J. Astrophys. Astr. (2001) **22,** 321–342

Real-Time Signal Processor for Pulsar Studies

P. S. Ramkumar* & A. A. Deshpande *Raman Research Institute, C. V. Raman Avenue, Bangalore 560 080, India.*
* *Present address: Intel Technology India Pvt Ltd, No. 65, 13th cross, III phase, JP Nagar, Bangalore 560 078.*

Received 2002 April 1; accepted 2002 May 8

Abstract. This paper describes the design, tests and preliminary results of a real-time parallel signal processor built to aid a wide variety of pulsar observations. The signal processor reduces the distortions caused by the effects of dispersion, Faraday rotation, doppler acceleration and parallactic angle variations, at a sustained data rate of 32 Msamples/sec. It also folds the pulses coherently over the period and integrates adjacent samples in time and frequency to enhance the signal-to-noise ratio. The resulting data are recorded for further off-line analysis of the characteristics of pulsars and the intervening medium. The signal processing for analysis of pulsar signals is quite complex, imposing the need for a high computational throughput, typically of the order of a Giga operations per second (GOPS). Conventionally, the high computational demand restricts the flexibility to handle only a few types of pulsar observations. This instrument is designed to handle a wide variety of Pulsar observations with the Giant Metre Wave Radio Telescope (GMRT), and is flexible enough to be used in many other high-speed, signal processing applications. The technology used includes field-programmable-gate-array(FPGA) based data/code routing interfaces, PC-AT based control, diagnostics and data acquisition, digital signal processor (DSP) chip based parallel processing nodes and C language based control software and DSP-assembly programs for signal processing. The architecture and the software implementation of the parallel processor are fine-tuned to realize about 60 MOPS per DSP node and a multiple-instruction-multiple-data (MIMD) capability.

Key words. Stars: neutron—pulsars; interstellar medium: dispersion—Faraday rotation; telescope: GMRT; instrumentation.

1. Introduction

Pulsars are highly magnetized rapidly-rotating neutron stars radiating predominantly linearly polarized electromagnetic radiation beamed along its magnetic poles. As the pulsar rotates, its radiating beam sweeps across the line of sight of the observer periodically, much like a rotating beacon, resulting in a periodic train of narrow pulses of broad-band radiation, which can be detected using a radio telescope. Different types of observable parameters, such as the pulse periodicity, its rate of change, the pulse

width, pulse energy, pulse shape and its structures, polarization of radiation received at different positions within the pulse, and their spatial and temporal variations are of great interest, besides the characterization of some of the properties of the interstellar medium through which the pulsar signal propagates. This paper describes the design, tests and preliminary results of a real-time parallel signal processor built to aid a wide variety of pulsar observations planned using the GMRT (Swarup *et al.* 1991; Deshpande 1995; Ramkumar 1998). Pulsar signals appear as a train of highly periodic pulses, with period typically in the range of a few milliseconds to a few seconds and are usually very weak and buried in the noise contributed by the background sky and the receiver of the radio-telescope (typically 20 to 40 dB below the noise level). During their propagation through the intervening medium, these signals are distorted due to dispersion, scattering and Faraday rotation (see Manchester & Taylor 1977; Hankins & Rickett 1975 for details). Also, due to the relative motion of the earth and the pulsar, the Doppler acceleration changes the apparent period of the pulses. To correct some of these distortions in real-time (as the telescope tracks the pulsar), and process the signal to enhance the signal-to-noise-ratio (SNR), the computations required demand rates of several Giga-operations per second. Many operations that are common in the methods used to correct each type of distortion were identified. These redundancies were removed and a common signal processing algorithm was designed to handle one or more of the corrections within a single framework. A parallel processing machine was designed based on ADSP 21020 processors, to implement the algorithm and perform the corrections in real-time and the algorithm was then fine-tuned to exploit the architectural advantages of the DSP chip.

2. Hardware architecture

The front-end receiver system (Swarup *et al.* 1991), which supplies data to the Signal Processor, consists of RF/IF sections for the signals (with a bandwidth of 32 MHz) received from two polarization channels of each of the thirty dishes. Each of these two bands is split further into two sub-bands and converted individually to baseband outputs, each 16MHz wide, and are processed separately in the FX system that follows. The Signal Processor for Pulsar Studies (SPPS) will use two identical systems to cater to the two sub-bands separately. The further description will detail the instrumentation for only one sub-band (the other being identical). The inputs to the SPPS appear as two parallel digital streams of complex voltages (corresponding to the frequency spectra of two orthogonal polarization channels). The input specifications of the SPPS for one sub-band are summarized below.

Signal source: Combined (sub-band) output spectra (Prabu 1997; Ramkumar 1998) from a selected number of GMRT antennas with 16 MHz band-width and two polarizations.

Nature of input: 256-channel complex frequency-spectrum (per polarization); data over frequency channels sent in time-multiplexed form at a rate of 16M samples/sec, new spectra available every 16 μs.

Data format: 1-bit sign, 8 bit magnitude each, for real/imaginary part of the complex voltage for any frequency channel, polarization.

The block diagram shown in Fig. 1 outlines the different sections of the SPPS for one sub-band.The SPPS has four basic blocks: A Data Input Module (DIM), DSP parallel-processing nodes, a Result Collection Module (RCM) and the Control and

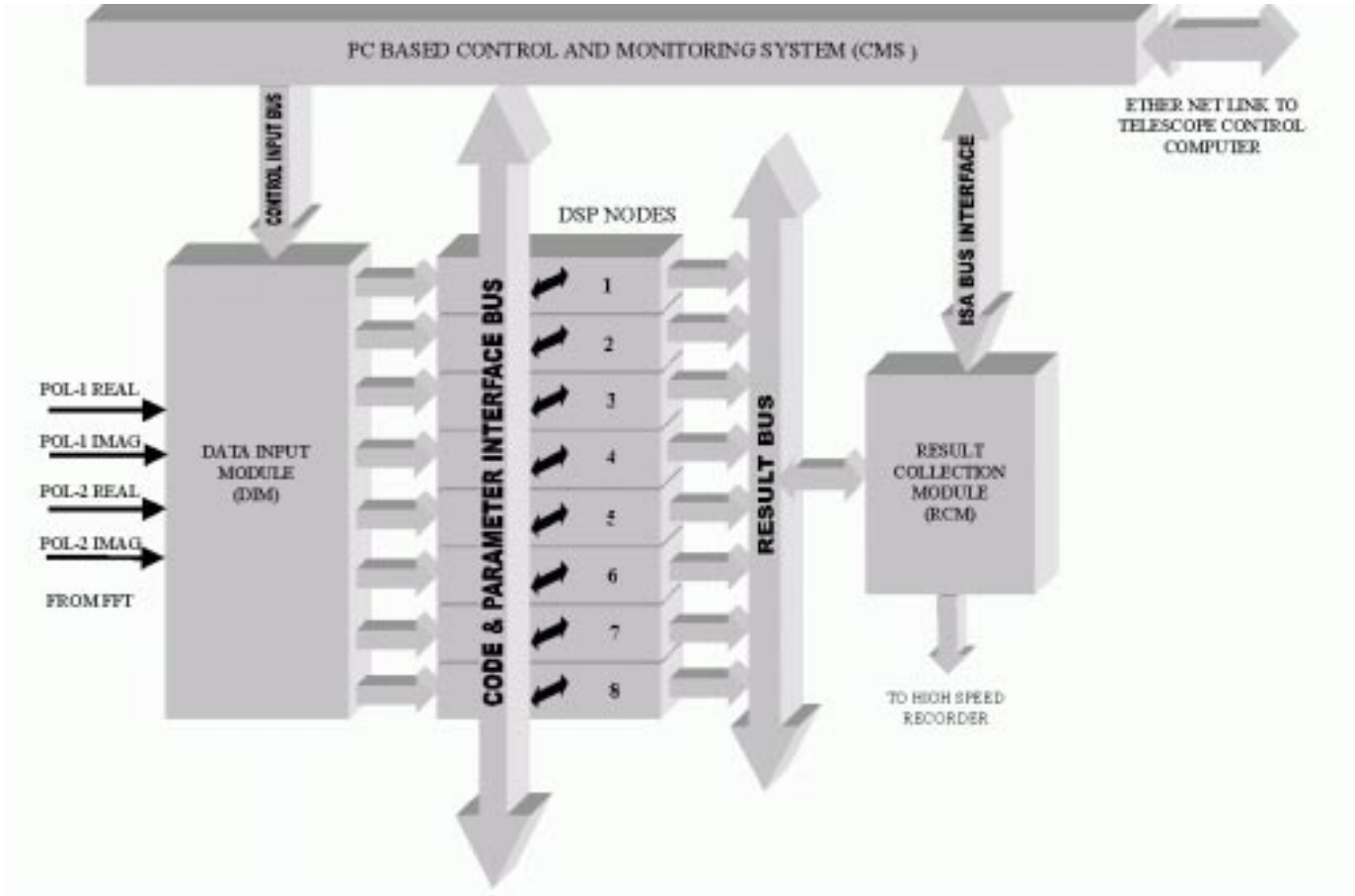

Figure 1. Basic blocks of SPPS of one sub-band (the other sub-band SPPS is identical, and the CMS is shared by both the sub-bands).

