To Library 78-114

Génie Electrique

Electrical Engineering

RESEARCH ON MICROPROCESSOR ARRAYS FOR SIGNAL PROCESSING AT HIGH DATA RATES

Final Report

C.V.W. Armstrong

Supply and Services Canada
Contract No. OSU-78-00099

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

RESEARCH ON MICROPROCESSOR ARRAYS FOR SIGNAL PROCESSING AT HIGH DATA RATES

Final Report

C.V.W. Armstrong

Supply and Services Canada
Contract No. OSU-78-00099

Research Report No. 3
MACS Project

March 31, 1979

CONTENTS

1. INTRODUCTION

This final report is divided up into a number of sections. The first section describes a simulator that was developed for the purpose of studying algorithms for the MACS processor. Following sections describe a set of radar and other related processing problems noted in the Interim Report [1]. These problems are as follows:

(1) plot-to-track correlation

(2) FFT processing

(3) partial plot correlation

(4) MTI processing.

A fifth topic, namely Kalman Filter processing, was not treated due to lack of time although some interesting research was referenced in the Interim Report.

The final section of this report summarises the main findings and attempts to identify those areas of future study which seem fruitful. The major results of this research are embodied in the simulator and these recommendations for future research and development associated with this array architecture. A number of possible changes in the design of the multitimicroprocessor array are noted in this conclusion.

2. MACS SIMULATOR

A simulator has been developed, written primarily in FORTRAN, which allows a microprogrammer to run a MACS microprogram for an array of any number of PEs. A complete report on this effort is provided in [2]. The simulator goes ahead and implements a number of proposed hardware changes and has been used to demonstrate the usefulness of these changes.

The simulator has been split into a number of programs in such a way that every physical device is associated with a particular subprogram, allowing any FORTRAN programmer to easily modify the architecture of the simulated MACS system by simply creating an appropriate program. The interaction between programs is as simple as reflected in a block diagram representation of the system.

Chapter 2 of this user manual describes the MACS architecture as presently implemented in the package of program modules forming the simulator. An overview of the general architecture is presented as well as an explanation of the processing element (PE) capabilities. Microinstruction fields are treated in great detail.

Chapter 3 explains thoroughly with many examples, how to input microprograms to the simulator and how to execute them. It also demonstrates how to program the top and bottom buffers, as well as the working memory. It shows how to run the simulator in a stand-alone mode, using the monitor to control the execution and debug the microprograms.

Chapter 4 is a technical description of the simulator, providing a better understanding of the simulator to the user, but mainly provided for the more sophisticated user who may want to modify the architecture or improve on the current simulation. In this chapter, all the programs, program variables and common blocks are explained in great detail and the interactions between program modules are examined.

3. PLOT-TO-TRACK CORRELATION

This section reports on the progress made in studying the problem of plot-to-track correlation. As noted in the interim report, it was decided to go back to first principles and refer to Sittler's original discussion on the association problem [3]. In particular, it was decided to investigate the one space dimension problem because as noted in the paper, the appropriate equations are valid for any number of space dimensions when the appropriate variables are replaced by vectors. In a microprocessor array of this sort, this generalization would be reflected in processing vectors rather than scalars where the same operations are involved. We would therefore see simply a reduction in throughput although the number of vector "elements" processed per unit time would remain the same.

3.1 Processing Problem

As developed from [3], there are basically three problems to be considered as suitable candidates for array processing. Before enumerating these problems, it is necessary to briefly describe the surveillance model under consideration.

The host computer must provide this surveillance model which relies on:

(a) a set of actual tracks

(b) a set of current plots

This must be updated from one association operation to the next based on the following assumptions:

(a) Tracks

(i) Track Starting

This is given by a Poisson model with parameter λ_o. This means that λ_o objects per unit time per unit area are generated at random. That is, this is a constant average rate for independent Bernouilli trials.

(ii) Track Length

This is provided by an exponential distribution with time constant τ_o.

