CHAPTER FOUR
Signal Processor for Pulsar Studies

This chapter presents the hardware and software design of a Signal Processor for Pulsar Studies (SPPS) and the implementation details. This system is designed to handle the real-time processing of pulsar signals, where most of the distortions caused by effects that are not intrinsic to the pulsar radiation mechanism are removed. The implementation taken into account the optimizations presented in the previous chapter. Each of the major blocks of circuitry was functionally simulated by identifying equivalent functions in software, before going ahead with the hardware implementation. After the system was built, test patterns were generated and fed to the actual system and the performance was checked at full speed. In the discussion of the design, it is assumed that the front-end systems of GMRT and ORT described in chapter 1 are available. However, while discussing tests and results, some modifications based on presently available front-end systems have been taken into account.

4.1. Specifications for the SPPS:

As discussed in chapter 1, pulsar signals are usually very weak and buried in the noise contributed by the background sky and the receiver. To cater to the wide range of periods and the flux densities of pulsars, the receiver must be extremely sensitive and should provide high time resolution. The specifications of SPPS were derived after incorporating the optimizations mentioned in chapter two. Table (4.1) describes the specifications of the front-end system.

Table (4.1)

Base-band	1,2,4,8 or 16 MHz per sub-band, 2 sub-bands per polarization, 2 polarizations.
Sampling rate and data representation at FFT input	Nyquist rate, 1-bit sign, 3 bit magnitude.
FFT output	256 channel complex frequency spectrum (per sub-band, polarization), generated with 512 input samples.
Array combiner output format	1-bit sign, 8 bit magnitude each, for real and imaginary part of any frequency, polarization, sub-band. Two parallel outputs (one per polarization) giving incoherent (detect & add) and phased array (pre-detection addition) combination.
Antenna masking at array combiner	Any set of antennas can be masked; two sub-bands can be used to operate at two different operating frequencies by masking off different set of antennas in each sub-band.

Specifications of the front-end system (Analog base-band system, Sampler, FFT, and Array Combiner)

The philosophy adopted in this design was aimed at reducing the size, complexity and cost while providing flexibility and operational simplicity. Two identical systems will be used to cater to the two sub-bands separately. The further description will be therefore on the details of instrumentation for only one sub-band, (the other being identical). With these points in view, the processor is considered to receive 256 complex spectral channels over 16 MHz base bandwidth in a time-multiplexed format from the two polarization channels. The block diagram shown in figure (4.1) outlines the different sections of the SPPS for

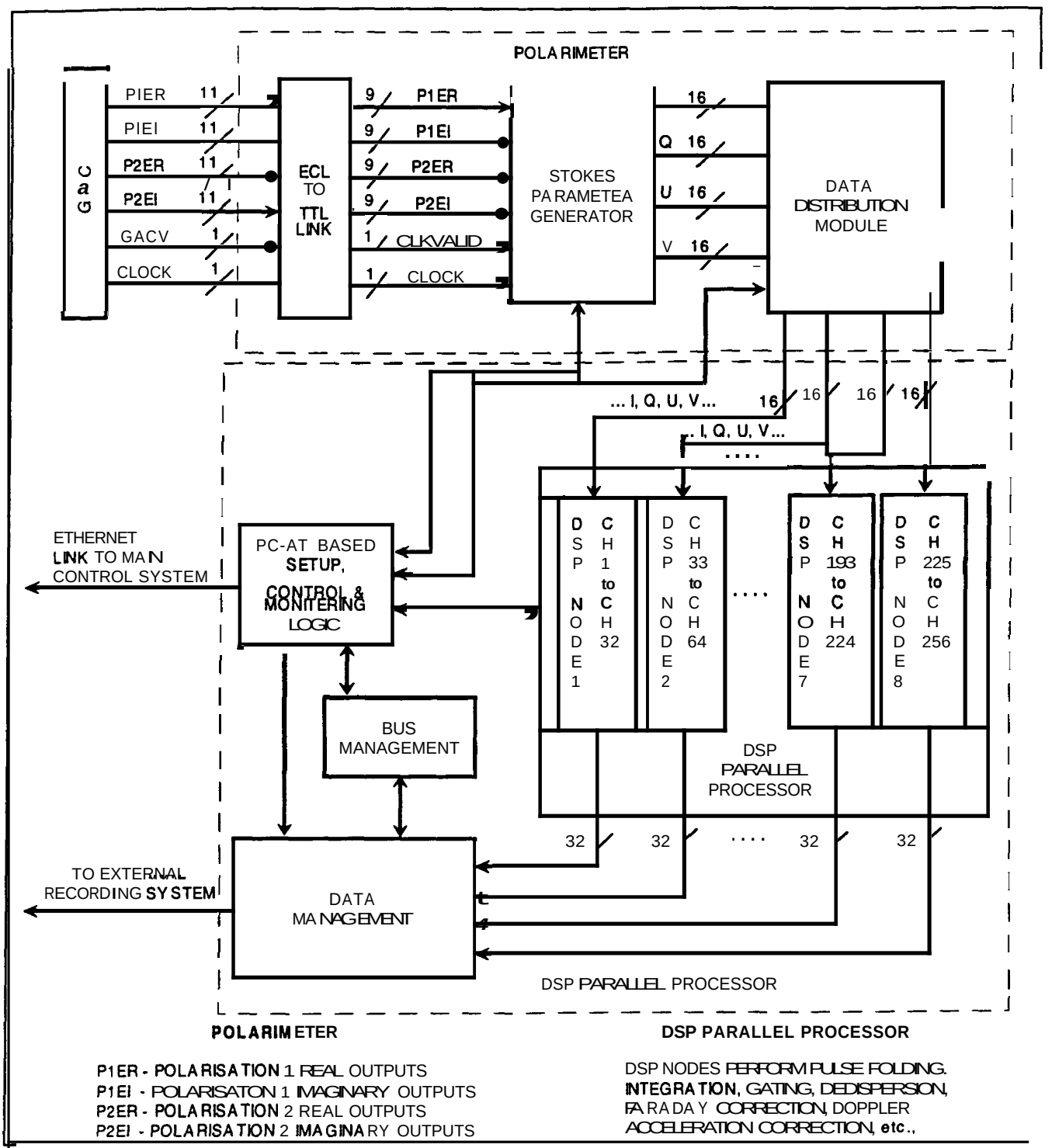


Fig. 4 1 Overall block diagram of signal processor for pulsar studies

one sub-band. The specifications of the inputs to the polarimeter are same as those mentioned in chapter three for the PSP with the phased array signals of two orthogonal polarizations. The two polarization feeds at the antenna are expected to be right and left circular terms at all operating frequencies of GMRT except at 1420 MHz, where linearly polarized feeds are used. The SPPS has two basic blocks. The first is a polarimeter module capable of producing Stokes parameters, where the total intensity, linearly polarized intensities and circularly polarized intensity are computed using the voltage samples of the two orthogonally polarized input channels. The second block is a 16-node DSP parallel processor. Each node is designed to handle all the Stokes parameters of a selected bank of 32 spectral channels produced at the output of the polarimeter. Each node performs jobs such as pulse folding, de-dispersion, Faraday de-rotation, Doppler correction, adjacent sample integration, pulse gating, data formatting, etc..

4.2. Implementation of Polarimeter:

The specifications of the polarimeter were drawn as listed in table (4.2) after incorporating the optimizations mentioned in chapter 2. In-depth studies of pulsar polarization demands full polarization information (meaning, the linearly polarized, circularly polarized and unpolarized intensity components of the pulsar radiation) with a time resolution limited only by the recording capabilities. Thus, the polarimeter is designed as a separate module, capable of producing polarization information in terms of Stokes parameters at full speed, limited only by the raw bandwidth of the receiver. The incoming data is accepted in the form of 256 spectral channels in time-multiplexed form with one complete spectrum every 16μs. In connection with the GMRT system, the data from the polarimeter needs to be processed further in terms of pulse-folding, gating, Doppler correction, etc., in real time. Since the effective data rate after calculation of Stokes parameters will be about 128 Mbytes per second, the data output needs to be split into multiple paths, so that separate modules in the following system can accept a smaller number of frequency channels at a slower rate and process the data independently in each module. Thus, a data distribution circuit is needed to follow the basic polarimeter. It is preferred to host this distribution logic also in the same board as the polarimeter since the data rate after the distribution logic will be slower and easier to transmit to subsequent modules.

As shown in the block diagram of figure (4.2), the polarimeter module has two major blocks, one for calculation of Stokes parameters, another for distributing the Stokes parameters of different frequency channels to eight different data output paths. Each path renders 4 Stokes parameters in each frequency channel and 32 frequency channels of respective channel groups in a time-multiplexed form.

Table (4.2)

INPUT TERMS	Digital data representing complex voltage samples from two orthogonal polarized (both linearly or both circularly polarized) antennas.
INPUT REPRESENTATION	Sign-magnitude numbers, 1-bit sign, 8-bit magnitude each for real and imaginary parts of the complex voltage samples.
NUMBER OF INPUT FREQUENCY CHANNELS (TIME-MULTIPLEXED FORM)	256 per sub-band.
NUMBER OF OUTPUT DATA PATHS	8 data paths, with the spectrum being split into sets of 32 consecutive frequency channels per data path. The Stokes parameters of respective group of 32 channels are sent in time-multiplexed form in each path.
TECHNOLOGY	FLEX EPLDs, PROM look-up tables.
OUTPUT DATA REPRESENTATION	Sign magnitude numbers, 1-bit sign, 15-bit magnitude for each Stokes parameter.
OUTPUT LINEAR POLARIZATION ANGLE ERROR LIMIT	10.5 deg.
OUTPUTDATARATEPERDATAPATH	128Mbytes per second after Stokes parameter calculation, 16 MWords per second per data path after splitting for 8 nodes.
POLARIZATION REPRESENTATION	In terms of Stokes parameters I,Q,U,V, (un-normalized) representing total power linearly polarized intensities and circularly polarized intensity.

4.2.1. Calculation of Stokes Parameters:

The basic equations relating the complex voltage samples to the Stokes parameters are as listed in table (4.3). The table also lists the number of multiplications and additions required for calculation of each Stokes parameter. For the sampling rate listed in table (4.1), the total computation rate will be 544 million operations per second (MOPS) for calculating all four Stokes parameters. Using the high-speed design

optimizations mentioned in chapter 2, this circuit has been implemented as a hybrid design involving the use of both look-up tables and dedicated high-speed logic built into EPLDs.

Table (4.3)

USING LINEAR INPUTS	MULTIPLICATIONS	ADDITIONS	No. of operations	PROCESS
$\langle I \rangle = \langle e_x e_x^* + e_y e_y^* \rangle$	8	5	13	Self Correlation Sum
$(Q) = \langle e_x e_x^* - e_y e_y^* \rangle$	8	5	13	Self Correlation Difference
$\langle U \rangle = \langle \mathrm{Real}(2\, e_x\, e_y^*) \rangle$	3	1	4	Real part of cross correlation
$\langle V \rangle = \langle \mathrm{Imag}(2\, e_x\, e_y^*) \rangle$	3	1	4	Imaginary part of cross correlation
TOTAL	22	12	34	
WITH CIRCULAR INPUTS				
$(I) = \langle e_R e_R^* + e_L e_L^* \rangle$	9	5	14	Self correlation sum
$\langle Q \rangle = 2\langle \mathrm{Real}(e_R e_L^*) \rangle$	2	1	3	Real part of cross correlation
$\langle U \rangle = 2\langle \mathrm{Imag}(e_R e_L^*) \rangle$	2	1	3	Imaginary part of cross correlation
$\langle V \rangle = \langle e_R e_R^* - e_L e_L^* \rangle$	9	5	14	Self correlation difference
TOTAL	22	12	34	

As shown in the block diagram of figure(4.2), the first stage of the circuit calculates the desired products using look-up tables and the second stage adds/subtracts these product terms to produce I,Q,U and V. With the specifications of table (4.3), at the first sight, it would seem that input number width (9 bits each) implies a total of 256 K possibilities. However, some combinations are redundant and can be removed. As an example, consider the situation with circularly polarized inputs. In the calculation of total power of each polarization channel, the sign bit can be ignored since the power is the sum of squares of the real and imaginary part. Also, the input numbers are in sign-magnitude form and hence, in general, the product of the magnitudes can be separately computed and the resulting sign is simply the exclusive-or (XOR) operation of the sign bits of the two numbers. Thus, only magnitudes need be fed to look-up tables, while simple gating provides the resultant sign. Considering the computations for all the four Stokes parameters, it is clear that dividing all parameters by a common factor or 2 is not going to affect the measurements. Using this fact, the

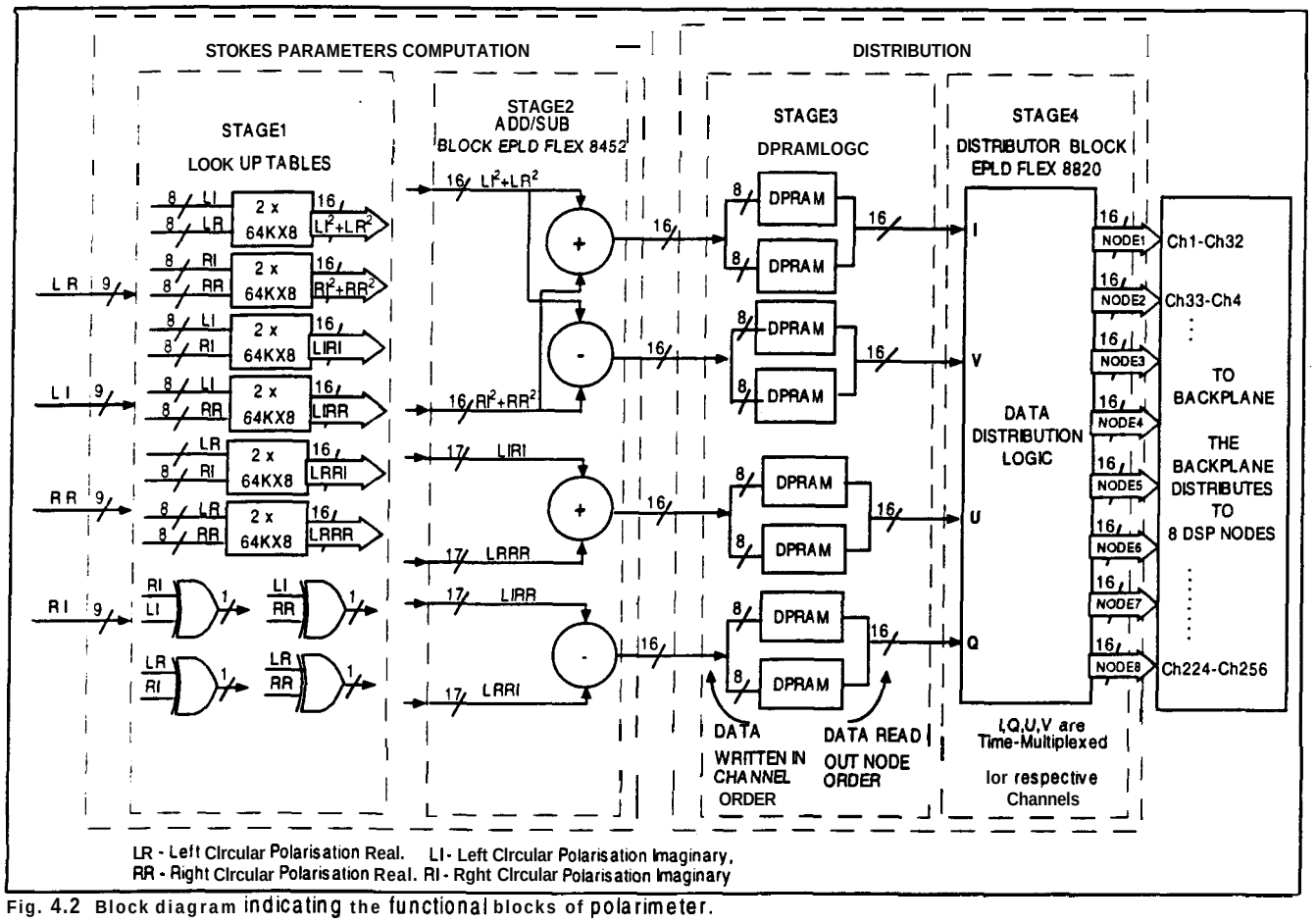


Fig. 4.2 Block diagram indicating the functional blocks of polarimeter.

pre-computation of product look-up tables rounds of the product terms to upper most 16-bits, so as to fit the result within the bus width of the EPROMs. Each EPROM is 8-bit wide, and the 16-bit products are stored in a split manner across pairs of EPROMs. The EPROMs chosen have registers within the chip, so that all the EPROMs latch the products on the rising edge of every clock, every 62.5ns. The "sign" of the corresponding results from XOR gates are also latched in an external register simultaneously. At this point the data width is too large for using look-up tables, so the next operation, namely 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 summing/subtraction requires four adder/subtractor logic modules within the two EPLDs, consuming about 3500 gates. The chosen EPLDs have dedicated fast carry chains available within the device, so that the design can perform the 17 bit additions/subtractions upto 20 MHz speed, using 7 pipeline stages. The routing has been optimized for speed and user defined pin-outs (so as to reduce the complexity in the PCB design).

4.2.2. Data distribution:

The EPLDs mentioned above deliver 4 Stokes Parameters, with 19-bit each, every 62.5ns. Every successive clock produces the Stokes parameters of the next frequency channel and a new spectrum starts after every 256 frequency channels. This means an effective data rate of about 40MBytes per second on each of the four data paths. This data is to be shared and processed in the next stage by a set of 8 DSP

nodes in each sub-band. Each node is designed to handle all of the 4 Stokes parameters of a set of 32 consecutive frequency channels. In such a case, it is difficult to handle the transmission of four wide, parallel busses operating at this speed and the data lines may be prone to interference and cross talk. Besides, the driving length of the bus has also got to be minimized at such speeds. Also, It is obvious that each node has to have a high-speed data link to receive the data and send it later to the DSP. It is difficult to provide four parallel ports and memories of 19 bits on each of the DSP nodes to accept the Stokes parameters in parallel and log them into a temporary memory. Alternatively, if the four Stokes parameters are time-multiplexed, things become worse because the data rate will then be 64M samples/sec! Also, since each node picks up data of a small set of consecutive frequency channels, the data rate is uneven in each node, meaning that the data arrives in bursts at high-speed, with long gaps between bursts (intervals of 8 times the burst length). A novel multiplexing scheme has been evolved to reduce the transmission speed and make it uniform for all nodes. The scheme is implemented as shown in the last stage of figure (4.2).

The data from the EPLDs, i.e., the Stokes parameters I,Q,U and V, are written sequentially into a separate set of DPRAMs, for each Stokes parameter, as they arrive, once in 62.5ns. Since the DPRAMs are available in multiples of 8-bit-width, the lower 2-bits of the EPLD outputs are dropped so that the data fits into 16-bits. This does not result in any significant loss of phase or amplitude information. Consecutive data samples corresponding to consecutive frequency channels gets written in sequential locations of the DPRAM through a port dedicated for writing. The other port of the DPRAMs is dedicated for "reading". Read-out is inhibited till the first spectrum of 256 channels is written into the DPRAMs. During the next spectrum, the read-out is performed simultaneously with the write operation. Since the two ports never address a common location simultaneously, there is no contention between the two ports for data access.

On the "reading" side, the channel order in which the data is read out is different, as shown in figure (4.3). The read address accesses locations 0,32,64, 224 in steps of 32. Then the address rolls back to 1,33,65,....225. This sequence of addressing continues till all channels of the first spectrum are read out in this order. This ensures that the data is read in such a way that every next data sample corresponds to the next DSP node, 32 channels away from that at the previous node. Thus, each node gets data at a uniform rate and the burst transfer is avoided. By the time all the 256 channels of one spectrum are read out, the next bank of 256 locations of the DPRAM would have been written by the write logic. At this point, the read address jumps to the second bank, write pointer moves to the beginning of the first bank. While the first bank now gets written , the second bank is read out.

This process continues, with the banks being switched after every new spectrum is written. The four Stokes parameters are latched in parallel into a block of parallel load, serial out shift registers.The shift registers are interconnected in such a way that when all the 4 Stokes parameters get loaded, the parameter U

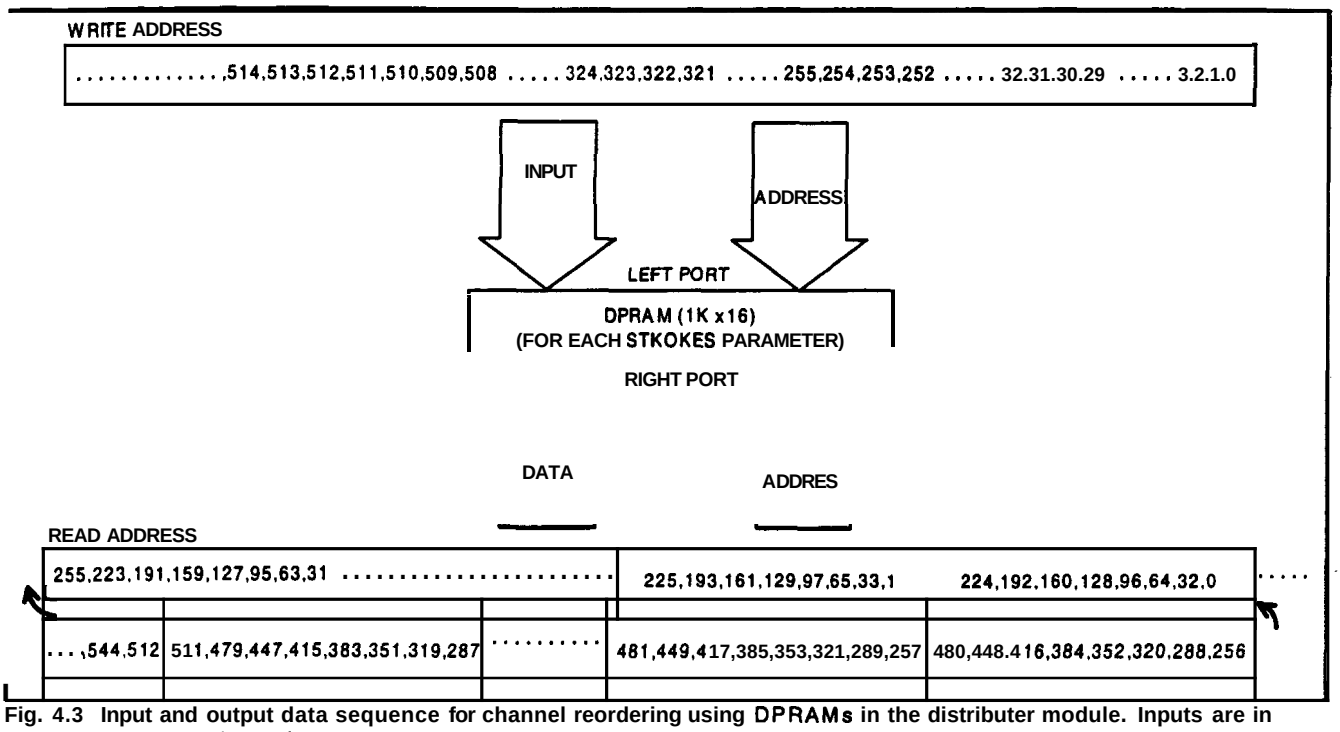


Fig. 4.3 Input and output data sequence for channel reordering using DPRAMs in the distributer module. Inputs are in increasing order of channels,outputs are in order of dsp nodes (32 channels-per node).

is immediately available to the output. Figure **(4.4)** shows a shift register for 1-bit data of all four Stokes parameters, going to one node. A counter based state sequencer logic generates a parallel load and shift states in synchronism with the clock, as shown in figure (4.4c) . The load and shift signals are mutually exclusive. When the valid data arrives, the load pulse is generated in synchronism with the clock. At this time, the next clock edge latches the shift registers with the Stokes parameters. Then the load pulse is laid passive and the shift pulse becomes active on alternate clocks for the next 7 clocks. During this phase, the

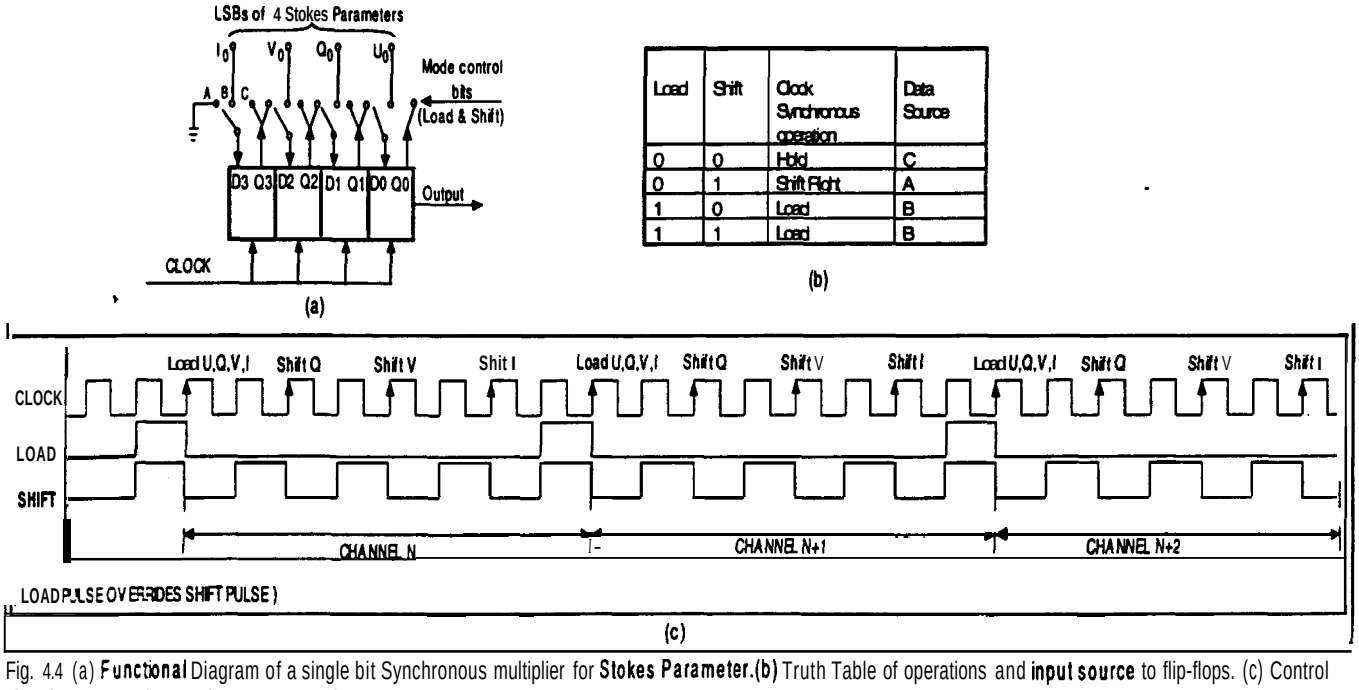


Fig. 4.4 (a) **Functional** Diagram of a single bit Synchronous multiplier for **Stokes Parameter.(b)** Truth Table of operations and **input source** to flip-flops. (c) Control signal sequence for synchronous operation.

shift pulse would have gone active thrice. The data shifts out of this register in the order U,Q,V,I, in every alternate clock. This means that, in effect, the Stokes parameters are time-multiplexed. One such register is dedicated to every bit in the 16-bit field of the data, so that a bank of 16 registers handles the data width of U,Q,V and I for one node (refer. figure (4.5)). The load and shift pulses are common to all registers of this bank.

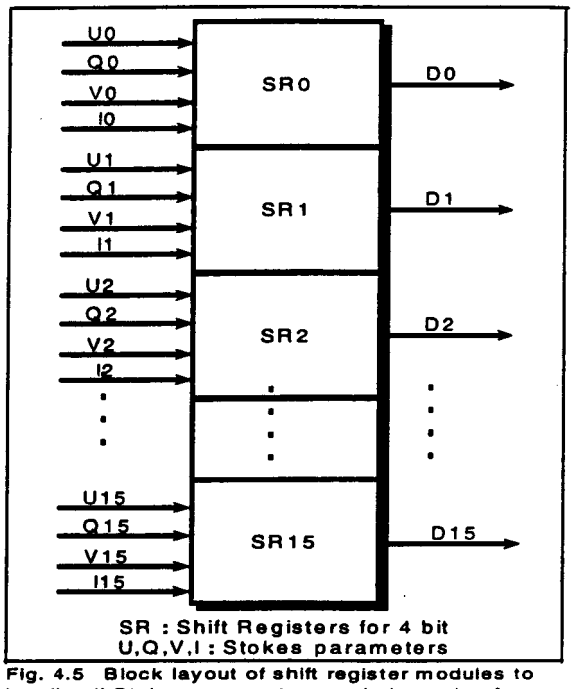


Fig. 4.5 Block layout of shift register modules to handle all Stokes parameters and channels of one DSP Node.

Eight such register banks are placed in parallel (figure 4.6), to handle data paths for eight nodes. The data paths for the Stokes parameters are connected in parallel to all these blocks. Load pulses are supplied to each bank in a staggered fashion, as shown in figure (4.7) and consecutive set of Stokes parameters are latched sequentially at different nodes. The load pulses are staggered from node to node but are synchronous with the data, which is read in node order, as mentioned earlier. The shift pulses for each block are also staggered just as the load pulses, to maintain synchronism with the logic states of respective banks. It is obvious that each bank gets loaded once in eight clocks (512 nsec) and shifts out U,Q,V and I every 128ns. After the initial pipeline delay, data will appear concurrently on all the eight data paths, but with a uniform speed of 8Mwords/second. Each data path is routed to a separate DSP node and a data write pulse is transmitted along with the respective data. The write pulses are also gated before transmission to ensure that they start after the pipeline delays in their respective data paths, so that the DSP gets write pulses only when valid data exists.

