_A}$ can be determined from an orthonormal basis of the column space of $\mathbf{A}$. If the columns of $\mathbf{Q}_1 = [\mathbf{q}_1 \quad \mathbf{q}_2 \quad \cdots \quad \mathbf{q}_n]$ form an orthonormal basis found by computing a QR decomposition of $\mathbf{A}$, then $\mathbf{P_A} = \mathbf{Q}_1\mathbf{Q}_1^H$. Either the modified Gram-Schmidt or a plane rotation method can be used to compute the QR decomposition using a triangular systolic array, although the Gram-Schmidt method is particulary well suited to this block processing case.

The solution given in equation (2) does not require the explicit computation and application of an optimal weight vector, and can be viewed as the block generalization of this same result for the scalar case.[2] Furthermore, the orthogonal projection $\mathbf{P_A}$ depends only on the auxiliary data matrix $\mathbf{A}$. It can be computed once and then applied to each main beam vector $\mathbf{y}$ in a multiple main beam system. The multiple main beam architecture consists of an auxiliary processor that computes the QR decomposition of $\mathbf{A}$, together with a set of main beam processors. Each main beam processor accepts $\mathbf{Q}_1$ and computes the residual $\mathbf{z}^{\mathbf{y}} = \mathbf{y} - \mathbf{Q}_1\mathbf{Q}_1^H\mathbf{y}$ for its m-vector $\mathbf{y}$.

The residual $\mathbf{z}^{\mathbf{y}}$ can be computed from the QR decomposition of the augmented matrix $\hat{\mathbf{A}} = [\mathbf{A} \quad \mathbf{y}]$, formed by adjoining the vector $\mathbf{y}$ to $\mathbf{A}$. If the modified Gram-Schmidt algorithm is applied to $\hat{\mathbf{A}}$, the residual is produced at the last stage of the algorithm (corresponding to the last column of $\hat{\mathbf{A}}$). The final normalization is omitted. Computing the residual in this way involves $m(n+1)^2$ complex multiply and accumulate operations (CMACs).

4. CONSTRAINED PROCESSING

In this section we describe an efficient partitioning for including constraints in the concurrent block least squares sidelobe canceller. The null-constrained least squares formulation is given by

$$\min_{\substack{\mathbf{w} \\ \mathbf{S}^H\mathbf{w}=\mathbf{0}}} \|\mathbf{y} - \mathbf{A}\mathbf{w}\|_2^2, \tag{3}$$

where $\mathbf{S}$ is an $n \times p$ matrix whose columns consist of p steering vectors in the constraint directions. We assume that $\mathbf{S}$ has full column rank p. To solve problem (3) using the blocking matrix approach,[1] vectors orthogonal to the column space of $\mathbf{S}$ are needed. A basis of such vectors can be obtained from the QR decomposition of $\mathbf{S}$. Specifically, if the $n \times p$ constraint matrix is decomposed as

$$\mathbf{S} = [\,\mathbf{Q}_{c1} \quad \mathbf{Q}_{c2}\,] \begin{bmatrix} \mathbf{R}_c \\ \mathbf{0} \end{bmatrix},$$

then the columns of the $n \times (n-p)$ matrix $\mathbf{Q}_{c2}$ form an orthonormal basis for the space orthogonal to the column space of $\mathbf{S}$. The matrix $\mathbf{Q}_{c2}$ corresponds to an appropriate blocking matrix.

The constraints are applied by forming the matrix product $\mathbf{A}\mathbf{Q}_{c2}$, thus eliminating signals from the constraint directions. The transformed auxiliary data matrix $\mathbf{A}\mathbf{Q}_{c2}$ can be viewed as the data matrix of an unconstrained sidelobe canceller with $n-p$ auxiliary antennas. The corresponding unconstrained least squares minimization problem becomes

$$\min_{\mathbf{u}} \|\mathbf{y} - \mathbf{A}\mathbf{Q}_{c2}\mathbf{u}\|_2^2, \tag{4}$$

where $\mathbf{u}$ is a vector of $n-p$ weights for the modified auxiliary data. This approach is a standard method for solving least squares problems with linear constraints.[3]

In a multiple main beam system, each main beam vector $\mathbf{y}$ will have a distinct constraint matrix $\mathbf{S}^{\mathbf{y}}$ with a corresponding $n \times (n-p)$ preprocessing matrix $\mathbf{Q}_{c2}^{\mathbf{y}}$. We use a superscript $\mathbf{y}$ to denote quantities that depend on the main beam. Thus, there are as many $m \times (n-p)$ transformed auxiliary data matrices $\mathbf{A}\mathbf{Q}_{c2}^{\mathbf{y}}$ as there are main beams. The straightforward approach for solving these distinct unconstrained sidelobe cancellation problems would require a QR decomposition of each $m \times (n-p)$ matrix. However, it is more efficient to apply the constraints after performing a QR decomposition on the auxiliary data matrix.