(iii) Track Positions

Once it has been determined (probabilistically) when and where a track commences and what its track length is going to be, it is considered to follow a general random track model of independent motions.

(b) Plots

A sequence of "observations" are made of this "actual" track. The time at which such observations are taken is given by a Poisson model with parameter λ_s. Again this is based on independent Bernouilli trials for each time increment. Although, at a given time, we know the "true" position, the measured position for a plot is influenced by an error distribution which is Gaussian with a Markov property:

$$f(x_{ik}|x_{i1},x_{i2},\ldots,x_{1,k-1}) = f(x_{ik}|x_{i,k-1})$$

$$= \frac{1}{\sqrt{2\pi}\,\sigma_{ik}}\, e^{-(x_{ik}-\hat{\mu}_{ik})^2/2\sigma^2_{ik}}$$

$$\left.\begin{array}{ll} \text{with} & \sigma^2_{ik} = \varepsilon^2 + \sigma^2_0\, t_{ik} \\ \text{and} & \hat{\mu}_{ik}\ (x_{i,k-1}) \end{array}\right\} \text{determined by appropriate tracking algorithms.}$$

x_{ik} = k^{th} measured plot position for i^{th} track

$\hat{\mu}_{ik}$ = expected k^{th} measured plot position for i^{th} track

σ_{ik} = standard deviation of the expected k^{th} measured plot position distribution

In addition, there are plots associated with false alarms which are also given by a Poisson model with parameter λ_N.

We are attempting simple chain associations (that is, we do not consider forking or crossing tracks) and with these simplifications, it was determined that the following two routines were candidates for PE array processing:

(1) Gate Association

For each track, the PE array receives parameters specifying a gate. All plots within that gate must be tagged.

(2) Maximum Likelihood Calculation

For each track and the plots tagged as within the gate for that track, the following calculation must be performed:

$$L_i = \ln \frac{\lambda_o \lambda_s}{\lambda_N (\lambda_s - \frac{1}{\tau_o})}$$

$$+ \overbrace{\sum_{k=2}^{n_i} \left[\ln \frac{\lambda_s}{\lambda_N \sqrt{2\pi}\ \underline{\sigma_{ik}}} - \frac{1}{2} \frac{(\underline{x_{ik}} - \underline{\hat{\mu}_{ik}})^2}{\underline{\sigma_{ik}^2}} \right]}^{①}$$

$$- (\lambda_s + \frac{1}{\tau_o}) \overbrace{(\underline{v_i} - \underline{u_i})}^{②}$$

$$+ \ln \left[\frac{\lambda_s e^{-(\lambda_s + \frac{1}{\tau_o})(T - \underline{v_i})} + \frac{1}{\tau_o}}{\lambda_s + \frac{1}{\tau_o}} \right]$$

Appropriate values for λ_o, λ_s, λ_N, τ_o and T (current time)must be provided. The underlined variables are provided for each plot (5 items per plot). ① is a partial sum which is already precalculated for the previous case. ② is also a partial sum if it is interpreted as $\sum_{k=2}^{n_i} t_{ik}$ (the some of the time intervals between observations). The maximum L_i generated by a plot must be used as a signal to tag that plot as the best choice.

The host computer can used this best value for L_i to further classify the track as being in one of the following categories:

(1) tentative

(2) initiated, unconfirmed

(3) confirmed, good

(4) poor

(5) dropping

(6) dropped (tentative)

The host computer could use the maximum likelihood value calculated to determine track initiation, track confirmation and track maintenance. This would involve using a number of threshold values which are calculated for each track. This operation could be done during the time the maximum likelihood operation is being performed in the PE array. In addition, all unassociated plots in the track file must be treated as tentative tracks, which would greatly increase the load on the PE array. Those plots not matched to tracks must be treated as false alarms.

The master program running in the host computer must perform the operations given in Figure 1. Progress did not reach the point where this program could be written and run but implementations of the appropriate PE microprograms were developed.

