

COMPACT SONAR SURVEILLANCE PROCESSING SYSTEMS

Tom Curtis¹, Michael Curtis²

Curtis Technology (UK) Ltd, Weymouth, Dorset DT4 7BS

¹ tomc@curtistech.co.uk

² michael.curtis@curtistech.co.uk

1. INTRODUCTION

Many military systems rely heavily on COTS (“Commercial of the Shelf”) technology to provide the required processing performance. The advantages of COTS implementations are lower development costs, shorter design cycles and added system flexibility.

This paper outlines a recently developed COTS system for sonar surveillance applications. It uses an open architecture and commercial operating system, together with programmable DSP cards, to realise a compact unit that has been deployed in a number of operational sonar applications. The system can be configured by the operator for a variety of surveillance and classification processing tasks and for data recording and replay/re-processing, for post event analysis. Functionally, the system provides the baseline processing to handle current generation surveillance arrays deployed by the UK Navy, as well as a number of enhancements to improve the flexibility and operational effectiveness of these types of system.

Although the following description concentrates on passive sonar applications, the system has also been used to provide active processing for a number of array configurations.

2. SYSTEM BACKGROUND

Many of the towed array surveillance sonar systems currently deployed by the UK Navy (for example ST2031 and ST2046) were developed by the Defence Evaluation Research Agency (now Qinetiq), in collaboration with UK industry, back in the early/mid 1980s [1]. The technology and systems architectures used to realise many of these sonars were similar. They were based on distributed networks of custom hardware processing modules [2], with a limited amount of programmability and flexibility, configured by micro-computer based controllers. The systems were installed in many platforms in the UK Fleet, both on surface ships and submarines, and have been in operational service for the past twenty years or so.

Twenty years have seen significant changes in the operational requirements for surveillance sonar systems, as well as major reductions in the number of UK platforms. The end of the Cold War has changed the perceived threat the Navy needs to counter [3]. This, coupled with reduced defence spending and a consequent narrowing of research focus, has resulted in the balance of sonar research moving away from SSBN-centric passive systems towards low frequency active systems [4] to counter the SSK threat.

However, there are still significant gains to be realised in passive sonar systems by updating the processing algorithms, improving system operability and integrating additional tools to help the operator classify and track potential targets. Whilst these enhancements can be retro-fitted into the existing hardware systems, it is often more cost effective to implement the complete system, together with the required enhancements, on a new hardware platform that integrates into current operational systems. The following sections outline one such new development and illustrate some of its potential.

3. SYSTEM DESIGN PARAMETERS

Since the system was to be configured using COTS components, a number of initial design decisions that were made to maximise the cost effectiveness of the overall system. It was decided to base the design on a 3U compact PCI (cPCI) chassis, rather than on VME, and to use Microsoft NT4 as the operating system.

Compact PCI is the “professional” version of the standard PCI interface specification [5] widely used in commercial PCs. It has the advantage of using the same interface chip sets as office PC clones but with a more reliable two-part connector and a passive back-plane. Various “flavours” of the cPCI interface specification support back-plane data bandwidths in the range 133 to 528 Mbytes/second with passive back-planes and in excess of 5 Gbytes/second with silicon fabric back-planes.

Adopting cPCI allows much of the system development to be carried out on low cost commercial PC clones using readily available PC software tools. It also allows shore-based replay/analysis/training facilities to be designed around office PCs, again reducing system costs. Whilst NT4 is not strictly a real-time operating system [6], it can be used in real-time applications by suitable design: this includes defining the correct priorities on critical software procedures and the judicious use of FIFO memory buffers to allow some elasticity in the overall processing schedule. These techniques ensure no data is lost during real-time processing and that overall integrity is maintained. However, small changes in processing latency will occur as the processing load varies. Typically these are of the order of a few milliseconds over a one second period for this particular application, so have a minimal effect on system performance, as far as the operator is concerned.