Table 1. Computations corresponding to different Stokes parameter derived from dual-polarization inputs.

Computation	Products	Additions	Stokes parameters for linear (circular) polarization inputs [e_1, e_2]
$e_1 e_1^* + e_2 e_2^*$	8	5	I (I)
$e_1 e_1^* - e_2 e_2^*$	8	5	Q (V)
Real($2e_1 e_2^*$)	3	1	U (Q)
Imag($2e_1 e_2^*$)	3	1	V (U)
Total = 34	22	12	

Monitoring System (CMS). The DIM links the front end to the parallel processor. It receives complex-voltage spectra from FFTs over differential-ECL links, computes the Stokes parameters (I,Q,U,V) for each frequency channel, and distributes the channels to the DSP processing nodes at a uniform rate. Each of the eight DSP processing nodes receives all 4 Stokes parameters of a selected band of 32 channels and performs operations such as pulse folding, Faraday de-rotation, dedispersion, Doppler acceleration correction, averaging of adjacent sample in time and frequency, pulse gating, data formatting, etc.. The RCM collects the results from each of the DSP nodes and renders the blocks of results at high speed to a fast recorder. In parallel, the same data are buffered to the CMS, which performs set-up, diagnostic and monitoring operations. The CMS also down-loads the processing codes and parameters to the DSP nodes, acquires and stores low-speed outputs (results) from the RCM, performs adaptive modifications to process parameters and feeds them back to the DSP nodes in real-time.

2.1 *Data input module*

As shown in the block diagram (see Fig. 2), the DIM has two major blocks, one for computation of Stokes parameters and another for distributing the Stokes parameters of different frequency channels to eight different data output paths. The effective data rate of the first block is about 128 Mbytes per second. The second block splits the data into multiple paths, so that individual DSP nodes in the following block can accept data for a smaller number of frequency channels at a slower, uniform rate and process the data independently in each module. The basic equations (e.g., Kraus 1966) relating the complex voltage samples (e_1, e_2) to the Stokes parameters are as listed in Table 1. For the specified bandwidth, the total computation rate amounts to about 544 million operations per second (MOPS) for calculating all four Stokes parameters.

This function has been implemented as a hybrid design involving the use of both look-up tables and dedicated high-speed logic built into EPLDs. The product of the magnitudes is calculated separately using EPROM-based look-up tables, while the resulting sign is from the exclusive-or (XOR) operation of the sign bits of the two numbers. The product terms are rounded to upper-most 16-bits to fit within the registered EPROMs. The addition and subtraction of the product terms, is performed using dedicated pipelined adders, designed and hosted within a pair of EPLDs. The outputs of these EPLDs represent the Stokes parameters I, Q, U and V. The summation/subtraction requires pipelined adder/subtractor logic modules (hosted in two EPLDs, consuming

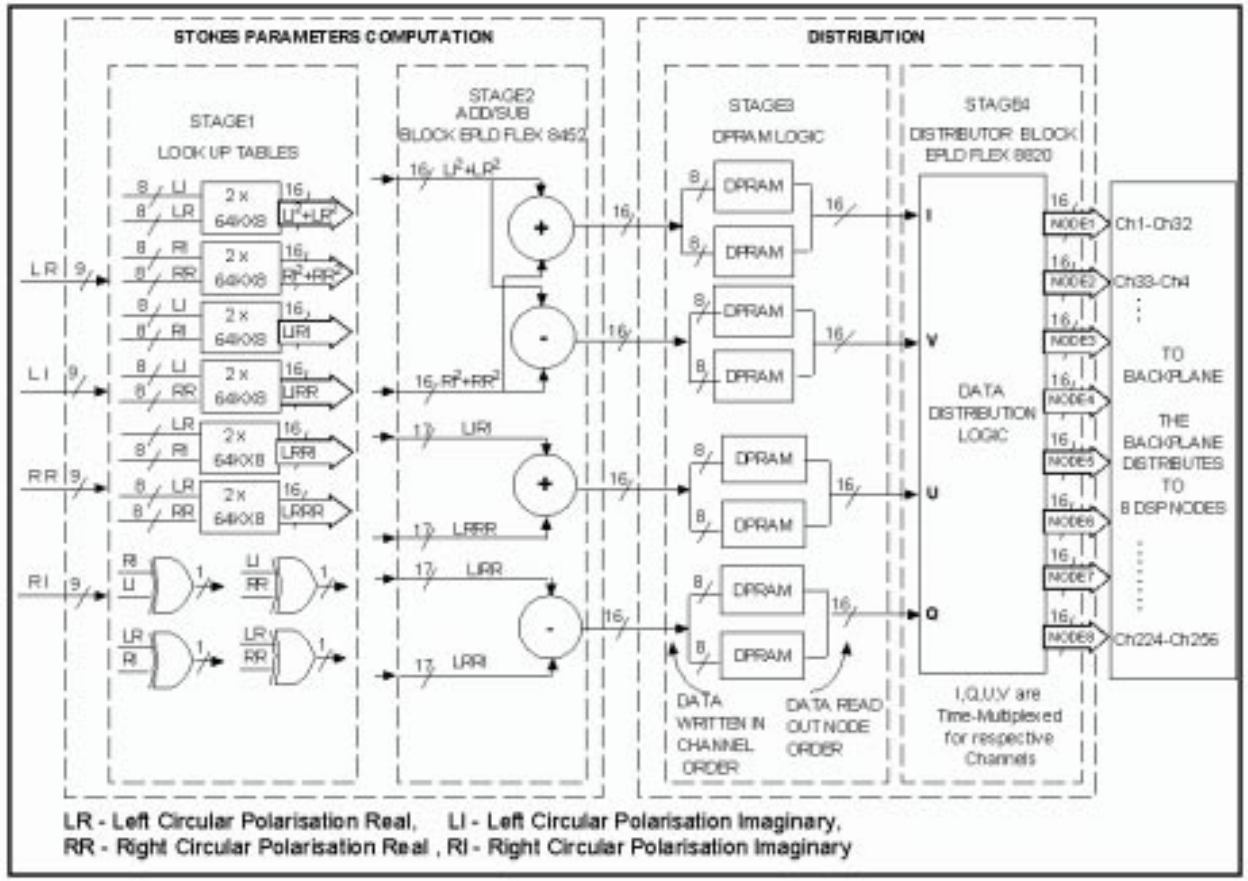

Figure 2. Architecture of Data Input Module.

about 3500 gates). A PASS-THROUGH mode is also provided, in which the look-up table and EPLD outputs are just replicas of their inputs, so that the raw data can be directly fed to further stages, if needed. Further, the 256 channels are split into eight paths of 32 consecutive channels, so as to supply to eight DSP nodes. To avoid burst - transfers to the DSPs, the Stokes parameters of various channels are written into a set of DPRAMs as they appear (1, 2, 3,..255, 1, 2.. etc.). After a full spectrum is written, the read out is started, in a node-sequential channel order (0, 32, 64, .. 224, 1, 33, 65,... 225, 2, 34,.. etc.). Thus the DSP nodes share the data at a slower, uniform rate. After the first spectrum is written, the read and write operations continue simultaneously without contention, in two alternating halves of the DPRAMS, through two ports of the DPRAMs. The four Stokes parameters of each node are latched in separate set of parallel-in, serial-out shift registers for each node, and are shifted out with the Stokes parameters time-multiplexed. After the initial pipeline delay, data appear concurrently on all the eight data paths, at a uniform speed of 8 Mwords/second. Write-strobe pulses are transmitted to each node along with the respective data and are synchronized to start after the pipeline delays in their respective data paths. This entire circuit is designed to fit in a pair of EPLDs (about 8000 gates), and is down-loaded into the EPLDs from a PC at setup time. Separate design files for different channel order, node order, etc., can be programmed, to route the DPRAM data to any node in any sequence.

2.2 *DSP nodes*

To perform all operations such as folding, adjacent sample integration, gating, Faraday de-rotation, dedispersion and Doppler acceleration correction, a suitable scheme is needed where the available memory resource can be redistributed between Stokes parameters, frequency channels and time frames depending on the type of observation. A parallel processing architecture based on DSP chips with an efficient signal processing algorithm was an optimal choice for design. Given below is a description of the architecture of one DSP-node (8 such nodes are to handle one sub-band of the GMRT) . The corresponding block diagram is shown in Fig. 3.