3.2 Implementation

Figure 2 gives the PE microprogram for gate association. As shown, this is indeed very simple as it applies to only one space dimension. (An increase in the number of space dimensions requires a corresponding increase in the number of PEs, one dedicated to each dimension).

Implementation in the PE array can only be justified if the plot information can be stored in the input buffer and accessed at the high data rate

```
(1) determine new tracks

(2) determine set of plots
Do for all tracks :
    (3) load track information into PE array

    (4) load buffer with plot information

    (5) run simulator

    (6) store tagged plot information
Do for all tagged tracks :
    (7) load track information into PE array

    (8) load buffer with tagged plot information

    (9) run simulator

    (10) store associated plot information

(11) make appropriate decisions

(12) display results
```

Fig.1. Master Program for Plot-to-Track Correlation

```
700114020   130    1;DATA FROM ABOVE IN Q
621715040   130    2;MAX(ER0)-Q,DISABLE IF NEG.
611615060  4130    3;Q-MIN(ER1),DISABLE IF NEG.
200015000   130    0;Q OUT BELOW IF ENABLE,GO BEGIN
    0        0       0     0;
    0        0       0     0;
    0        0       0     0;
    0        0       0     0;
    0        0       0     0;
    0        0       0     0;
    0        0       0     0;
    0        0       0     0;
    0        0       0     0;
    0        0       0     0;
    0        0       0     0;
    0        0       0     0;
```

Fig.2. Gate Association Microprogram

implied by this short microroutine. Otherwise the host computer can prescreen the plot information and allow the PE array to simply perform the maximum likelihood operation.

Figure 3 gives the PE microprogram for the likelihood calculation. Five PEs are required for this operation. Note that a likelihood value is calculated once every microcycle of 16 microinstructions.