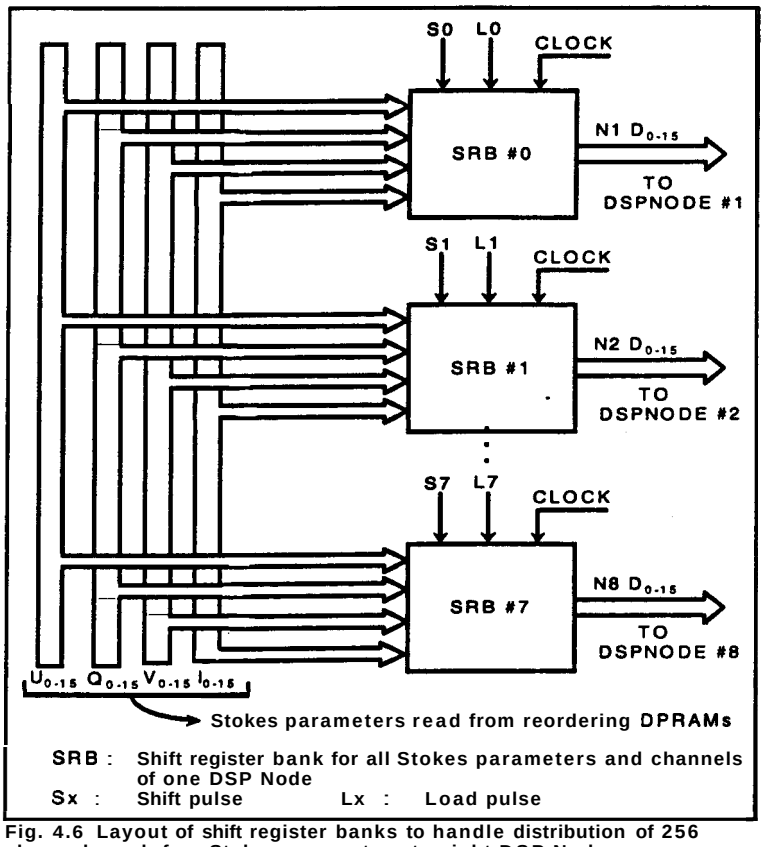


Fig. 4.6 Layout of shift register banks to handle distribution of 256 channels and four Stokes parameters to eight DSP Nodes.

By using this method, we need eight independent buses of 16-bits each, but all of them operating at 8Msamples per second. This reduction in transmission speed and making the transfer at a uniform rate has advantages in terms of being able to drive longer line lengths, better utilization of channel bandwidth,

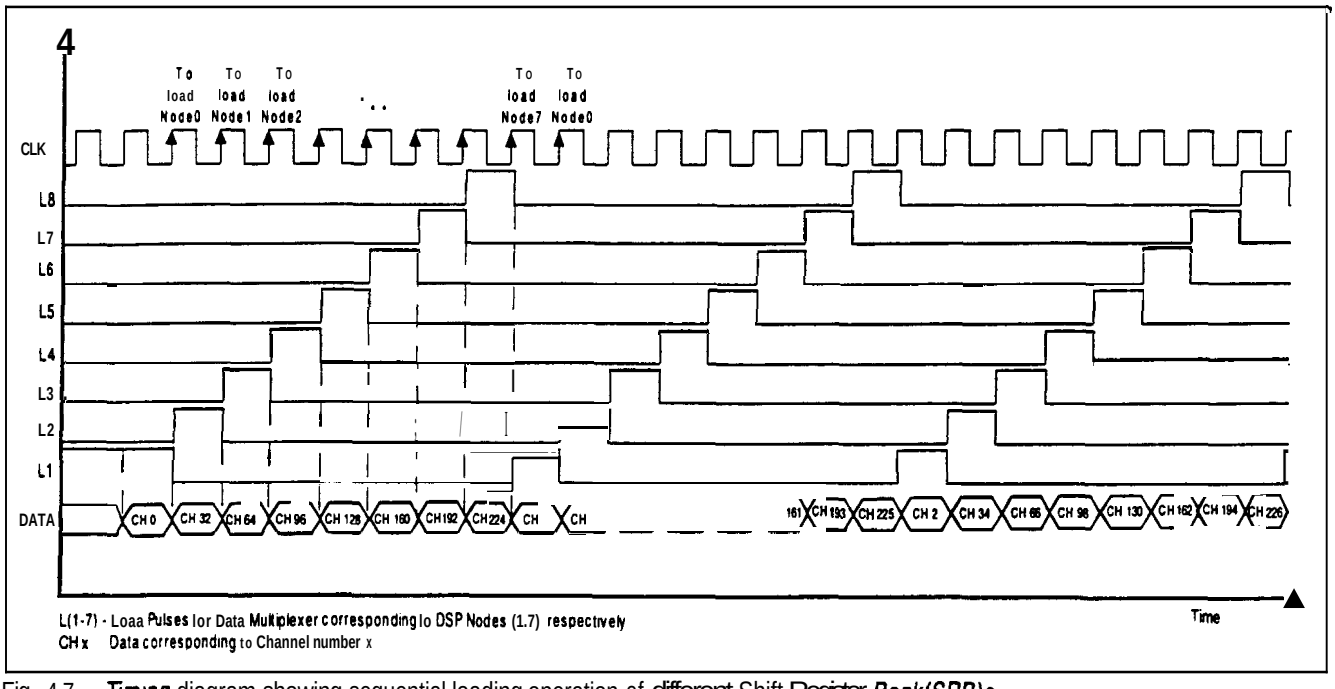


Fig. 4.7 Timing diagram showing sequential loading operation of different Shift Register Bank(SRB)s.

simplification of parallel interface on the DSP nodes, besides allowing a choice of lower-speed components. Although, this results in doubling the number of lines that carry the outputs from the polarimeter module, it is not a major hurdle since a multi-layer PCB backplane is used to carry these lines with sufficient isolation. To summarize, the polarimeter module takes in complex voltage sequences of 256 frequency channels of two orthogonal polarizations, computes the Stokes parameters and distributes them across eight DSP nodes with 4 Stokes parameters of 32 channels per node, sent in a time-multiplexed format, along with suitable write pulses to latch the data at the respective nodes. The data rate at the input of each node is 8 Mwords per second (16-bit data path).

4.3. Design of DSP Parallel Processing System (DSP-PPS):

As mentioned earlier, the total data rate is too high to perform all the operations on every data point using a single processor. Various alternative architectures were considered to handle the real-time computational load posed by the signal processing tasks. Obviously, the aim was to minimize the number of components and have a modular design such that the data flow and control can be highly organized.

4.3.1. Specifications for DSP-PPS:

From the considered optimizations mentioned in chapter 2, some factors were clear with which the specifications for such a system could be laid, as indicated in table(4.4).

Table (4.4)

Number of DSP Nodes	16 (8 per sub-bands).
Number of input frequency channels	32 channels per node.
Input data format	16-bit sign-magnitude format.
Output data format	32-bit signed, 2's complement format
Input Polarization information for each frequency channel	Stokes parameters I,Q,U,V.
Input data rate for consecutive spectra	Upto 8Mwords/sec/node
Options for one or more functions to be performed on the input data	Dispersive delay correction, Faraday de-rotation, Channel collapsing, adjacent sample integration, pulse folding, Doppler acceleration correction, pulse gating, data formatting for I/O.
Result recording systems	PC/AT based recording system and Canadian S2 recorder system
Maximum output data rate limited by recording system	Upto 32 KB/sec on PC based DCS and 8MB/sec on S2 recorder.
Polarization accuracy limited by instrumental noise and ionospheric turbulence	~3 deg.
Minimum number of output frequency channels	16
Maximum number of additions before recording in each data point of resultant profile	65536 additions, from any combination of channel collapse, adjacent sample integration and folding operations.
Maximum time resolution per profile	16μs.

Specifications for DSP-PPS.

Depending on the type of observation, the requirements of frequency and time-resolution, number of Stokes parameters required, minimum number of pulse folds to be performed, etc., were analyzed to maximize the performance and determine the corresponding memory size for temporary storage of data during the processing and the associated maximum data rates at the outputs. The maximum time resolution and number of channels are, of course, limited to that provided by the FFT module. Also, the results are standardized to fit into 32-bit numbers, irrespective of the type of processing performed. If finer time resolution is required, the output data can be formatted to fit into smaller numbers (fewer bits per sample), by simple changes in software, to achieve higher output data rates.

The processing required for average profile studies is computationally most complex, since it involves all operations such as folding, adjacent sample integration, gating, Faraday de-rotation, de-dispersion and Doppler acceleration correction. This needs a scheme where the available memory resource can be redistributed between Stokes parameters, frequency channels and time frames depending on the type of observation. Even though the implementation of all the above functions is possible using a cascade of individual blocks of dedicated logic circuits, it is not easy to rearrange/bypass different blocks depending on the type of observation. Alternatively, the use of a microprocessor based system suits well to provide such a flexibility. In such a system, one has to simply change the program code for the processor to handle a different processing task. The hardware optimization and programming skills can exploit the architecture of the processor to enhance the speed performance of such a system. Noting the possible optimizations mentioned in chapter 2, it was found that building a parallel processing architecture based on DSP chips was an optimum choice. The common signal processing algorithm developed in section (2.1.4) is highly suitable to a DSP-chip architecture. After a survey of the available DSP chips, ADSP 21020 from M/s Analog devices was found to suit well for this type of application.

4.3.2. Architecture of a Single DSP Node:

As mentioned above, some types of observations need that all the functions mentioned be performed on every data sample, demanding an extremely high-speed of computation and data communication. After optimization, the architecture shown in figure (4.8) evolved. Given below is a description of the architecture of one DSP-node (16 such nodes are to handle the total GMRT band-width).

- The input data are available for the DSP node from a memory located on the program memory (PM) side. The data flowing from the polarimeter get 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 polarimeter to the DSP node is unidirectional and the writing/reading operations are independent as well as are 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 polarimeter and the read side is connected to the DSP node. As explained earlier the polarimeter sends out Stokes parameters of 32 channels in a time-multiplexed fashion in 16-bit sign-magnitude format. However, the DSP micro-processor expects data in a 32-bit, fixed point, 2's complement format. To convert the formats and enhance the data width, a set of EPROMs are used as look-up tables, to accept the input data as its address and deliver the corresponding converted data.

- The parameters required for processing (such as Faraday rotation correction phase, Bin phase increment, etc.) are available from a separate memory on the PM side. The instruction code for the DSP node is also located in this memory. It will be necessary to update the process parameters on-line, without disturbing the processor in DSP node. It will be convenient to have 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 (DPRAM, 25ns access time) block is chosen to hold the 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.

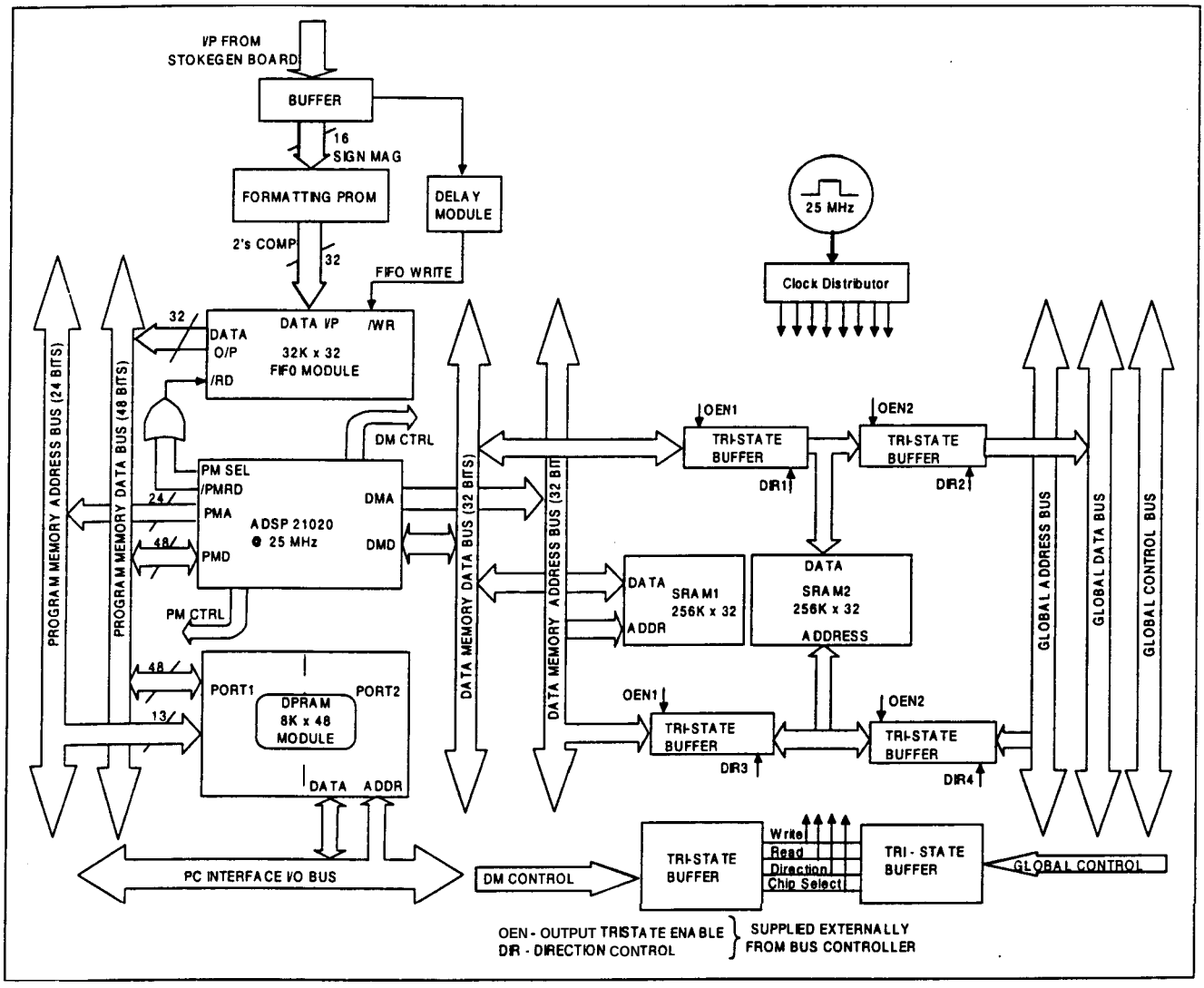


Fig. 4.8 Block level architecture of one DSP node

- A memory block in the data memory side of the DSP node holds the temporary results during the processing of the data. This memory has to be fast enough to match the DSP read and write speeds without requiring any wait cycles and is dedicated to the processor. This is realized in the form of a 32-bit SRAM module designated SRAM-A, (25ns access time) which is interfaced to the data memory (DM) side of the DSP node. This memory is logically organized into two halves (banks). One half is used for processing and after the processing is complete for one set of data, the second half is used for processing the next set of data. Meanwhile, the result can be read out from the first half, so as to empty the first half

before the processing in the second bank is complete. Then the processing is done on a fresh data set in the first half while the results in the second half are transferred.

- 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 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 on the DSP node side are turned-on while those on the output bus side are tri-stated. This connects SRAM-B to the DSP node. After the results are written, SRAM-B (about 25ns access time) is disconnected from the DSP node by tri-stating the buffers on the processor side. 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 DCS is explained later. Even though the flow of results is always from the DSP node to the DCS, 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. By providing this, 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 and hence is beyond the scope of this thesis.

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.

4.3.3. Software architecture of a DSP node:

As mentioned in section (2.2.3) of chapter two, the chosen DSP chip provides a modified Harvard architecture. As mentioned in the subsequent example, such an architecture enables parallel executions of several operations. It is therefore important to identify independent operations and to sequence the algorithm in such a'way as to maximize the number of operations that can be executed in parallel. By trial and error method, a number of models of implementation were tried out and the final algorithm was arrived at. The total operation 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. Within the processing routine, the code has been organized into a ***core-loop*** and an ***outer shell.*** The hardware architecture of the DSP node mentioned above was arrived at in an effort to match the requirements of the common signal processing algorithm derived in section. (2.1.4) of chapter 2. An additional ***Boot*** routine has been developed which runs as soon as the system is switched on, to perform basic house keeping functions, before the signal

processing routine is loaded and executed. Given below is an explanation of the algorithm, in accordance with the desired operation-flow during processing.

4.3.3.1. BOOTING THE SYSTEM:

When the machine is switched on, it is necessary to execute a general diagnostic routine to perform "health checks" on each node, under the supervision of a central controller. In this system, the functions of the central controller are implemented on a PCIAT platform, with suitable interface through parallel I/O ports hosted on the ISA bus of the PCIAT. Specifically, the parallel ports are I/O mapped on the PC bus, and their outputs are grouped to form the address, data, control and status buses which connect to the DPRAMs in the DSP nodes, as shown in figure (4.9). 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, requiring 5 additional address bits. Consecutive DPRAMs are organized as pages, and for each sub-band this results in a total of 24 pages. By this method, the data bus needs to be only 16-bits and the address bus only 13 bits wide. To access any DPRAM, its page number is initially written into a control register. A decoder circuit then uses the registered value and selects the DPRAM to which further communication is linked. The selected page remains connected to the PC till the page number is updated to a different value. A portion of this control register is also used to set and reset the interrupts to individual nodes, so that the selected one or more nodes can be reset under software control.

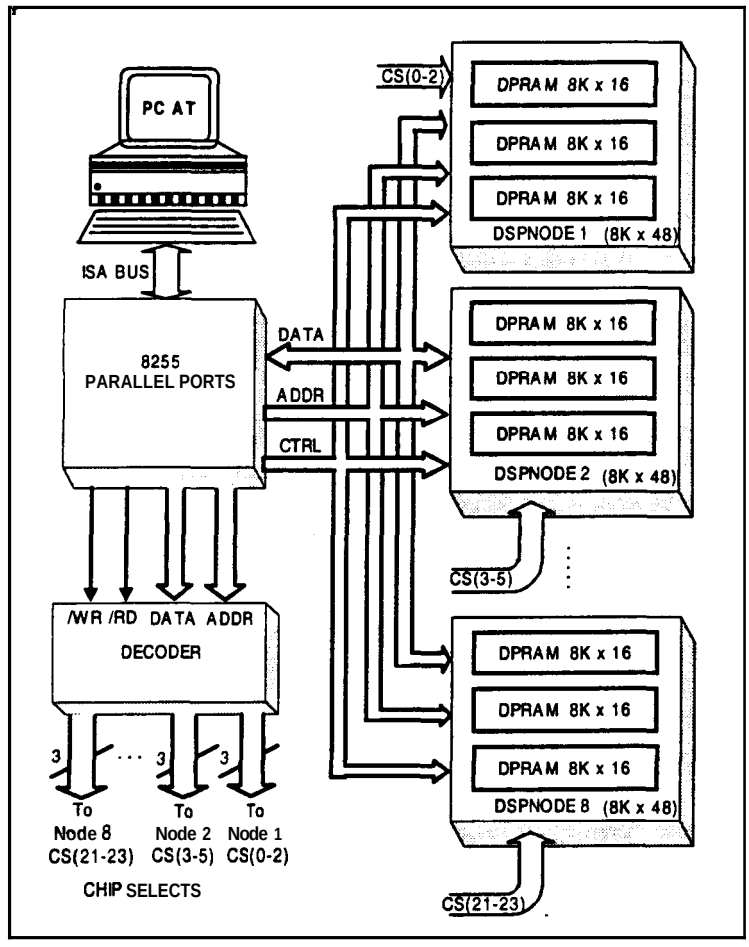


Fig 4.9 Page layout of DPRAMs for linking control PC to DSP nodes.

The entire control software running on the PC is written in the C language. Figure (4.10a) outlines the flow of operations on the control PC. Under the control of this routine, the PC initially keeps all DSP nodes under the 'reset" condition and writes standard test patterns into the DPRAMs, reads them back and cross checks and locates mismatches if any. These errors are reported to the user. Three test patterns are chosen for tests of different aspects: a) an alternating of all 0s and all Ones (00 and FF hex) is written to check if there are d ving problems when the chips have to source/sink full load; b) an alternating sequence of values (AA

and 55 hex) such that alternate data line are high while remaining are low, so as to check the possibility of any shorts/coupling between adjacent data lines; c) a ramp sequence (0 to FFFF hex) so as to check the response of the data path for different combinations of data values at the required speed. Only those nodes that pass all the three tests are selected for further interaction, i.e., a shortlist is formed out of the available nodes. The program then reads the file containing 48-bit instruction codes that are to be down loaded to the DSP nodes and splits each instruction into three sections of 16-bits each, so that each section fits into a DPRAM. These codes are then written into the correct pages such that each set of three consecutive pages forms the correct instruction code sequence for a particular DSP node. These codes are read back and cross checked before letting the DSP node execute the code. The code is a boot routine written for initial setup and diagnostics to be conducted by the DSPs. After loading the code, the reset on the selected DSP nodes is released so that the nodes can execute the boot code.

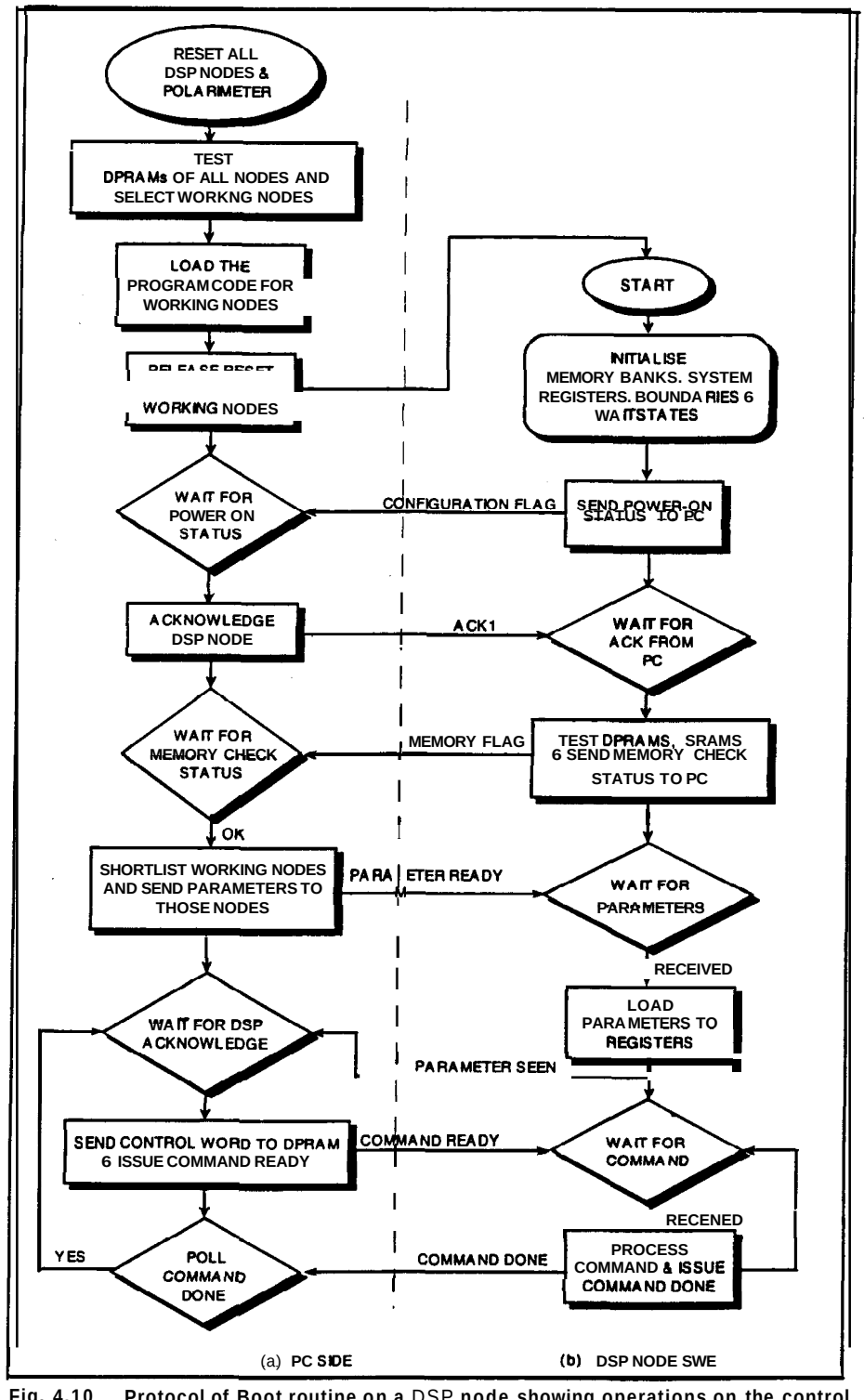
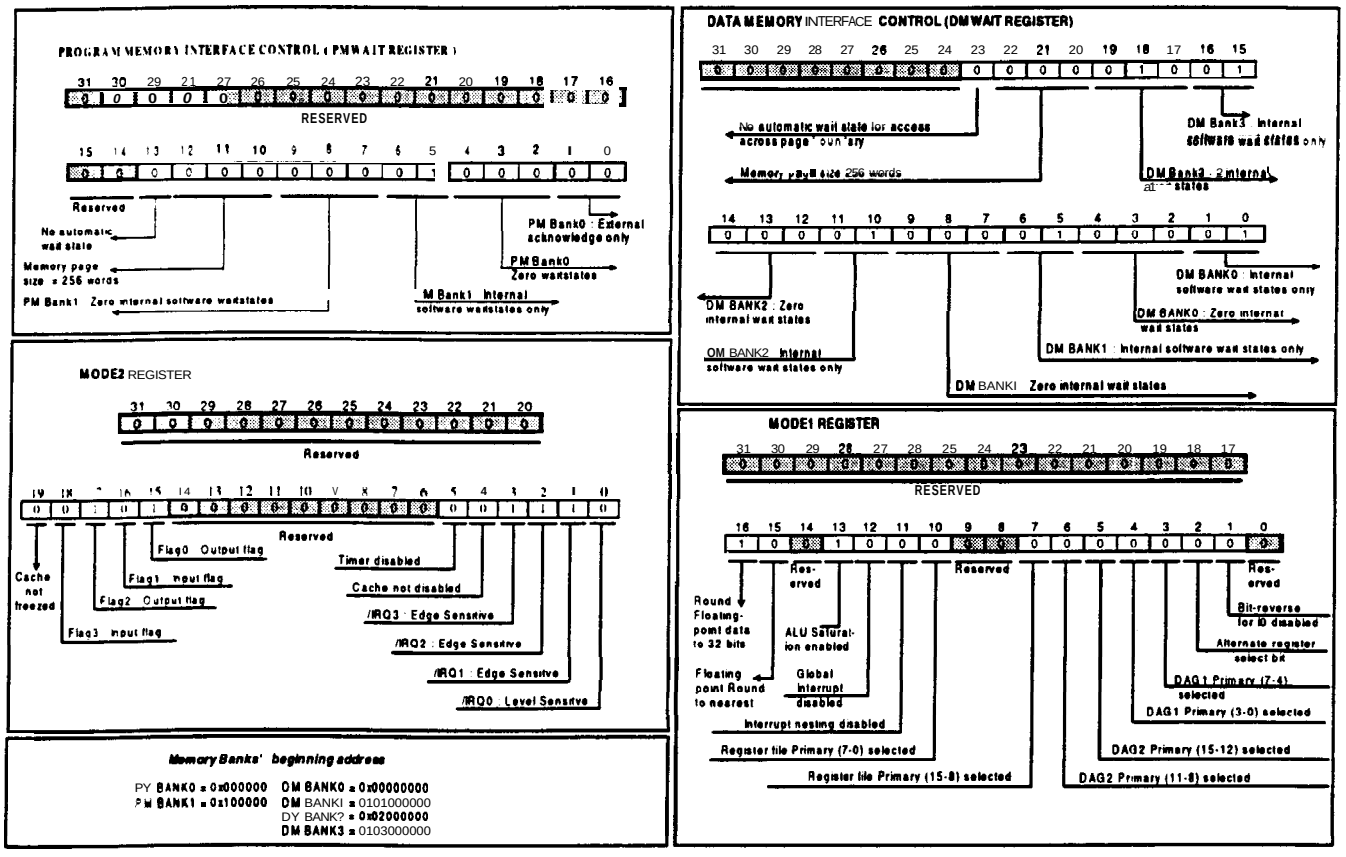


Fig. 4.10 Protocol of Boot routine on a DSP node showing operations on the control PC (a) and the DSP node (b).

The steps followed by the DSP boot routine is shown in the flow chart of figure (4.10b). The architecture of the DPRAM provides for generation of two interrupt signals. One interrupt signal gets generated to the left port when a fixed location is written from the right port and gets cleared automatically

when this location is read out from the left port. Another location similarly generates an interrupt to the right port when written from the left port and automatically clears when it is read from the right port. These interrupt flags are linked to the status port on the PC side and to an interrupt line on the DSP chip. Initially, the DSP sets its internal control registers to a chosen configuration, indicated in table (4.5). After this, the status register is

Table (4.5)



read and written in a chosen semaphore location in the DPRAM. This operation automatically raises an interrupt signal from the DPRAM chips. Meanwhile, the PC would be polling for the DPRAM interrupt from each node, by reading the status bus. Once it senses an interrupt flag, it reads the semaphore and decodes the status to check if a standard configuration is identified in all nodes. The selected-node list on the PC is further short-listed to contain only those on which the standard configuration is identified. A semaphore indicating an acknowledgment for having received the status is then sent to each node. The DSP nodes would be waiting till they receive an interrupt generated by the DPRAM (generated by the PC when a semaphore is'sent to the DSP). Once the interrupt is sensed, the interrupt service routine performs a dummy read of the dedicated DPRAM location just to clear the interrupt signal and returns to the main routine. In the main routine, the DSP starts local memory checks. It checks SRAM-A and SRAM-B by writing and reading the three patterns mentioned earlier. Then it checks the DPRAM region meant for holding the parameters. It maintains a register to note the success or failure of these tests by setting appropriate bits to logic one or zero depending on the status of each test. The exact allocation of the bits is shown in figure (4.11).

Once the memory checks are completed, the resulting status code is sent to the PC in the same way as mentioned above, and the DSP waits for an acknowledge form the PC. The PC reads out the status of memory check in each node, and further shortlists the table of selected good nodes and writes it into a file and then acknowledges the DSP. After receiving the PC-acknowledge, the DSP node polls for a command semaphore, which can be sent by the PC indicating that the DSP can call a subroutine located at a given location. The starting address of the subroutine is written by the PC into a standard location of the DPRAM, which the DSP reads out to determine where to call. This feature is built in to support execution of any user program after the initial boot sequence. The PC should have loaded the routine to be called into the DPRAMs before giving the command semaphore. The subroutine is expected to end with a return code, which returns control to the Main polling program of the DSP, where it continues to wait for the next command from the PC. This boot sequence is executed primarily to perform an initial health check for the entire machine by the control PC and to identify the good nodes, so that it can communicate further only with the selected nodes.