- The input data are available for the DSP node from a memory located on the program memory (PM) side. The data flowing in gets written into this memory at a steady speed, while the processor reads them out as and when required by the processing algorithm. Since the data flow from the DII to the DSP node is unidirectional and the writing/reading operations are independent as well as being at different speeds, this memory is chosen to be a 32-bit FIFO block. The write port of the FIFO block is interfaced to the DII and the read side is connected to the DSP node.
- The parameters required by the DSP nodes for processing are available from a separate memory on the PM side. The instruction code for the DSP node is also located in this memory. This allows the necessary update of the process parameters on-line, without disturbing the processor in the DSP node, and provides a common control block for all DSP nodes from which the instruction code and process parameters can be down loaded and updated on-line. The controller must be able to read back and check what it has written into this memory, before letting the DSP use it. The controller-DSP node communication protocol requires many semaphores. These semaphores may be bi-directional and are also located in the same memory for simplicity. A 48-bit dual port RAM block is chosen to hold the

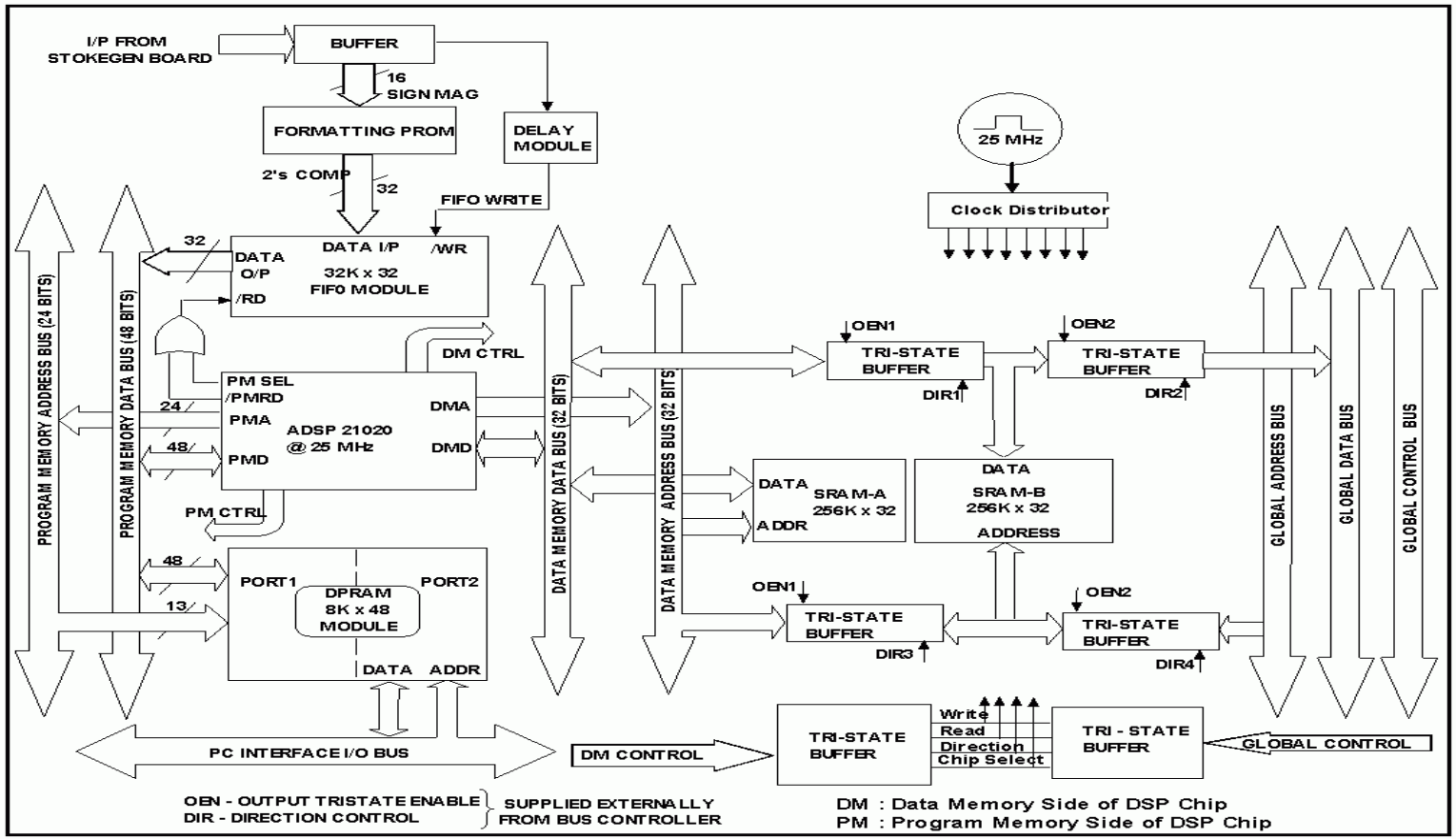

Figure 3. Architecture of a single DSP node.

instructions, parameters and semaphores. One port of the DPRAM is connected to the controller, while the other is interfaced to the PM side of the DSP node.

- A memory block in the data memory side of the DSP node holds the temporary results during the data processing. This memory has to be fast enough to match the DSP read/write speeds without requiring any wait cycles and is dedicated to the processor. This is realized in the form of a 256K location, 32-bit SRAM module designated SRAM-A which is interfaced to the data memory (DM) side of the DSP node. This memory is logically organized into two halves (banks) which alternate so that when one half is used for processing the previous results available in the other half can be read out by another task.
- The processed results in one of the two logical partitions of SRAM-A are copied into a continuous block of another SRAM module designated SRAM-B. This module is interfaced to the DM side of the DSP through tri-state buffers, such that it can be attached to or detached from the processor bus under software control. This SRAM is connected through another set of tri-state buffers to an output bus, common to all DSP nodes. When the DSP node wants to copy results from SRAM-A to SRAM-B, the buffers only on the DSP-node side are turned-on. After the results are written, SRAM-B (about 25 ns access time) is disconnected from the DSP node by tri-stating these buffers. Control logic ensures that the output-bus buffers are enabled only when the DSP node is not accessing SRAM-B and when the data acquisition system needs to acquire data from this particular node. The handshake between the DSP node and the RCM is explained later. Even though the flow of results is always from the DSP node to the RCM, SRAMs are chosen instead of FIFOs to provide flexibility for inter-node communications, such that the results of one node can be read and processed further by another node, if required, for other types of processing. With this feature, the architecture provides a loosely coupled, multiple instruction, and multiple data (MIMD) parallel processing system, which could be a very advantageous configuration for a variety of applications. The handshake for inter-node communications has been worked out, but is not necessary for the present requirements of pulsar signal processing. A memory mapped port on the DM side of the DSP node brings all status flags to the processor, which can be polled during processing.

Each DSP node operates at a 25 MHz clock rate. The code memory of all nodes are physically bussed together as shown in Fig. 4. All nodes are controlled by a central controller, implemented on a PC/AT platform, with suitable interface through parallel I/O ports hosted on the ISA bus of the PC/AT. The parallel ports are grouped to form the program bus, consisting of address, data, control and status buses which connect to the DPRAMs in the DSP nodes. The DPRAMs are 16-bit wide with 8K locations each, needing a 13-bit address bus and 16-bit data bus. There are 24 DPRAMs per sub-band and are organized as separate pages. Initially, the page number is latched in to a page-selector and subsequent addresses refer within a selected DPRAM until the page number is changed.

2.3 *Result collection module*

Two hand-shake signals, namely BUS-FREE and ENGAGED, are dedicated to each node for communication between the RCM and the nodes. Whenever a DSP node has its results ready in its SRAM-B, it activates its BUS-FREE line indicating that the RCM