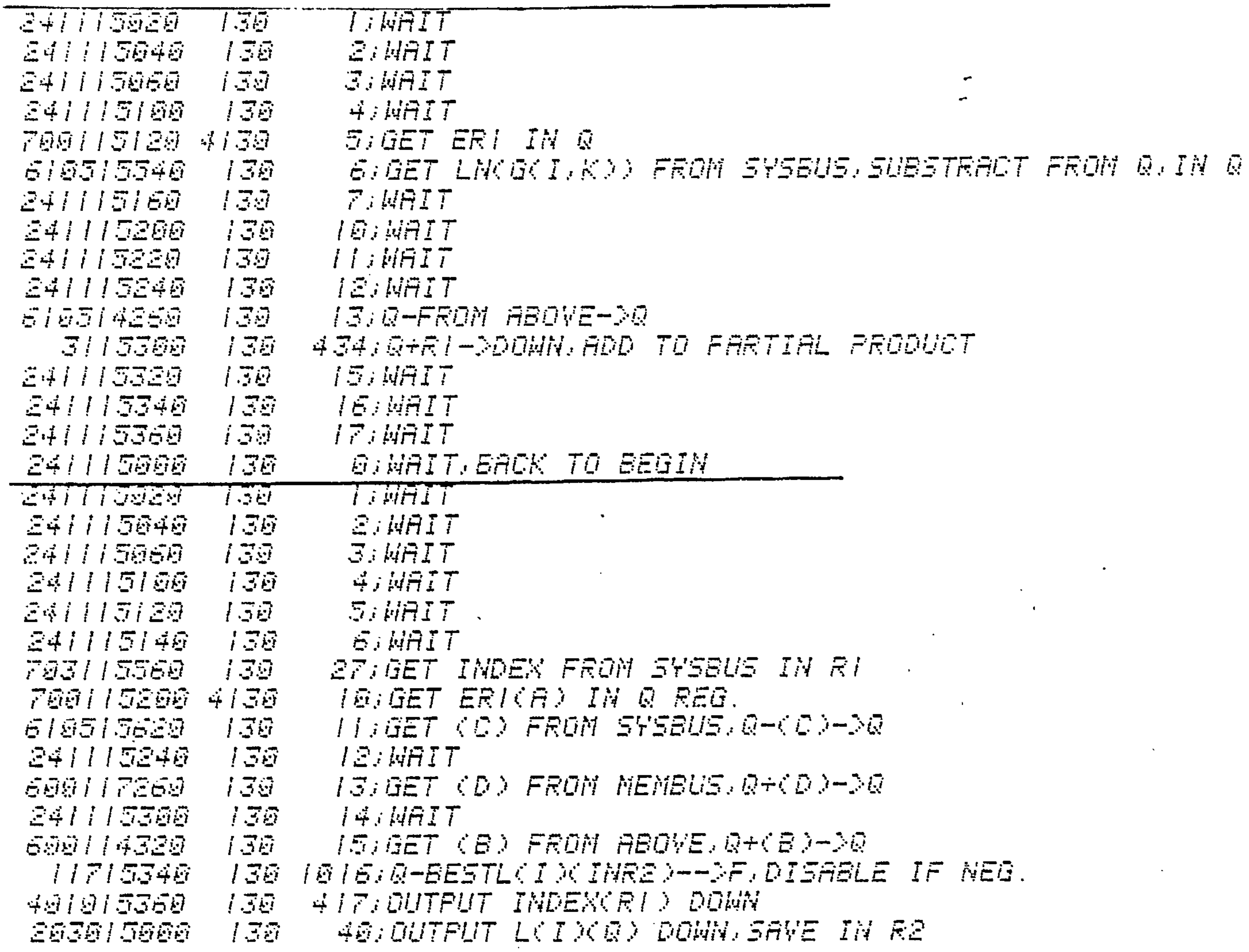
3.3 Evaluation

Firstly, note the performance that can be achieved with these microprograms. The gate association routine requires 4 microinstructions. Thus, 1000 plots can be processed in 24 μsecs and for 200 tracks, the complete operation would take 4.8 milliseconds. This is a rule-of-thumb calculation assuming a 60 nsec clock cycle time. For the maximum likelihood function, assuming 20 plots within the gate, each track requires 19 μsecs and for 200 tracks, the complete operation takes 3.84 milliseconds.

A number of hardware modifications are necessary for this implementation to be practical. A hardware multiplier is required because of the complexity

```
3 24115020   130      1;WAIT
 70315040 4130    22;GET T FROM ER1 INTO R1
 51515460  530  403;GET V(I) FROM SYSBUS,T-V,MULTIPLY *BY ER2
 24115100  130     4;WAIT FOR MULTIPLICATION RESULT
 24115120  130     5;WAIT FOR MULTIPLICATION RESULT
 70126140  130     6;RESULT IN & TO MEMBUS FOR EXPONENTIAL
 70117160  730     7;RESULT FROM MEMBUS,MULTIPLY BY ER3
 24115200  130    10;WAIT FOR MULTIPLICATION RESULT
 702115220201130  111;WAIT & GET ER4 IN R4
 50126240  130 2012;RESULT + R4->MEMBUS FOR LN
 70017260  130    13;RESULT FROM MEMBUS(LN) IN Q
 24115300  130    14;WAIT
 24115320  130    15;WAIT
 24115340  130    16;WAIT
 24115360  130    17;WAIT
 24115000  130     0;WAIT,BACK TO BEGIN
 24115020  130     1;WAIT
 24115040  130     2;WAIT
 70015460  130     3;GET V(I) FROM SYSBUS IN Q
 24115100  130     4;WAIT
 61051552 0 330    5;GET U(I) FROM SYSBUS,V-U->R,MULTIPLY BY ER1
 24115140  130     6;WAIT FOR MULTIPLICATION RESULT
 24115160  130     7;WAIT
 70013220 0 130   10;RESULT TO SYSBUS
 24115220  130    11;WAIT
 24115240  130    12;WAIT
 24115260  130    13;WAIT
 24115300  130    14;WAIT
 24115320  130    15;WAIT
 24115340  130    16;WAIT
 24115360  130    17;WAIT
 24115000  130     0;WAIT,BACK TO BEGIN
 70015420  130     1;GET X(I,K) FROM SYSBUS IN Q
 61051544 0 130    2;GET U(I,K) FROM SYSBUS,X-U IN Q&ER1
 20011506 0 320    3;SQUARE RESULT,Q*ER1
 70115500  130     4;GET 1/G2(I,K) FROM SYSBUS TO ER2
 24115120  520     5;WAIT FOR MULTIPLICATION RESULT
 70511614 0 130  106;DIVIDE RESULT BY 2 IN R4
 30111516 0 530  107;MULTIPLY R4*ER2
 24115200  130    10;WAIT
 24115220  130    11;WAIT
 70111624 0 130   12;RESULT DOWN TO NEXT PE
 24115260  130    13;WAIT
 24115300  130    14;WAIT
 24115320  130    15;WAIT
 24115340  130    16;WAIT
 24115360  130    17;WAIT
 24115000  130     0;WAIT,BACK TO BEGIN
```