In view of the tight time-scales required for the system development, a rapid application development tool-set, Borland DELPHI [7], was used to develop the system software. In the main, this software controls DSP resource configuration, data archiving, the display formats and MMI interfacing, with the heavy “number crunching” for the DSP algorithms off-loaded onto programmable DSP. The MMI system uses a standard Windows tabbed notebook graphics unit interface (GUI) that allows operators to set-up processing parameters and to pull up different processing bands and options in an intuitive manner.

3.1 System Architecture Background

Distributed architecture DSP systems have been deployed in UK Naval applications since the early 1980s [1,2]. Many of these systems used signal flow architecture [8] to realise compact, self-scheduling distributed systems at minimum cost. It was decided to capitalise on this design experience for this program and to migrate signal flow architectures onto current generation COTS hardware.

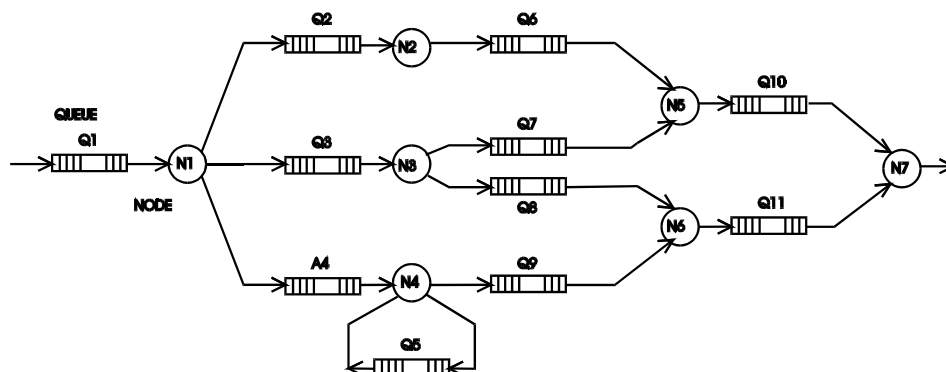


Figure 1 – Schematic of a Generic Signal Flow Graph

A schematic of a generic signal flow graph is shown in Figure 1: the nodes, labelled N_x , represent the signal processing functions performed on the data, whilst the queues, labelled Q_x , are the inter-node communication channels. This type of flow graph is used to define the detailed signal processing algorithm flow performed on the data. It can be realised either on general purpose computers, using software to instantiate the nodes and queues [8], or on a homogeneous distributed system [2], where some (or all) of the node resources are realised using hardware blocks.

The corresponding generic hardware realisation of such a signal flow system is shown in Figure 2. It used a high-speed bus as the shelf back-plane, with a number of dedicated hardware resources interfaced to it, via FIFO-based standard bus interface units. The hardware resources are configured using the control/monitoring bus to define the processing schedule. Private bus connections can also be implemented for direct higher-speed transfer between hardware processing modules, if needed. Communications between shelves is provided by fast serial data links. This is the basic form of architecture used for sonars ST2031, ST2061 and ST2046.