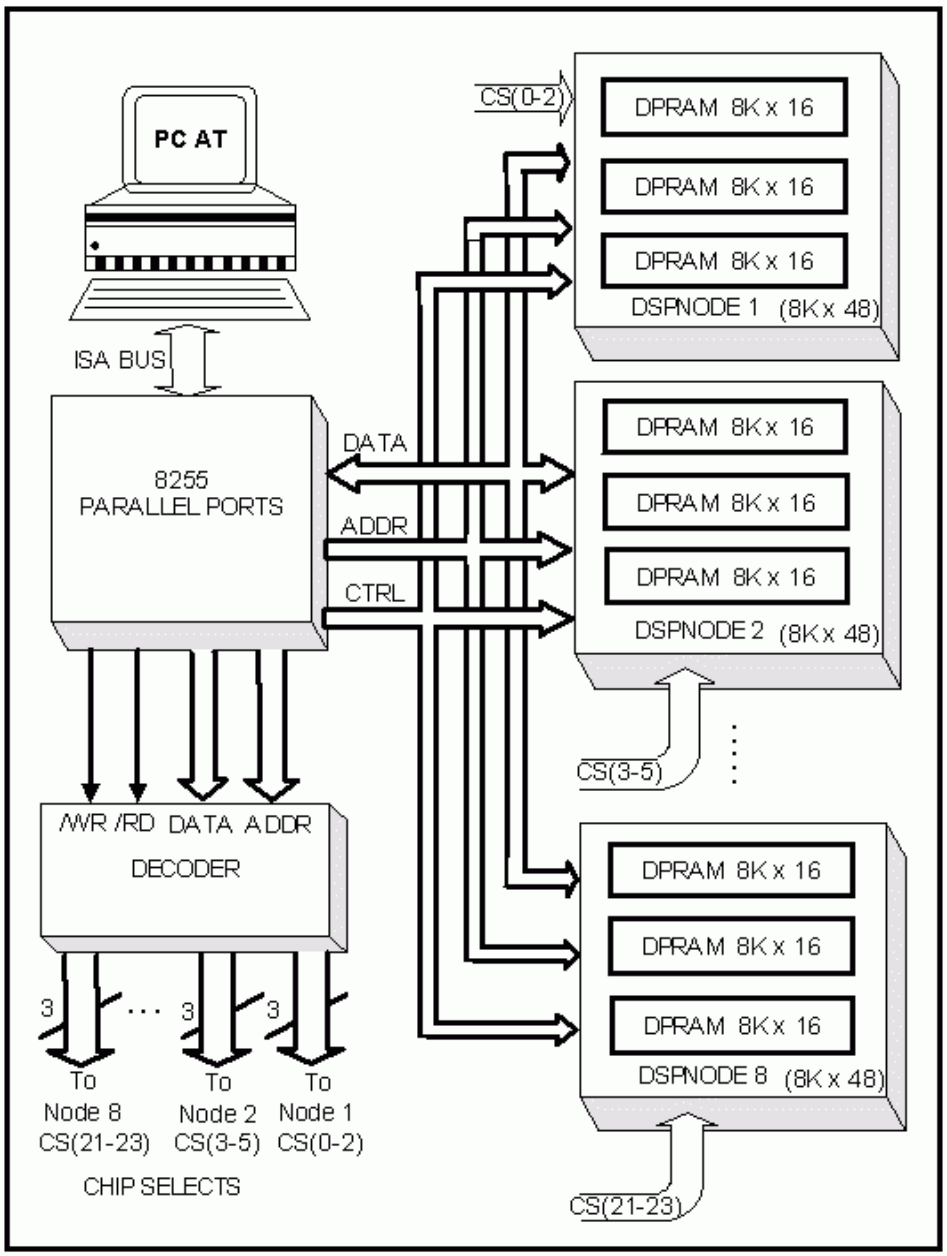

Figure 4. PC I/O interface to code memory of DSP node.

can access its SRAM-B. The RCM polls the BUS-FREE line and if found active, it sets the ENGAGED line to indicate to the DSP that it has taken over the bus and SRAM-B. Then it sends a sequence of addresses on the result-address-bus within a preset range. After asserting each address, it sets a chip-select signal (called OE) for that node, upon which the data from the SRAM flows onto the “result-bus". The RCM latches this data

into a 32-bit register and then sequences the address further. After completing the range specified for each node, it deactivates the ENGAGED line, indicating to the DSP node that it has finished transactions with its SRAM-B. This signal is tied to a hardware interrupt of the DSP chip, and an interrupt is issued whenever the ENGAGED line goes from active to deactivated state. An interrupt service routine in the DSP routine gets activated immediately and removes the BUS-FREE signal, disengages the SRAM-B from the result bus, and reclaims the connection to the memory. Meanwhile, the RCM polls the BUS-FREE line of the next node to set, after which it acquires data from that node in a similar manner as explained above. This process goes on, and suspends only when all nodes deactivate their BUS-FREE lines. After each 32-bit result gets latched in the RCM, it is split into two halves of 16-bits each and is loaded into a local FIFO block (32 K x 16). The FIFO block is memory mapped onto the PC via ISA bus. The PC monitors the FIFO flags and initiates a block transfer to the hard disk from the FIFOs whenever the half-full flags are set. The acquisition is stopped when the control PC finds that the time specified for ending the observation is reached.

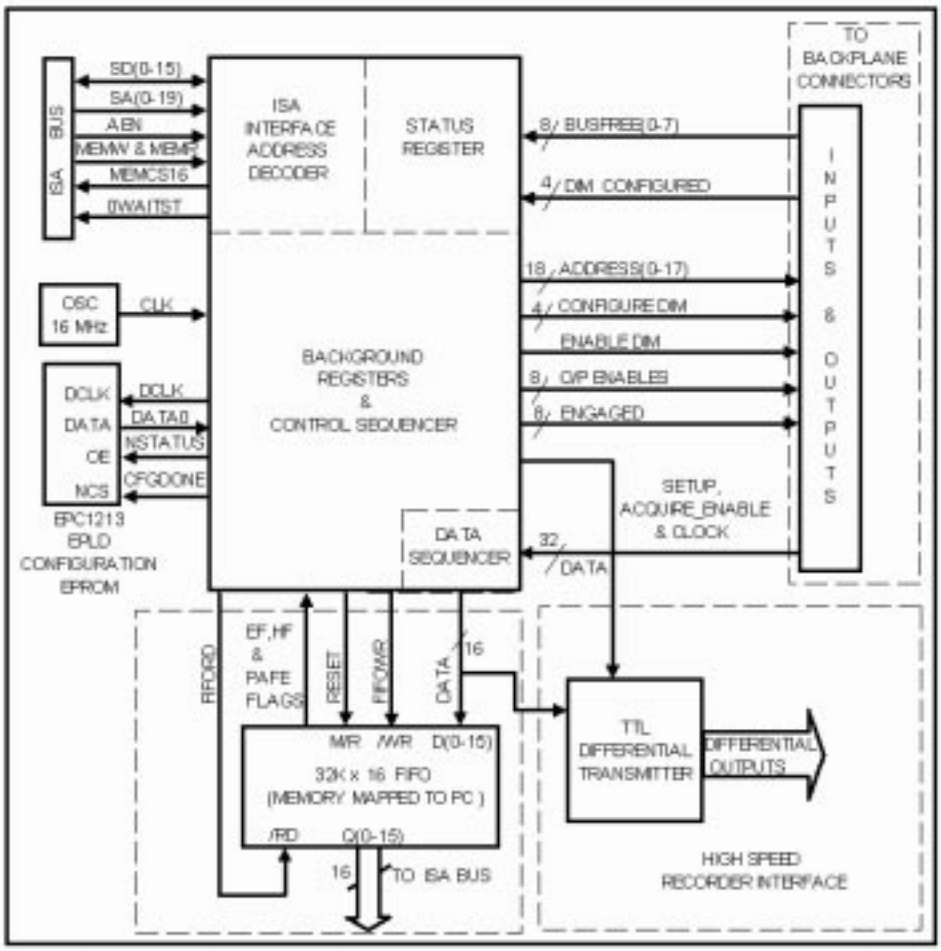

Figure 5. Architecture of Result Collection Module.

The block diagram of the RCM is shown in Fig. 5. The DCS is memory mapped through the PC's ISA bus. A space of 32 KBytes is mapped to the FIFO bank from which the results filled by the DCS can be read out. In the upper half, many control registers are mapped. A decoder generates the appropriate read/write pulses when the PC presents the respective address of different registers or memory, depending on the read/write signals of the ISA bus. The DCS is capable of working in a zero-wait state, 16-bit data transfer mode of ISA BUS. To specify the range of memory locations in each node from where results are to be collected, two registers, called lower- and upper-bound registers, are to be loaded with the start and end address of the result block in SRAM-B (the results are expected to fall in the same address range in all nodes). Through a programmable counter, the speed of acquisition from DSP nodes can be varied from 32 Kbytes/sec to 8 Mbytes/sec. Some times all of the eight nodes are not used or may have to be accessed in a different order to collect the results. To indicate the number of nodes and the node sequence for collecting results, a separate 32-bit register is loaded by a pattern, which uniquely defines every combination of number of nodes and node sequence. A 29-bit counter, clocked continuously at 16 MHz, forms the basic control sequencer. The lower most 8-bit section of the counter can be used to program the data acquisition rate. The next section of the counter chain is a 3-bit state counter. For every location to be read out from any DSP node, the eight states of this counter are decoded to produce the control signal sequence to complete reading out one 32-bit location, and loading it into two 16-bit locations of the local FIFO bank of DCS. The 16-bit words going to the FIFO bank are brought out in parallel along with a write strobe through an ECL differential link to a connector, which can be linked to a remote recorder for acquiring data at speeds higher than the PC's capability. The next stage in the counter chain is an 18-bit address counter, which generates an address sequence for read-out from the DSP nodes in a preset address range (maximum of 256 K locations). The next section in the counter chain is a Node sequencer, based on a set of shift registers. The node sequence and the number of nodes are loaded initially into background registers and the node sequencer cycles through this sequence and order. The state sequencer uses the node number to route the control signals to the appropriate node. The entire logic explained above has been built into a single FLEX EPLD chip (4000 gates).

3. Signal processing algorithm

It can be assumed that the input complex spectra are obtained with sufficient frequency and time resolution, and that consecutive spectra can be arranged to form a matrix, with the profile in each channel being located in fixed number of memory locations, say $N_{\mathrm{per-chan}}$ (which is equal to N_{bins} when folding at the pulsar period is performed). Consecutive spectra come in at time intervals of T_{frame} seconds. The basic operations required for various corrections are grouped as addition of incoming data to the matrix at suitably indexed locations. The calculation of index for various corrections is explained below:

a) De-dispersion (channel based data alignment): Due to interstellar dispersion, the arrival time of pulsed radiation at different frequencies (f_k) within the observed bandwidth is different and the delay difference ($\Delta\tau$) can be calculated relative to

the arrival of the pulse at the highest frequency ($f_{\rm high}$), and expressed in units of the sampling interval as,

$$\Delta N(f_k) = \frac{\Delta\tau}{T_{\rm frame}} = \frac{\beta.{\rm DM}}{T_{\rm frame}}[{f_k}^{-2} - {f_{\rm high}}^{-2}] \quad (1)$$

where DM, the dispersion measure, is the column density of electrons along the sight-line to the pulsar, and β is a known constant. This offset $\Delta N(f_k)$ can be subtracted from the time index and the data sample can be added into the memory location addressed by the modified time index. This way, as the data of different frequencies get written into the memory, the dispersive delay is automatically compensated and the matrix will contain aligned pulse profile sequences. If pulse folding is not required, the relative delay between channels is compensated by just skipping the corresponding number of samples in respective channels initially, before starting to write/add the incoming samples in the output data matrix. In such a case, the matrix data (over the allocated $N_{\rm per-chan}$ locations per output channel) are read out well before they can be overwritten by new data.