Suppose $\mathbf{A} = \mathbf{Q}_1\mathbf{R}_1$ is a QR decomposition of the auxiliary data matrix $\mathbf{A}$. Then write $\mathbf{A}\mathbf{Q}_{c2}^{\mathbf{y}} = \mathbf{Q}_1\mathbf{R}_1\mathbf{Q}_{c2}^{\mathbf{y}} = \mathbf{Q}_1\mathbf{V}^{\mathbf{y}}$, where $\mathbf{V}^{\mathbf{y}} = \mathbf{R}_1\mathbf{Q}_{c2}^{\mathbf{y}}$. The QR decomposition of $\mathbf{A}\mathbf{Q}_{c2}^{\mathbf{y}}$ can be completed by performing a QR decomposition of $\mathbf{V}^{\mathbf{y}}$, a smaller matrix with dimensions $n \times (n-p)$. If $\mathbf{V}^{\mathbf{y}} = \mathbf{Q}_1^{\mathbf{y}}\mathbf{R}_1^{\mathbf{y}}$ is a QR decomposition, then $\mathbf{A}\mathbf{Q}_{c2}^{\mathbf{y}} = (\mathbf{Q}_1\mathbf{Q}_1^{\mathbf{y}})\mathbf{R}_1^{\mathbf{y}}$ is a QR decomposition of $\mathbf{A}\mathbf{Q}_{c2}^{\mathbf{y}}$. The residual is given by

$$\mathbf{z}^{\mathbf{y}} = \mathbf{y} - \mathbf{Q}_1\mathbf{Q}_1^{\mathbf{y}}\mathbf{Q}_1^{\mathbf{y}H}\mathbf{Q}_1^H\mathbf{y}. \tag{5}$$

For k main beams, the straightforward approach requires $kmn(n-p)$ CMAC operations for the k matrix-matrix products $\mathbf{AQ}_{c2}^{\mathbf{y}}$, and $km(n-p+1)^2$ CMACs to determine the residuals using the QR decompositions of the augmented data matrices.

The approach described in this section requires mn^2 CMACs for the initial QR decomposition of the auxiliary data matrix, kmn CMACs to form the inner products $\mathbf{Q}_1^{\mathrm{H}}\mathbf{y}$, $kn^2(n-p)$ CMACs for the matrix-matrix products $\mathbf{R}_1\mathbf{Q}_{c2}^{\mathbf{y}}$, $kn(n-p+1)^2$ CMACs to form the main beam projections $\mathbf{Q}_1^{\mathbf{y}}\mathbf{Q}_1^{\mathbf{y}\mathrm{H}}\mathbf{Q}_1^{\mathrm{H}}\mathbf{y}$, and kmn CMACs for the matrix-vector products required to form the final residuals.

The above operation counts indicate two points where the approach in this section will reduce the number of operations required—the k matrix multiplications by $\mathbf{Q}_{c2}^{\mathbf{y}}$ and the k QR decompositions involve substantially smaller matrices, since n is usually considerably smaller than m. In fact, for $p=1$ and $m=3n$ (resp. $m=4n$), the approach that separates the auxiliary and main beam processing requires fewer operations than the straightforward approach even with one main beam when $n>8$ (resp. $n>6$). The savings increase as the number of main beams increases.

5. FAST ALGORITHM AND ARCHITECTURE

In this section we further reduce the complexity of constrained processing by exploiting a structured blocking matrix. Forming $\mathbf{V}^{\mathbf{y}} = \mathbf{R}_1\mathbf{Q}_{c2}^{\mathbf{y}}$ and computing its QR decomposition to obtain $\mathbf{Q}_1^{\mathbf{y}}$ both require order n^3 CMACs. However, the special structures of the matrices $\mathbf{R}_1$ and $\mathbf{Q}_{c2}^{\mathbf{y}}$ allow us to obtain $\mathbf{Q}_1^{\mathbf{y}}$ using Givens or fast Givens plane rotations in order n^2 operations.

The matrix $\mathbf{R}_1$ is upper-triangular, and the matrix $\mathbf{Q}_{c2}^{\mathbf{y}}$ can be computed using plane rotations, as in recursive least squares,[4] to have the form

$$\mathbf{Q}_{c2}^{\mathbf{y}} = \begin{bmatrix} \mathbf{T} \\ \mathbf{U} \end{bmatrix},$$

where $\mathbf{T}$ is $p\times(n-p)$, and $\mathbf{U}$ is an $(n-p)\times(n-p)$ upper-triangular matrix. Consequently, the matrix $\mathbf{V}^{\mathbf{y}}$ has the same special form as $\mathbf{Q}_{c2}^{\mathbf{y}}$ and can be reduced to upper-triangular form by a sequence of $p(n-p)$ plane rotations, each of which takes order n operations. Since p is much less than n, this QR decomposition requires order n^2 operations.