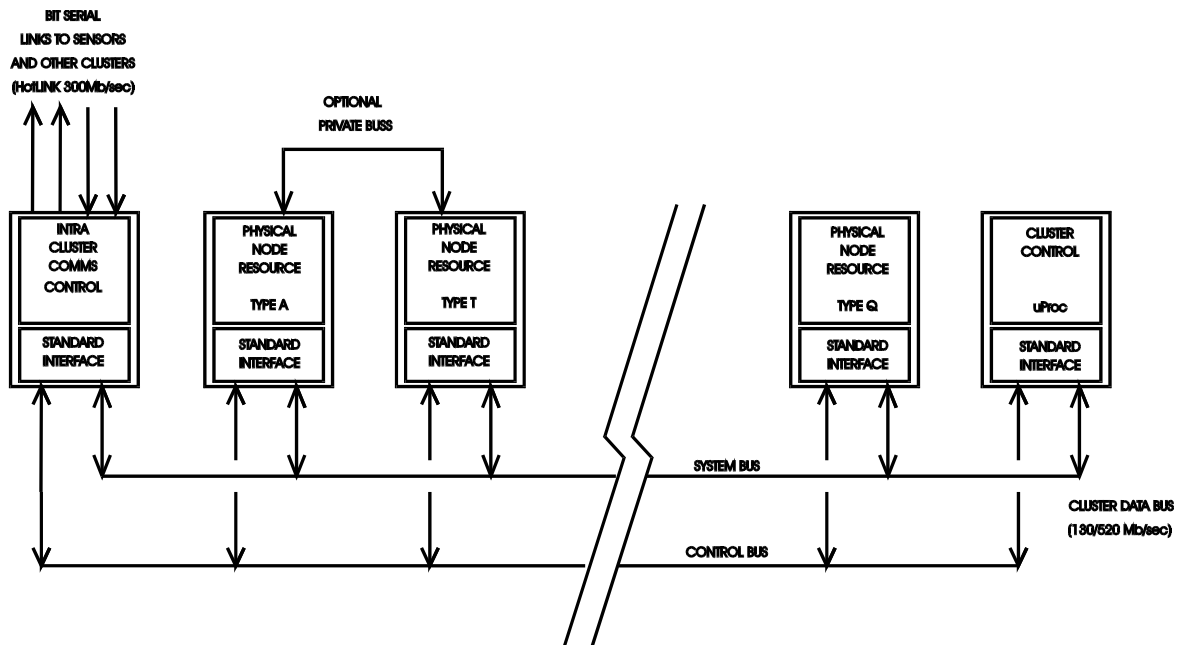


Figure 2 – Schematic of a Generic Hardware Implementation of a SFG

It is an easy step to migrate this distributed architecture approach to current generation COTs hardware. The system and control/monitoring bus functions can be provided using the memory and I/O space on a PCI bus. This allows the use of commercial PCI chipsets for system bus to hardware resource local bus interfacing. It also allows PCI bus extension via PCI-to-PCI bridge devices to be included, if needed for larger systems. (PCI-to-VME bridges are also available for legacy VME system support). The private bus connections can use the accepted Front Panel Data Port (FPDP) standard [9] and the dedicated hardware resources can be provided by FPGAs, programmable DSP parts and standard microprocessors.

3.2 System Hardware Details

A schematic of the DSP chassis and card layout used in the hardware system is shown in Figure 3. The unit is based on a standard 4-slot, 3U cPCI chassis. It is configured around a commercially available single card computer, the Gespac PIII-SYS 650E [10]. This provides the