However, when pulse folding is required, each frequency channel is allocated $N_{\rm bins}$ locations to hold its average profile (over a pulse period P). Here, for a given input sample of $i^{\rm th}$ time frame and $k^{\rm th}$ frequency channel, the linear memory destination address can be found by the relation $A(i,k) = (k.N_{\rm bins}) + {\rm REM}[(i - \Delta N(f)).Ph_{\rm inc}/N_{\rm bins}]$, where the first term $(k.N_{\rm bins})$ forms the base address (B_k) of the corresponding frequency channels and the REM operator extracts the remainder of the operand ratio. The remainder in the second term indicates that the result will be modulo $N_{\rm bins}$, ensuring wind back into the $N_{\rm bins}$ space for the $k^{\rm th}$ channel. The $Ph_{\rm inc}$ denotes the input frame interval in units of the profile bin-width $(P/N_{\rm bins})$.

b) Spectral integration (channel summation): If there are $N_{\rm ch-in}$ input frequency channels that are to be grouped into $N_{\rm ch-out}$ bunches of adjacent channels that are added together, then the memory can be split into $N_{\rm ch-out}$ banks of $N_{\rm bins}$ each. It is reasonable to let $N_{\rm ch-in}$ be a binary multiple of $N_{\rm ch-out}$. Then the input and output channel indices (i.e., k and j respectively) are related as $j = INT(k.N_{\rm ch-out}/N_{\rm ch-in})$, where both indices start from zero. The destination index in the matrix for any given output channel j is given as $A(i,j) = (j.N_{\rm per-chan}) + {\rm REM}[(i - \Delta N(f_k)).Ph_{\rm inc}/N_{\rm per-chan}]$, where k = 0 to $N_{\rm ch-in} - 1$, j = 0 to $N_{\rm ch-out} - 1$, i = 0 to $N_{\rm per-chan} - 1$. In cases where profile folding is also required, the index is given by a similar expression where $N_{\rm per-chan}$ is replaced by $N_{\rm bins}$.

c) Pulse folding: If a pulsar has an apparent period P and the interval between consecutive frequency spectra is $T_{\rm frame}$, then the number of time samples within one period is simply $R_{\rm bins} = (P/T_{\rm frame})$. This $R_{\rm bins}$ value is generally a real number, consisting of a fractional and an integer part (a default choice for $N_{\rm bins}$). Since the profile has to be hosted only in an integer number of locations ($N_{\rm bins}$), the fractional part is the residual time width that has to be accounted for. A pulse-phase (in units of profile bin-width) pointer is incremented every time a new sample (i.e., spectral frame) arrives and its integer part is used to address the memory as already indicated in the index computations above. The phase increment per data sample (frame) can be estimated as $Ph_{\rm inc} = (N_{\rm bins}/R_{\rm bins})$, and is ≤ 1. In general, when there are multiple frequency channels, this phase increment is common to all frequency channels. When folding is not required, the $Ph_{\rm inc}$ would be equal to unity, unless integration over successive time-samples is required.

d) Doppler correction: The apparent period at a given epoch and observing location may be different from the "true" period of the pulsar due to the Doppler shifts in the observed pulse frequency because of the relative motions of the Earth and the pulsar and is estimated using standard algorithms. The correction of interest amounts to compressing or stretching the profile (before folding) in a direction opposite to that due to the Doppler effect, so as to ensure that every new profile is in phase with the old ones as folding progresses, and thus avoids time-smearing of the details within the folded pulse. Once the new period P_{new} is known, the compression or expansion can be implemented by changing the phase increment (Ph_{inc}) value adaptively, as ${Ph_{\text{inc}}}^{\text{new}} = {Ph_{\text{inc}}}^{\text{old}}.(P_{\text{old}}/P_{\text{new}})$. Typically, the interval between updates in Ph_{inc} values will be in the order of a few seconds, to ensure that the phase error never accumulates to more than a sample interval.

e) Successive time-sample integration (smoothing): This integration can be implemented in a simple manner while folding by just changing (reducing) the number of bins (N_{bins}) into which the profile has to be fit, thus reducing Ph_{inc} correspondingly. Then the suitable value for N_{bins} would be the integer part of ($R_{\text{bins}}/N_{\text{smooth}}$), where N_{smooth} is the number of time samples to be integrated. In the absence of folding, such an integration can be effected by defining $Ph_{\text{inc}} = 1/N_{\text{smooth}}$.

f) Pulse gating: A time window within a pulse period is specified as that containing the pulse (plus more), so that the data within the on-pulse window are processed retaining the required resolution, when the samples outside the window can be rejected. Initially, the window width is set equal to the full period. The input data are folded until a significant deflection above noise (corresponding to the pulse-peak) is detected, and the window is then shrunk around the "on-pulse" region to a suitable width. This adaptive phase-locking of the window is automated under software control. Within the window, the matrix address is extracted from the phase pointer every time it gets incremented. However, during the off-pulse window region, the phase increment proceeds, but the index address is jammed to a constant value, so that all the time frames outside the window get added onto a single bin, which may be ignored.

It is clear from the above discussion that all these operations can be clubbed together, by simply manipulating the index to the time-frequency matrix and adding the incoming sample to the matrix at that index.

g) Faraday de-rotation: Initially, a known approximate value of Rotation Measure (RM) is used to find the differential Faraday rotation of the polarization position angle at a given channel (frequency f_k) with respect to one of the edge channels (say, at f_{high}) in the band as

$$\theta(f_k) = c^2.\text{RM}.({f_k}^{-2} - {f_{\text{high}}}^{-2}) \tag{2}$$

where c is the speed of light. Then the rotation θ can be corrected for by equivalently rotating the phase of the Stokes parameter combination $Q + jU$, in each frequency channel, by an amount ϕ = -2θ, such that $(Q+jU)_{\text{corrected}} = (Q+jU)_{\text{observed}}(\cos\phi + j\sin\phi)$. Thereafter, the time averaged Stokes parameters Q and U can be used to find the residual error, so that the corrections can be adaptively changed.

h) Parallactic angle correction: In case of an alt-azimuth mount telescope (at latitude L), the parallactic angle change as a function of Hour Angle (HA) can be expressed as

$$\theta_{\text{par}} = \tan^{-1}\left[\frac{\cos(L).\sin(HA)}{\cos(\delta).\sin(L) - \sin(\delta).\cos(L).\cos(HA)}\right] = -(\phi_p/2) \tag{3}$$

where δ is the declination of the source. The parallactic angle is corrected for, by rotating the phase angle of $(Q + jU)$ by ϕ_p in all channels, since it is purely a geometrical effect and is independent of frequency.

4. Software architecture

A BOOT routine has been developed to perform the basic house-keeping functions such as memory checks, allocation of logical memory partitions for data, code and parameters, identifying and short-listing working nodes, and forming the node-sequence, etc.. After booting the DSP nodes, the control PC loads the signal processing code and associated parameters into the dedicated regions of DPRAMs and releases the “reset" for only the short-listed nodes. Each DSP starts executing the outer shell of the signal processing task immediately. The DPRAM memory space is logically partitioned into Code, Semaphore and Parameter space. The parameter space is further partitioned into several tables and values. Initially, the DSP node clears its SRAM memories and initializes its FIFOs and disables the bus transaction for SRAM-B and establishes a header table in SRAM-B. The header is recorded along with the results, to help in tracking any data loss. The DSP then loads some ‘constants’ required for processing into its computational registers. As a next step, the DSP initializes its internal Data Address Generation (DAG) index registers with pointers to the FIFO, SRAMs and the different tables within the parameter space of the DPRAM. The beginning addresses of the profiles in each output channel are pre-stored into a circular table to allow fast generation of pointers while processing. Two such tables are established to contain the starting addresses in the two alternate processing banks of SRAM-A. A register is initialized to contain the total number of folds to be performed before switching the SRAM-A banks for fresh processing. Once these initial configurations are setup, the DSP sends a config-done semaphore to the Control PC through its DPRAM ports. Upon receiving the semaphore, the PC enables clocks to flow to the entire system. The DSP nodes await the onset of their FIFO half-full flags and then invoke further processing. The signal processing routine was split into two tasks, one for data processing and another for data communication to update process parameters on-line and send the results to a data recording system. To facilitate fast context switching, two sets of registers, called the primary and alternate registers, are used, to hold data corresponding to Task 1 and Task 2 respectively.