Even with the special structure, the multiplication of $\mathbf{R}_1$ and $\mathbf{Q}_{c2}^{\mathbf{y}}$ requires order n^3 operations. The new approach (outlined below) preserves the advantage afforded by the Givens technique by never explicitly forming all of $\mathbf{V}^{\mathbf{y}}$. Instead we shall compute only the elements of $\mathbf{V}^{\mathbf{y}}$ needed to obtain the rotations, namely the $n-p$ diagonal elements $v_{jj}^{\mathbf{y}}$ and the $p(n-p)$ subdiagonal elements $v_{lj}^{\mathbf{y}}$, for $l=j+1$ to $j+p$, that are to be zeroed out by the Givens rotations. The rotation parameters are applied to the pertinent rows $\mathbf{r}_j$ and $\mathbf{r}_l$ of $\mathbf{R}_1$, but need not be applied to elements of $\mathbf{V}^{\mathbf{y}}$ other than $v_{jj}^{\mathbf{y}}$. With this modification the total number of operations required to compute $\mathbf{Q}_1^{\mathbf{y}}$ drops to order n^2.

FAST ALGORITHM. *Let $\mathbf{r}_1^{\mathrm{H}},\ldots,\mathbf{r}_n^{\mathrm{H}}$ be the rows of $\mathbf{R}_1$ and $\mathbf{q}_1^{\mathbf{y}},\ldots,\mathbf{q}_{n-p}^{\mathbf{y}}$ be the columns of $\mathbf{Q}_{c2}^{\mathbf{y}}$. The following procedure produces $\mathbf{Q}_1^{\mathbf{y}}$ in factored form.*

```
for $j = 1$ to $n-p$
    $\mathbf{v}_{jj}^{\mathbf{y}} \leftarrow \mathbf{r}_j^{\mathrm{H}}\mathbf{q}_j$
    for $l = j+1$ to $j+p$
        $\mathbf{v}_{lj}^{\mathbf{y}} \leftarrow \mathbf{r}_l^{\mathrm{H}}\mathbf{q}_j$
```

```
compute rotation parameters $c, s$
$\mathbf{v}^{\mathbf{y}}_{jj} \leftarrow [c \quad s] \begin{bmatrix} \mathbf{v}^{\mathbf{y}}_{jj} \\ \mathbf{v}^{\mathbf{y}}_{lj} \end{bmatrix}$
$\begin{bmatrix} \mathbf{r}^{\mathrm{H}}_{j} \\ \mathbf{r}^{\mathrm{H}}_{l} \end{bmatrix} \leftarrow \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} \mathbf{r}^{\mathrm{H}}_{j} \\ \mathbf{r}^{\mathrm{H}}_{l} \end{bmatrix}$
store rotation parameters
```

The constrained main beam processor is formed by incorporating an integrated matrix multiplication-plane rotation processor into the unconstrained main beam processor. The integrated processor, which is illustrated in figure 2, is a two-dimensional array made up of $n - p$ columnar subarrays. Each columnar subarray consists of a column of $p + 1$ cells (circles in figure 2) that compute the inner product v_{ij} of a row $\mathbf{r}_i$ of $\mathbf{R}_1$ with a column $\mathbf{q}_j$ of $\mathbf{Q}^{\mathbf{y}}_{c2}$, followed by a column of p cells (squares) that compute and apply the rotation parameters. The rows of $\mathbf{R}_1$ are entered at the left and across the bottom of the array; the columns of $\mathbf{Q}^{\mathbf{y}}_{c2}$ are entered at the top of the array. The updated rows of $\mathbf{R}_1$ are passed out of the array from the rotation cells across the bottom and at the right side. By augmenting the matrix $\mathbf{R}_1$ with the column vector $\mathbf{Q}^{\mathrm{H}}_1\mathbf{y}$, formed as in the unconstrained case, the array can compute $\mathbf{Q}^{\mathbf{y}\mathrm{H}}_1\mathbf{Q}^{\mathrm{H}}_1\mathbf{y}$, which is passed out at the bottom of the array as the last components of the rows $\mathbf{r}_i$ for $i = 1$ to $n - p$. The computed rotation parameters must be passed out of the array to a buffer (not shown in the figure) so that the Hermitians of the rotations can be applied in reverse order to form $\mathbf{Q}^{\mathbf{y}}_1\mathbf{Q}^{\mathbf{y}\mathrm{H}}_1\mathbf{Q}^{\mathrm{H}}_1\mathbf{y}$. The residual in equation (5) can then be formed as in the unconstrained case.