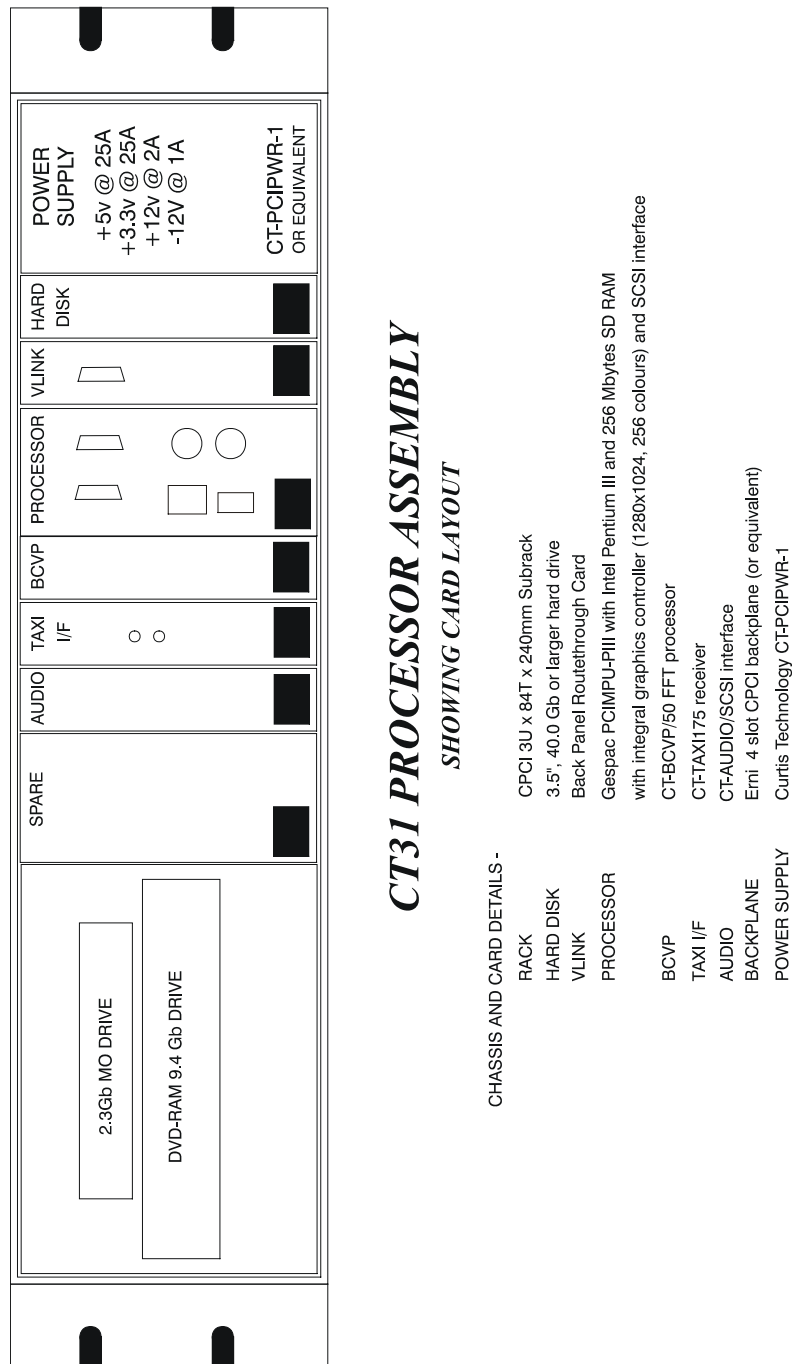


Figure 3 – DSP System Configuration

standard PC system interfaces and resources (IDE and SCSI, mouse, RS232 and Ethernet Tbase10/100 interfaces, an X VGA video adapter - 1280x1024 pixels by 16M colours, memory, RTC, etc). Apart from the CPU card, the system uses three other OEM COTS boards, viz. an array interface card, a programmable DSP card and an AC97 compatible audio sound card. Also

included in the rack are standard bulk storage devices for raw data archiving, data logging and for storing screen dumps.

The array interface card (the TAXI I/F board) connects the array terminal unit to the system via dual FDDI (175 MBPS) or HOTLINK (400 MBPS) serial links. It also provides 64 Mbytes of high speed buffer memory and some supporting control and processing logic, using FPGA devices.

The DSP card (the BCVP) uses the Sharp LH9124/9320 vector processor DSP chip set [11,12], again with high speed buffer memory and FPGA logic. This card is described in more detail in the following paragraphs.

The AC97 audio system is provided for sonar aural output and also provides system interfaces to a DVD-RAM drive for raw data archiving and an MO drive for screen dumps. This combination also provides DVD movie and CD playback for training (or other) purposes.

3.2.1 The Complex Vector Processor (BCVP) - Figure 4.

The BCVP is a compact, programmable DSP card for high throughput, front-end signal processing operations, such as beamforming, filtering and spectral analysis.

The board integrates two basic units:

- i. the motherboard containing the PCI interface chipset and associated FIFO data storage for high speed data transfer, local bus interfaces to DSP modules and some control logic for configuration and scheduling

and

- ii. DSP modules (up to 4 per motherboard) based on the Sharp chip set and banks of high speed synchronous SRAM that perform the high throughput number crunching.



Figure 4(a) - cPCI BCVP Processing Card

Both commercial and Compact PCI versions of the mother-board are available, so the DSP board can be used in any system that supports a PCI bus interface, for example most Intel based PCs, DEC alphas, etc. Kernel-mode PCI bus-master device drivers were developed that allow the card to be used with most PC operating systems (e.g. DOS, Windows 95, Windows NT4, Windows XP, etc).

The processing module (Figure 4b) has a pass-based architecture, where data flow and processing operations are set up to process large contiguous blocks of data. This pass based architecture, coupled with the kernel mode bus-master drivers, allows very straightforward processing software to be written in a variety of high level languages (e.g. from C, Visual C++, Pascal, Delphi, etc). For example, only three lines of Delphi code are needed to program the card to calculate a 4k-point complex fast Fourier transform (FFT), with a further two lines to load and unload data to/from the processor.