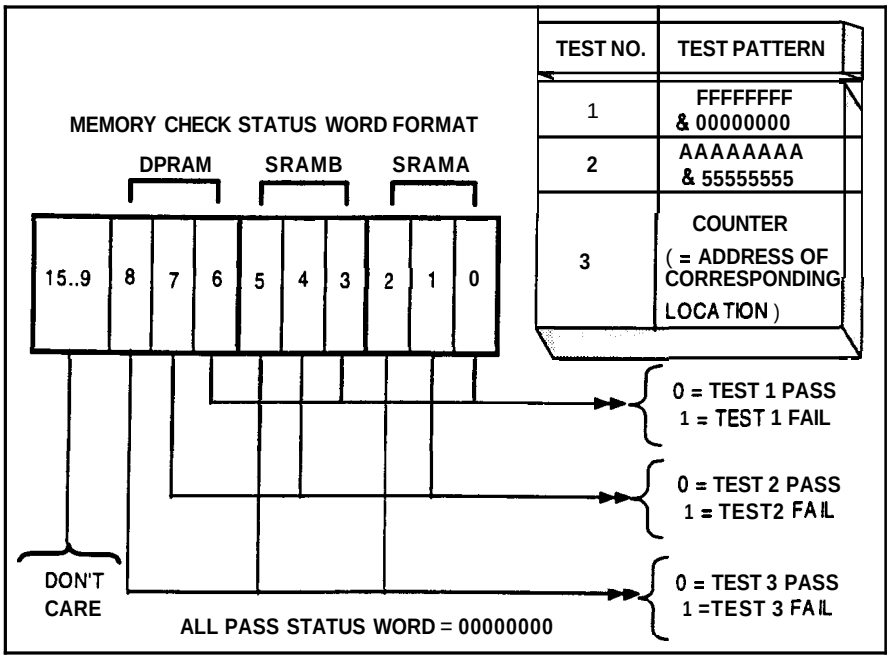


Fig. 4.11 Status word format for self check of DSP node

4.3.3.2. PROCESSING PROGRAM SHELL:

Once the "good" nodes are identified and selected by the control PC, it resets all nodes again and loads the signal processing code and associated parameters into dedicated regions of DPRAMs and it releases the reset for only the selected nodes. The DSP starts executing the outer shell of the signal processing task immediately. The steps of the outer shell routine are outlined in the flow chart of figure (4.12a). The DPRAM memory space is logically partitioned into Code, Semaphore and Parameter space. The parameter space is further partitioned into several tables and values. The layout of the three blocks is shown in figure (4.13a). In figure (4.13b), the partitions of the parameter space are shown. For clarity, the various tables in this area are described along with the code execution, in the following discussion.

Initially, the DSP node clears its SRAM memories and initializes its internal register, the FIFOs and disables the bus transaction for SRAM. Then, it establishes a header table. While reading the results, the data acquisition system is to read out the header also, so that the header serves as a marker between the data from different nodes, channels and Stokes parameters. This will help in case of any data loss during

recording, to identify and skip the bad or discontinuous blocks of results. To establish this header pattern, the DSP needs information as to how many output channels are required and how many memory locations are allocated to fit in one period for every channel. This information is supplied by the PC AT in the NBINS_ADR and NOCH_ADR locations of the parameter space. With this information, the DSP computes the memory interval between successive channels and Stokes parameters and writes these as parts of the headers. The header length is two locations long, in the pattern shown in figure (4.12b). The DSP then proceeds to load some of its computational registers with some constants required for processing the pulsar signals. The register allocation shown in figure (4.14), is done with care so as to minimize the number of dedicated registers and also adapt to the processor restrictions regarding the usage of different registers. Two sets of registers, called the primary and alternate registers, are used, to hold data corresponding to TASK-1 and TASK-2 respectively.

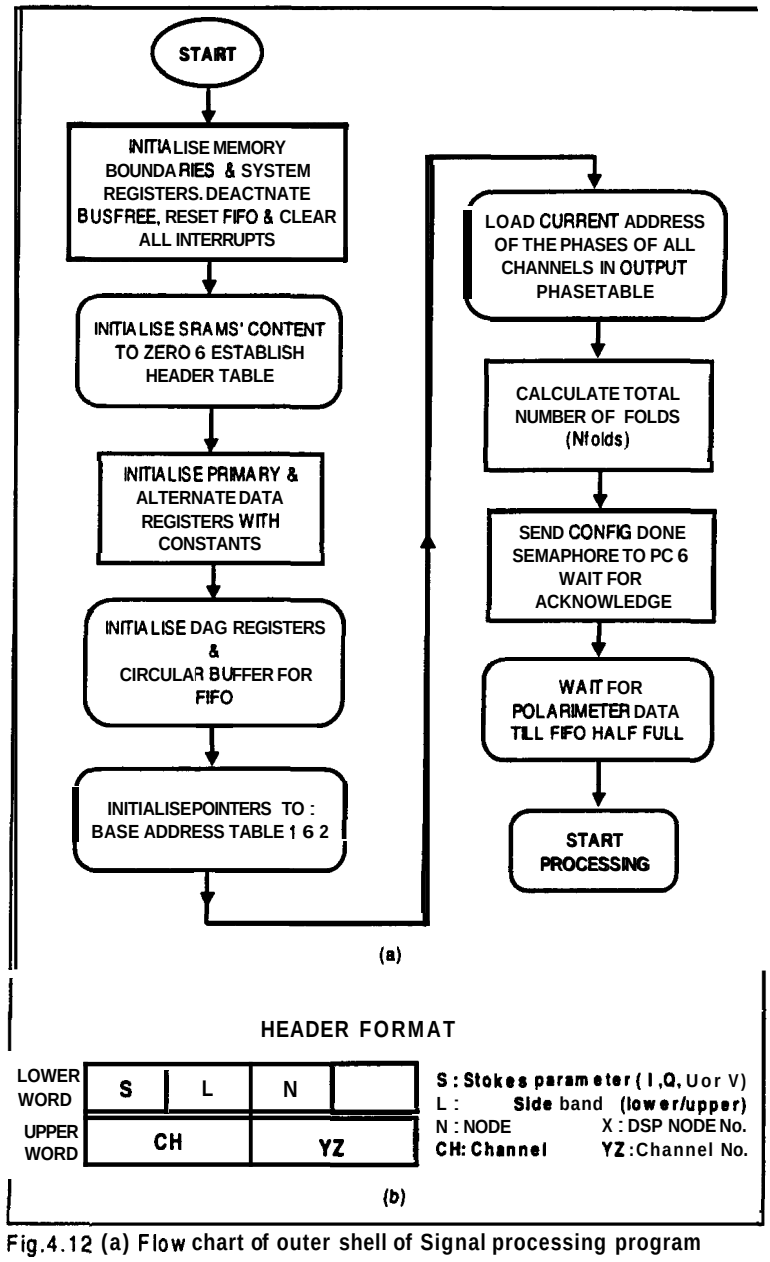


Fig.4.12 (a) Flow chart of outer shell of Signal processing program (b) Header pattern.

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 FIFO is identified as a circular buffer and its length is defined to the processor as half of the actual FIFO sire. This is just to restrict the processor address within a specific region beyond which the pointer should roll back to the beginning of the region and generate an internal interrupt, to the DSP chip indicating that processing one half of FIFO data has been completed. A table of addresses where the profiles in each output channel in SRAM-A are to begin is pre-calculated to allow fast generation of pointers while processing. The Beginning address of any output frequency channel is given by:

$$BEG\_CH_i = (SRAMA\_BEG + BANK\_OFFSET + MARKER\_SIZE) + (MARKER\_SIZE + NBINS) * j$$
$$(i = 0 : Nchin - 1, j = 0 : Nchout - 1) \qquad 4.1$$

As mentioned above, the number of channels to be combined, i.e., the ratio of the number of input channels to the number of output channels, determines the number of increments in i for every increment in j. The processor reads out the starting address every time a data sample is to be processed. Since the data of consecutive frequency channels arrive sequentially, consecutive locations of the table can be read to get the beginning address of corresponding channels. The processing region alternates between the two banks of SRAM-A for every new set of results produced. This requires two sets of base-addresses corresponding to the starting addresses in the two banks of SRAM-A. These two base-address tables are designated as PTR_TO_SET1_CH_BEG_ADR_TAB and PTR_TO_SET2_CH_BEG_ADR_TAB. A separate two location circular table designated as PTR_TO_BEG_ADRS_TAB_TAB, is declared where the two locations contain the beginning addresses of the two address pointer tables. It facilitates fast bank switching during processing, when, for example, the processor is finished with the processing of one bank of SRAM-A, it starts processing fresh data in bank-2, leaving bank-1 results ready for read-out. In addition to all these tables, a separate table is needed to contain the current phase of the profile which as the successive spectra are processed. This table is called OP_PH_TAB. A register is then initialized to contain the total number of folds to be performed before switching the SRAM–A banks for fresh processing. This is given by

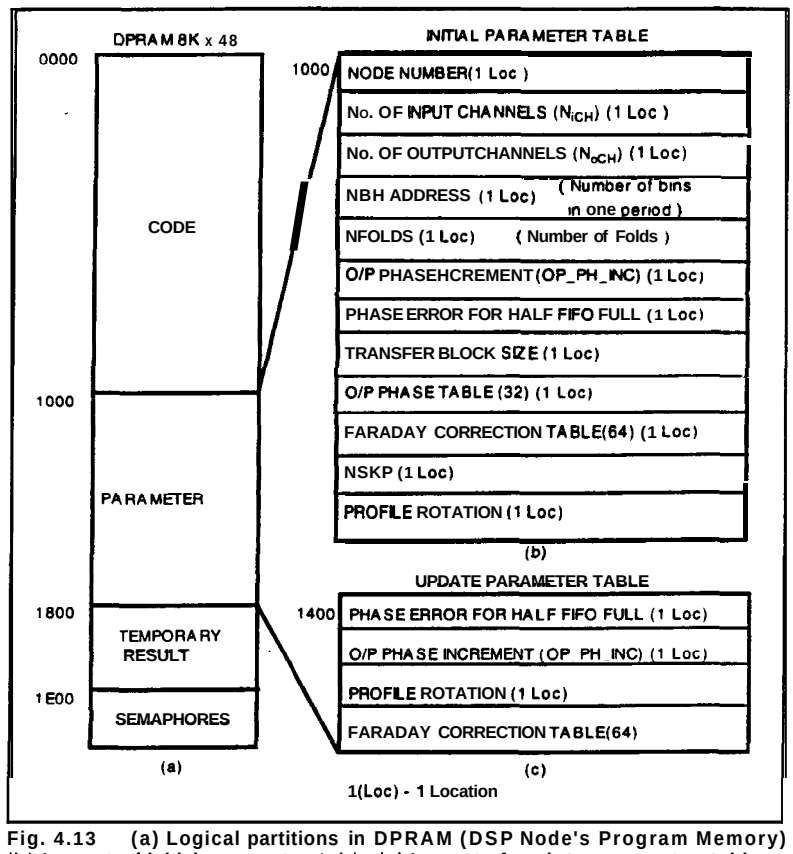


Fig. 4.13 (a) Logical partitions in DPRAM (DSP Node's Program Memory) (b) Layout of initial parameter table, (c) Layout of update parameter table.

$$N_{folds\_total} = N_{folds} \cdot N_{chin} \qquad 4.2$$

This is required so that the DSP can decrement this count every time it finishes one fold in any channel and eventually reaches a zero count, upon which the memory bank is switched. Once these initial configurations are setup, the DSP sends a config-done semaphore to the Control PC through its DPRAM ports and waits for an acknowledge. The Control PC in the meanwhile would have been polling for the config-done semaphore. Upon receiving the flag, it sends acknowledge to all the selected DSP nodes. The PC then enables the gate which allows clocks to flow to the entire system, commencing from the first channel of the next immediate FFT frame that arrives from the array combiner. Upon receiving the acknowledge from the PC, the DSP will start polling the FIFO half-full flags. As the data gets filled into the FIFOs from the polarimeter, half full flags get generated, which invoke further processing. Initially the algorithm for three

operations, namely, index calculation, data processing and fold management are explained sequentially. Later, it is demonstrated how the code for the three operations is parallelized, to save the number of instruction cycles.

4.3.3.3. TASK-1: This task is for processing the data from the FIFOs using SRAM-A for temporary storage of intermediate results and using the parameters from DPRAMs. This is also called the CORE routine, since the code packed into this task is performing a highly optimized, parallel instruction sequence which loops for every set of Stokes parameters received.

DATA REGISTERS

GENERAL PURPOSE	PRIMARY SET	R0,R1,R4,R5,R8,R10,R11,R12, 13,R15
	ALTERNATE SET	R0,R3,R7,R8,R9,R10,R14
DEDICATED REGISTERS	PRIMARY SET	R2 -- O/P Phase Increment value
		R3 -- No. of I/P Channels (NiCH)
		R6 -- Scalefactor for integerization of O/P Pulse Phase --(16 bit right shift) R7 - No. of tins in a profile (NBINS) R9 - NBINS left shifted by 16 bits R11 - Result Base Address R14 - 2 x No. of I/P channels
	ALTERNATE SET	R1 -- Parameter ready semaphore
		R2 -- Pulse phase error limit
		R4 -- Mask Pattern for Semaphore
		R6 -- Scale factor for Intigerization of current phase
		R11 -- NBINS left shifted by 16 bits
		R12 -- Quantization error in pulse phase for every FIFO Half full
		R13 -- Pulse phase error accumulator
		R15 -- DPRAM Interrupt Flag

DATA ADDRESS GENERATOR (DAG) REGISTERS

DEDICATED REGISTERS	PRIMARY SET	B4 -- Result Address table
		B8 -- Address of offset for pulse phase rotation
		B10 -- Pointer to Faraday Correction factors
		B11 -- O/P Phase table address
		B15 -- FIFO Bank Beginning Address
		L15 -- Half FIFO size
	ALTERNATE SET	B0 -- SRAM A beginning
		B1 -- SRAM B beginning
		B8 -- Address of Profile Rotation value

Fig. 4.14 Initial allocation of Registers for various parameters in Pulsar Signal Processing.

Figure (4.15) shows the flow of operations for index calculation. For every input sample set of 4 Stokes parameters, the current position of the pulse point within the profile corresponding to that channel is read from the OP_PH_TAB table. The current phase value has a 32-bit representation in which the upper 16-bits form an integer memory address and the lower 16-bits represent a fractional position within a memory bin. The values of current phase are initially loaded by the control PC with offsets compensating the de-dispersion delay gradient, calculated as explained in section (2.1.4 (a)) of chapter 2. These offsets correct the relative delay gradients only within the 32 channels of each node. The gross delay of each node's highest frequency channel is calculated relative to the first channel of the first node in units of the sampling interval as

$$N_{skip_K} = \frac{\tau_{diff_K}}{T_{frame}} \qquad 4.3$$

This is fed to respective nodes, so that each node can skip that many initial samples before beginning the process. This will align the phase of the profile between nodes, as shown in figure (4.16).

For subsequent data samples of any channel, the corresponding current phase position pointer has to be incrernented by the OP_PH_INC value and stored back in OP_PH_TAB. The pointer to OP_PH_TAB

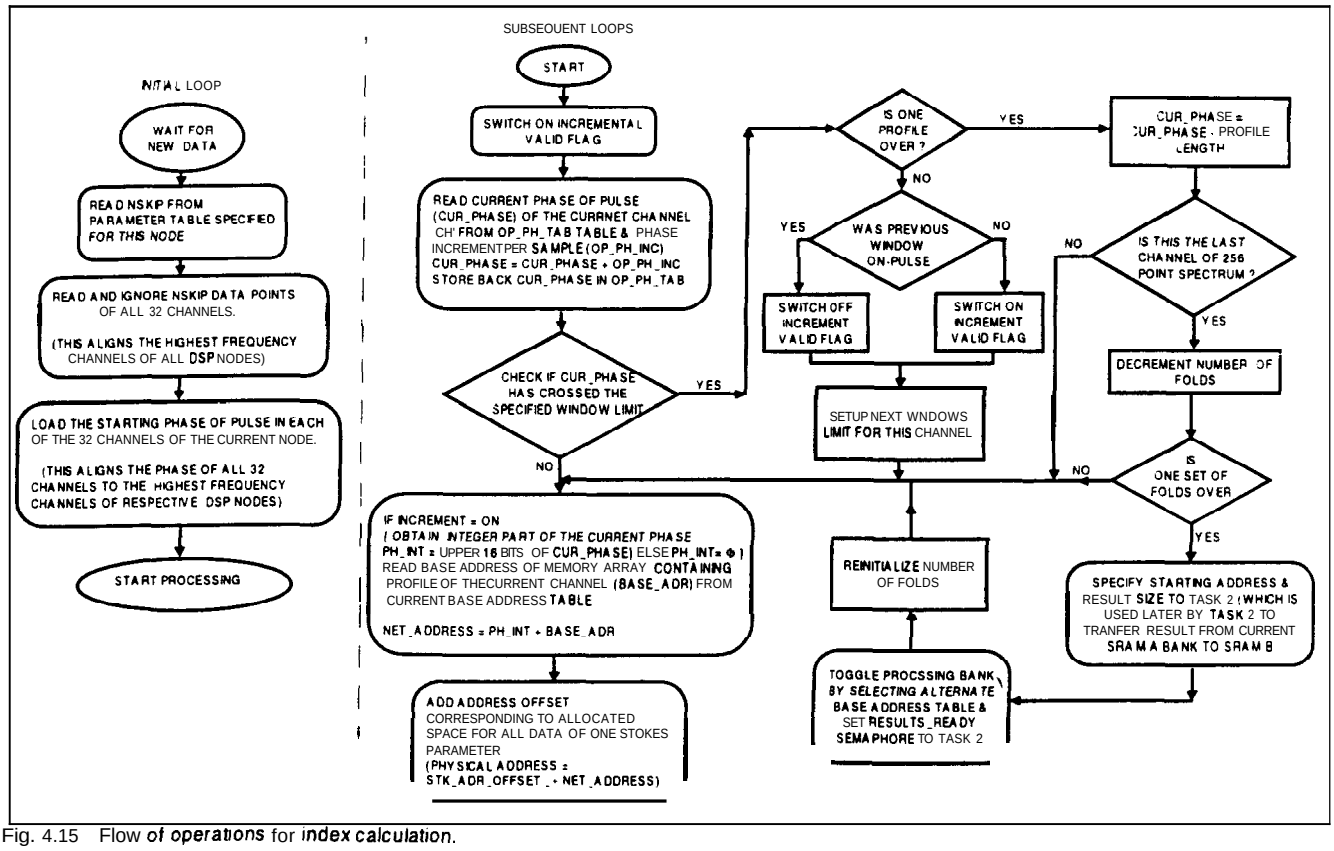


Fig. 4.15 Flow of operations for index calculation.

automatically moves to next channel after writing back to this table, but does not change when the value was read out earlier (this is achieved by using two different modify registers, one initialized to zero and one with unity, while accessing the table). As mentioned earlier, every channel has a designated base address in the base address table, from where the profile of that channel has to start in SRAM-A. During normal operation the base address is added to the integer part of the phase value (figure (4.17)), to get the actual location in the time-frequency matrix, which corresponds to the data point obtained from the input.

However, if the pulse gating mode

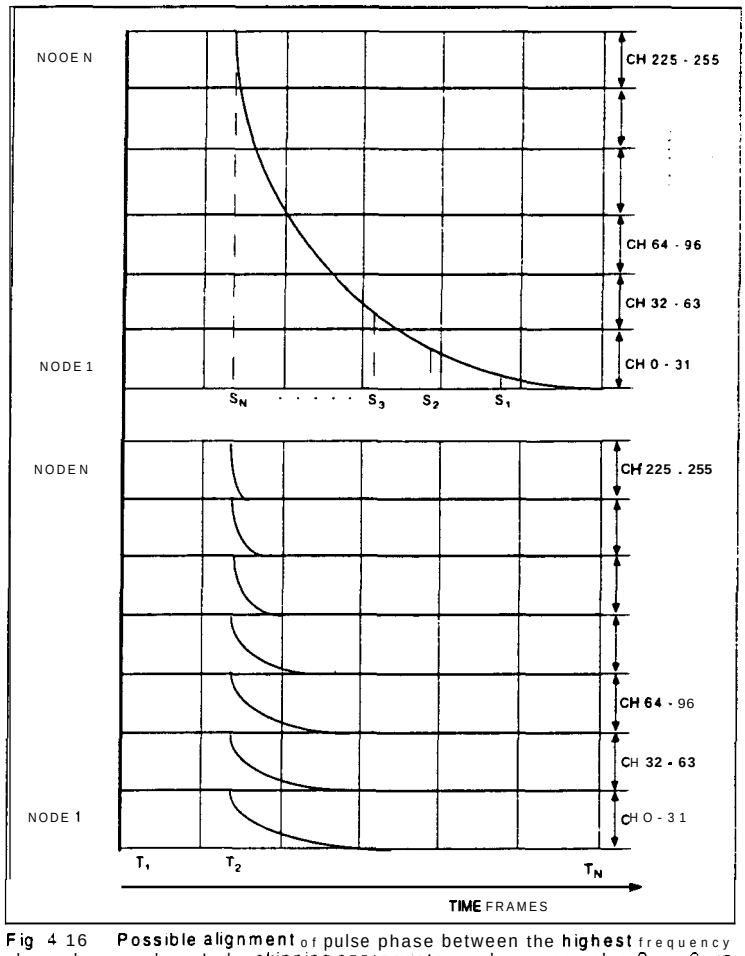


Fig 4 16 Possible alignment of pulse phase between the highest frequency channels of each node by skipping appropriate number of samples S_1 ... S_n in respective nodes

of operation is chosen then the integer part of the phase value is forced to zero just before adding to the base address whenever the phase of the pulse falls outside the specified on-pulse window. This will result in addition of any bin falling in the off-pulse region to be added to a fixed solitary memory location, thereby saving the available memory to host more on-pulse samples. This zeroing operation is performed after the current output phase is incremented by OP_PH_INC and stored back in OP_PH_TAB, so that the progression of phase continues unaltered. The base address is a 32-bit absolute address of the memory and the base address pointer locates the old value corresponding to the incoming channel. The base address pointer auto-increments on every read so as to point to the base address of the next channel. The allocation of SRAM memory space for the **4** Stokes parameters is shown in figure **(4.18).** As can be seen, once the absolute address of the data point of one Stokes parameter is computed as mentioned above, the locations of corresponding data of other Stokes parameters are at equal, constant offsets. The offset corresponding to the current Stokes parameter, being processed is added to the absolute address so as to get the final physical address of the current Stokes parameter in SRAM – A.

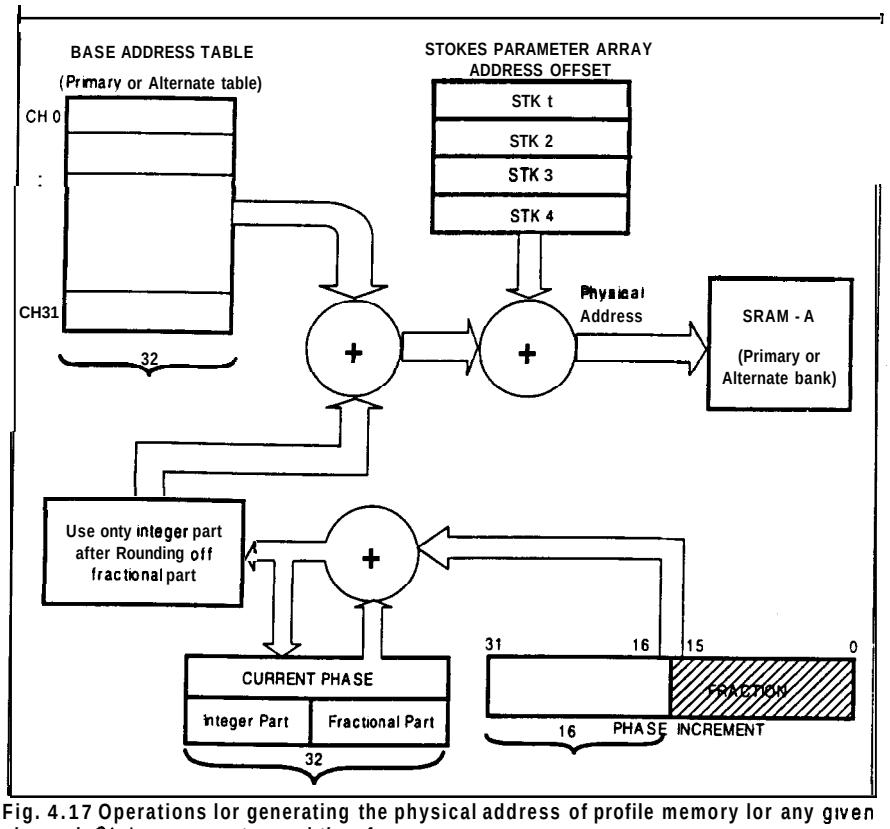


Fig. 4.17 Operations lor generating the physical address of profile memory lor any given channel, Stokes parameter and time frame

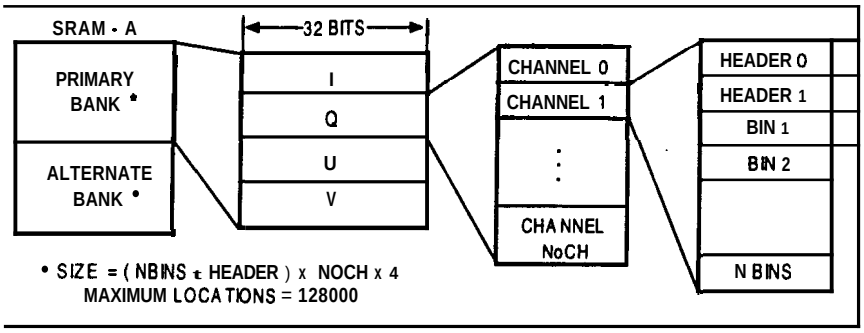


Fig. 4.18 Allocation of SRAM for primary and alternate banks of Stokes Parameters, frequency channels and profiles.

The data of parameters I and V are simply added to the profile data from the SRAM-A locations, but parameters U and Q (which represent the linear polarization component), are to be multiplied with a phase factor for Faraday de-rotation. The real and imaginary coefficients of phase factor Φ are read out from the FAR_COR_TAB which is a circular buffer memory allocated for sequentially storing phase rotation values of

different channels. The complex number (Q+jU) is then multiplied by $\left(\Phi_R + j\,\Phi_I\right)$. The real and imaginary parts of the product are then stored back into SRAM-A , at Q and U result locations respectively.

After writing the results of the additions back to SRAM-A, the current position pointer is compared to a value equal to the phase specified as the limit to the current window (on-pulse or of-pulse window). If the phase has crossed this limit, then it is checked whether the phase had completed one period. If not, a flag is toggled to indicate the switch over of the window from on-pulse to off-pulse region (or vice versa) so as to allow or zero down the integer phase value during the respective regions as explained above. On the other hand, if the phase pointer has completed a period then the phase pointer is wrapped around by subtracting Nbins from the current value. Also, in such an event, a fold counter is decremented whenever the processing of one period worth of data of any channel is completed. The fold counter is compared with zero every time it is decremented to see if the prescribed number of folds are not yet reached and then the process loops back to the above mentioned steps. Once the required number of folds are completed, the banks are switched and suitable semaphores are set up for TASK-2 to start transferring the results from the present bank. TASK-1 sets up this information regarding the result size and starting location from where the results have to copied and sent to DCS unit by TASK-2. After this, TASK-1 loops back to begin a new fold in the alternate bank with fresh data. The current position values are not disturbed during this transition, so that the results of successive sets of fold results are phase - synchronous. The folding proceeds until data of one half FIFO size is read out, when the index register crosses the specified length and rolls back to the beginning address of FIFO block. An internal interrupt is automatically generated upon this event and the interrupt service routine branches to TASK-2 after clearing the interrupt. The process returns from Task 2 only when the next block of input data is ready for processing. Upon return from Task 2, Task 1 continues operation from exactly where it had branched to task 2. The registers for operations are from the primary data registers of the DSP chip for TASK-1, while the alternate set of data registers is dedicated to the execution of TASK-2.

4.3.3.4 TASK-2:

a) Pulse Phase error correction: The events of TASK-2 are shown in the flow chart of figure (4.19). The interrupt service routine branches to TASK-2 and the alternate set of data registers is selected. As mentioned earlier, the current position value is of a format where the upper 16-bits correspond to an integer bin address and the lower 16-bits to a fractional bin. This means that the position is accurate only to $\left(\frac{1}{65536}\right)$ of a bin. The phase-lag residue accumulates upon every position increment, and slowly accumulates to about half a bin. Beyond this, the error in current position would smear out the profile noticeably as the folding proceeds. Typically, several thousands of pulses may be folded together before recording. Since the error does not grow significantly within half FIFO of data, it is convenient to pre-calculate the error per half FIFO of data and the control PC feeds this number to individual DSP nodes during the setup time. Every time TASK-2 gets evoked after half a FIFO of data is processed and the error per half-FIFO is added to an error accumulator register. This value is then compared to a preset error limit-register. If the phase lag error

exceeds the 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 position, to force a new position from which the folding can proceed, so that the profile gets rotated within the Nbins span. This makes it easier to monitor, in case the epoch of observation was such that the profile is split between the beginning and end portions of the Nbins space. The rotation value is fed by the PC and can be altered on-line. The current position of some channels may have just crossed their Nbin space (and incremented the fold counter) and after subtraction the phase may go back to the previous bin. This condition is checked and for any channel the condition is true, the fold counter is decremented and the current position is adjusted to rotate back across Nbins space. During subtraction of the error, it is possible to subtract only the upper 16-bits of the fractional bin space, and a small residue may be left behind in the error accumulator.