TASK 1: This task is the core routine for processing the data from the FIFOs using SRAM-A for temporary storage of intermediate results and using the parameters from DPRAMs. The code packed into this task is a highly optimized, parallel instruction sequence which loops for every channel. The data of parameters I and V are directly added to their old profiles in SRAM-A, but parameters U and Q are multiplied with a Faraday correction factor read from a table and then added to their old profiles in SRAM-A. In each iteration, the ‘phase’ index is checked for completion of a fold. On completion of a fold, the phase pointer is wrapped around by subtracting N_{bins} from the current value and the fold count is decremented. Once the required number of folds are completed, the banks are switched and information is set up for Task 2 regarding the result data size and starting location from where the results have to be copied to SRAM-B and sent to RCM. Then, Task 1 loops back to begin a new fold in the alternate bank with fresh data. Successive folds continue in a phase-synchronous manner. The folding proceeds

until the data of one half FIFO size is read out, and then the FIFO address register rolls back to the beginning address of the FIFO block. This generates an internal interrupt automatically and the interrupt service routine branches to Task 2 after clearing the interrupt. The process returns from Task 2 only when the FIFO half-full flags indicate the availability of the next block of input data. On return from Task 2, Task 1 continues operation from exactly where it had branched to Task 2.

TASK 2: This task performs the phase error correction, profile rotation, parameter update and transfer of results.

a) Pulse Phase error correction: The current phase pointer has its upper 16-bits corresponding to an integer bin address and the lower 16-bits to a fractional bin. Thus the indicated position is accurate only to (1/65536) of a bin. The residual error accumulates with every sample, and has to be corrected to avoid pulse-smearing during folding. The phase error is separately accumulated in a register and when it exceeds a preset limit, then the accumulated error is subtracted from the current position for all channels. At this point another facility is provided to add an integer value to the integer part of the current phase, to force a new position from which the folding can proceed, so that the profile gets rotated within the N_{bins} span, if required. The rotation value is fed by the PC and can be altered on-line.

b) Transfer of results: Each time Task 2 is executed, the value of 'residual' result size is examined. If it is a non-zero number, then the DSP node immediately deactivates its BUS-FREE flag to indicate to the RCM unit that it is going to access the SRAM-B and disables the bus-buffers of SRAM-B, enables the buffers on its own side. The results are transferred in small enough blocks such that the time required for each of these transfers does not exceed the intervals between two FIFO-half-full events. The DSP updates the 'residual' size in successive transfers. When the residue eventually reaches zero, the DSP buffers are disconnected and the bus-side buffer enabled, and the BUS-FREE flag is activated, to indicate to the RCM that the results are ready to be collected. As the data gets transferred from SRAM-A to SRAM-B, the old locations of SRAM-A are initialized to zeros so that, consequently, the bank can be re-used for a fresh set of folds by Task 1.

c) Parameter update: The parameters used by the DSP may change with time and are updated periodically. The control PC computes the parameters and the time at which the update is needed. At such a time, the PC stores the parameters in an update table in the DPRAMs of all nodes and sets up a semaphore to the DSP. This semaphore is written at a dedicated DPRAM location which automatically generates an interrupt to the DSP. The DSP may be in the middle of processing data, so the interrupt service routine just sets a flag, indicating that there are new values. During Task 2, this flag is checked for and if found 'set', the corresponding tables used routinely by the DSP during processing are changed using the update-table. An acknowledgment semaphore is then sent to the control PC to indicate that the new parameters have now been accepted by the node.

After completing the above tasks, the DSP waits in a loop polling for the FIFO-half-full flag to get set again. Once the flags appear, the DSP switches the alternative set of registers to primary set, returns from the interrupt service and continues with the execution of Task 1, from where it left on receiving its internal interrupt.

4.1 *Time budget*

Each DSP node runs at 25 MHz, executing one instruction every 40 ns and processes 8 Msamples per second. This means that on the average only about three instruction cycles are available for every data point to perform all the functions mentioned in Task 1 and Task 2! To speed up the code the instructions were heavily parallelized. The architectural advantages of the DSP chip that were exploited in implementing the signal processing code are:

- Zero-overhead branching and looping,
- Address-based interrupt generation,
- Intelligent caching to use effectively the 3 bus architecture,
- Parallel multiply,
- Add and data access instructions,
- Single-cycle task switching,
- Circular-buffer addressing with auto increment,
- Rolled coding for core loop,
- External hardware interrupt service for I/O handshake with DCS, and
- Use of on-chip decode lines to interface zero-wait-state memories.

With these optimizations, the core loop takes 11 cycles to process 4 Stokes parameters and perform the index calculations. This means about 2.75 cycles, or 110ns per data point. Since a new data point is available every 125ns, the savings is only about 0.375 cycles, or 15ns per data point. This loop iterates “half FIFO size" number of times before getting an internal interrupt, and the net time ‘saved’ within such intervals is given by X = half-FIFO-size * 15 (nsecs). This time must be sufficient to execute the outer loops of Task 1 and all operations of Task 2. Considering the computational load for folding, de-dispersion, Faraday correction, adjacent sample integration, channel integration, Doppler correction of a 1-millisecond period pulsar (in a fast binary),

Table 2. Time bugdet for the tasks and estimation of free-cycles.

Operation/Task	No. of cycles per half FIFO
Task 1	
Core loop	$11 \times 4096 = 45056$
Outer loop 1 overheads (after 1 profile is over)	$3 \times 32\text{ch} \times 2$ periods $= 192$ cycles
Outer loop 2 overheads (after N-folds are over)	8×2 period $= 16$ cycles
TASK 1: Total cycles required	45264 cycles
Task 2	
Error correction	600 cycles
Transfer of results (block of 256 locations)	820 cycles
Parameter update	240 cycles
TASK 2: Total cycles required	1650 cycles
Free cycles	4350 cycles

the number of cycles consumed by these instructions is outlined in the following table. A set of 4 FIFOs are chosen to form a 32-bit memory of 32K locations so that the time saved for these operations will be about 6000 instructions cycles between every two FIFO half-fulls which is sufficient for all the remaining operations. To include pulse gating, two operations are added in each outer loop 1. This overhead is well within the limits of the available free time.

5. Tests and results

Initially, a digital ramp pattern starting at value 0 and increasing up to 255 was fed repeatedly to the DII module at 16 Msamples/sec, such that each of the 256 frequency channels gets a constant number every 16 μsec, and adjacent channels have the ramp pattern. The polarimeter outputs were obtained satisfactorily on all 8 output paths at full speed. Subsequently, two DSP nodes were populated and used for tapping the DII data to check the signal processing algorithms and the communication links. The Stokes parameters of 64 frequency channels from DII were time-averaged independently to check for long-term stability and the outputs were obtained satisfactorily. Then, a periodic pulse-pattern with 1-sample pulse-width and period of 256 time frames was fed and the DSP nodes were run to fold the pulses for 16384 periods with correction for different RM values (while the input pattern does not simulate any of these effects). The resultant profile reflected the corrections put in, as an opposite handed rotation of the “linear polarization" position angle across the band as expected. Similarly, tests were made using a range of DM values and the resultant profiles showed corresponding de-dispersion delay gradients across the band as expected.

While the GMRT telescope was getting ready, the SPPS was interfaced to the Ooty Radio Telescope (ORT) to conduct the primary field tests. Since the ORT is a single polarization telescope, the FFT outputs of the north and south halves of the ORT antenna array (same polarization) were connected to two polarization input channels (e.g., treating them as they were dual circular polarization channels). In this mode of connection the Stokes parameters calculated by the polarimeter do not represent polarization characteristics, but represent physically different terms, namely, the Stokes I gives the sum of the total powers from the two halves of the telescope, V gives the difference in the power from North and South halves and $(Q + jU)$ represents the complex correlation between the North and South array voltages. This method of connection was sufficient to allow us to critically evaluate the performance of the machine even though the ORT does not possess dual polarization facilities. The two DSP nodes together tap a total of 64 channels out of the 256-point spectrum corresponding to a total of 4 MHz. The power levels of the North and South halves were matched by adjusting the gains of respective receivers such that the average spectra of parameter V was minimized. The difference in the arrival time of the signals in the two halves is reflected in the cross-correlation phase, defined as $\phi = \tan^{-1}(U/Q)$, showing an apparent phase gradient across the band. From this measurement, the delay was equalized at the IF stage of the two receivers. After phase equalization, a strong pulsar (B1749-28, Average flux density $\sim$1.3 Jy) was observed and the pulses were de-dispersed and folded on-line. The resultant profile showed that the deflections corresponding to the pulse are about the same in I and Q parameters, while the contributions in V and U are very small, indicating that the gains and phases were matched adequately.