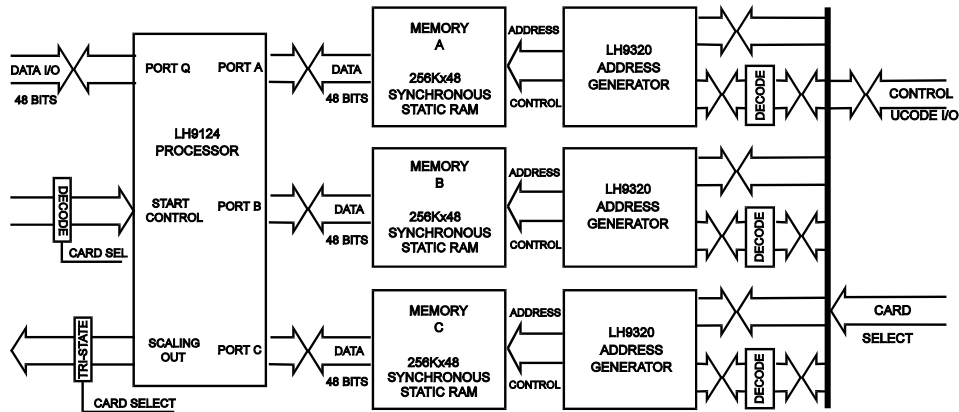


Figure 4(b) – Reverse Side of BCVP Processing Module

3.2.2 Module Architecture.

A block schematic of the module is shown in Figure 5: the core DSP engine consists of a complex, block floating point processor that is optimised for block-orientated algorithms and array processing. It uses bi-directional, multi-port data flow, so high speed synchronous static RAM can be used for working data and coefficient storage and high speed FIFOs for input/output data queuing.

The module has been designed to handle large transform sizes relatively easily. Forty eight megabits of memory are used to support the DSP chip - data ports A, B and C each interface to up to 256k complex words of synchronous static RAM, whilst the Q port interfaces to a 256k complex word bi-directional FIFO (on the motherboard) for data I/O.



STACKING BCVP PROCESSOR MODULE SCHEMATIC

Figure 5 – BCVP Processor Module Schematic

The flexibility of the data routing on the module allows the use of fast synchronous SRAM, rather than the dual port memory that limited previous similar system designs [13]. This allows much larger memory sizes and faster access times than previously and processing block sizes up to 256k-point complex can be handled directly. (The DSP chipset itself can handle up to 1M point transforms but the block size on the current implementation is limited to 256k by the amount of memory fitted.)

The DSP core in the LH9124 is a pass-based processor, where each function op-code and directional data flow op-code are valid for one complete pass of the data - the function op-code requires a related data flow op-code that specifies the data sources and destinations. So, the data flow op-code defines how data moves internally through the execution unit, whilst the function op-code defines the algorithm performed on the data.

The module handles high level processing functions such as filtering, spectral processing, correlation, modulation and de-modulation. The power of the execution unit lies in its ability to perform transforms very efficiently, particularly radix-16 based FFTs. The op-codes that the unit implements are compact and operate at the macro level, supporting DSP, complex arithmetic, vector arithmetic, vector logical and a number of general purpose functions as embedded operations. The unit provides either fixed (24 bit real + 24 bit imaginary) or block floating point arithmetic (24 bit + 24 bit + 8 bit block exponent). A summary of the data-flow paths and the execution unit op-codes are shown in Table 1.