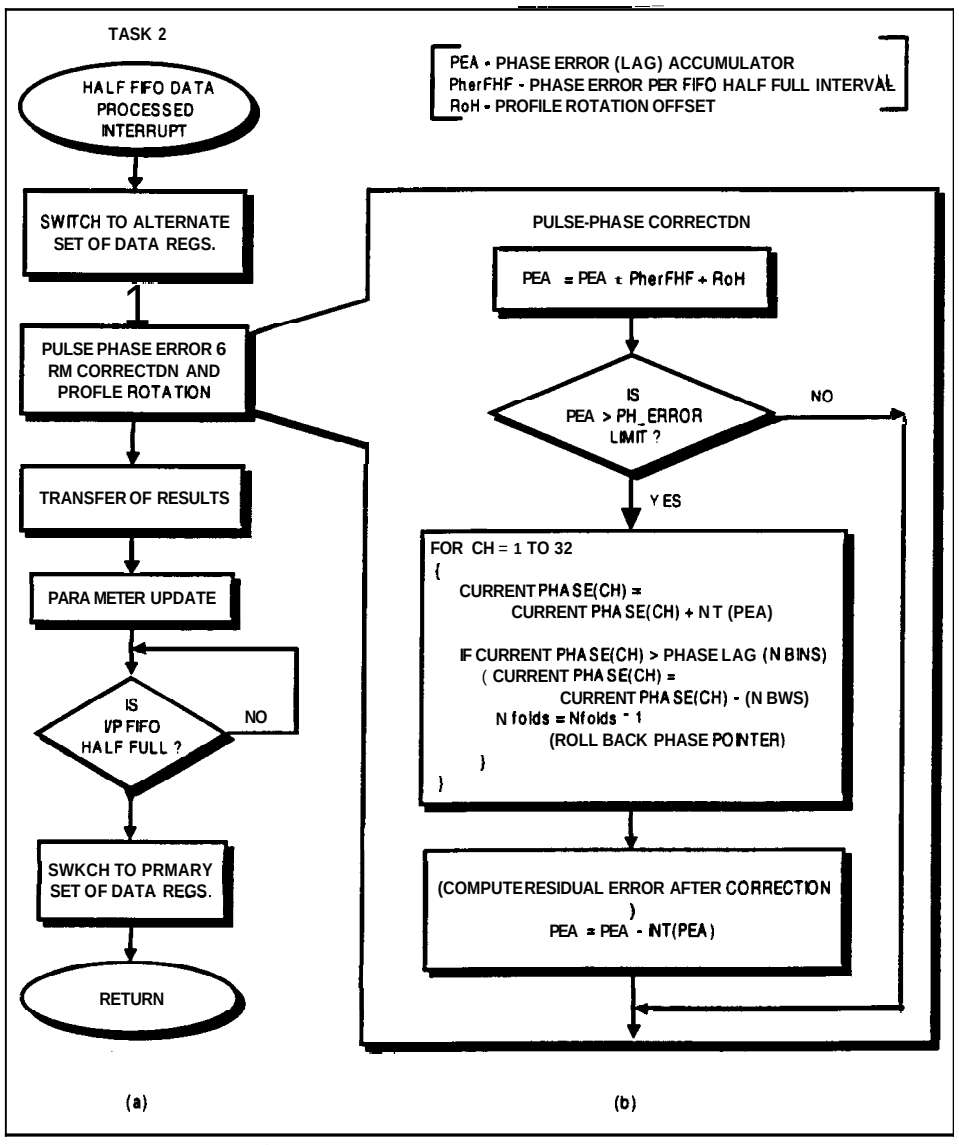


Fig . 4.19 (a) Flowchart of TASK 2 operattons, (b) Pulse Phase Error Correctton operatton.

b) Transfer of results: The sequence of operations during result transfer is shown in figure (4.20a). Each time TASK-2 executes, it reads the value of residual result size from a dedicated location in DPRAMs and checks whether it is zero. If so, it assumes that there are no results to be transferred and the transfer routine is bypassed. However, if the result-size is a non-zero number, then the DSP immediately deactivates a BUS-FREE flag, to indicate to the DCS unit that it is going to access the SRAM-B for copying results from SRAM-A, so as to avoid contention. The DSP then disables the bus-buffers of SRAM-B, disconnecting it from the bus and enables the buffers on its own side, allocating the SRAM-B to itself. It is not possible to transfer the whole of the result block within a single execution of TASK-2, and the transfers have to be in smaller blocks, such

that they are performed in the intervals between two FIFO-half-full events. After relevant timing calculations, an optimum block size is found to be about 256 locations per service of TASK-2. Therefore, the DSP finds out whether the result–size is greater than this block size and in such a case transfers one block of data, subtracts that block size from result–size and writes back to the dedicated location of DPRAM so as to use this information in the next iteration of TASK-2. If the residual size is less than one block size, it transfers the entire data available for the transfer residue and clears the "result" locations in DPRAM to zero. Only in this case, the DSP buffers are disconnected and the bus-side buffer enabled, and the BUS-FREE flag is activated, to indicate to the DCS that the results are ready to be collected. Thereafter, the SRAM-B is free for use by the DCS for the entire interval till the next set of results gets ready. The ratio of processing time to transfer time is high enough to maintain this protocol without any data loss. As the data gets transferred from SRAM-A to SRAM-B, the old locations of SRAM-A are filled with zeros, so that when this bank gets used for a fresh use (folding) by TASK-1, no residual values are left behind which may contaminate the next set of results.

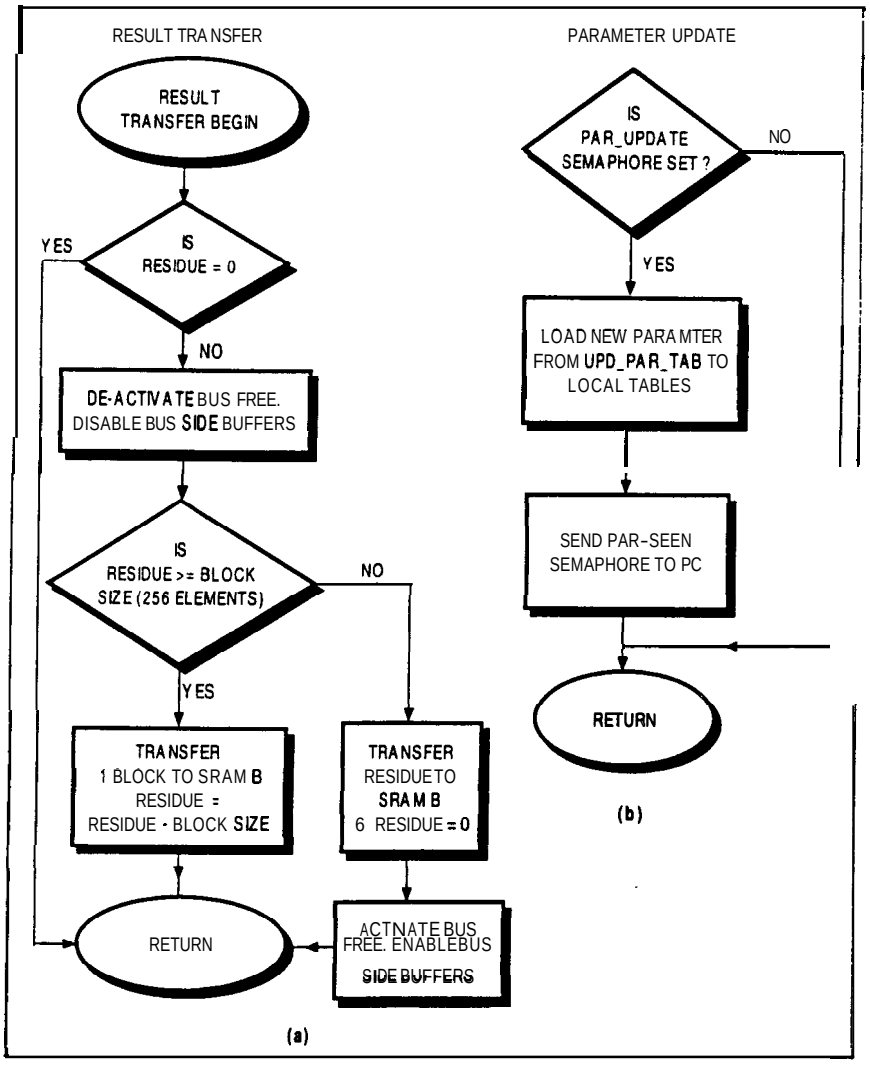


Fig : 4.20 (a) Sequence of operations in transfer of results. (b) Flow of operations for parameter update.

c) Parameter update: Figure (4.20b) shows the sequence of operations during a parameter update cycle. During the initial part of an observation, one may also need to rotate the profiles within their Nbins space iteratively, to arrive at a comfortable profile-layout. Also, some effects such as parallactic angle & Faraday rotation, Doppler acceleration change with time. The parameters used by the DSP will change correspondingly and need to be updated suitably once in a while. The control PC handles the jobs of computing the new parameters and the finding out the time at which the update is needed. At such a time, the PC uses a dedicated set of DPRAM locations called UP–PARJAB and loads the updated parameters into this table. The PC then sets up a semaphore at a dedicated location of the DPRAM indicating that new

parameters have been down loaded. This semaphore generates an interrupt to the DSP. The DSP may be in the middle of processing data, and may not have time to load the new values immediately. It suffices to simply take note of the indication that there are new values and load them later in the next TASK-2 interval. So the interrupt service routine corresponding to the DPRAM interrupt sets up a local flag in a register to indicate the availability of the new parameters and returns from the interrupt service to continue with the processing. During TASK-2, this registered flag is checked and if new parameters are available, they are transferred from the UPD-PAR-TAB table to corresponding tables used routinely by the DSP during processing. The layout of UPD-PAR-TAB table is shown in figure (4.13c). 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 two tasks, the DSP then 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 had left upon receiving its internal interrupt.

4.3.4. Exploiting Parallelism in Operations:

In each DSP node, while the data is being read out of the FIFOs and processed (say, folded), the polarimeter would keep filling the data into the FIFOs. To avoid eventual overflow of the FIFO, the DSP has to process every data point at a rate faster than the fill-in rate. The optimizations in the software routine and the hardware architecture explained above formed the basis for selecting 8 DSP nodes to handle one sub-band. This permitted a data of 8Msamples per second (including 4 Stokes parameters and 32 frequency channels) in each node. The DSP runs at 25 MHz, executing one instruction every 40ns. The data arrives once every 128ns. **This means that on the average only about 3 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 exploiting all architectural advantages of the processor chip. The architectural advantages exploited in the signal processing code are listed below:

1. Zero overhead branching and looping
2. Address based interrupt generation
3. Intelligent caching to use effectively the 3 bus architecture
4. Parallel multiply, add and data access instructions
5. Single cycle task switching
6. Circular buffer addressing with auto increment
7. Rolled coding for core loop
8. External hardware interrupt service for I/O handshake with DCS

The part of the algorithm which consumes maximum time is the core loop. This portion has been implemented completely in parallel code. For fastest execution, it is required that all external memories associated should run with access times ≤30 ns. If address decoding is done externally, the additional

propagation delay of the decoder imposes even lesser access time (–25 to 20ns) for reliable operation. To avoid this, the on-chip decode lines have been used, which get activated automatically within a range of addresses pre-programmed into the chip at the boot time. These lines can therefore be used directly as chip selects for memory chips, alleviating the need for external decoders, thereby relaxing the memory speed and, even more so, the associated cost. However, the DSP chip provides only two decoded select lines on PM side and four on the DM side, so the layout of the memory modules had to be carefully arranged to balance the requirements of processing speed and features of the chip. With the architecture explained above it is possible to achieve the best parallelization in the instruction code, and the memory access for data and code. However, the FIFOs do not posses a chip select line and require that the read/write lines toggle between every successive access. The on chip decode lines of the DSP remain low (active) and do not toggle if consecutive address accessed fall within the same range specified for the respective decode lines. To ensure that the chip-select toggles, the core loop parallel instructions have been written in such a way that after every FIFO access, the next instruction has an access to the DPRAM for some parameter. The chip-select lines for the DPRAM & FIFO have mutually exclusive ranges of address. Even though this optimization is used presently, for the general case, an option has been provided to use a fast (5ns) decoder with chipselect and read lines, so as to derive a decoded read-pulse as an alternative for FIFO read lines. The processor has some restrictions in register allocations in the parallelized code. Hence the register usage and allocation is a non-trivial problem and this configuration was arrived at after many iterative simulations. Even though the manufacturer provides a C language compiler for the DSP chip programs, it does not effectively parallelize the code to suit our needs and hence generates a slower code for execution. Also, the assembly instructions are fairly high level and easy to use, so the entire DSP routine was coded in assembly language. The parallelized core algorithm is displayed below:

In this implementation, the inner-most 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 interrupted by internal interrupt, and the net time saved is given by

$$X = HALF\_FIFO\_SIZE * 15ns; \qquad 4.4$$

This time must be sufficient to execute the outer loops of TASK-1 and all operations of TASK-2. The number of cycles consumed by these instructions is outlined in table(4.6). The table displays the estimates of the computational load for folding, de-dispersion, Faraday correction, adjacent sample integration, channel integration, Doppler correction of a 1 millisecond period pulsar.

Considering the time needed for these computations, a set of 4 FIFOs are chosen to form a 32-bit memory of 32K locations. Then, 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. The above routine accounts for all operations except pulse gating. To include pulse gating, two operations are added in each outer loopI. This overhead is well within the limits posed by the available "free" time.

Table(4.6)

TASK1: Core loop	11 x 4096 = 45056 cycles per half FIFO
Outer loop1 (after 1 profile is over) overheads	3 x 32 ch x 2 periods = 192 cycles per half FIFO
Outer loop2 (after Nfolds are over) overheads	8 x 2 period = 16 cycles per half FIFO
TASK1: Total cycles required per FIFO half-full	45264 cycles
TASK2: Error correction	602 cycles
Transfer of results (block of 256 locations)	813 cycles
Parameter update	235 cycles

4.3.5. Data Collection system (DCS):

Two modes of data acquisition are required to receive the results from all the DSP nodes of each sub-band. In one mode, the results flow at a reasonably slow rate (upto 64 KBytes per second), and are acquired locally by the control PC, for typical observations and diagnostic runs. In the second mode, the results may be generated at a high-speed (upto 8 MBytes per sec) and are needed to be acquired continuously at this rate. To handle these two modes, a local PC-based DCS card has been developed to read out data from the nodes at a programmable speed. It delivers the results in parallel to the PC for lower speed acquisition, and to a separate connector, where a separate high-speed DCS (S2 -Canadian recorder is planned to be used) can be interfaced to acquire the data during high-speed transfers. The data acquisition process is shown in the flow diagram of figure (4.21).

Each node processes its data in parallel as explained earlier and makes the resultant data available in its own SRAM-B, for collection of results by a central DCS. Two hand-shake signals, namely BUS–FREE and ENGAGED, are dedicated to each node for communication between the DCS and the nodes. Whenever the DSP needs its SRAM-B, it deactivates its BUS–FREE line, indicating that the DCS cannot access its SRAM-B. Whenever the DCS needs to collect data from SRAM-B of any node, it first polls the BUS-FREE flag and upon finding it active, it activates 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-data-bus". The DCS 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 DCS starts polling the BUS-FREE line of the next node and keeps waiting till that flag gets set. after which it acquires data from that node in a similar way as explained above. This process of acquisition goes on, and suspends only when all nodes deactivate their BUS-FREE lines. After each 32-bit result gets latched in the DCS , it is split into two halves of 16-bits each and is loaded into a local pair of 32KByte FIFOs. The FIFOs are memory mapped onto the PC via ISA bus. They generate flags to the PC to show whether they are empty, half-full or full. The PC monitors these flags once in a while and initiates a transfer to the hard disk from the FIFOs whenever the half-full flags are set.

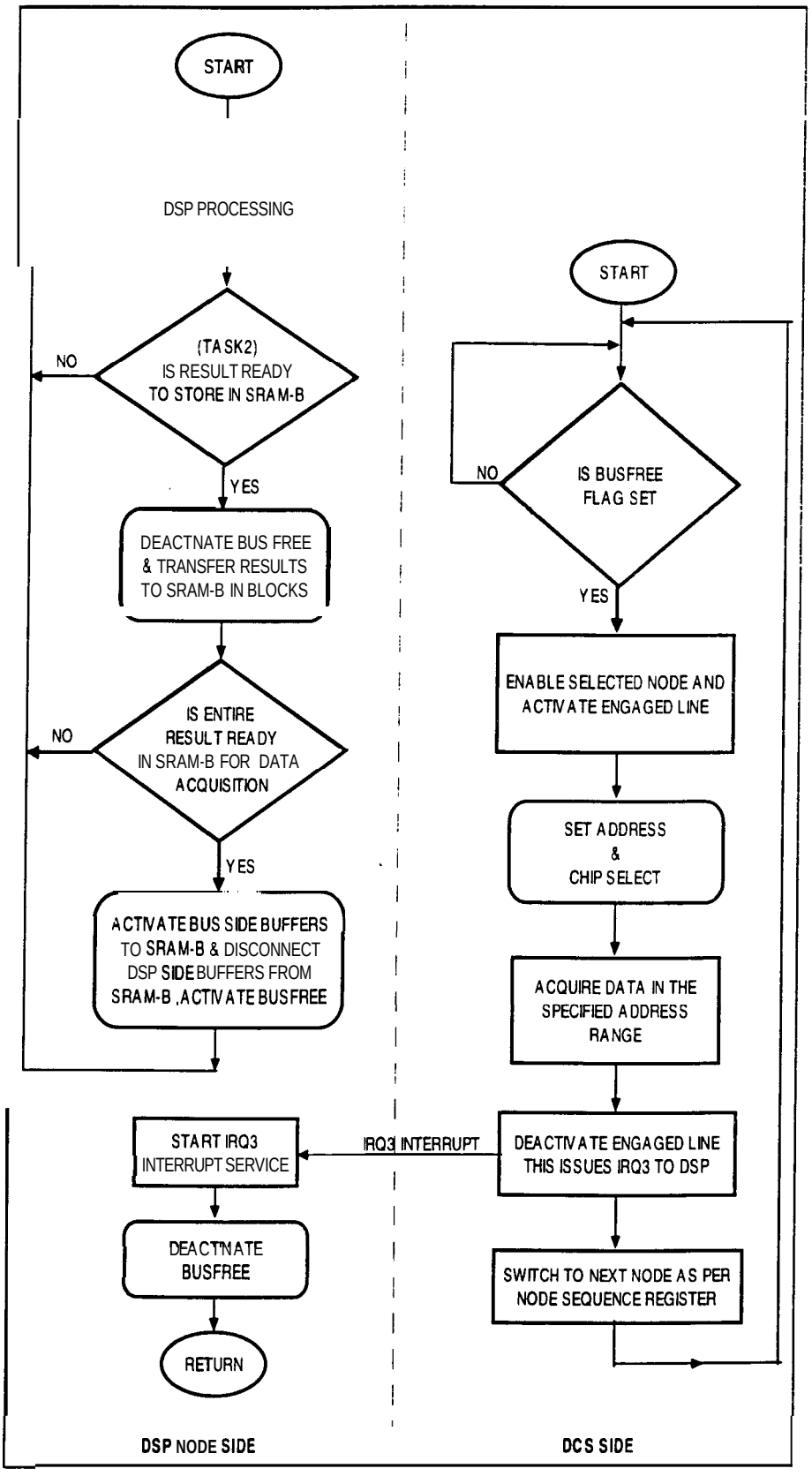


Fig. 4 2 1 The data acquisition process by DCS.

Note that the data acquisition is but one of the tasks of the main control routine that runs on the control PC. The acquisition is stopped when the control PC finds that the time specified for ending the observation is reached.

Hardware Design:

The block diagram of the circuit is shown in figure (4.22). The DCS is memory mapped through the PC's ISA bus in the address region D0000 to EOOOO hex. The lower half of this space (32KBytes) is mapped to the FIFO bank from which the results filled by the DCS can be read out. In the upper half, many registers are mapped as listed in table (4.7). 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 indicate to the PC that it is capable of transfer in this mode, the decoder generates OWS and MEMCS16 signals whenever it senses a valid address within the above range. The PC can then transfer a 16-bit data word within 300 ns. 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 starting and ending address of the result block in SRAM-B (the results are expected to fall in the same address range in all nodes). To indicate the required acquisition speed, another register is used.

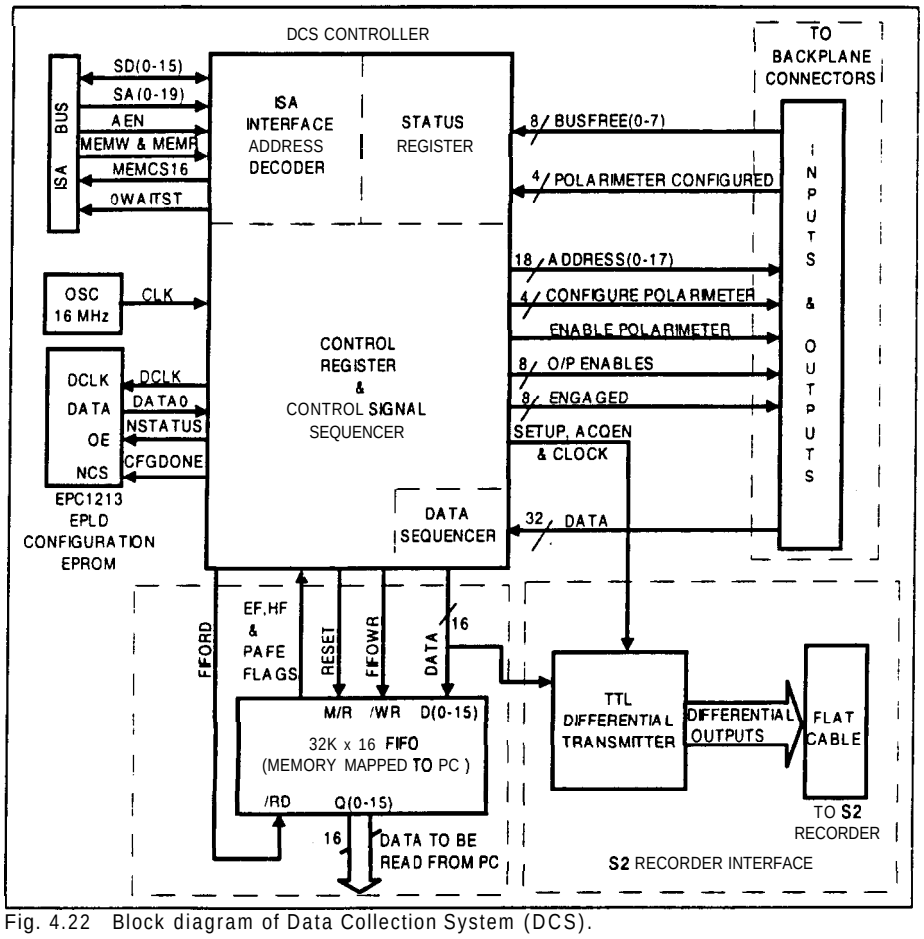


Fig. 4.22 Block diagram of Data Collection System (DCS).

Table(4.7)

MEMORY MAPPING OF DCS	
ADDRESS	ASSIGNMENT
D0000 - D7FFF	FIFO (32K x 16)
D8000 & ABOVE	Registers
D8000 & D8002	Result starting address in DSP Node
D8004 & D8006	Result ending address in DSP Node
D8008 & D800A	Node-sequence Shift Registers
D800C	Control Register
D800E	Load pulse for Address Counter
D8014	Status Register

Some times all of the eight nodes may not be used, and therefore or otherwise the nodes 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 code pattern. This code is uniquely defines every combination of number of nodes and node sequence.

The operations during acquisition are implemented in the form of a state sequencer. A 29-bit counter forms the basic control sequencer. This counter is clocked continuously by a 16 MHz clock, derived from a crystal oscillator chip. Initially, the counter is disabled from counting by a power-on reset signal generated by a simple R-C circuit. The counter can be enabled by setting the ENRUN bit in the control register (the format of the control register contents is shown in figure (4.23). Usually, the counter is enabled after setting up all

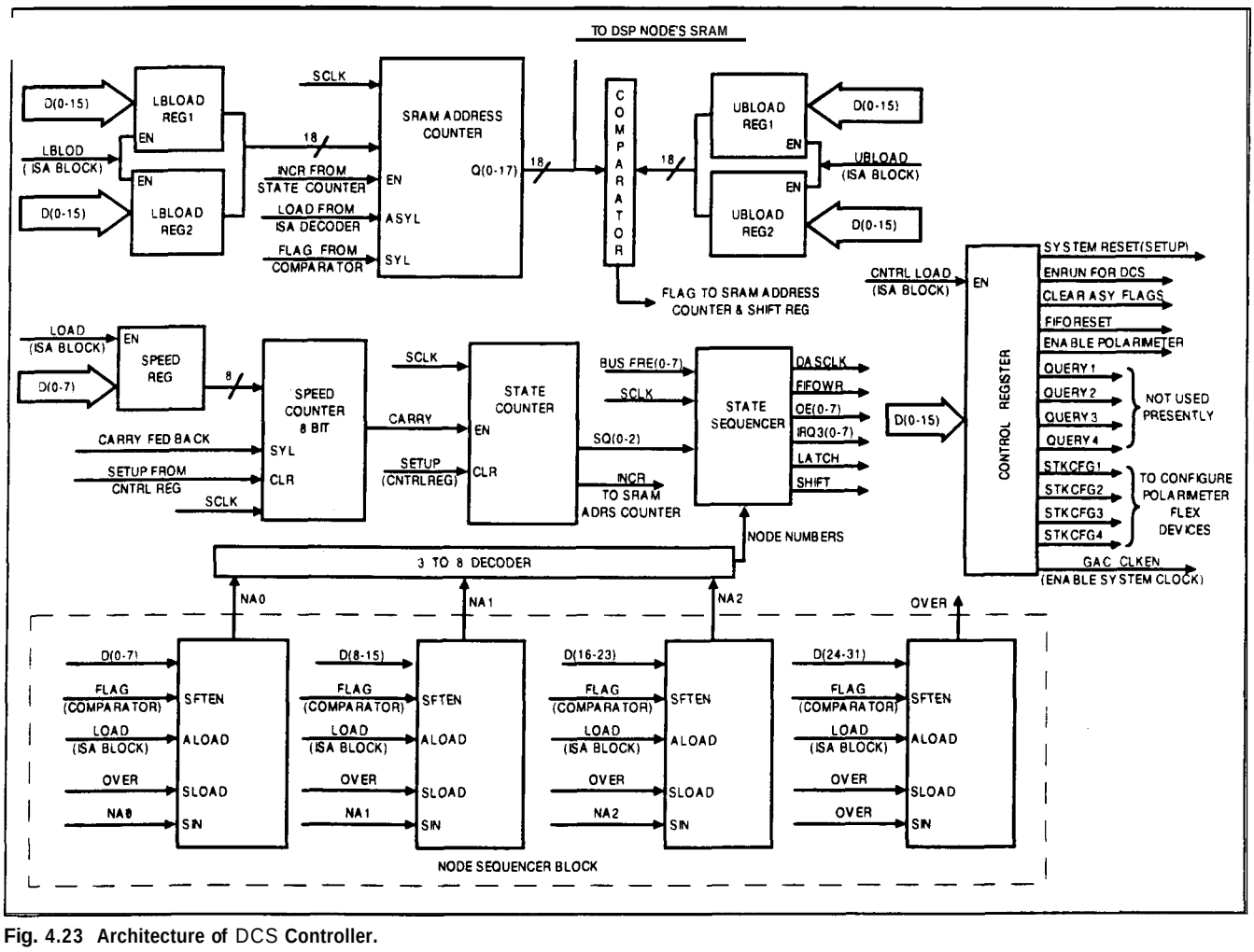


Fig. 4.23 Architecture of DCS Controller.

parameters for the DCS. The layout of the counter is shown in figure (4.23). The lower most 8-bit section of the counter can be used to slow down the sequencing at a programmable rate. This part of the counter chain (called speed counter) can be loaded with the speed select number. This section counts up, generates a carry and reloads itself from the speed select register again. Depending on the value in the speed select register, the rate of carry generation varies. The carry flag of the speed counter is used to enable counting of the next section of the counter chain, namely state counter. This portion of the counter is 3-bit wide. For every location

to be read out from any node, the eight states of this counter are decoded to produce the control signal sequence to complete reading out one 32-bit location, splitting it and loading it into two 16-bit locations of its local FIFO bank. Depending on the pre-set value of the speed counter, the sequencer reads one number from a node within 512ns to 128μs.

The control signal sequence is as shown in figure (4.24). During the first state, the output enable

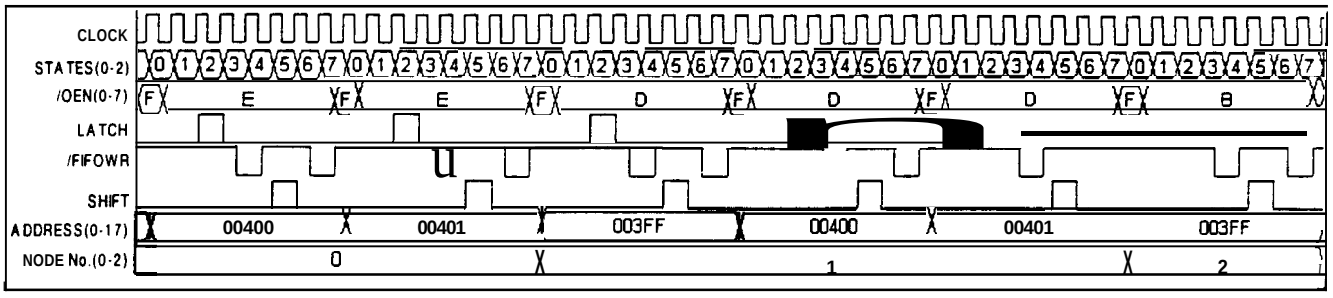


Fig. 4.24 Control signal sequence for operation of DCS.

signal and address are issued allowing the result of a selected location to flow from the selected node onto the result bus. During the next state the 32-bit result is loaded into a shift register. The lower 16-bits of this register are mapped onto the inputs of FIFOs. During the third state a write pulse is generated for the FIFOs to record the lower 16-bits. During the fourth state a shift pulse is issued to the register to push the upper 16-bits to the lower 16-bit frame, so that it appears as inputs to the FIFOs. In state 5, a write pulse is issued to the FIFOs again, for recording the portion corresponding to the upper word of the result. The 16-bit FIFO inputs and the associated write pulse are brought out in parallel through an ECL differential link to a connector, which can be linked to a remote DCS for recording data at speeds higher than the PC's capability. In state 6, the bus and the FIFO signals are deactivated. In the seventh state, the BUS-FREE signal of the selected node is polled with a suitable gating logic before starting the next transfer. If the selected node has not issued a BUS-FREE, the bus-side signals are deactivated and the state sequence pauses, till the BUS-FREE is issued by the node. If the BUS–FREE is sensed, then a carry is generated to the next stage of the counter chain, so as to enable one increment on the next clock. For this reason, the node sequence of data collection has to be in the order of result generation, since each node skips a given number of input samples to compensate for inter-node dispersive delay, for example the node handling the highest frequency channel delivers its results earlier then the other nodes. Thus, the node order has to be correspondingly chosen for DCS. Also, non-working nodes have to be excluded in selecting the node sequence, otherwise the DCS will hang waiting for response.