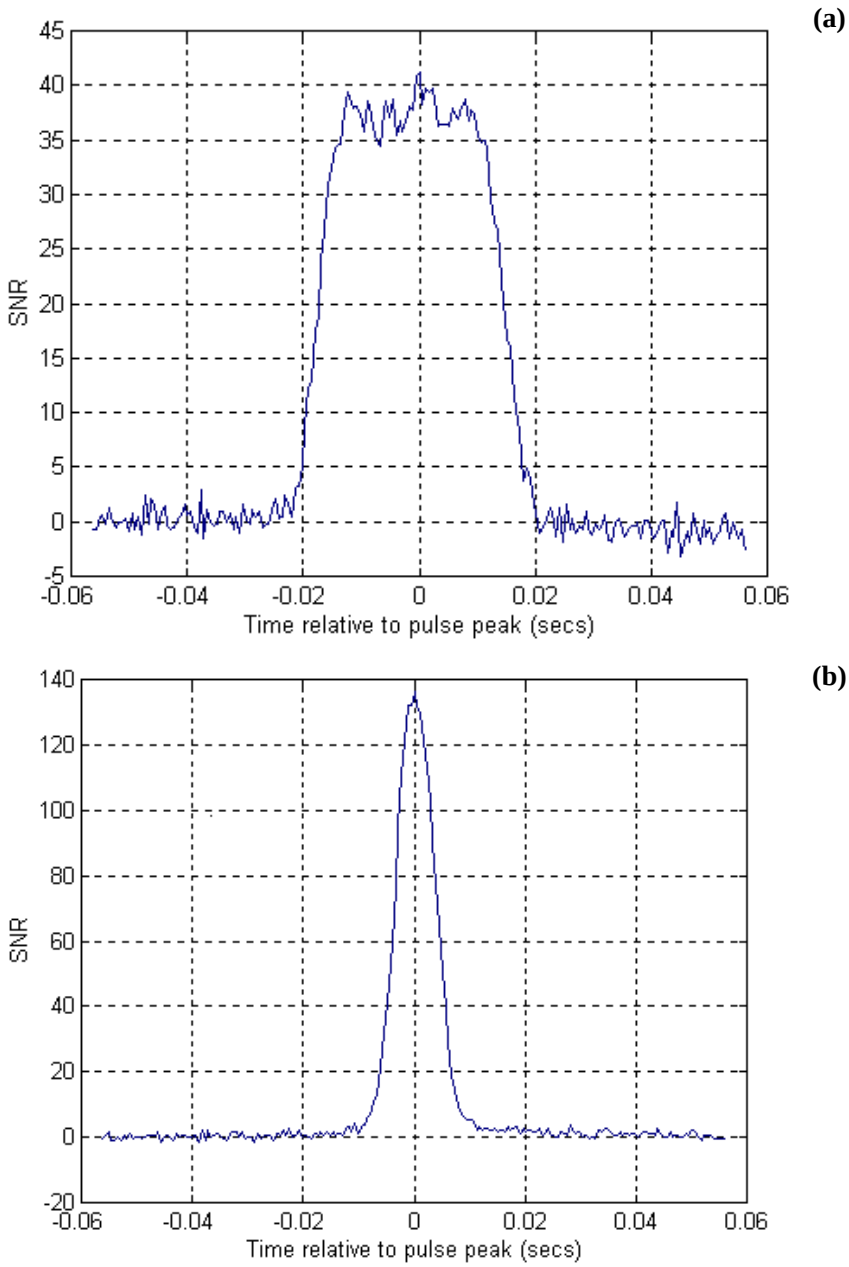

Figure 6. Folded total-power (I) profile of pulsar B1749-28 observed on 5th October 1997 using the Ooty Telescope. (period $\sim$ 0.562 s; DM $\sim$ 50.88 pc/cc; $S_{average} \sim$ 1.3 Jy; equivalent pulse width $\sim$ 7.5 ms; No. of pulses averaged = 580; No. of adjacent samples integrated per time bin in the profile = 27; 64 channels averaged after dispersion correction) Note: The delay difference across the observed band at 327 MHz is about 64 ms. The plots in the top **(a)** and bottom **(b)** panels show the results of the runs without and with the on-line dispersion correction respectively. In both cases the pulse-phase error was allowed to accumulate without correcting it. Significant smearing due to lack of this correction is still present in (b).

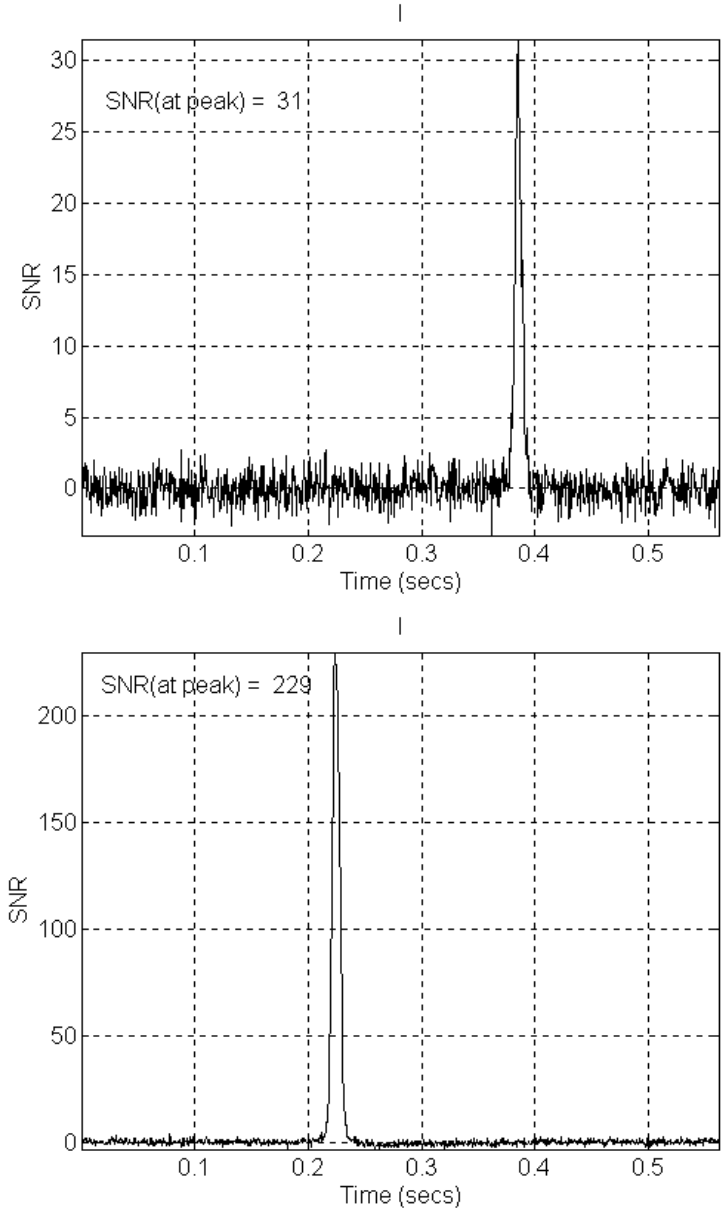

Figure 7. Folded profile (signal-to-noise ratio versus pulse phase) of pulsar B1749-28 observed on 13th December 1997 using the Ooty Telescope. Here, the time resolution is ~ 550 μs (27 sample smoothing) and both dispersion and pulse-phase error corrections were enabled during folding. No. of folds: (top) 18 and (bottom) 1160.

The clock was derived from a programmable frequency generator, which was phased locked to a stable reference signal from a Rubidium oscillator to obtain high accuracy (about 1 part in 10^{12}) and stability. As mentioned earlier, due to finite representation of the pulse phase (32 bit), the residual error will accumulate as the folding pro-

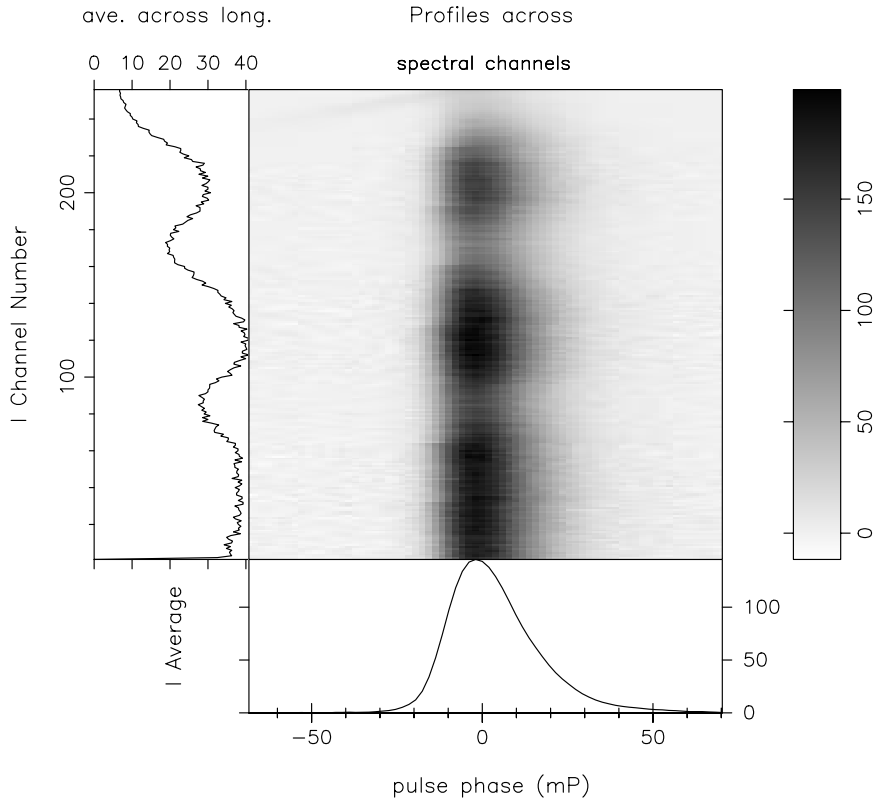

Figure 8. Average pulse profiles (across a 16-MHz band at 610 MHz; 256 channels) from an observation of the Vela pulsar (B0833-45) using one dish of GMRT are shown in the central panel and the average over the entire spectral band is given in the bottom panel. The profiles shown are for the Stokes parameter I (uncalibrated), where the relative delays due to dispersion (DM $\sim$ 68 cm^{-3}pc) have been corrected. The x-axis scale is in milli-periods where the pulsar period is $\sim$ 89 ms (with 512 bins across the period). The left panel shows the apparent intensity variation across the spectral channels, and reflects primarily the spectral gain response of the analog (RF/IF) system. The polarization data (for all the 4 Stokes parameters) were recorded after on-line folding over 3000 pulse periods ($\sim$270 seconds integration). The two polarizations channels input to the system were dual-circular.