REFERENCES

1. L. J. Griffiths and C. W. Jim, "An Alternative Approach to Linearly Constrained Adaptive Beamforming," *IEEE Transactions on Antennas and Propagation*, Vol. AP-30, pp. 27–33, 1982.

2. C. R. Ward, P. J. Hargrave, and J. G. McWhirter, "A Novel Algorithm and Architecture for Adaptive Digital Beamforming," *IEEE Transactions on Antennas and Propagation*, Vol. AP-34, pp. 338–346, 1986.

3. G. H. Golub and C. F. Van Loan, *Matrix Computations*, Baltimore, MD: The Johns Hopkins University Press, 1983.

4. S. Haykin, *Adaptive Filter Theory*, Englewood Cliffs, NJ: Prentice Hall, 1986.

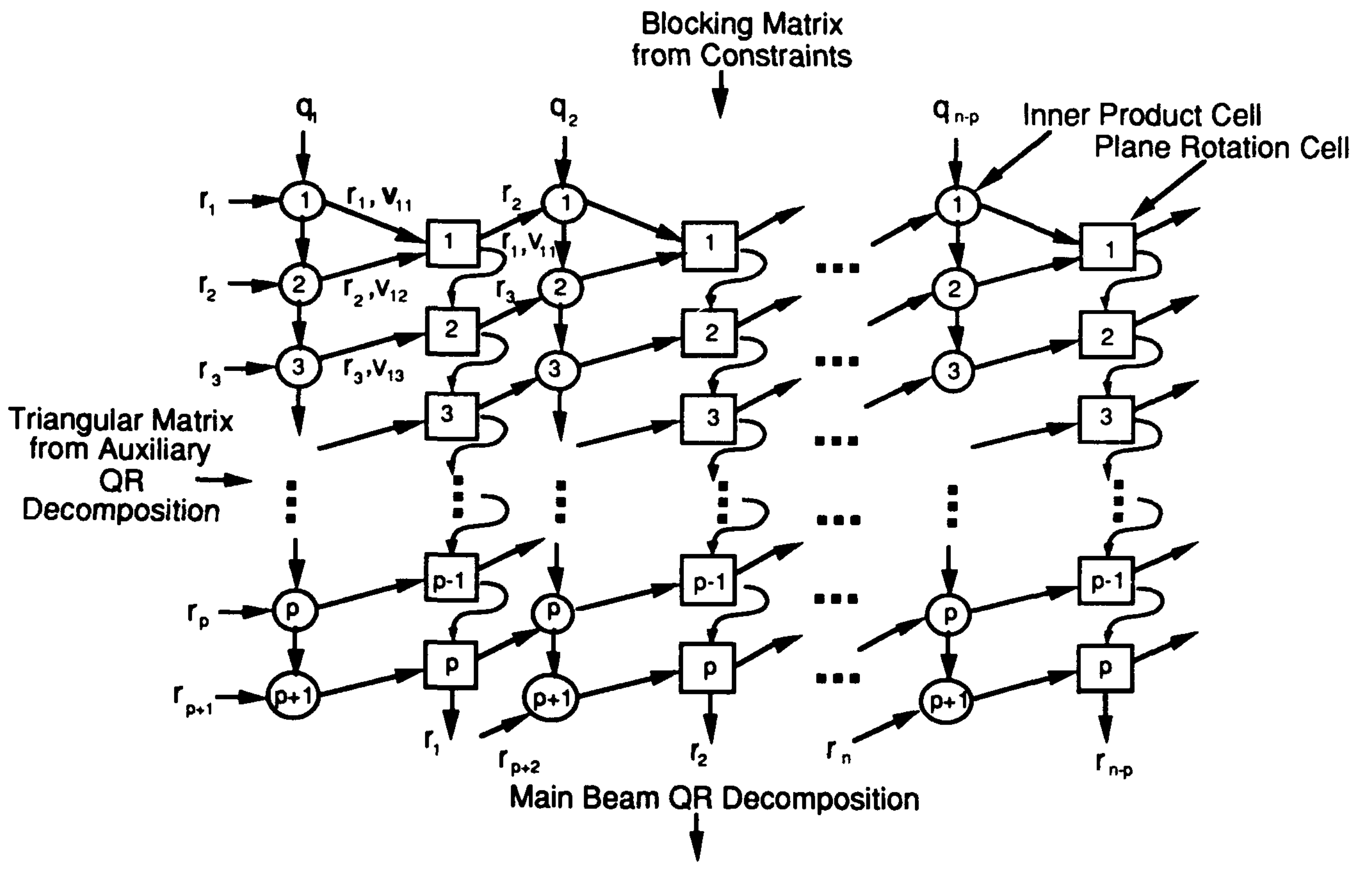


Figure 2. Two-Dimensional Integrated Matrix Multiplication–Plane Rotation Processor