The next stage in the counter chain is an 18-bit address counter block. This portion of the counter is loaded initially with the starting address. Every time the state sequencer generates a carry, this address increments to the next location on a given node. This address is sent out on the bus during the first state of a transfer cycle, as mentioned earlier. The address is also supplied to a comparator which checks whether the current address has reached the last location of the specified range in the UBLOAD register. If so, it generates a flag which allows the address generator to self-load the starting address in LBLOAD register, so as to begin transferring a block of results from a similar address range from the next node. This flag also

enables one increment of the next section which is a 3-bit Node sequencer. This node sequencer goes through a maximum of eight node addresses and then the entire transfer cycle of all nodes starts all over again. At a given time, a lesser number of nodes (<8) may be used for processing, and the node order may not be always sequential. Thus the sequencer has to be capable of generating the states for effecting result transfers on the selected set of nodes in the sequence desired. To provide this feature, the a separate set of shift registers are used and are inter-connected in the form shown in figure **(4.23).** The node sequence and the number of nodes are encoded in the form discussed above and are loaded initially in to the shift register and a separate background register. (The node sequence codes are listed in table (4.8). Every time the address range specified for one node is exhausted, this node sequencer receives one "shift" pulse, pushing the next node number in the sequence, onto the state decoder. The state sequencer uses this node number to generate the control signals on the appropriate node. Depending on the code, the self-load enable flag becomes active during the next shift after the desired number of nodes are serviced. Upon receiving this flag, the node sequencer is reloaded from the alternate register and a new cycle of transfer begins starting from the first node in the preset sequence.

Table(4.8)

32 BIT DATA PATTERN	NODE SEQUENCE
00000000	0 NODES
00000000	NODE 1 ALONE
00000001	NODE 2 ALONE
01000002	NODES 1 & 2
00000100	NODE 3 ALONE
01000200	NODES 1 & 3
01000201	NODES 2 & 3
03000402	NODES 1,2 & 3
00000101	NODE 4 ALONE
01000202	NODES 1 & 4
01000203	NODES 2 & 4
03000406	NODES 1 , 2 & 4
01000302	NODES 3 & 4
03000604	NODES 1, 3 & 4
03000605	NODES 2.3 & 4
07000C0A	ALL 4 NODES

This process continues and in parallel, the flags of both FIFOs are checked for synchronism in generation of half-full, empty and full flags. If there is any asynchronism between the two FIFOs, a flag is set up to indicate this condition to the PC. The flags of both FIFOs are ganged together and brought to the PC via a status buffer. The PC can monitor these flags once in a while, and initiate a transfer to the hard-disk

whenever it finds the FIFO is half-full. The FIFO is chosen to be 32K X 16-bits, so that every hard-disk transfer cycle collects half the FIFO size, i.e. 32KBytes. The PC does other jobs besides the acquisition, resulting in a sustained data recording rate of about 64 KBytes per second. The entire logic explained above has been built into a single FLEX EPLD chip, using about 123 flip-flops and 4000 gates, in a 208 pin surface mount package. Apart from this EPLD, the rest of the components on the DCS module are the two FIFOs and the buffers to drive several data address and control lines.

4.3.6. Parallel Architecture of Nodes:

With each node having the above architecture, all the eight nodes that handle one sub-band are physically bussed together as shown in figure (4.25). All nodes are controlled by the central PC/AT, designated as the "controller". The programs for the controller are developed in C language and run under MS-DOS. There are three busses distributed to the nodes on a common backplane: a) the data input bus b) the program bus and c) the result bus. The data input bus is a separate set of data lines and an associated write pulse for every node from the polarimeter, as explained in the discussion of the polarimeter in

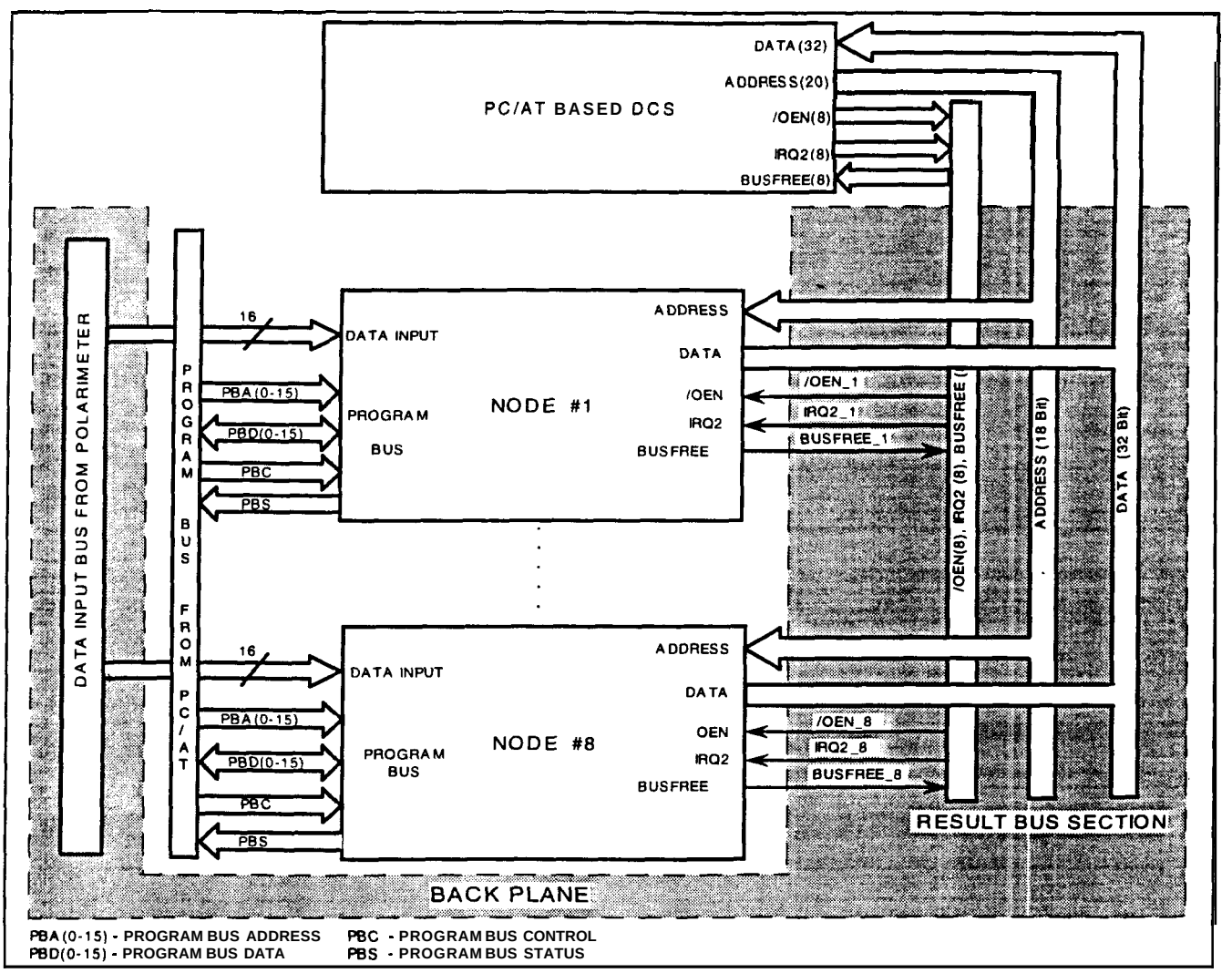


Fig. 4.25 Parallel architecture of eight nodes for one sub-band.

section(4.2.2). The program bus consists of DPRAM address, data, control and status lines. They are used to

access the DPRAM pages of individual nodes, as explained in the section covering boot-routine. The third bus, namely the Result bus, is designed to transmit results from each DSP node to the DCS. Physically, the result bus is interfaced to the DCS card, while the program bus is interfaced to the PC via a pair of 8255 PPI chips hosted on another PC-add-on I/O card. Each set of four nodes is hosted on one PCB. The DCS is designed to be a PC-ADD-ON card. Almost all devices are chosen to be surface mount packages for providing additional routing space and facilitating mounting space for ICs on both sides of the board.. The DCS card is a 6-layer PCB while the polarimeter is a two layer PCB. The DSP module is an eight layer board. The PCBs were designed with separate ground and VCC planes and orthogonal routing of interconnections in consecutive layers, and such other component placement and routing optimizations to enhance the speed performance of the boards.

Some additional signals are brought out of every DSP node, so that, if required, the nodes can be interconnected in a functionally sequential manner. Under the control of a different bus-management system and an alternative backplane, one node can communicate to another by transferring data/semaphores through its SRAM-B. The bus protocols for such a loosely coupled parallel processor architecture have been worked out, but are not particularly useful for pulsar signal processing under the present specifications. Hence, at present it suffices to only mention that with such a protocol, the same DSP nodes can be hooked together to perform as a loosely coupled, MIMD parallel signal processor. This is a useful spin-off of the present work through its details are out of the scope of this thesis.

4.4. System Software:

The control PC handles the following jobs:

Offline jobs:

a) Pre-calculation of process parameters to be supplied to DSP during observation.
b) Communicate with the main telescope control computers over Ethernet links to set / receiver parameters, antenna parameters, etc., and get time information for synchronizing the telescope activity with the pulsar signal processor.

Online jobs:

a) Setup/diagnosis of the polarimeter, parallel DSP processor and DCS modules.
b) Down-loading programs and parameters and real-time modification of parameters in the DSP nodes.
c) Data acquisition and recording.
d) Real-time display of average profiles.
e) Monitoring current time and starting/ending the observation at specified start/end times respectively.

4.4.1. Offline Jobs:

The control PCs of the two sub-bands are connected together through Ethernet links. This link extends to the other control computers at the telescope through an elaborate network. The basic communication method used to send/get information from the telescope control computers is the standard SOCKET COMMUNICATION mechanism. While the telescope control machines are workstations running unix, the control PCIAT is under a DOS/WINDOWS system. Socket communication system calls are built into the UNIX operating system, where as on the PC, a separate commercial package runs to provide the same protocols with similar syntax. The message is sent as a series of packets of information in the format shown. When the IP address field is sensed by the destination computer, the communication proceeds by standard handshaking of TCP/IP protocols to pass the information. The user specifies his observation details at the main telescope control computer. This computer has a ready-made routine that analyzes the nature of observation and forms templates for parameters to be sent to various subsystems of the telescope. Each template is communicated over Ethernet to its appropriate destination computer/subsystem, which setup their systems accordingly and flash back acknowledgements. After receiving this, the control computer of the SPPS acknowledges the telescope control computer and proceeds to organize the setup of the SPPS.

As a next step, the control PC has to calculate several parameters based on the information received, to feed the DSP nodes and setup various parts of the polarimeter, DSP nodes and the DCS. This routine, named as PRE_CAL, performs all the computations required for this purpose and loads the parameters in a particular order into a parameter file. The functional flow of the precal routine is shown in the flow chart of figure **(4.26).** This routine expects that the observation details received from the telescope control computer

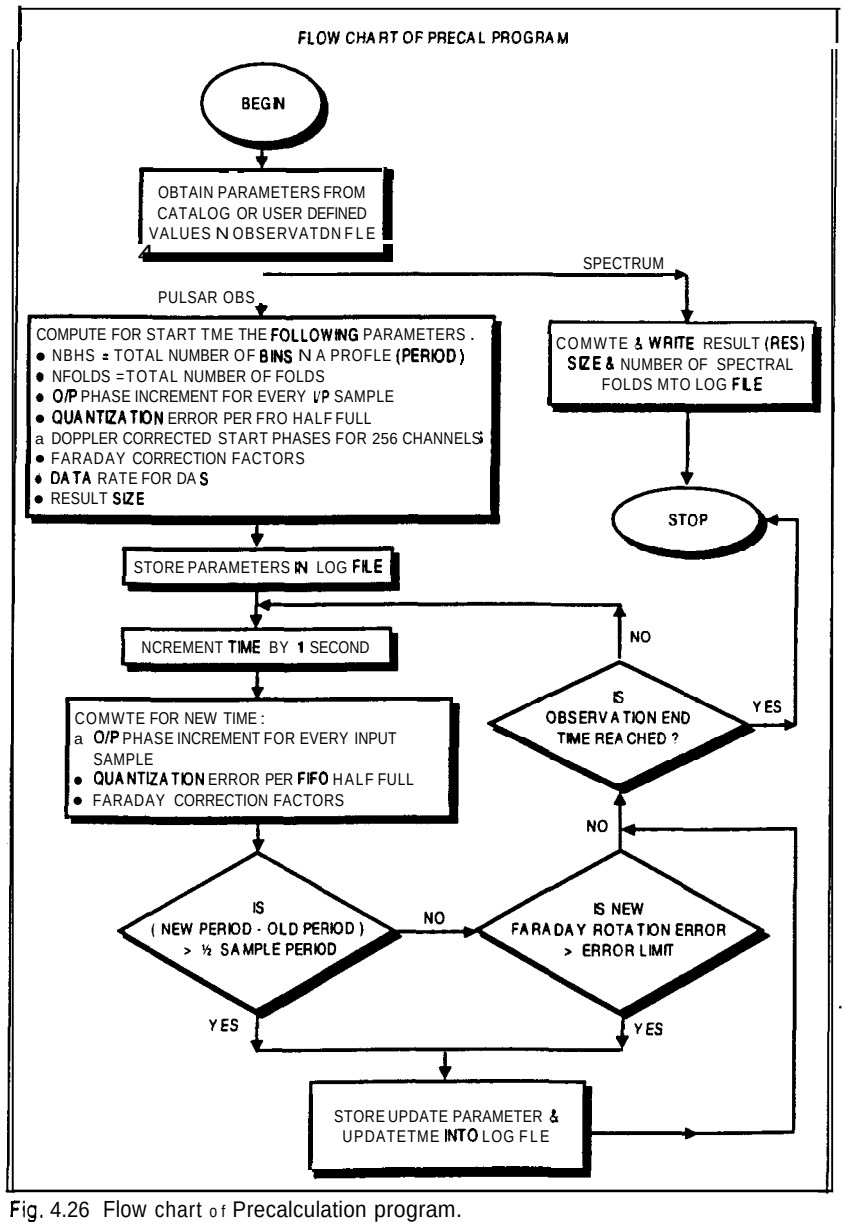


Fig. 4.26 Flow chart of Precalculation program.

are in a specific file name, in the format shown in figure (4.27). If the obs-file indicates external parameters, then the values mentioned in the observation file are forced into the corresponding parameters instead of

pulsar	//-observation
internal	//_read_from-catalog
79.6875e-9 326.0 8.0 32	//_sampling_interval,center_frequency(MHz),bandwidth(MHz). Number_of_output_channels_per_node
0740-28 1000	//_pulsar_name,Number_of_bins_per_profile
1997 12 9 8 14 30.0	//_obs-year,month,day,hour,min,secs
1997 12 9 8 26 50.0 L	//_end_year,month,day,hour,min,secs,upper_or_lower_sideband
262144	//_Number_of_spectra_to_be_integrated_before_results_ready (SPECTRAL MODE)
73.77 150.43 0.166755	//_DM,RM,PERIOD
07 42 43.71	//_RA-HOUR,MIN,SECS
256	//_No_of_time_samples_per_fft
32000	//-maximum_number_of_additions_before_results_are_stored
10	//_Number_of_profile_folds
Y	//_is_RM_Correction_required_?(Y/N)
Y	//_is_DM_Correction_required_?(Y/N)
Y	//_is_FOLDING_required_?(Y/N)
Y	//_is_PARALLACTIC_Angle_correction_required_?(Y/N)
N	//_is_GATING_required_?(Y/N)

User's Note:

1. If spectral mode is used, the last entry after band type is Nspectra and the entries after this entry are ignored.
2. If pulsar mode is used entries after band type are Nspectra,dm,rm,period, ra-hour,min,second parallactic angle. Nspectra does not carry any meaning in this context, but is just read and ignore by the program, so put any dummy entry in its field(do not delete the field!).

Fig. 4.27 Configuration file specifying observation details.

calculating them from the values based on the catalog. This helps during any diagnostic runs in forcing some values and monitor the response of the system. In the usual case, the parameters are calculated taking the epoch at the starting time of the observation as the reference. The calculations use the equations explained in section(2.1.4). In each iteration of the loop, the observation time is incremented by one second and the parameters are recalculated. Whenever the change in the parameters concerning parallactic angle and Doppler acceleration correction exceeds a preset limit, then the new parameters are taken as the reference and also copied into the parameter file along with information about the time at which the new parameters are to be updated into the DSP nodes. This process iterates in a loop until the incremented time reaches the specified end time limit. The parameter file is then closed and the program exits.

4.4.2. Online Jobs:

Once the pre-calculation is over, the control routine can be invoked. This program opens the parameter file and reads out the start time, end time and the parameters that should be set up at the beginning of the observation. It follows the steps outlined earlier in section (4.3.3.2), to setup the EPLDs in the polarimeter, and to download code and parameters into the DSP-nodes. The boot routine (which is independently run at the time of switching the machine on) should have identified and noted the identity of nodes that are working satisfactorily into a config file. The control routine reads out this information, forms a suitable code and loads it into the DCS EPLD, so as to sequence the DCS to read from only selected nodes.

It programs all the parameters of DCS and then sets the time to "enable clock to the system" and the system starts running. After this, the control routine enters a loop. Three tasks are executed in the loop. Firstly, the data acquisition subroutine is called. This routine polls for active FIFO half-full flag, and if it finds it "true", then a transfer is initiated from the DCS FIFOs to the hard disk, under the supervision of the hard disk controller. If the FIFO half-full flags are not set, then this step is skipped. As a next step, the time for the next update of parameters is compared with the current time. If the update time has been reached, then a subroutine is called to read the values from the parameter file and to update the parameters in the corresponding update tables in the DPRAM pages of all DSP nodes. If the update time has not been reached, this step is skipped. Then, it checks if the end time has been reached. If the end-time has not been reached, it loops back again to see if next set of data has filled half of the FIFO, to acquire further. Otherwise, the program exits after closing all related files. Every time a new block of data is read out from the DCS, one set of profiles in all frequency channels is summed up and displayed on the screen, for monitoring the pulse profiles online.

4.5. FFT Module:

The polarimeter is designed to work primarily with the FFT front-end of GMRT. However, during the testing phase of SPPS, the FFT system and array combiner themselves were undergoing tests and modifications and their performance was being evaluated. In the meanwhile, as a standby arrangement, a separate FFT module was developed to allow stand-alone test setup for SPPS. It has a different internal architecture as compared with the one at GMRT, but produces the same type of spectra in an identical format. Although this module serves as a standby for the current tests, it can be used independently in some other system which needs high-speed generation of Fourier spectra.

The block diagram of figure (4.28) displays the logic blocks of the FFT module. The core of the FFT module is an ASIC chip from M/S GEC-PLESSEY. This is a FFT chip with many built-in features such as internal twiddle-factor generation, programmable FFT length (16/ 64/256/1024 points) and window generation and multiplication, etc. All these features can be chosen by loading an appropriate control word into the FFT chip during a reset cycle. For our application, the configuration is frozen to handle 256 point complex transform. The chip has two separate buses for real and imaginary parts of the input data and a similar pair of buses for its spectral outputs. All data busses are 16 bit wide. The path for imaginary-part data also serves as the configuration word bus during a reset cycle. Since the input is a Real signal, the imaginary-part input is to be zeroed down when data is passed, and the same path has to contain the configuration word during reset. To simplify the this, a single bit is driven high only during reset. This bit is wired through buffers to all the configuration bits that should be high for the chosen configuration, while the rest of the bits of imaginary inputs are permanently zeroed-down. Thus during reset, the config word appears at the imaginary-part inputs, at all other times the path has zeros. Two FFT chips are used with identical setup to handle two polarization channels (even though the Plessey chip itself offers dual real transform mode, it is slower and not used here).

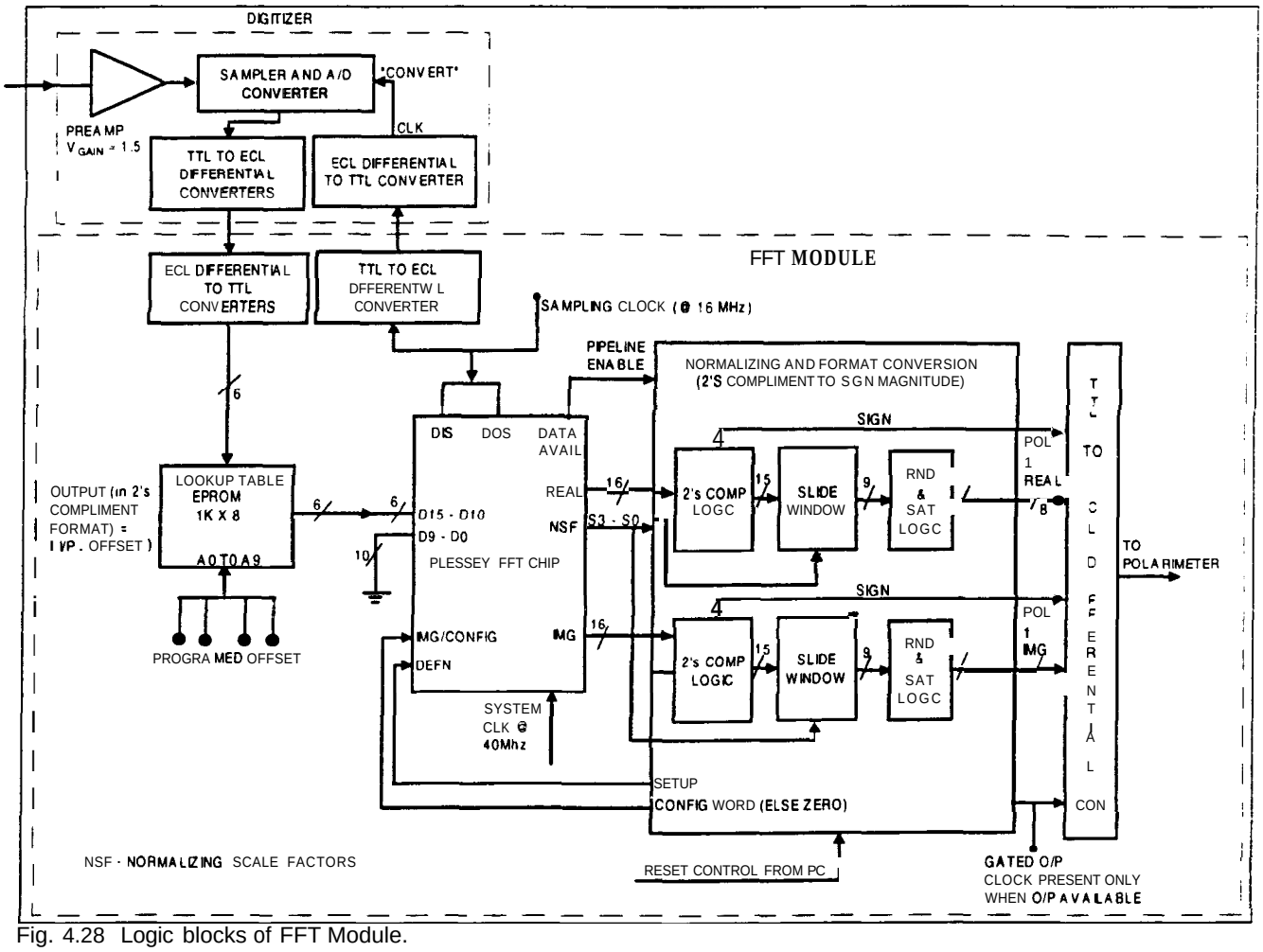


Fig. 4.28 Logic blocks of FFT Module.

The reset cycle follows the sequence as shown in figure **(4.29).** Once the reset is released, the FFT chip will be ready for accepting new inputs on every clock pulse given to the data Input strobe (DIS). The chip is pipelined internally, so that after an initial pipeline delay of 256 clocks, the output spectra will be available within the FFT chip, but will appear at the outputs in a time-multiplexed form, on every clock pulse given to the Data Output Strobe (DOS). In our application, the DIS and DOS are tied to the same clock, so that new data can be loaded while old FFT spectra is being read out. After the initial pipeline delay, a new time sample is accepted on every clock, while at the same time the next spectral channel of the previous transform is produced at its outputs simultaneously. An A/D converter is used to input digitized samples to this FFT module "in a 6-bit, offset-binary format, (with 0 representing the maximum negative voltage and 3F (hex) representing the maximum positive voltage). To make this compatible to the FFT input format (which is 16-bit, two's complement format), the samples are passed through PROM look-up tables, and then supplied to the FFT chip.

If the input signal is weak, there will be poor contrast between the signal and the quantization noise generated within the chip. Also, if there is any DC offset in the input data the spectral power of the first channel leaks to some extent into the adjacent channels during FFT computation due to finite precision, thereby corrupting the measurements in those channels. To avoid these effects, the PROMs chosen have

additional address bits which are mapped to a set of jumpers. These jumpered bits can be preset high or low, to select different tables within the PROMs. Each table is programmed to subtract a different DC bias from the

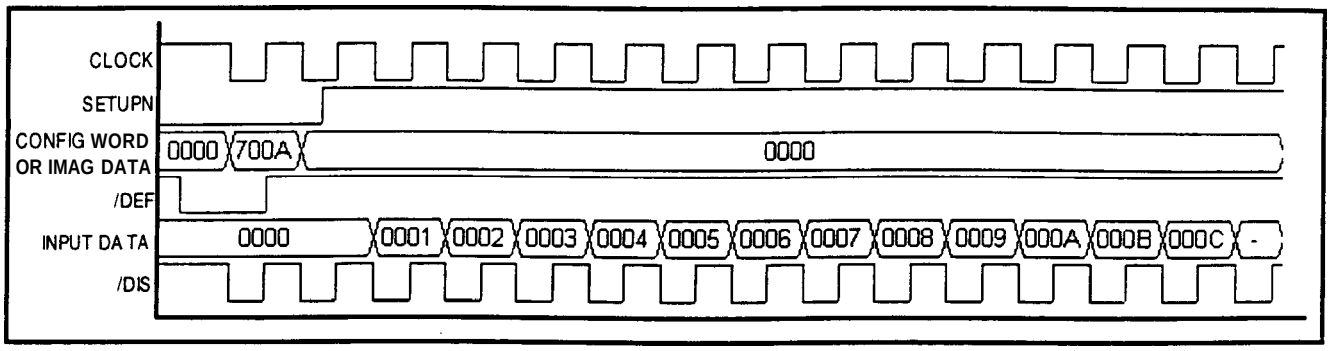


Fig. 4.29 Sequence for a Reset cycle of FFT module.

input numbers, in addition to the format conversion. If there is any inherent DC bias in the analog system, which is carried through by the A/D converter, it can be removed by selecting a suitable portion of the table. Since the input frame is only 6 bit wide, the PROM output is connected to the higher most significant bits of the FFT input bus, and the remaining lower bits are frozen to zero. By doing this, it is ensured that the inputs are scaled up as much as possible so that the noise generated due to finite precision of the twiddle-factors and the logic within the FFT chip is small compared to the input signal.

The Polarimeter accepts its inputs in nine-bit sign-magnitude format (with 1-bit sign and 8-bit magnitude). The 16-bit outputs from the FFT are to be rounded off to match this width. This reduction in result width will not affect the signal to noise ratio when the 8-bits of magnitude chosen are from a field within the 16-bit frame, such that the output signal strength is enough to fill this field but does not over flow. This is implemented by converting the two's complement format outputs of the FFT chip to a sign- magnitude format and passing the magnitude part of the numbers through a 9-bit sliding window. The window position can be chosen depending on the average strength of the analog signal. Of the nine bits of magnitude selected, the number is rounded off to 8-bits by adding the LSB to the upper 8-bit frame. This rounding off ensures that there is no dc bias in the numbers due to this step. An output strobe/clock is generated synchronous with the FFT output, so that the polarimeter can use it as a clock to latch its inputs. The clock is gated with the "data available" (DAV) flag of the FFT chip, so that the clocks flow only when valid data comes out of the FFT chip. The FFT chip configuration, output format conversion, sliding window logic, round-off logic, output strobe generation are all implemented in a single EPLD (ALTERA 7128). The data and strobe outputs of the EPLDs are passed through ECL-differential drivers and brought to connectors with a pin-out identical to the array combiner link. Since the Polarimeter accepts two polarizations, two identical modules of the above architecture are built, one to handle each polarization. The spectral purity of this system is mainly dependent on the performance of the FFT chip. At a sampling rate of 16MS/sec and FFT chip clock of 40MHz, a new spectrum is produced every 20.4μsec. Even though this is slower compared to the GMRT FFT (16.25μs), the data is essentially clocked into the polarimeter in a burst at the rate of 16 Mega-samples/sec, so that the

speed of clock in the polarimeter remains same for purposes of evaluation. Also, the DSP nodes run on their own 25MHz clock, and process the data at the same speed, once every block of data is presented to them.