gresses, which is corrected periodically. The data of pulsar PSR 1749-28 was processed (dedispersion + folding) with and without the phase-error correction and a significant change in the pulse shape was observed (see Fig. 6). After this, the phase increment value was updated every second so as to adapt to any changes in the period. Further observations were conducted in this mode. To test the spectral-integration function, pulsar PSR 1749-28 was observed with dispersion correction and all frequency channels added together and folded over the pulsar period. Such observations were repeated for different lengths of time and a corresponding improvement in signal to noise ratio (SNR) was as expected (see Fig. 7). Subsequently, many other pulsars were observed covering a wide range of periods, flux-densities and DMs (Ramkumar 1998).

Subsequent tests checked another feature of the instrument, namely, the pulse-gating operation. Also, the feedback from these tests was used in improving the performance

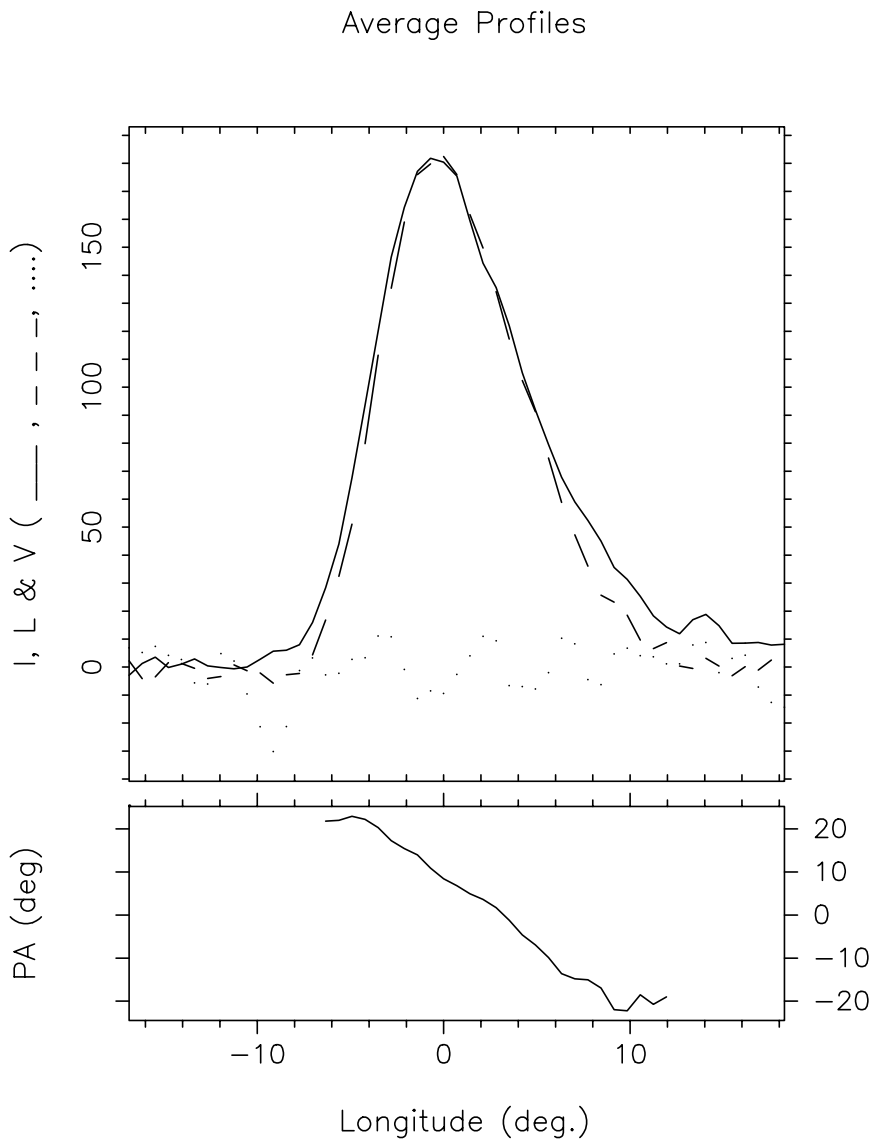

Figure 9. A full set of Stokes parameters for one of the 256 channels from the observation mentioned above (i.e. Fig. 8) is used to compute profiles in total intensity (Stokes I), the linear and circular polarized intensities (L & V) and are shown in solid, dashed and dotted lines respectively. The lower panel shows the corresponding position angle (PA) profile. Please note that these profiles are shown only as an example of typical raw data output from the processor and are not corrected for any (complex) gain differences between the input polarization channels (and their possible coupling). However, the high degree of linear polarization and the PA sweep rate apparent in the raw data are generally consistent with known polarization properties of the Vela pulsar (see, for example, Radhakrishnan & Cooke 1969). The PA profile shown includes the offsets due to Faraday rotation and the (feed) parallactic angle.

of the system. While an enhanced version of the SPPS system was being reproduced to handle full 32 MHz bandwidth at GMRT, the two-node 4-MHz system was used for conducting further tests with the dual polarization antennas of the GMRT array. A result from a series of test observations with the final full bandwidth system commissioned later at GMRT, and now available for observations, is shown in Figs. 8 and 9. A user-friendly software package for the post processing of pulsar data recorded by this processor has been developed.

The system can be used, in general, as a networked, high-speed parallel (SIMD OR MIMD) signal processing computer for many other signal processing applications. Program development tools provided by third-party vendors can be used to develop software in C/Assembly languages and downloaded to a chosen set of nodes. The resources of each node are scalable in terms of the depth of a FIFO and SRAM modules. Provision has been made for additional dynamic RAM interfaces on both the programme and data memory sides of the DSP chip allowing for larger size applications. A separate shared, scalable memory module can be added and shared by all the nodes, if required, by suitably designing a memory bank with an interface that mimics the hand-shakes of a DSP slave node. With such an interface, the memory bank can be treated as an additional node of the parallel processor and this memory can be directly accessed by other DSP nodes in the system. The PCBs are capable of performing at higher clock rates and the frequency of operation can be scaled up with the availability of suitable devices. With minimum modifications in the control software, the DSP nodes can be configured to execute together and perform either parallel (identical) tasks or different tasks/jobs that can be arranged for parallel execution.

Acknowledgements

It is a great pleasure to acknowledge individual and collective contributions from all the members of the GMRT-pulsar project team at the Raman Research Institute and the staff at the Ooty Radio Telescope. We also thank V. Radhakrishnan, G. Swarup and V. Balasubramanian for their constant support, encouragement and for many useful discussions. PSR gratefully acknowledges many useful discussions with K. Kishan Rao.

References

Deshpande, A. A. 1995, Proceedings of the 6th Asia Pacific Regional Meeting of IAU (1994), Supplement to *J. Astrophys. Astr.*, **16**, 225.

Hankins, T. H., Rickett, B. J. 1975, in *Methods in Computational Physics*, (New York: Academic Press, 1975), Vol. 14, page 56.

Kraus, J. D. 1966, *Radio Astronomy* (McGraw-Hill Book Company: 1966).

Manchester, R. N., Taylor, J. H. 1977, *Pulsars*, (San Fransisco: W. H. Freeman and Co.)

Prabu, T. 1997, *Array Combiner for GMRT*, M.S. Thesis, Dept. of Electrical Communication Engineering, I.I.Sc., Bangalore, India.

Radhakrishnan V., Cooke D. J. 1969, *Astrophys. Lett.*, **3**, 225.

Ramkumar, P. S. 1998, *Real-Time Signal Processing Instrumentation for Search and Studies of Pulsars*, Ph.D. Thesis, Regional Engineering College, Warangal, India.

Swarup, G., Ananthakrishnan, S., Kapahi, V. K., Rao, A. P., Subramanya, C. R., Kulkarni, V. K. 1991, *Curr. Sci.*, **60**, 95.