Mnemonic	Data Flow
RAWB	read from port A, write to port B
RAWC	read from port A, write to port C
RAWQ	read from port A, write to port Q
RBWA	read from port B, write to port A
RBWC	read from port B, write to port C
RBWQ	read from port B, write to port Q
RCWA	read from port C, write to port A
RCWB	read from port C, write to port B
RCWQ	read from port C, write to port Q
RQWA	read from port Q, write to port A
RQWB	read from port Q, write to port B
RQWC	read from port Q, write to port C

Table 1(a) - Data Flow Operations and Mnemonics

Op-Code	Mnemonic	Processor Operation
\$00	BFLY16	Radix 16 Butterfly
\$01	BFLY4	Radix 4 Butterfly
\$02	BFLY2	Radix 2 Butterfly
\$03-\$06	{RESERVED}	
\$07	BRFT	Dual real FFT separation pass
\$08-\$0B	{RESERVED}	
\$0C	CMAG	Complex magnitude squared
\$0D	CMUL	Complex multiply
\$0E	{RESERVED}	
\$10	CADD	Complex Add
\$11	CSUB	Complex Subtract
\$12	VMUL	Vector multiply
\$13	VABS	Vector absolute value
\$14-\$18	{RESERVED}	
\$19	VPAS	NOP
\$1A-\$1B	{RESERVED}	
\$1C	MOVD	Move data from (A,B or Q) to (A,B,C or Q)
\$1D	MOVC	Move data from C to (A,B or Q)
\$1E	{RESERVED}	
\$1F	LOOP	Re-start control sequence

Table 1(b) - Processor Operations and Mnemonics

Programmable address generators (AGs) on the module drive the memory banks on the A, B and C ports of the processor – the AGs generate the required address sequencing to implement various algorithms for processing block sizes from 2 to 1M points. Table 2 outlines some of the types of addressing sequencing supported by the AGs.

The BCVP AGs can support over 150 address patterns [12]. These define the data store address sequences for many signal-processing operations, for example:

Radix 2, radix 4, radix 16 and mixed radix FFTs
Digit reverse for FFTs
Double length/two at a time real data only FFT separation pass
Decimation/Interpolation
FIR Filtering
Sonar Beamforming

Table 2 – Address Generation Operations

Sonar data is fed to and from the unit via the PCI bus: the bus-master PCI interface allows data to be written and read from the unit at up to 132 Mbytes/second. 256k complex word bi-directional FIFOs are used on the card, so that data loading/unloading and processing can be performed concurrently.

The BCVP/PCI interface card (Figure 6) has been designed to be signal flow driven: when there is sufficient input data in the input FIFO and enough room for results in the output FIFO, then the process is triggered. The algorithm calculated with the data is defined using a tag or token in the input data stream block header to point to a sequence of op-codes and direction control codes. These are pre-loaded into the control FIFO, via the control/monitoring bus. The system is most efficient when processing multiple channels of data, for example for LOFAR processing, where the processing parameters on individual blocks of sonar data, for example vernier zoom bandwidths, are controlled, via the tag, on a channel-to-channel basis. Examples of some high-level BCVP control software and some DSP benchmarks are given in Table 3.