4.6. Tests and Results:

Before going ahead with the hardware implementation, the entire functioning of the SPPS was simulated with appropriate test data on a clock-by-clock basis by writing equivalent software programs. Using this simulations, the algorithms were optimized for speed and code size and then translated into equivalent hardware and the DSP assembly-language software. The polarimeter was built to handle 256 frequency channels and 4 Stokes parameters (corresponding to one sub-band of GMRT). A counter based pattern generator was designed and fit into two pin-compatible EPLDs to generate a digital RAMP pattern and a pulse

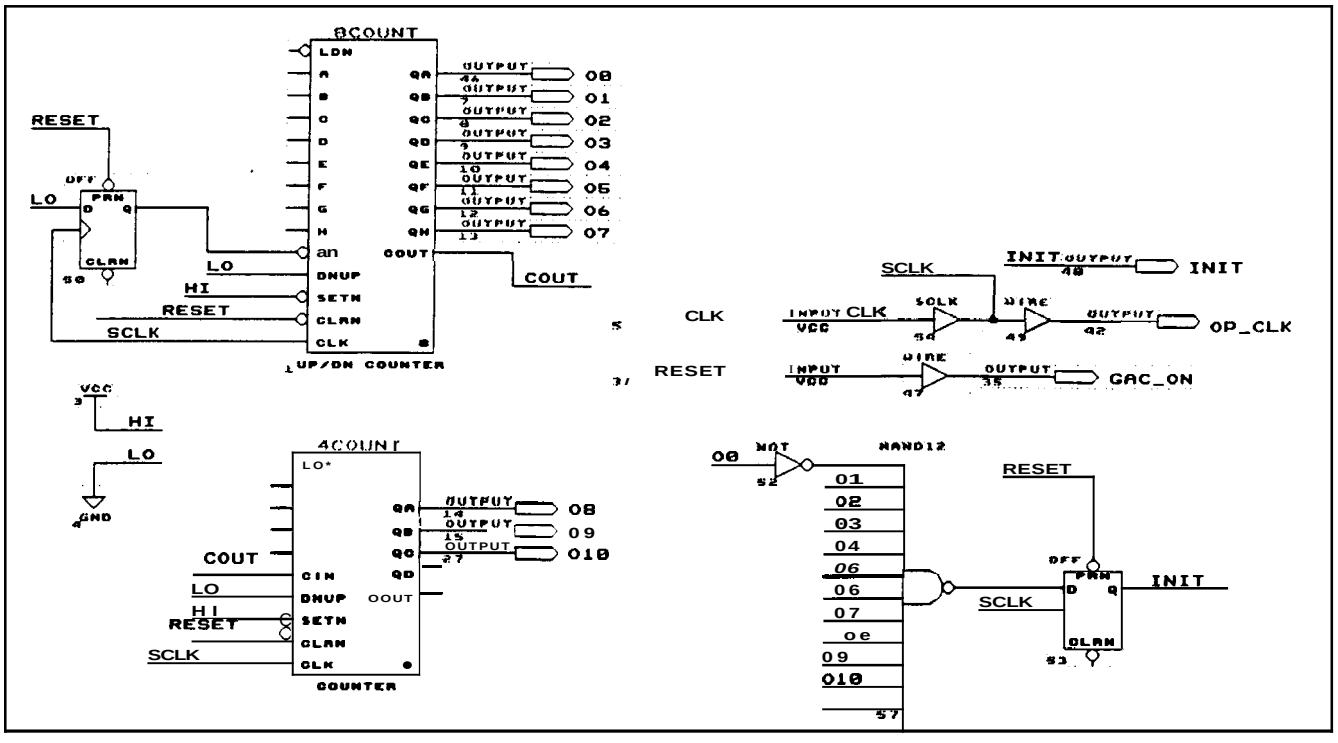


Fig 4.30 EPLD circuit for Ramp pattern Generation.

pattern as shown in figures (4.30) and (4.31) respectively. The pulse pattern is 256 sample period, pulse width of one sample, zero DM and the pattern is common to all the 256 frequency channels (corresponding to 256 consecutive clock) as shown in figure (4.32). The ramp pattern starts at value 0 and increases upto 255 and repeats as shown in figure (4.33). This will result in a unique value being repeated for every channel, on each spectra. These patterns were fed to the polarimeter at 16 Mega samples/second and the outputs of different sections of the polarimeter were monitored on a logic analyzer. Some adjustments had to be made to correct the digital signal shapes and delays of the clock's distribution to different sections of the hardware. After these adjustments, the polarimeter outputs were identical to the simulation results. The polarimeter outputs obtained for a RAMP input pattern are shown in figure (4.34) along with the inputs of individual DSP nodes of one sub-band. The PCB for 2 DSP nodes was populated and used for checking the signal processing algorithms and the communication links along with the data collection system.

Initially, a spectral averaging program was loaded into the DSP node. This program accepts

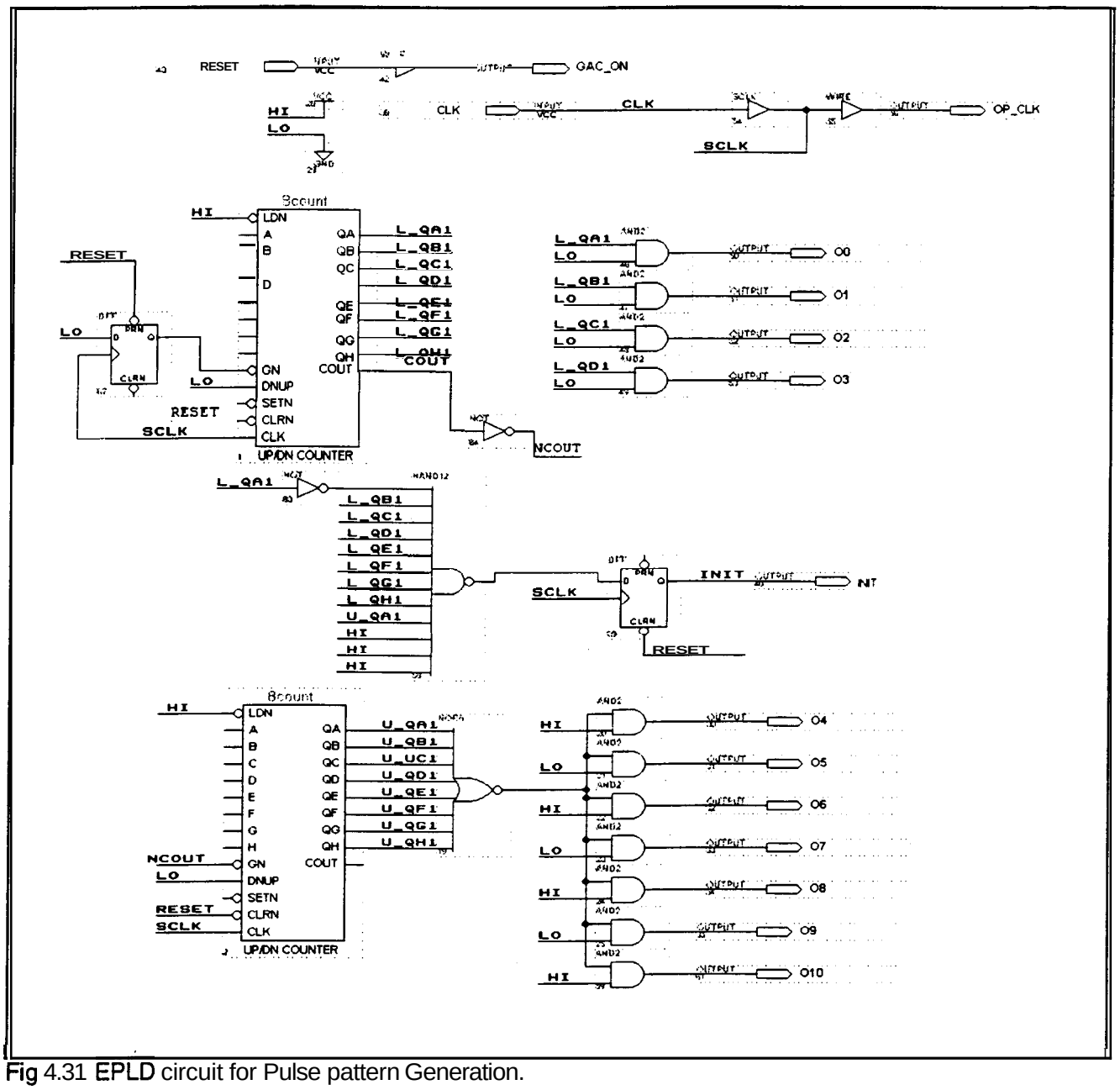


Fig 4.31 EPLD circuit for Pulse pattern Generation.

consecutive spectra of all Stokes parameters and integrates them for a programmable number of time frames and then sends it to the data collection system. By feeding the RAMP pattern to the polarimeter inputs the average spectra of all Stokes parameter was obtained for upto 262144 folds. The averages matched the original pattern of an individual spectrum, thus indicating the system operation is as desired. Figure (4.35) shows the corresponding average spectra.

The FFT module was interfaced to the polarimeter along with the A/D converter and a sinusoid signal of a known frequency was injected and 262144 spectra were averaged. The experiment was repeated for different carrier frequencies. Figure (4.36) shows these results.

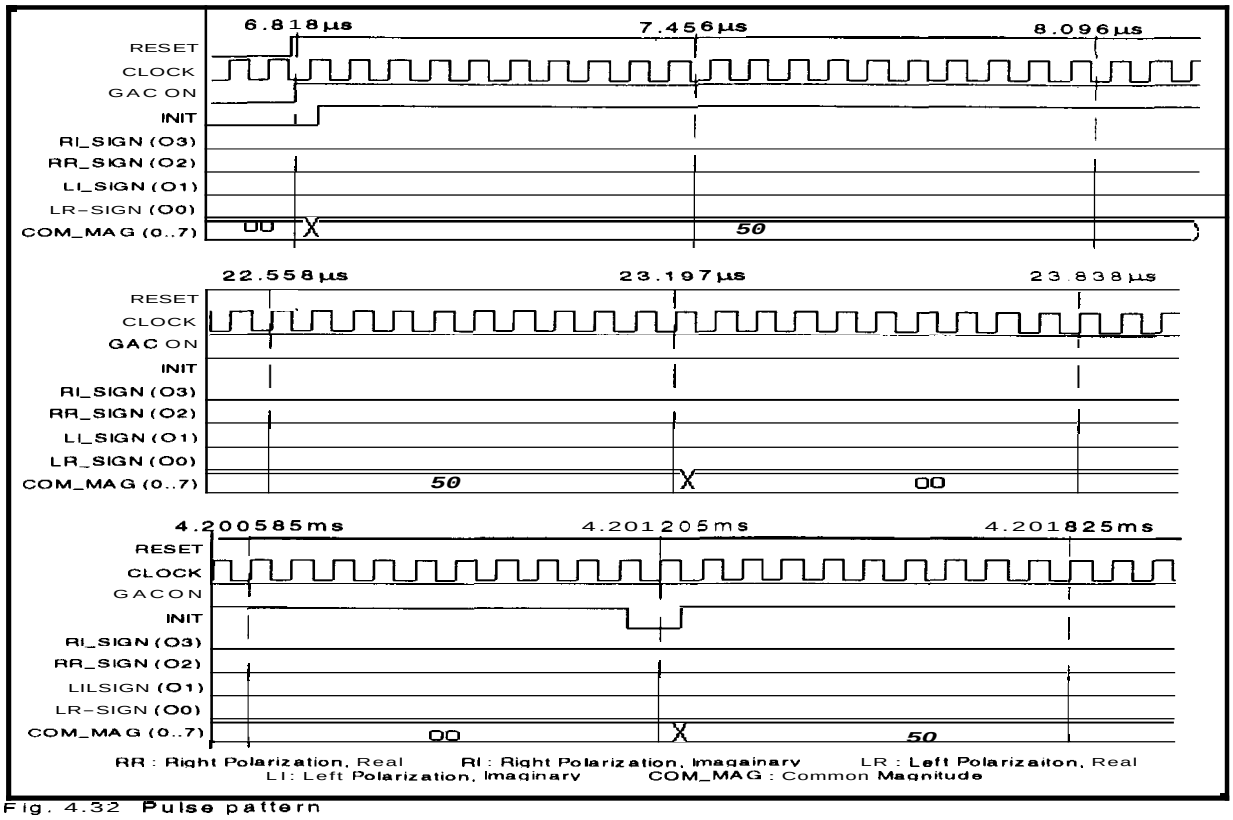


Fig. 4.32 Pulse pattern

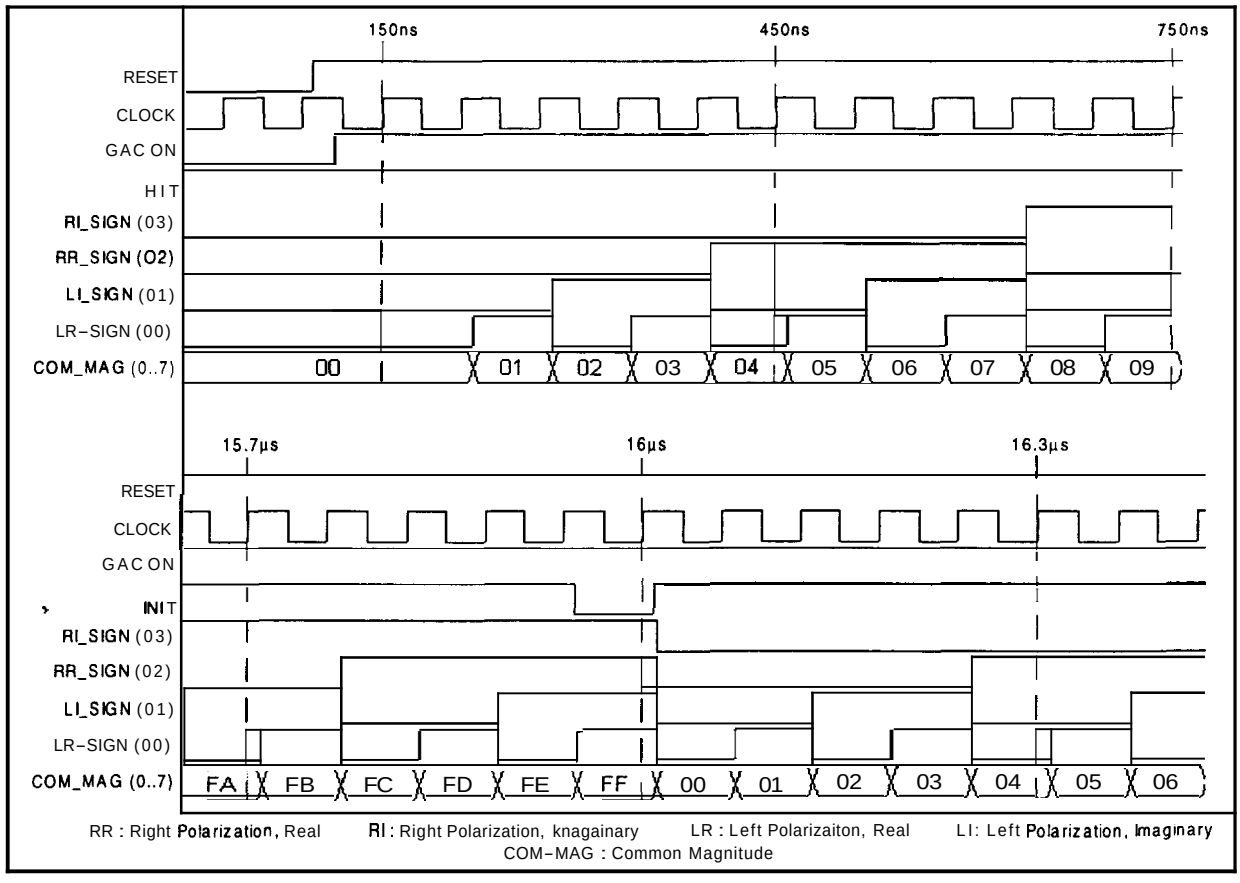


Fig. 4.33 Ramp pattern

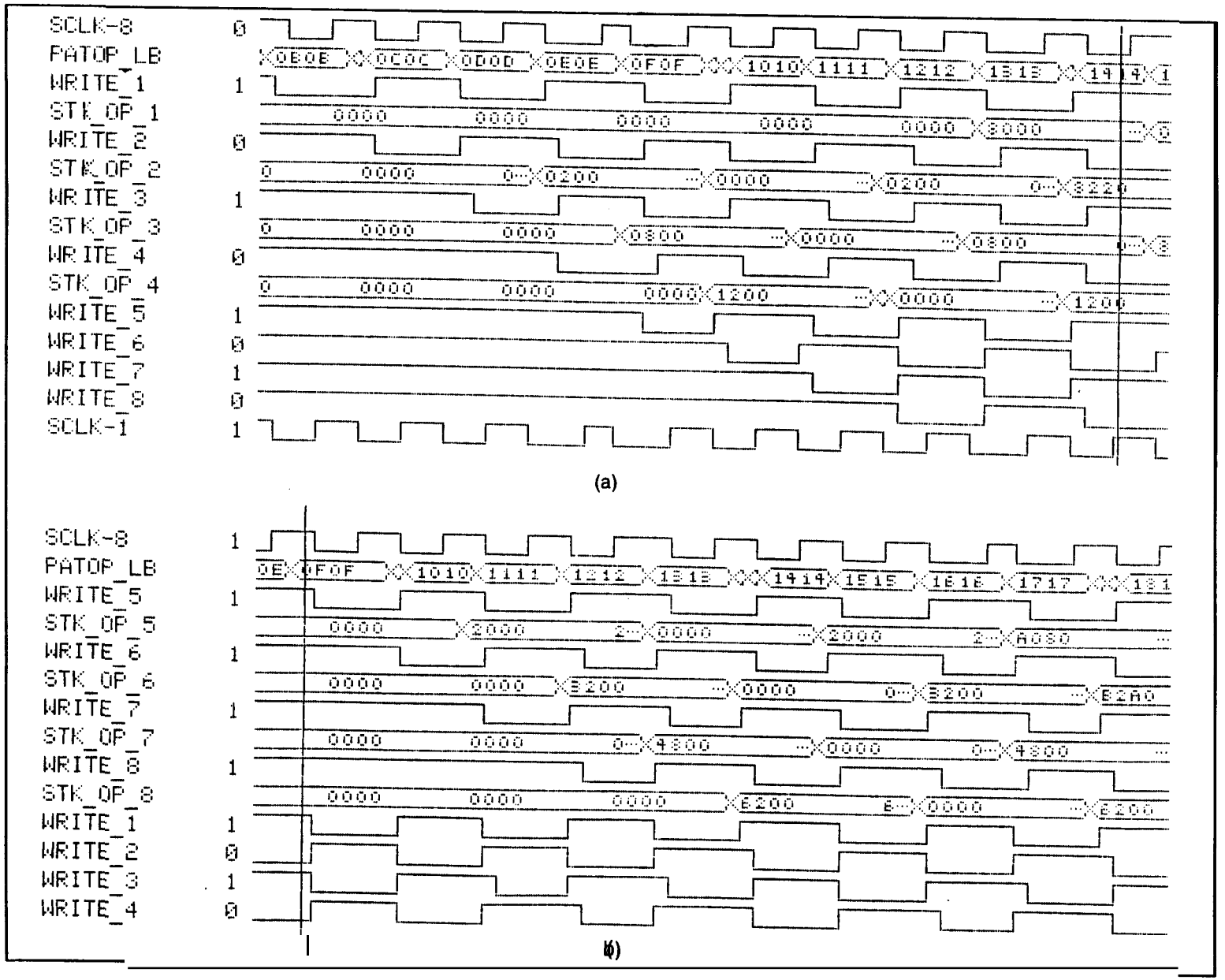


Fig. 4.34 Polarimeter outputs for 8 DSP nodes obtained for a ramp input pattern at 15 mhz clock. (a) Outputs for Nodes 1 to 4. (b) Outputs for Nodes 5 to 8. PATOP_LB is the input pattern sequence to the Polarimeter. STK_OP_[1..8] are the data, WRITE[1.8]] are strobe pulses to latch the data into respective DSP nodes

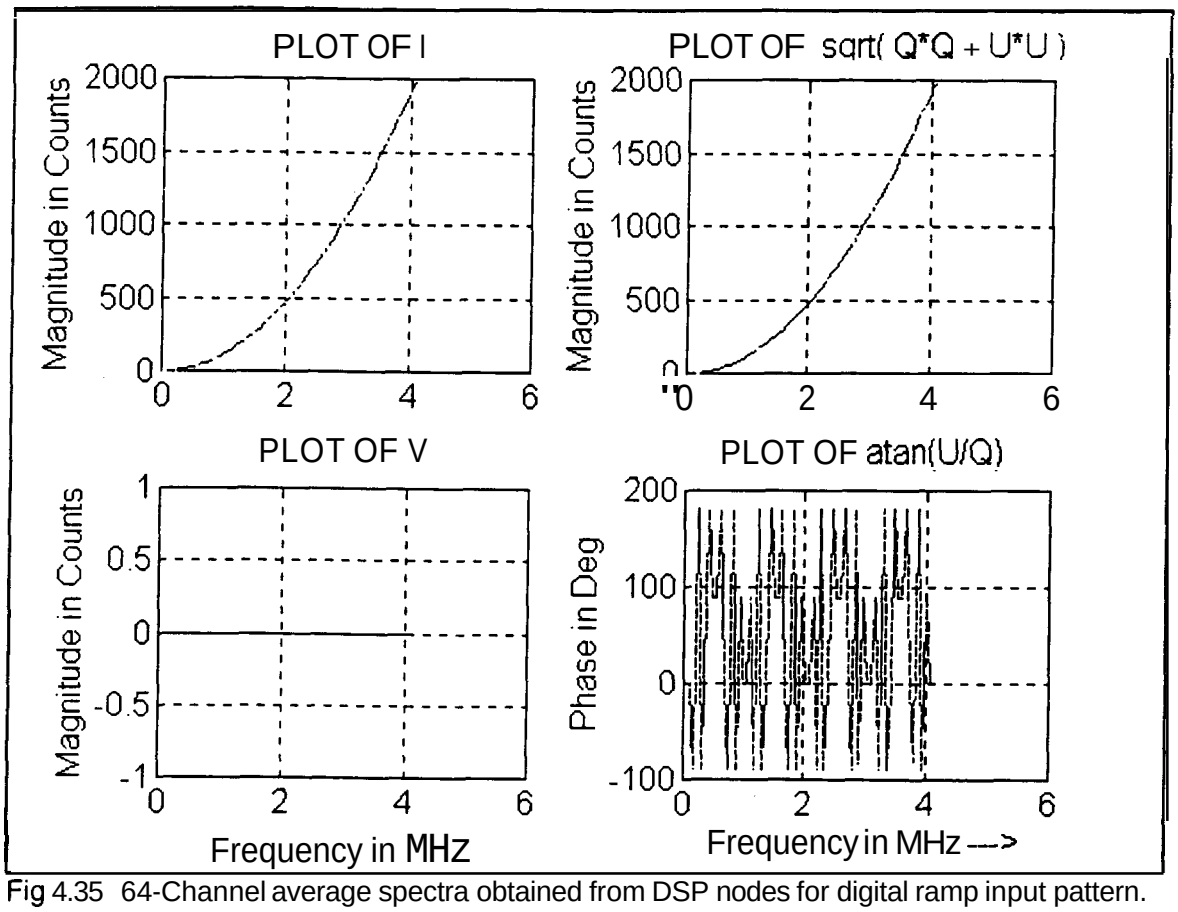


Fig 4.35 64-Channel average spectra obtained from DSP nodes for digital ramp input pattern.

After the "spectral average" tests gave satisfactory results, the two DSP nodes were loaded with the pulsar signal processing program. The FFT block was removed and the Digital pulse pattern generator was connected directly to the polarimeter input. The DSP pre-computation routine was used with DM, RM set to zero and the period specified as 256 time frames. The configuration and the signal processing part were down loaded to the DSP nodes from the control PC and DSP nodes were run to fold the pulses for 16384 periods. The folded profile was found to match the simulation exactly.

As a next step, different RM and DM values were specified to the system even though the input pattern does not simulate any of these effects. The system would then blindly correct the data for the indicated effects (by RM, DM) during the processing, assuming the presence of these effects and the resultant profile would reflect the corrections put in. The dispersion correction of such data would show up an opposite delay gradient in the pulse profile across the band. Figure (4.37a,b,c) show the results with DM=0, 2 & 4 respectively. Faraday De-rotation shows up as an opposite handed rotation of linear polarisation band across the band. Figure (4.38) shows the results of tests with RM of 0, 600 respectively.

The machine was moved to the field station at the Ooty Radio telescope to conduct further tests with real pulsar signals (The GMRT is undergoing calibrations, modifications and stability tests at present and it is better used a proven telescope to check the performance of this machine). The test setup at ORT is shown in figure (4.39). The signals from the entire array are split into two parts corresponding to the North and the South halves of the array respectively. Each signal was passed through an independent IF receiver system and the corresponding base-band voltages were fed to a pair of sampler/A-D converters. The digitized outputs from each sampler was fed to a FFT module to produce 256 point complex spectrum. The FFT

spectra corresponding to the North and South halves of the array were fed to the polarimeter as though they corresponded to two polarizations of the telescope (ORT is a single polarization telescope). For these further tests, the instrument was connected to the FFT system and the sampler inputs were derived from the ORT base-band receivers. The sampling interval was set at 16megasample/sec. The FFT was fed with a 40MHz

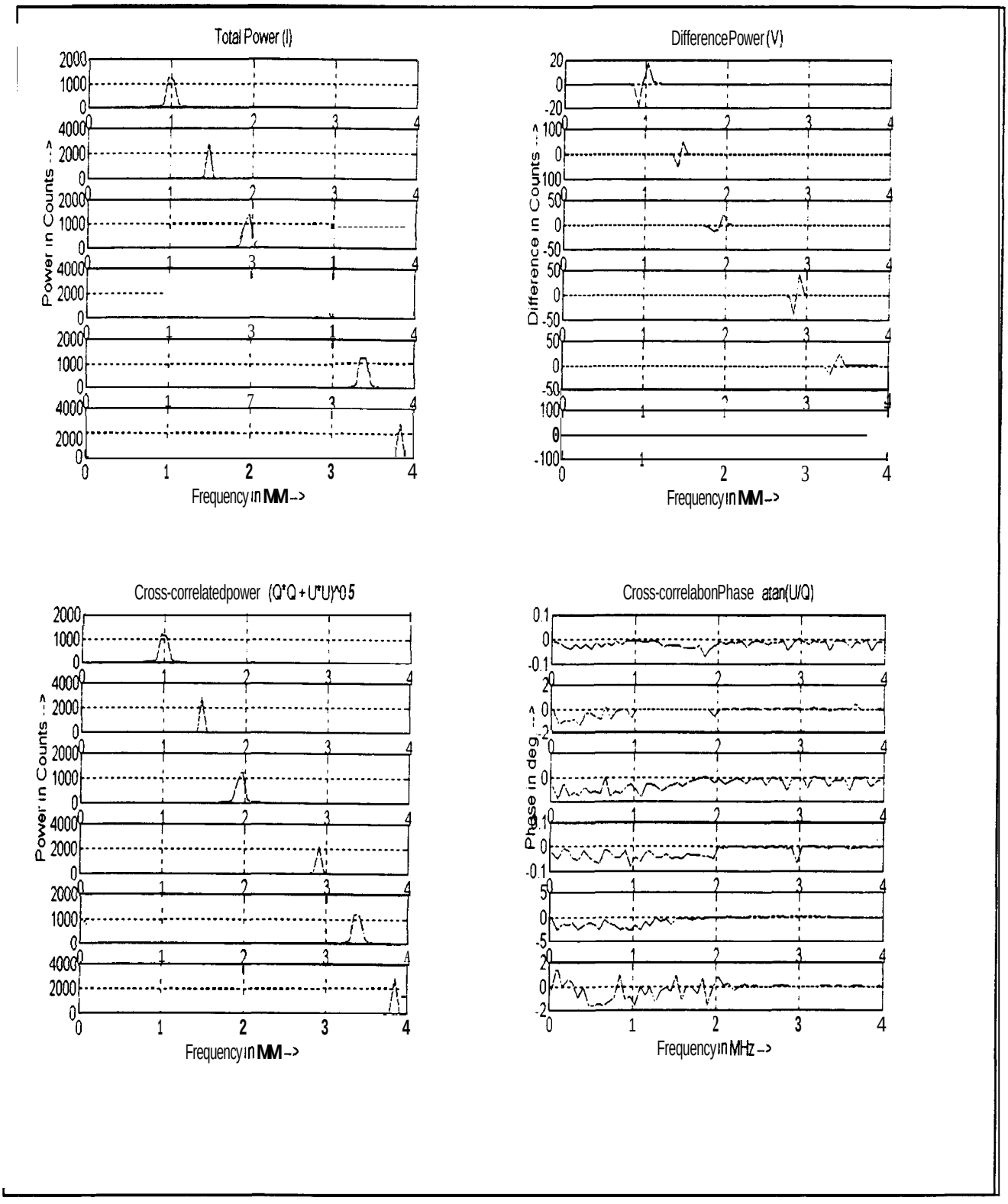


Fig 4.36. 64-channel average spectra obtained for sinusoidal analog input signal at different frequencies.