Fig.3. Maximum Likelihood Calculation Microprogram



```
 241115020   130      1;WAIT
 241115040   130      2;WAIT
 241115060   130      3;WAIT
 241115100   130      4;WAIT
 700115120 4130       5;GET ER1  IN Q
 610315540   130      6;GET LN(G(I,K)) FROM SYSBUS,SUBSTRACT FROM Q,IN Q
 241115160   130      7;WAIT
 241115200   130     10;WAIT
 241115220   130     11;WAIT
 241115240   130     12;WAIT
 610314250   130     13;Q-FROM ABOVE->Q
   3115300   130   434;Q+R1->DOWN,ADD TO FARTIAL PRODUCT
 241115320   130     15;WAIT
 241115340   130     16;WAIT
 241115360   130     17;WAIT
 241115000   130      0;WAIT,BACK TO BEGIN
 241115020   130      1;WAIT
 241115040   130      2;WAIT
 241115060   130      3;WAIT
 241115100   130      4;WAIT
 241115120   130      5;WAIT
 241115140   130      6;WAIT
 703115560   130     27;GET INDEX FROM SYSBUS IN R1
 700115200 4130      10;GET ER1(A) IN Q REG.
 610515620   130     11;GET (C) FROM SYSBUS,Q-(C)->Q
 241115240   130     12;WAIT
 600117260   130     13;GET (D) FROM MEMBUS,Q+(D)->Q
 241115300   130     14;WAIT
 600114320   130     15;GET (B) FROM ABOVE,Q+(B)->Q
  11715340   130  1016;Q-BESTL(I)(INR2)-->F,DISABLE IF NEG.
 401015360   130   417;OUTPUT INDEX(R1) DOWN
 203015000   130     40;OUTPUT L(I)(Q) DOWN,SAVE IN R2
```

Fig.3. (cont.) Maximum Likelihood Calculation Microprogram of the likelihood calculation. A possible implementation of this device is shown in Figure 4. Note that the input to the multiplier comes from both the external memory and the microprocessor. The output from the multiplier is provided to the PE input multiplexer and it is assumed that a delay of 2 clock cycles is involved in this operation. This hardware multiplier is used in the microprogram for the maximum likelihood calculation and has been implemented in the MACS simulator.

Secondly, in order to support the exponential and logarithm calculations required by the maximum likelihood calculation, the working memory is used as a set of two lookup tables. The value to be used in the calculation of the exponential or logarithm is placed on the working memory bus and one clock cycle later, the required "looked-up" value is provided on this same bus. In order to do this, the working memory must be sufficiently intelligent to use the first value as an address and simply access a word for output on the bus. Such intelligence has been assumed and incorporated into the MACS simulator. It is absolutely necessary in order to make the maximum likelihood calculation feasible.