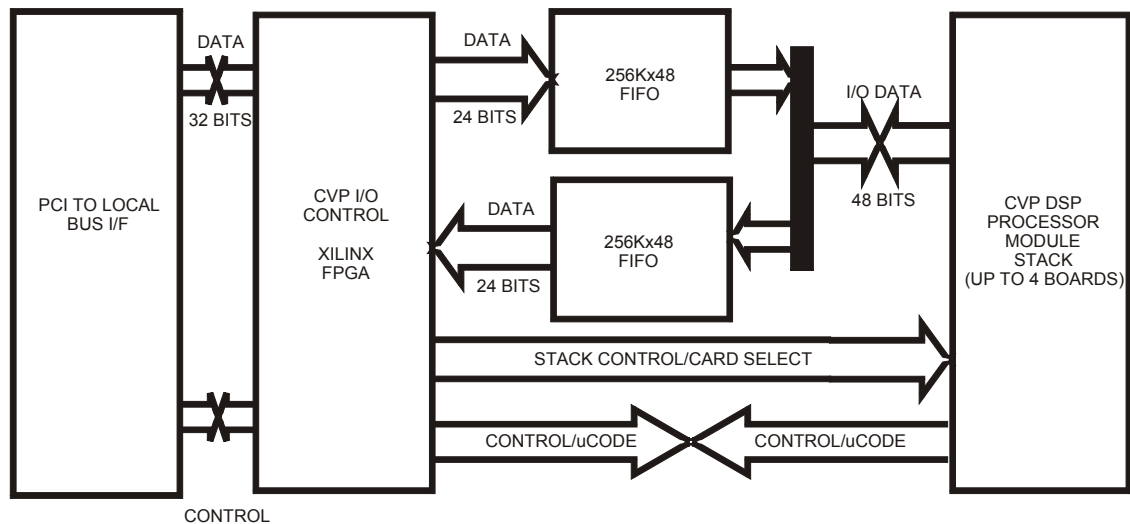


Figure 6 – Motherboard Schematic

<i>DumpCode(Operation</i>	<i>,Dataflow,AAddress,BAddress,CAddress)</i>			
DumpCode(MOVD	,RQWA ,RBF0	,NOP	,NOP);
DumpCode(BFLY16	,RAWB ,BF160	,BF160	,TF160);
DumpCode(BFLY16	,RBWA ,BF161	,BF161	,TF161);
DumpCode(BFLY16	,RAWB ,BF162	,BF162	,TF162);
DumpCode(MOVD	,RBWQ ,NOP	,BF160	,NOP);

Table 3(a) - Example DELPHI Code to Program BCVP for 4k point FFT

This code controls the device driver and sets up processor address generators and control logic for the five passes used to load data into the module, perform the transform and unload the results. Once the micro-code has been loaded, when sufficient data is fed to the Q input store and there is sufficient room in the Q output FIFO for the processed results, the operations defined by the code are performed in sequence. So 4k points of data are read from the Q input FIFO and written to RAM store A in digit reverse order on the first data pass. The three passes are performed to calculate the 4k point, radix 16 FFT, with data ping-ponged between RAM stores A and B. The final pass reads the complex spectral data read from RAM store A and writes it to the Q output FIFO.

1k Complex FFT	38.4 uSeconds
4k "Two at a Time" Real FFT	256.0uSeconds
Beamforming (1k complex points, 32 elements, 32 beams)	13.1 mSeconds
Vernier Processing (typical value, 1024 complex O/P points)	89.6 uSeconds

Table 3(b) - Some BCVP Sonar DSP Benchmarks

Individual units are networked together via T-base 100 to provide multiple operator stations, with raw and processed data from each unit being accessible across the network.



A typical screen dump from the system, looking at four contiguous beams in the wide-band mode, is shown in Figure 8.

Development systems using compatible processing modules, based on the FPGA DSP engine outlined in a companion paper [14], and cPCI active fabric back-planes, have demonstrated orders of magnitude increase in real time system throughput for next generation applications.