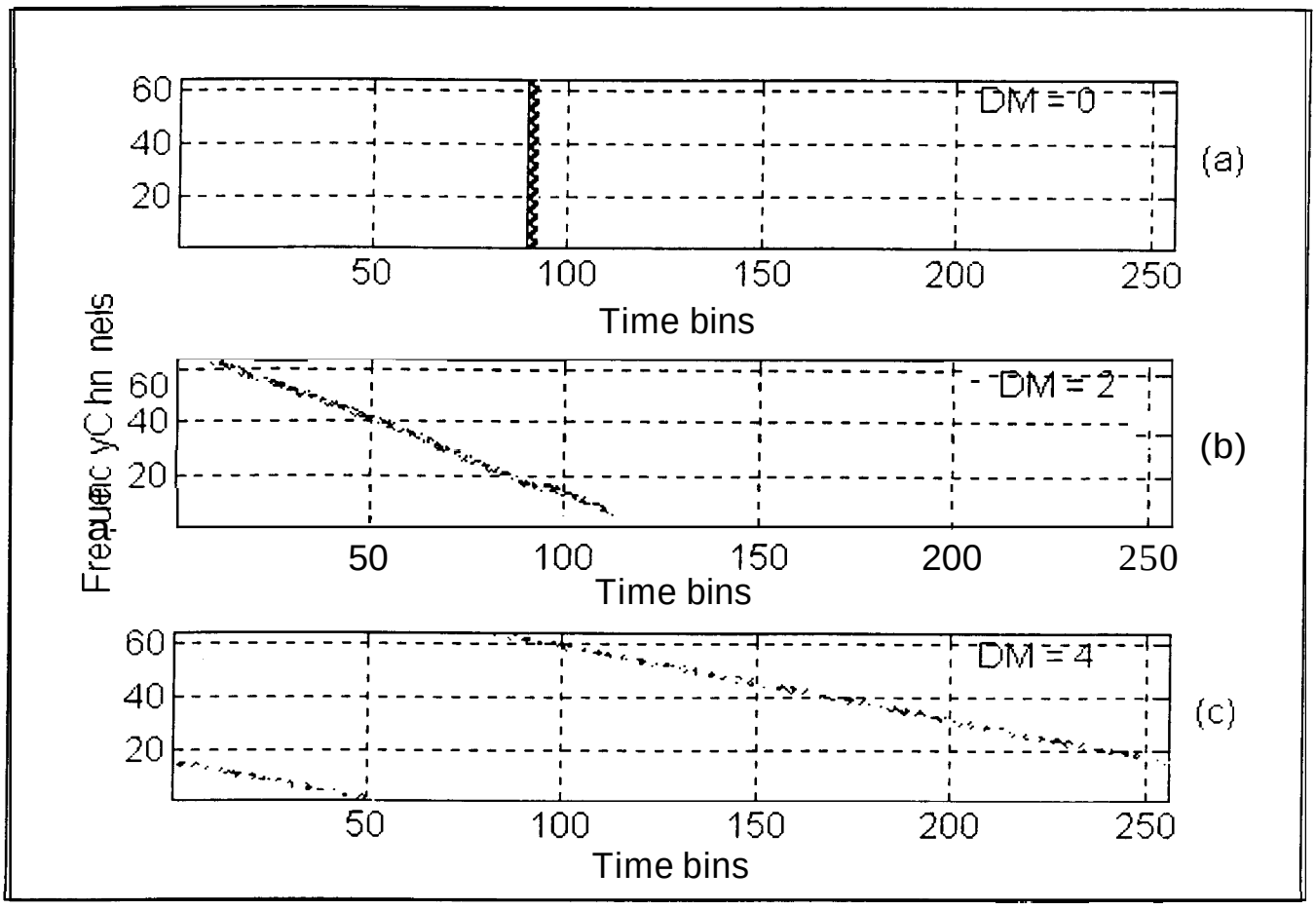


Fig. 4.37 Contour plot of total intensity as a function of time and frequency for digital pulse Input pattern. Process parameters : pulse period = 256 samples. number of frequency channels = 64. number of folds = 1000, input DM = 0 , Corrected for DM = 0, 2 and 4 in trials a, b and c respectively.

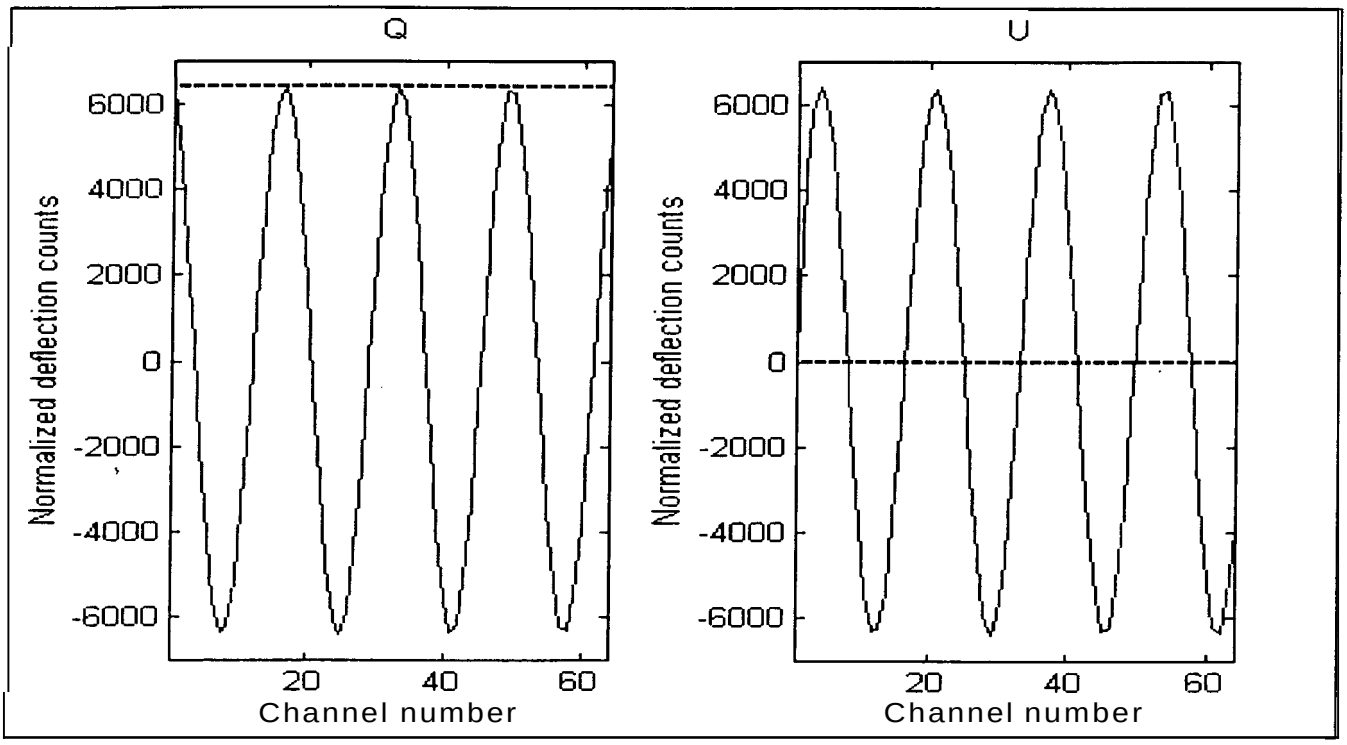


Fig. 4.38 Plot of Q & U Stokes parameters as a function of frequency for digital pulse input pattern. Process parameters : pulse period = 256 samples, number of frequency channels = 64. number of folds = 1000, input DM = 0, RM = 0 , Corrected for RM = 0 (dash line) & 600 (solid line).

clock so as to produce consecutive 256 point complex spectra every 20.4 microsecs. With this setup, the FFT rejects at its inputs, about 5µsec worth of data once every 20.4µsec, since it takes this time to compute the FFT. However, the FFT results are read out at 16 Megasample/sec to the polarimeter. The local oscillator signal that is used in the ORT IF receiver system was amplitude modulated, so that, depending on the local oscillator power, the noise power at the input of the sampler gets modulated. The DSP nodes were loaded with the pulse folding program and the pulse period was set to twice the period of LO modulation. The DSP nodes fold the data over this period for a given number of folds in each spectral channel. The corresponding folded results were checked to see if the profiles smeared as the number of folds were increased. Since the clock period was known only to a small accuracy initially, substantial smearing took place. An estimate of the smearing was used to arrive at a more accurate estimate of the clock frequency and this improved the shape(width) of folded profiles. Figure (4.41) shows some folded profiles obtained during these tests. After several iterations of these experiments the instrument was used to observe pulsars with the same setup except that the LO is fixed at a constant power of +7 dBm (without amplitude modulation).

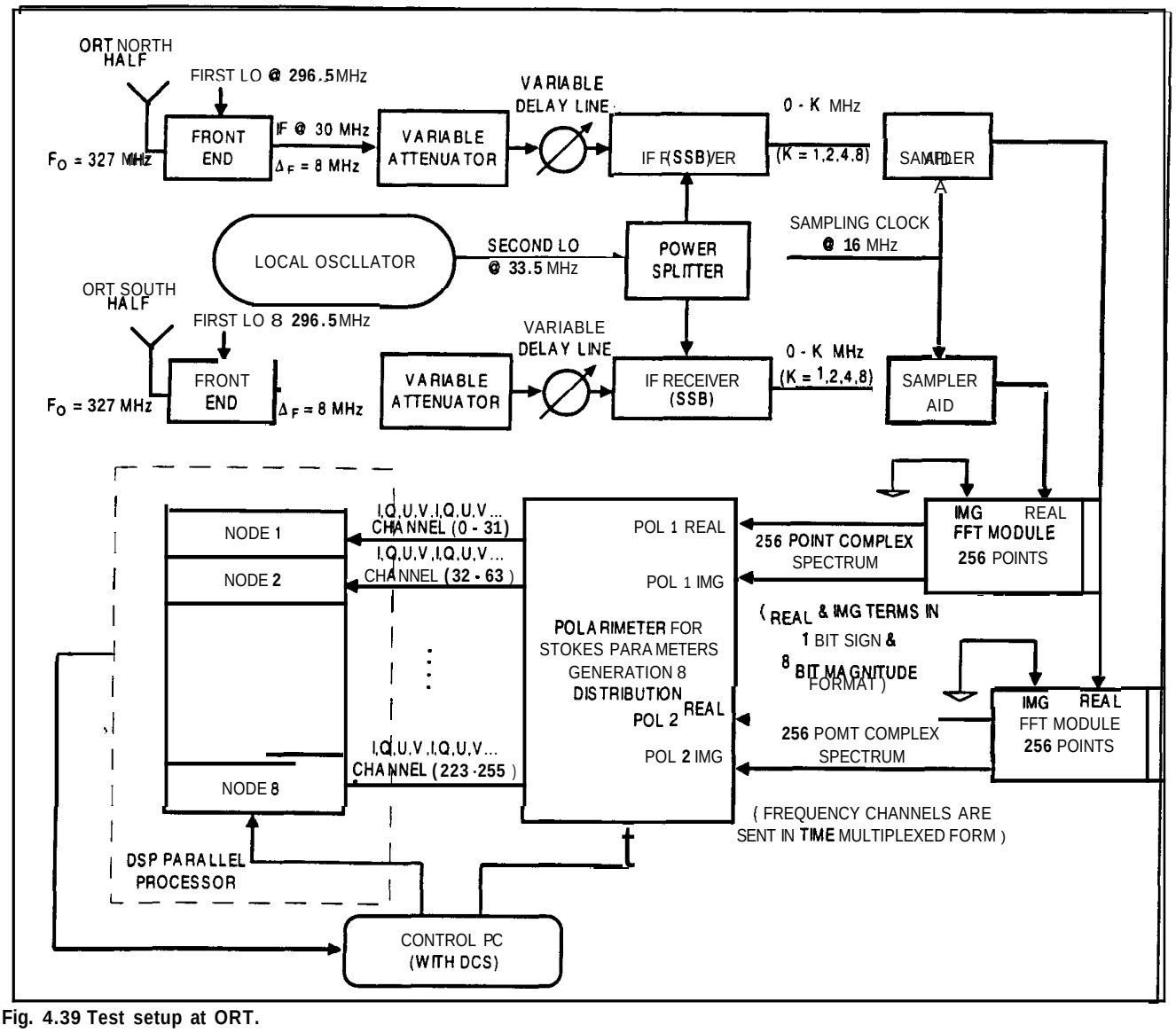


Fig. 4.39 Test setup at ORT.

In this mode of connection the Stokes parameters calculated by the polarimeter do not represent polarization characteristics, but represent physically different terms as shown in table (4.9).

Table (4.9)

I	$E_n E_n^* + E_s E_s^*$	Total power
V	$E_n E_n^* - E_s E_s^*$	Difference power
$\frac{Q+jU}{2}$	$E_n E_s^*$	Complex correlation of North & South voltages.

As can be seen the Stokes parameter 'I' corresponds to the total power detected from the array, parameter 'V' corresponds to the difference of the power spectra from the two halves of the array (due to possible gain differences) and the parameter $\left[\frac{Q+jU}{2}\right]$ represent the complex cross correlation of the signals of the North and South Halves across the spectrum. This method of connection is sufficient to allow us critically evaluate the performance of the machine even though the ORT does not posses dual polarization facilities. The band-width chosen at the receiver was 4 MHz even though the sampling rate was kept at 16 Megasamples/sec. Due to this the signal is over sampled but this does not affect the measurement since the two DSP nodes together tap a total of 64 channels out of the 256 points spectrum corresponding to a total of 4 MHz.

Gain Equalization:

Gains of the North and South receivers were adjusted such that the difference in their powers was minimized. This was apparent in the parameter 'V' obtained through the polarimeter and the DSP nodes by performing a spectral average of 262144 folds. The gain difference for different attenuation setting were measured and minimized as shown in figure (4.40). The results of figure (4.40a) are obtained with the North half connected to both of the IF receivers. Figure (4.40 b) shows the South half connected to both of the receiver channels. Figure (4.40c) shows the adjustment when the North and South halves connected to the two receiver channels respectively.

Phase Equalization:

The total lengths of the signal paths from the telescope front-end receivers to the base-band outputs are not identical for the North and South halves of the antenna. Due to this difference in the arrival time of the signals .

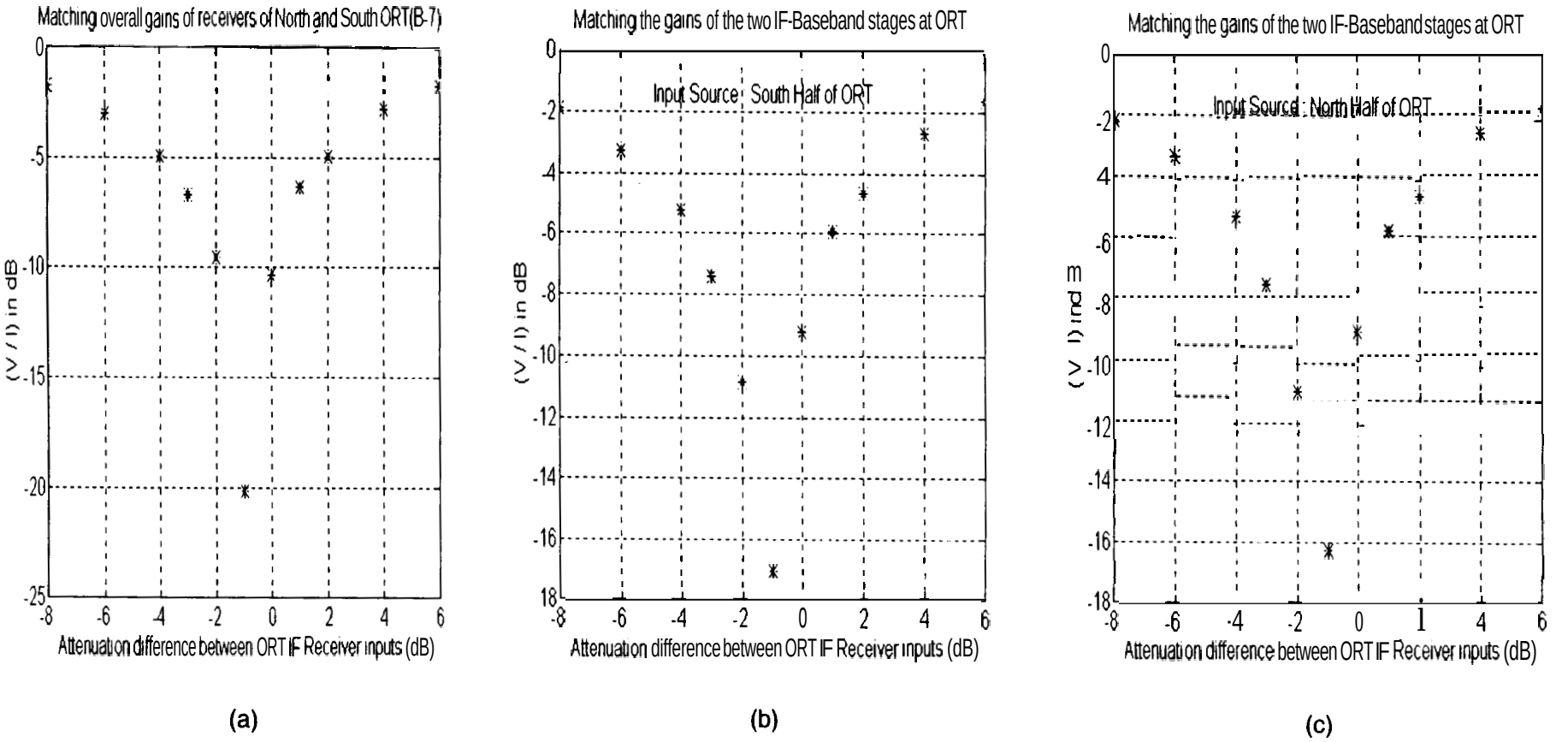
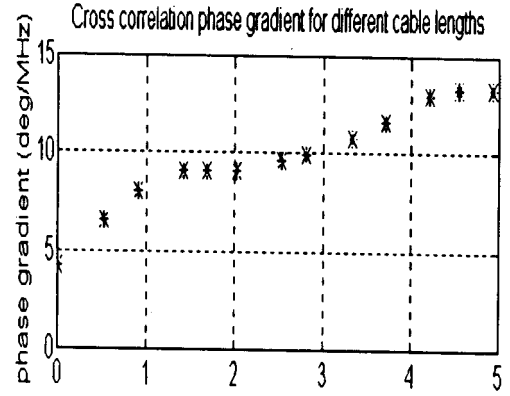
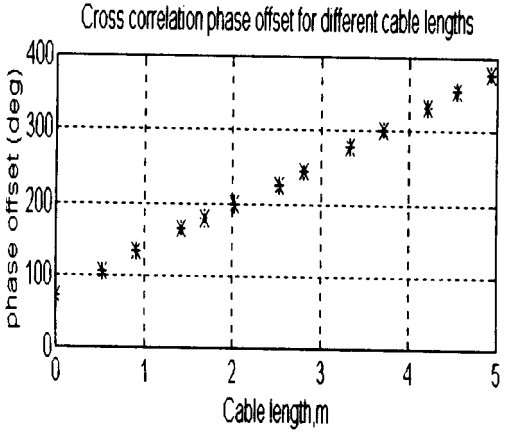


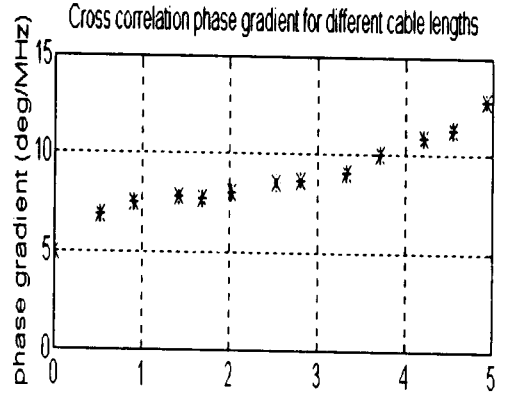
Fig. 4.40 Shows the difference in the power spectra of the two receiver polarization inputs channels. In (a). North half of ORT is connected to one polarization input and the South half is connected to the other. In (b). both receivers get a common input at the IF stage. from the South half of ORT. In (c). both inputs are common again. but with the source being the North half of ORT. While (b) and (c) depict the differences in the gain between the two IF-baseband stages. (a) compares the overall response of the two halves of the antenna and the respective receiver systems.

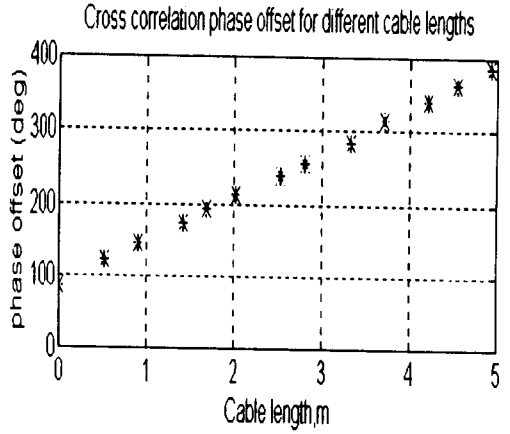




Common Input Source : North Half of ORT

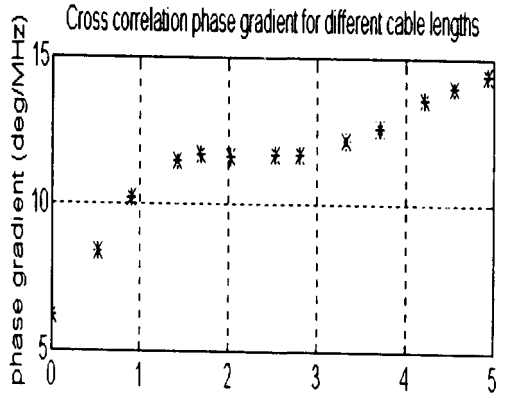
(a)

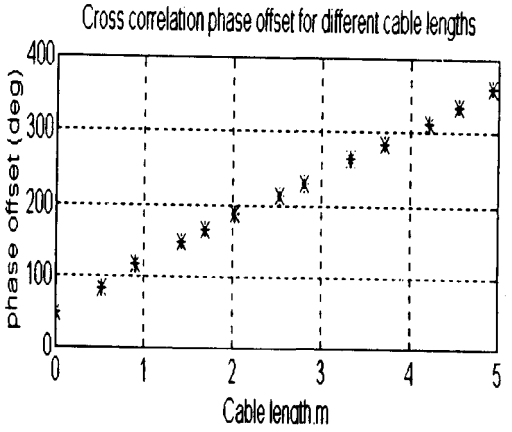




Inputs : Pol1 = North Half,
Pol2 = South Half of ORT

(b)





Inputs : Pol1 = South Half,
Pol2 = North Half of ORT

(c)

Fig. 4.41 Observed differences in the phase offsets and gradients between the two polarization inputs as a function of relative path-length difference . The phase-offsets convey the path-length difference already existing in the ORT. (a) shows the differences when both polarization inputs are derived by connecting a common IF input to both IF receivers, while (b) and (c) show the differences when the two inputs are derived from the two halves of the ORT array.

at the receivers of the two halves is offset as a function of the spectral frequency. **As** a result, the cross-correlation phase, defined as $@ = \tan^{-1}(U/Q)$, will show a gradient across the band. Some experiments were tried to minimize the path-length difference by inserting suitable cables at the IF stage of the two receivers. Figure (4.41) shows the observed phase gradient and offset phase at the edge of the band, as a function of the relative delay inserted, when the antenna was pointed to a strong continuum source, The results of figure (4.41a) are obtained with the North half connected to both of the IF receivers. Figure (4.41b) shows the South half connected to both of the receivers. Figure (4.41c) shows the adjustment when the North and South halves connected to the respective receivers.

After the phase equalization, a strong pulsar (PSR 1749-28, Savg - 1.3 Jy) was observed and the pulses were folded on-line. The resultant profile displayed in figure(4.42) shows 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 less , indicating that the gains and phases are closely matched [(V - 0) & (I-Q ; U - 0) respectively].

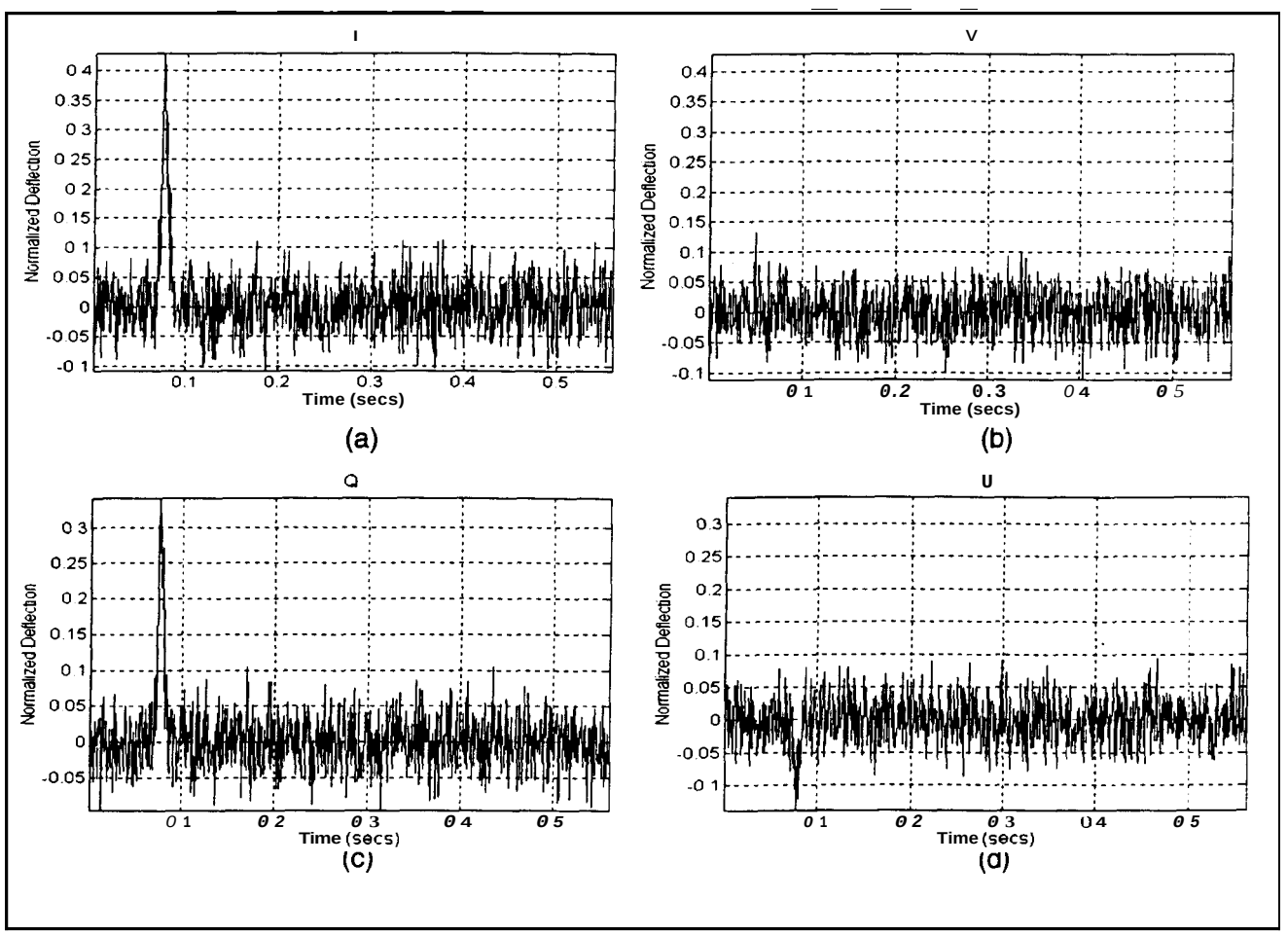


Fig.4.42 Pulsar PSR 1749-28 observed on 12-12-1997. Period - 0.56 secs, pulsewidth –7.5 msecs, DM - 50.88, Savg - 1.3 Jy. No. of adjacent samples added per time bin in the profile - 27, time resolution in profile - 0.56 msec. No. of folds = 4.

These results are similar to those obtained from earlier tests with the being derived inputs from digital pulse patterns [Fig(4.43)], where the magnitude and sign of all input terms are alike.

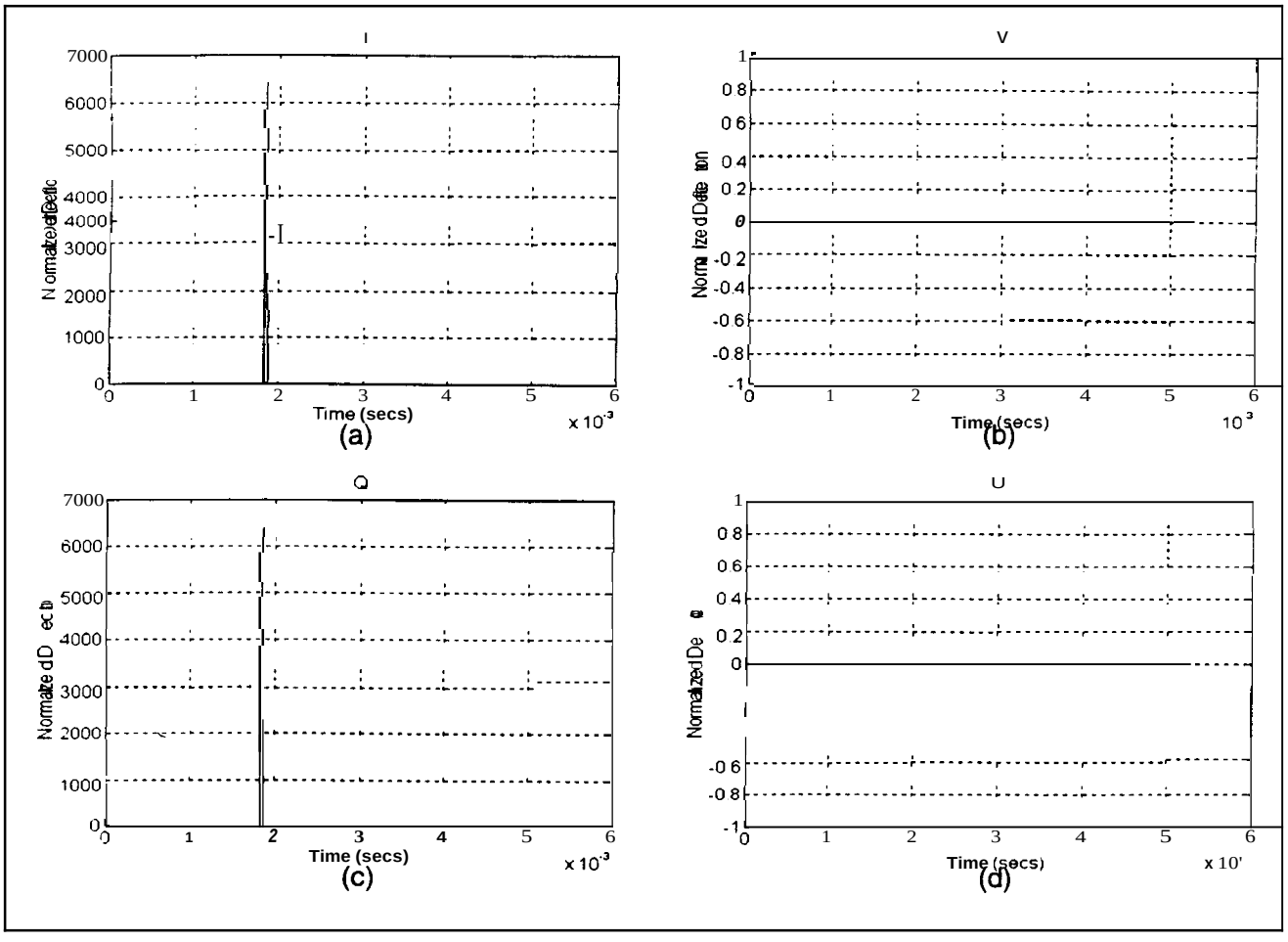


Fig.4.43. Folded average profile with identical digital pulse pattern inputs to both polarization channels, Period: 256 samples, same pattern input to all frequency channels without any dispersive delays. 1000 periods folded, no adjecent sample integration, all frequency channels averaged without dedispersion.