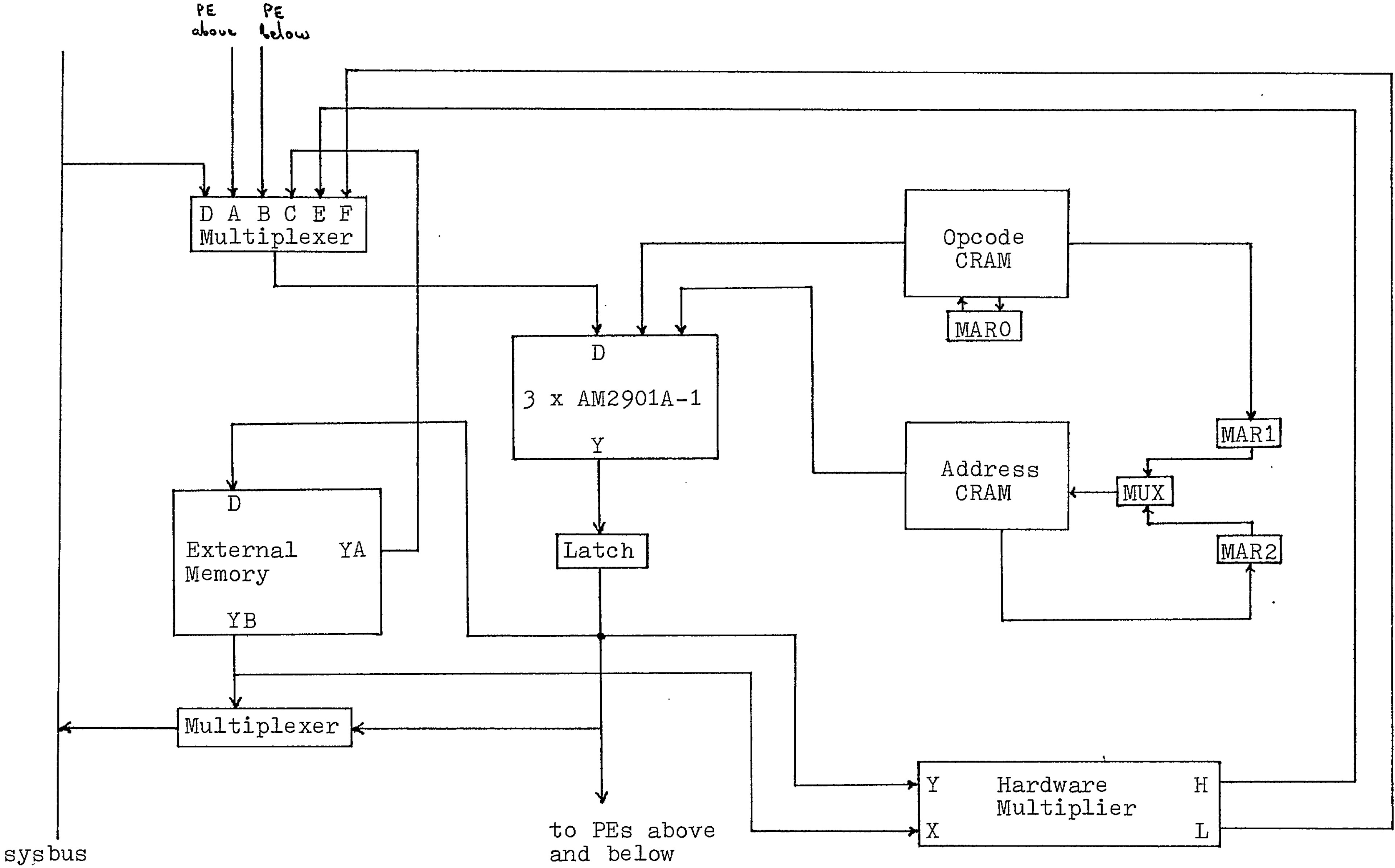


Fig.4. MACS Processing Element with a Hardware Multiplier

4. FFT PROCESSING

4.1 Processing Problem

As noted in the interim report, the philosophy followed here was considerably influenced by the research of Groginsky and Works [4]. The basic computation which must be performed is:

$$x' = x \pm yW^z$$

where W^z = a complex constant

and x,y,x' = complex variables.