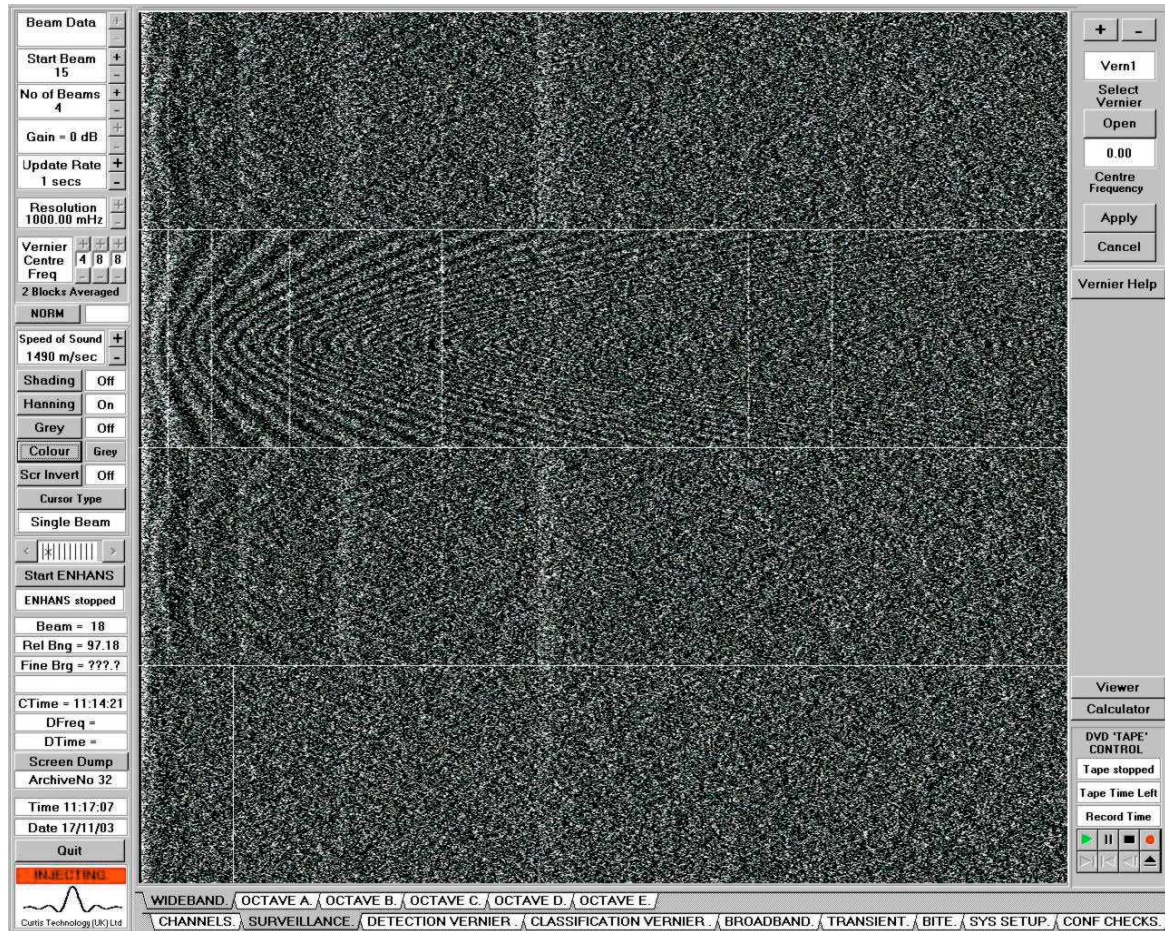


Figure 8 – System Display Format

© Copyright Curtis Technology (UK) Ltd, 2004.

REFERENCES

- [1] "ARE Pioneers Desk-Top Sonars", R Pengeley, Defence Electronics and Computing, IDR3/1989
- [2] "Control Ordered Sonar Hardware – COSH: A Distributed Processor Network for Acoustic Signal Processing", T E Curtis, A G Constantinides and J T Wickenden, Part F, Proc IEE, 1984.
- [3] "The Submarine Threat", at http://www.mod.uk/dpa/projects/sub_threat.htm
- [4] "Sonar 2087 Project", at <http://www.mod.uk/dpa/projects/project.htm>
- [5] "PCI Hardware and Software", E Solari and G Willse, pub by Annabooks, San Diego, USA, 1996.
- [6] "Inside Windows NT", H Custer, pub by Microsoft Press International, 1998.

Proceedings of the Institute of Acoustics

[7] See for example, "Delphi Nuts and Bolts for Experienced Programmers", G Cornell and T Strain, pub by Osborne McGraw-Hill, 1995.

[8] "A Common Operational Software (ACOS) Approach to a Signal Processing Development System", Proc ICASSP, Boston, 1983.

[9] "Front Panel Data Port Specifications", pub by VITA Standards Organisation (VSO), 1998.

[10] "Gespac PCISYS-PII/III Processor", Data Sheets pub by Gespac, Geneva, Switzerland.

[11] LH9124 Digital Signal Processor Users Guide, Sharp Electronics Corporation, Camas, WA, USA, 1992.

[12] LH9320 Memory Management Unit, Users Guide, Sharp Electronics Corporation, Camas, WA, USA, 1995.

[13] "A Fast 32-bit Complex Vector Processing Engine", A J Kerr and T E Curtis, IOA Conference on Sonar Signal Processing, Loughborough, 1989.

[14] "High Performance Signal Processing", T E Curtis and M J Curtis, IOA Conference on Sonar Signal Processing, Loughborough, 2004.