Further to these tests, the performance of some factors that determine the time resolution in the folded profile was tested. Initially, the clock frequency was known only to an accuracy of about 1 part in 10^5, using a crystal oscillator based clock source. Later, the clock was derived from a programable frequency generator, which was phased locked to a stable reference signal from a Rubidium oscillator. A PC-add-on digital frequency counter was built and integrated into the system, to monitor the clock frequency periodically, using a reference signal from the same Rubidium oscillator. With this arrangement. the clock was determined to a much higher accuracy (better than 1part in 10^{12}). Using this estimate, the folding parameters were updated and the observation was repeated on pulsar 1133+16. The results displayed in figure(4.44). The sharpness of the profile shows significant improvement in the time resolution, while the signal to noise ratio may not improve correspondinly, since the intrinsic stength of the pulses may vary with time.

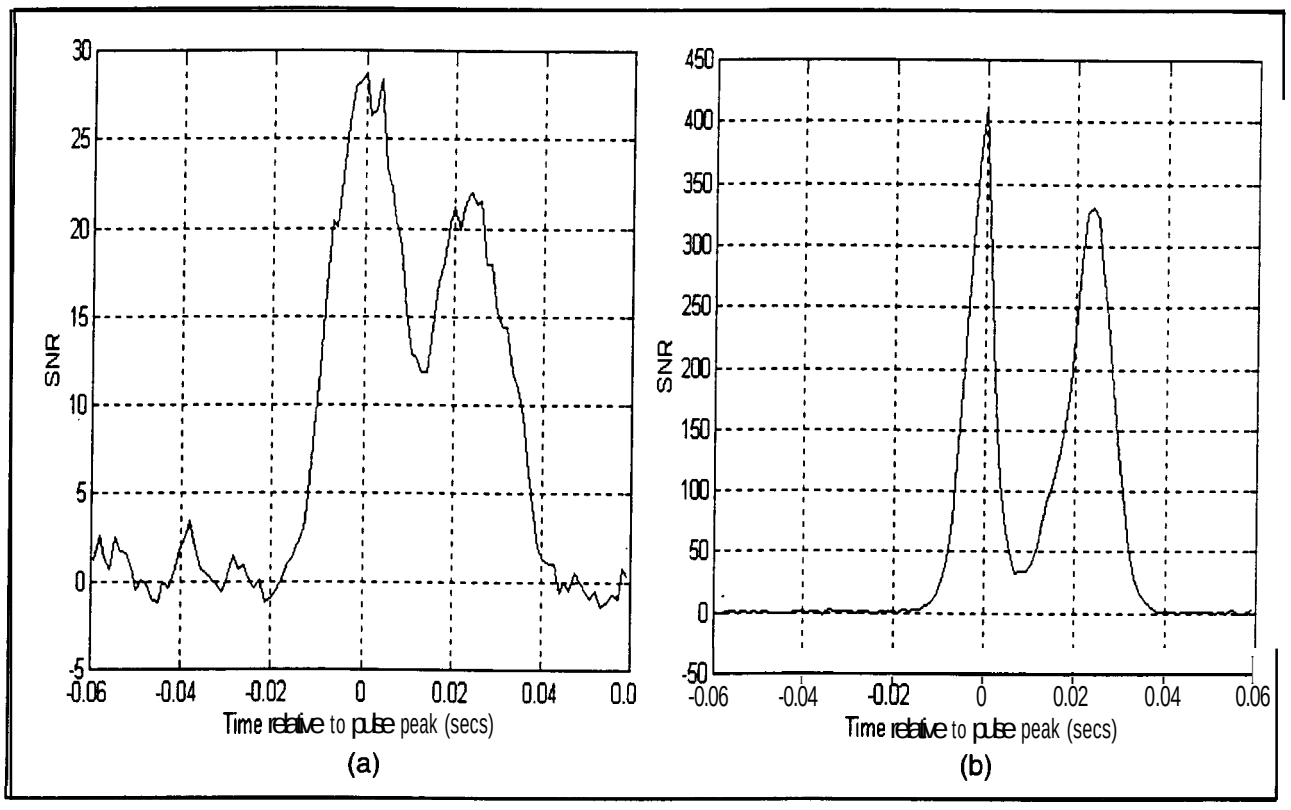


Fig. **4.44. Profile of pulsar PSR 1133+16. Pulse smearing due to uncertainity in assumed frequency of** FFT **system clock (40 MHz). Accuracy in estimate of FFT system clock frequency was about Ipart in 10^5 in (a) and batter than 1 part in 10^{10} in (b) [desired FFT clock frequency = 40 MHz] Number of periods folded in both cases = 1099.**

As mentioned earlier, during pulse-folding, the pulse phase is tracked by a 32-bit pointer, in which the top 16 bits represent the integer pulse-bin number in the profile, while the lower 16 bits represent the fractional phase within a bin. The residual error in fractional phase (beyond the least-significant-bit of the 16-bit fractional phase pointer) will accumulate as the folding progresses. The DSPs are. programed to periodically correct this error by keeping track of this accumulated error and subtracting the residual phase offset whenever it accumulates to about half a bin of the profile(this is optional). To check the performance of this function, the pulsar PSR1749-28 was observed with and without phase-error correction and the significant change in the pulse shape was seen. Figure (4.45) shows the profiles obtained without and with correction respectively. The residual error in pulse-phase that accumulates for every raw-sample in this case is about five parts in 10^5. After the timing parameters were properly estimated, on-line update of pulsar period was enabled so as to feed new parameters from the control PC to the DSP nodes once every second. Further observations were conducted in this mode.

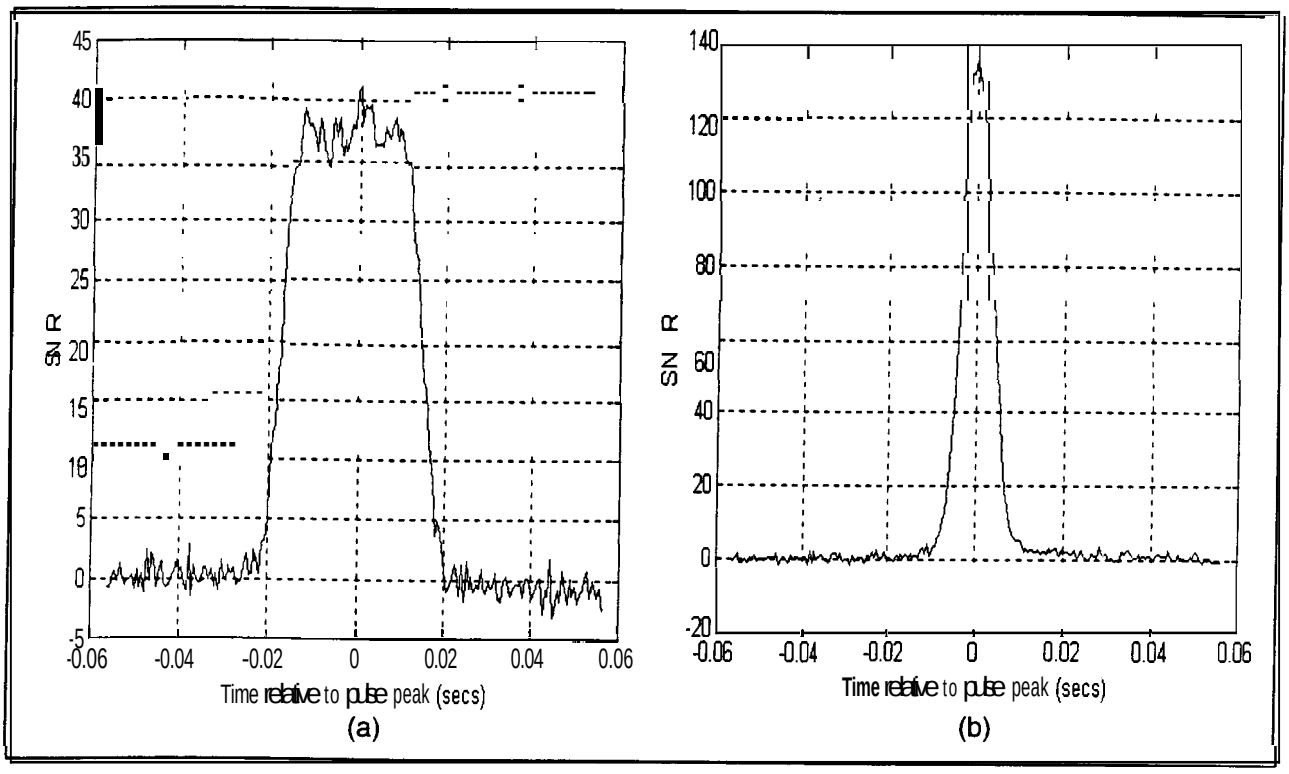


Fig. 4.45. Folded total-power (I) profile of PSR 1749-28 observed on 05-10-1997. Period = 0.562 secs, DM = 50.88, Savg = 1.31 mJy, equivalent pulse width = 7.5 msec, No. of folds = 580, No. of adjacent time samples integrated per time bin in the profile = 27, 64 channels averaged after DM correction. In plots (a) and (b) folding progresses without and with online correction respectively, for the accumulated pulse phase error due to finite representation of pulse phase. Significant smearing can be observed in the absence of this correction in (b).

Further, the dispersion correction function was tested by observing pulsar 1749-28, with and without on-line correction for the delay gradient accross the band, and the corrected pulse profiles were folded independently in each frequency channel and recorded. Figures (4.46 (a)) and (4.46 (b)) show the frequency-time-intensity plot obtained with and without dispersion correction respectively, around the on-pulse region. Such measurements can indeed be used to get better estimates of the DM by inspecting the residual delay gradients after the correction, when suitable operating frequencies and bandwidth are available. In the present observations using the two halves of the ORT array(which is a single polarization(linear) telescope), it is difficult to associate any residual gradient entirely to a corresponding difference in the value of DM. This is because a similar pattern may also be produced by the misalignment of the polarization vector of the incident

g n a l with respect to the antenna at different pulse-longitudes(due to the combined effect of Faraday rotation and the intrinsic sweep of the position angle within the pulse).

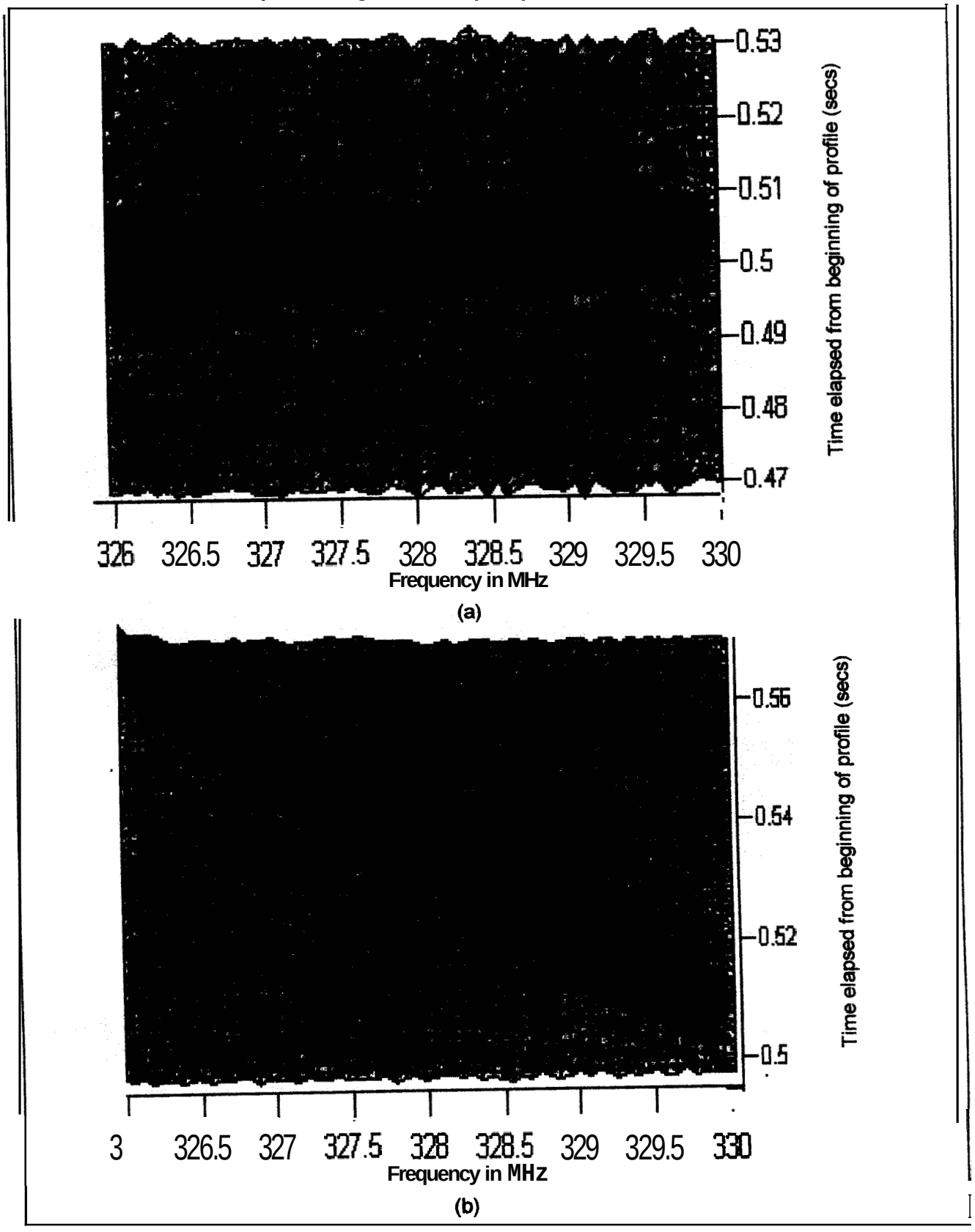


Fig.4.46 Folded total Power (I) profile of Pulsar PSR 1749-28 observed on 05/10/1997, with DM Correction (a) and without DM Correction (b), pulse-period ~ 0.56 secs, pulse width ~ 7.5 msec Savg ~ 1.3 Jy, DM = 50.88, Number of Folds = 145, Number of Frequency Channels = 64

After these tests, pulsar PSR 1749-28 was re-observed in a mode in which the dispersion corrected frequency channels were added together during folding process. Such observations were repeated for different lengths of time and a corresponding improvement in signal to noise ratio(SNR) was noted. Figure (4.47) shows the these results.

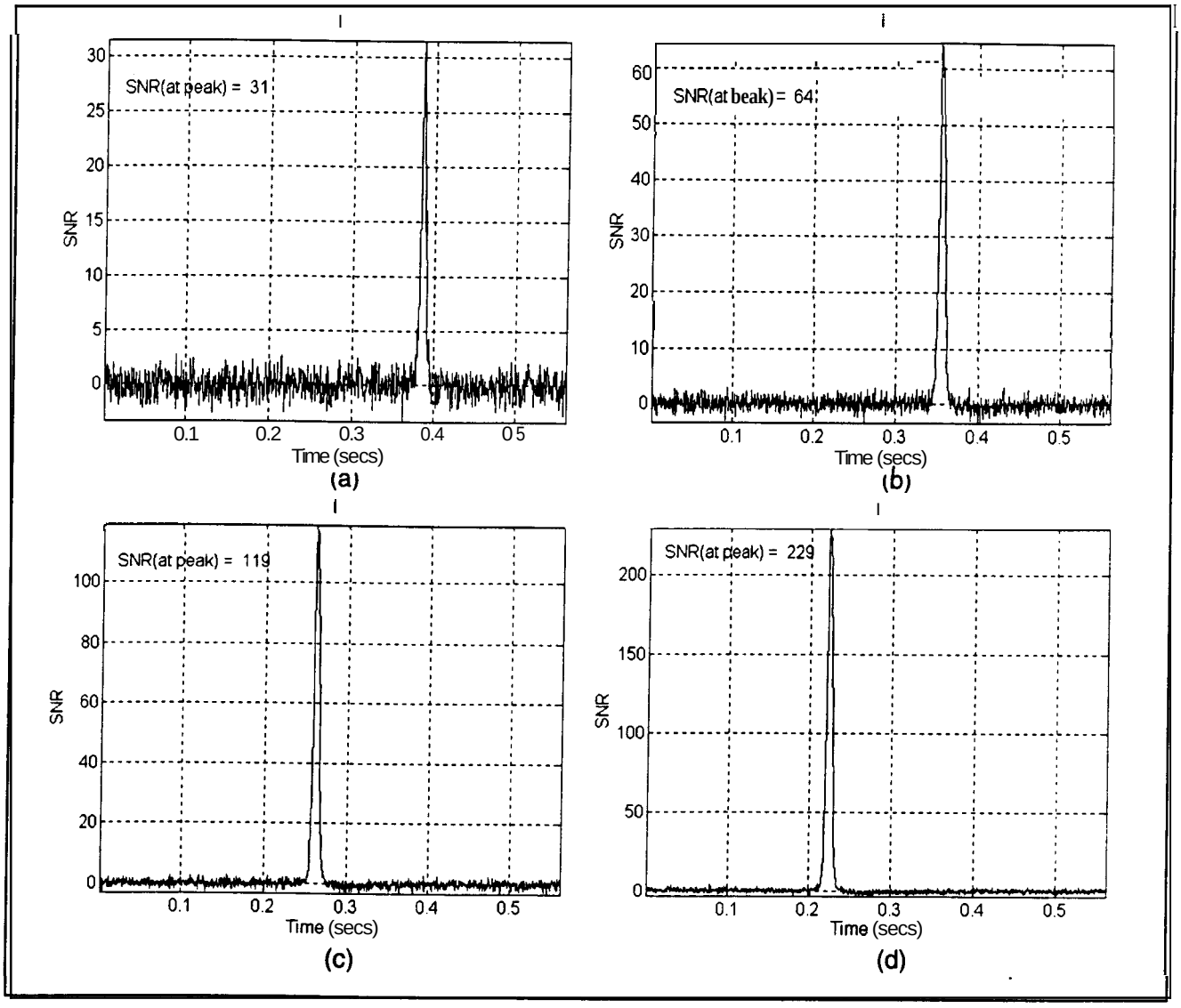


Fig.4.47. Signal to Noisde Ratio(SNR) change for different integration of pulsar PSR 1749-28 observed on 13-12-1997. Period -0.56 secs. DM -50.88, Savg -1.3 Jy, Time resolution in folded profile - 550 micro-secs, - 27 adjacent samples integrated per time bin in profile, 64 frequency channels corrected for dispersion and averaged. No. of Folds : (a) 18, (b) 73, (c) 290 and (d) 1160

Subsequently, many other pulsars were observed covering a wide range of periods, flux-densities and DMs. Figure(4.48) shows the total power (I, in units of SNR) profiles obtained from some of these observations. Pulsars with long periods were chosen to test the adjecent-sample integration function, while these with short period (< 20msec) were observed with full time resolution(20.4 microsecs per bin). Some pulsars with low flux-density were chosen to check the stabilty of the instrument during long-term pulse-folding, typically over an hour. Some pulsars with reasonable high DM (> 50) were chosen to check the channel-alignment and frequency-integration functions. The parameters of these observations are

summarized below. In common, the dedispersion and period update was performed and all the channels were averaged together covering a bandwidth of 4 MHz.

	PSR Name	Period (secs)	Savg (mJy)	DM ($cm^{-3}pc$)	Time resolution (msecs)	Total Folding time (min)
a.	80331+45	0.269	5.16	47	0.269	87
b.	B0609+37	0.298	17.64	26.7	0.298	174
c.	B0628-28	1.244	221	34.36	1.244	1.4
d.	B0740-28	0.167	335	73.7	0.167	5.5
e.	81133+16	1.18	333	4.84	1.18	5.4
f.	81702-19	0.298	33	22.92	0.298	86.7
g.	82016+28	0.558	314	14.17	0.558	2.7
h.	J0437-4715	0.006	600	2.65	0.0204	0.2

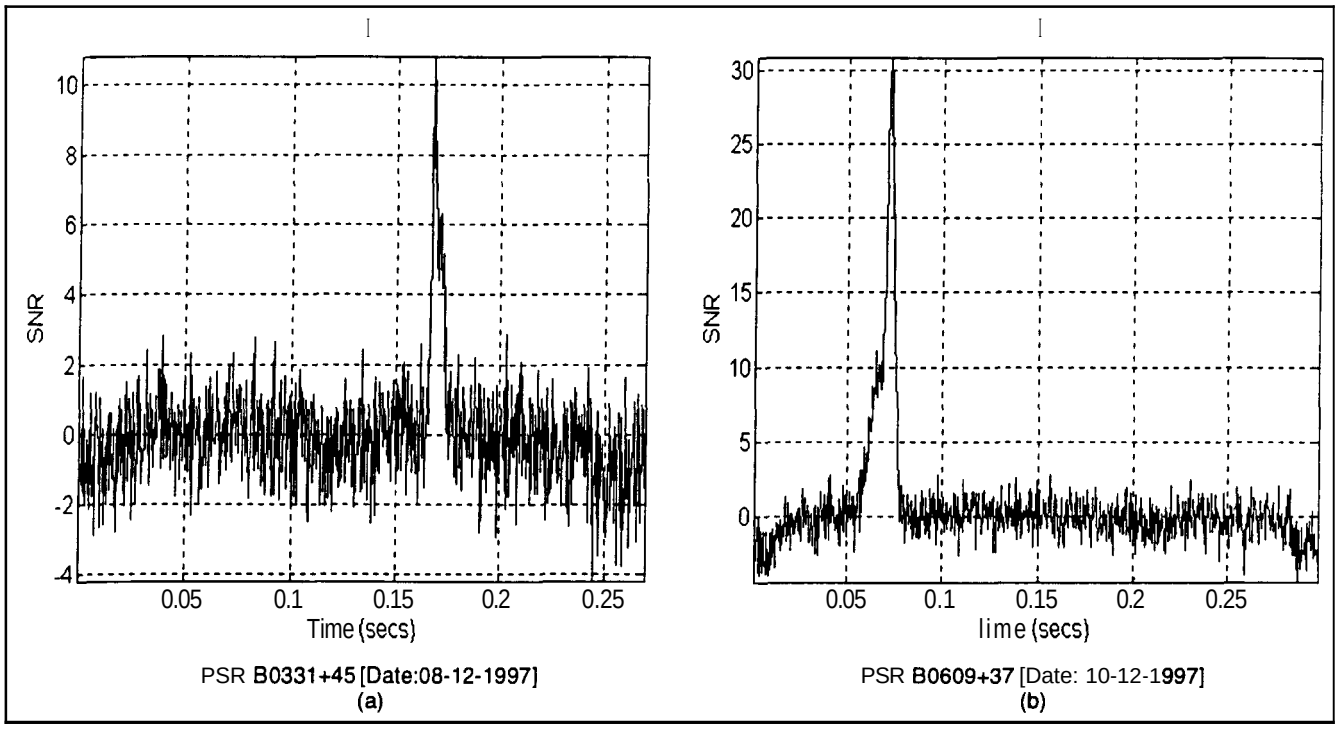
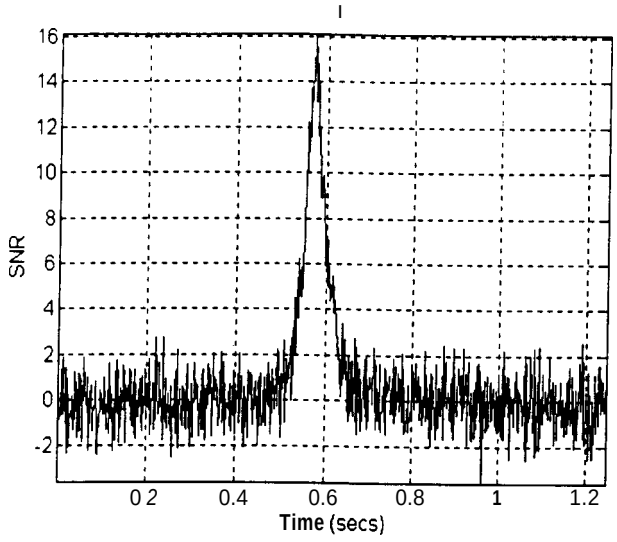
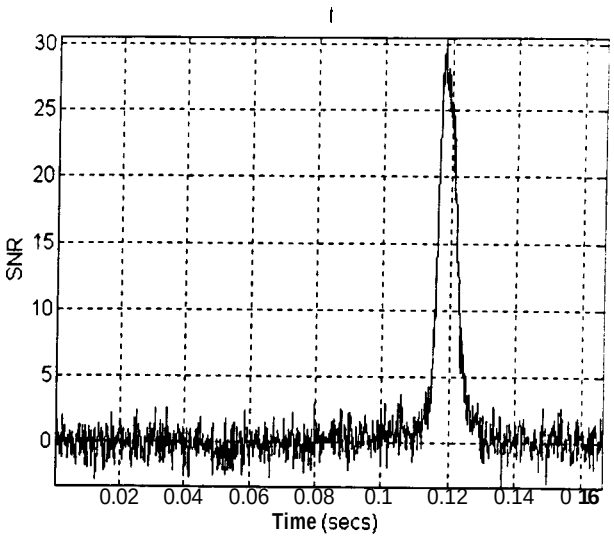


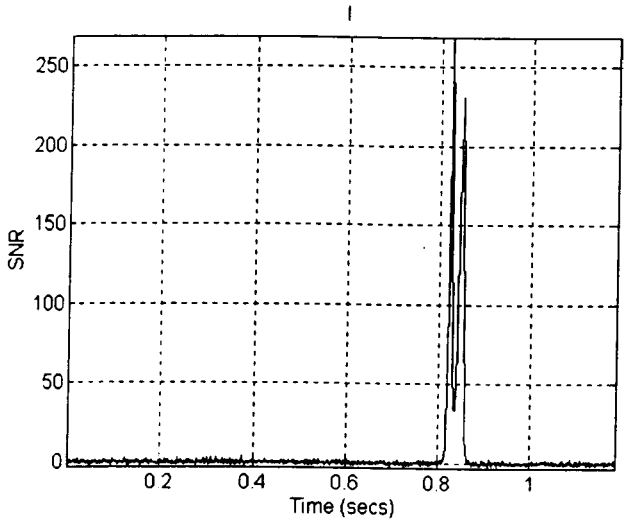
Fig. 4.48



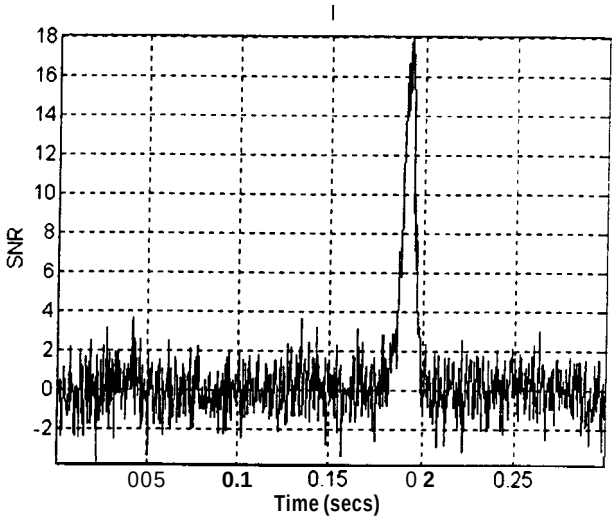
PSR 80628-28 [06-12-1997]
(c)



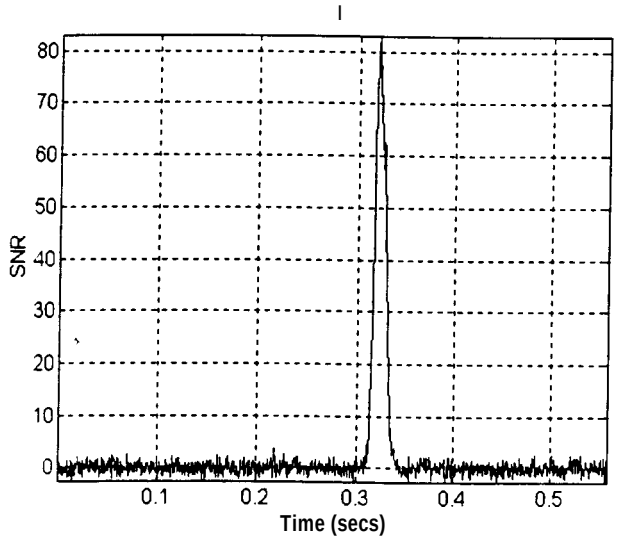
PSR 80740-28 [Date: 09-12-1997]
(d)



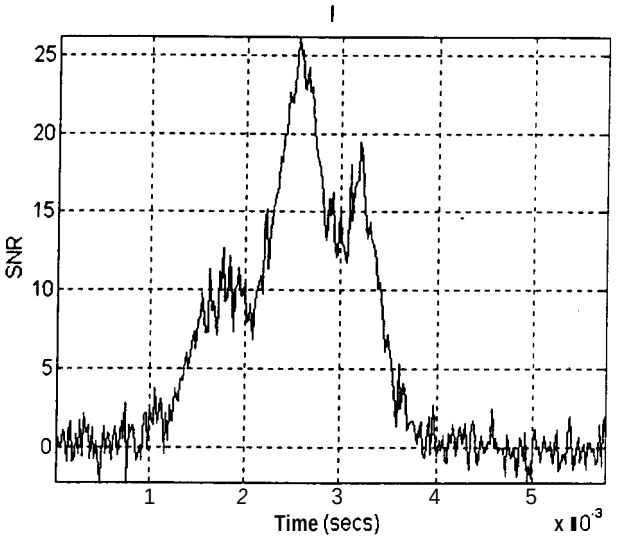
PSR 81133+16 [Date: 07-12-1997]
(e)



PSR 81702-19 [Date:08-12-1997]
(f)



PSR B2016+28 [Date: 13-12-1997]
(g)



Binary pulsar J0437-4715 [Date:26-10-1997]
(h)

Fig. 4.48. (Continued)

Currently, the tests are being continued to check another feature of the instrument, namely, the pulse-gating operation. Also the feedback from the tests have been used in improving the perfomance of the system. An enhanced version of the SPPS system is being reproduced to handle full 32 MHz bandwidth at GMRT. In the meanwhile, the current two node, **4** MHz system is planned to be used for conducting further tests with the dual polarization antennas of the GMRT array. With the Plessey FFT front-end, the SPPS is also capable of operating with any other telescope having suitable bandwidth.