In addition to a simple mechanism for calculating this function, it is necessary to move data around, basically to select where x and y are to come from in the memory and to determine where x′ is to be stored in this memory. Various interconnection schemes can be employed and the approach followed in [4] is to employ shift registers instead of an interconnection network.

The PE array is most amenable to exploitation of such shift registers since data movement from one PE to the next is similar in concept and because of the implicit shifting capability within the PE. A

PE can be used to provide a shifter and if more than 16 words are to be "shifted", more than one such PE can be employed. Figure 5 describes this concept.

In order to simply the analysis of the possible use of a PE array for FFT processing, it was initially assumed that only read computations were involved. That is, x, y, x′ and W^z were real values and only real addition and real multiplication were used.

4.2 Implementation

Figure 6 shows a 15 microinstruction routine. Significantly, in order for a number of PEs to perform FFT processing, each PE performs one stage of the FFT, and each PE uses exactly the same microroutine. The only difference lies in the values of a set of three counters, which are stored in external memories. By changing these values, the number of points can be increased or decreased.

This microprogram makes use of the fact that 16 words of internal memory can be used to provide a shifter. If more words are required due to the size of the computation, additional PEs are used simply for storage and shifting capability without performing any computation.

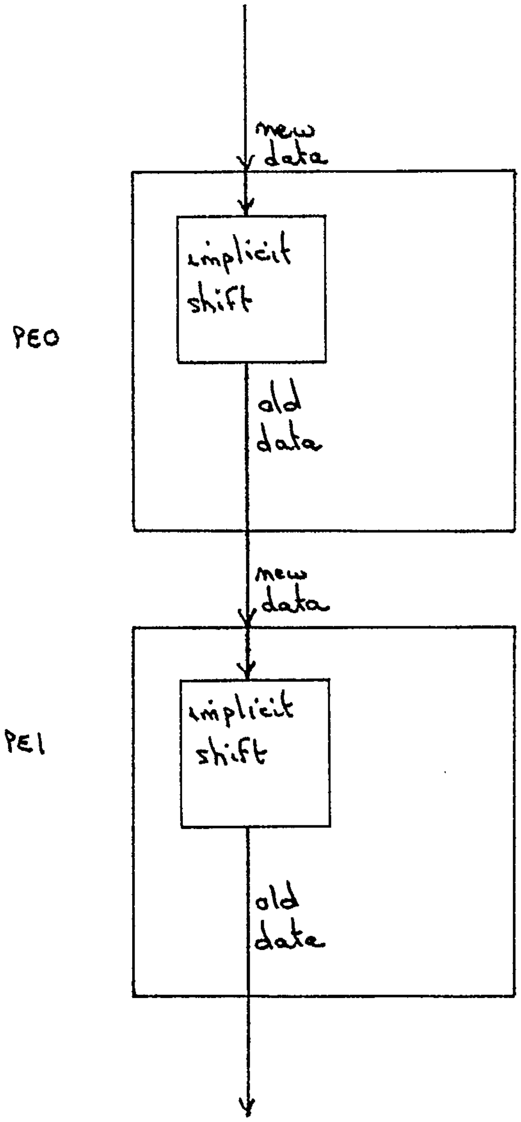


Fig.5. Shifting Capability of the PE Array

```
7016 102031560¦¦     1;CCOUNT=CCOUNT+1;DISABLE PE IF NEGATIVE
7310 144021160¦¦   422;INPUT ROTATION VECTOR FROM SYS.BUS TO E.M.
7010 106025560¦¦  1043;RESET CCOUNT
7311 110014170¦¦  1464;EMDATA TO PE BELOW (OUTPUT)
7300  120  170¦¦  2105;PE ABOVE TO Q REGISTER (INPUT)
3310  140  760¦¦  2526;IM TO EMDATA
7016 1160 4360¦¦  3147;ACOUNT=ACOUNT+1,DISABLE IF NEGATIVE
7310 1200 1171¦¦  3560;MULTIPLY ROTATION VECTOR WITH Q REG.
2310  220  170¦¦     0;NO-OP
5010 2240  760¦¦     0;ADD IM WITH RESULT,STORE IN EMDATA
5100 2260  170¦¦     0;SUBTRACT RESULT FROM IM, STORE IN Q
7016 130010560¦¦     0;BCOUNT=BCOUNT+1,DISABLE IF NEGATIVE
7010 1320  360¦¦     0;RESET ACOUNT
7010 134024560¦¦     0;RESET BCOUNT
2331    0  130¦¦     0;ENABLE PE/Q TO IM/ADDR RAM LOOP BACK
   0    0    0¦¦     0;
```

Fig.6. FFT Processing Microprogram

An important fact is that in order to perform the necessary computations, the rotation vector elements W^z must be calculated and made available to each PE. A potential data transmission problem arises. Figure 7 shows for a 16 point FFT, the rotation vector values to be used by each stage (PE) and also notes when the new rotation vector values need to be made available to the FFT stage. It is significant that for this implementation, there are no "clashes" and the working memory can be used to place this information on the system bus - one new value being placed on the bus for one specific "slot" of one microcycle.

4.3 Evaluation

The results of this study showed the potential for using an architecture of this form in order to process an FFT with the possibility of supporting a high input data rate - one point per microsecond, for example - and matching this with a high output data rate. The pipeline nature of the implementation allows the next block of data to be introduced into the PE array after the current block without any delay or interference and with the same processing rate maintained. The PE array operation need only be stopped if the

Stage 1	Stage 2	Stage 3	Stage 4	Data Required
W_0				W_0
W_0				
W_0				
W_0				
W_0	W_0			W_0
W_0	W_0			
W_0	W_0	W_0		W_0
W_0	W_0	W_0	W_0	W_0
			W_4	W_4
		W_4		W_4
		W_4	W_2	W_2
	W_4			W_4
	W_4		W_6	W_6
	W_4	W_2		W_2
	W_4	W_2	W_1	W_1
W_0				W_0
W_0			W_5	W_5
W_0		W_6		W_6
W_0		W_6	W_3	W_3
W_0	W_0			W_0
W_0	W_0		W_7	W_7
W_0	W_0	W_0		W_0
W_0	W_0	W_0	W_0	W_0

Fig.7. FFT Processing Rotation Vector Requirements

number of points to be processed must be changed. If this is necessary, the PE microprograms need not be changed but only the values of three counters stored in external memory of each PE.

The major problem is the data storing and computational power of the PEs. Only real data has been considered. In moving across to the complex data case, two problems arise.

(1) The amount of data to be transmitted and stored in each PE doubles.

(2) The number of operations involved in addition doubles and more than quadruples for multiplications.

After studying the implications of these requirements, two possible developments are suggested:

(1) The PE control memory size can be increased. There is no way that the FFT microprogram can be expanded to cope with complex calculations without requiring a microcycle of more than 16 microinstructions. The data rate is considerably reduced with this approach.

(2) The PE data manipulation facilities can be doubled so that the PE array becomes a "complex data" machine. This is an attractive alternative due to the fact that microprocessor slices are being used at present and what is required is only an increase in the number of such slices. Figure 8 shows a possible structure where the only interaction between "real" and "imaginary" streams is in the complex multiplier. This design requires no change in PE microprograms and would provide the capability of handling two parallel real streams for even higher data rates of real computations.

A major limitation which drastically increases the number of PEs required for large FFT calculations is the small size of the internal memory. Another suggested improvement is to increase the size of the PE external memory and provide the implicit shifting capability for addressing the external memory be using appropriate fields in the address section of the control memory. More storage than is available with the 2901 architecture is required and this seems one of the ways that this can be achieved.

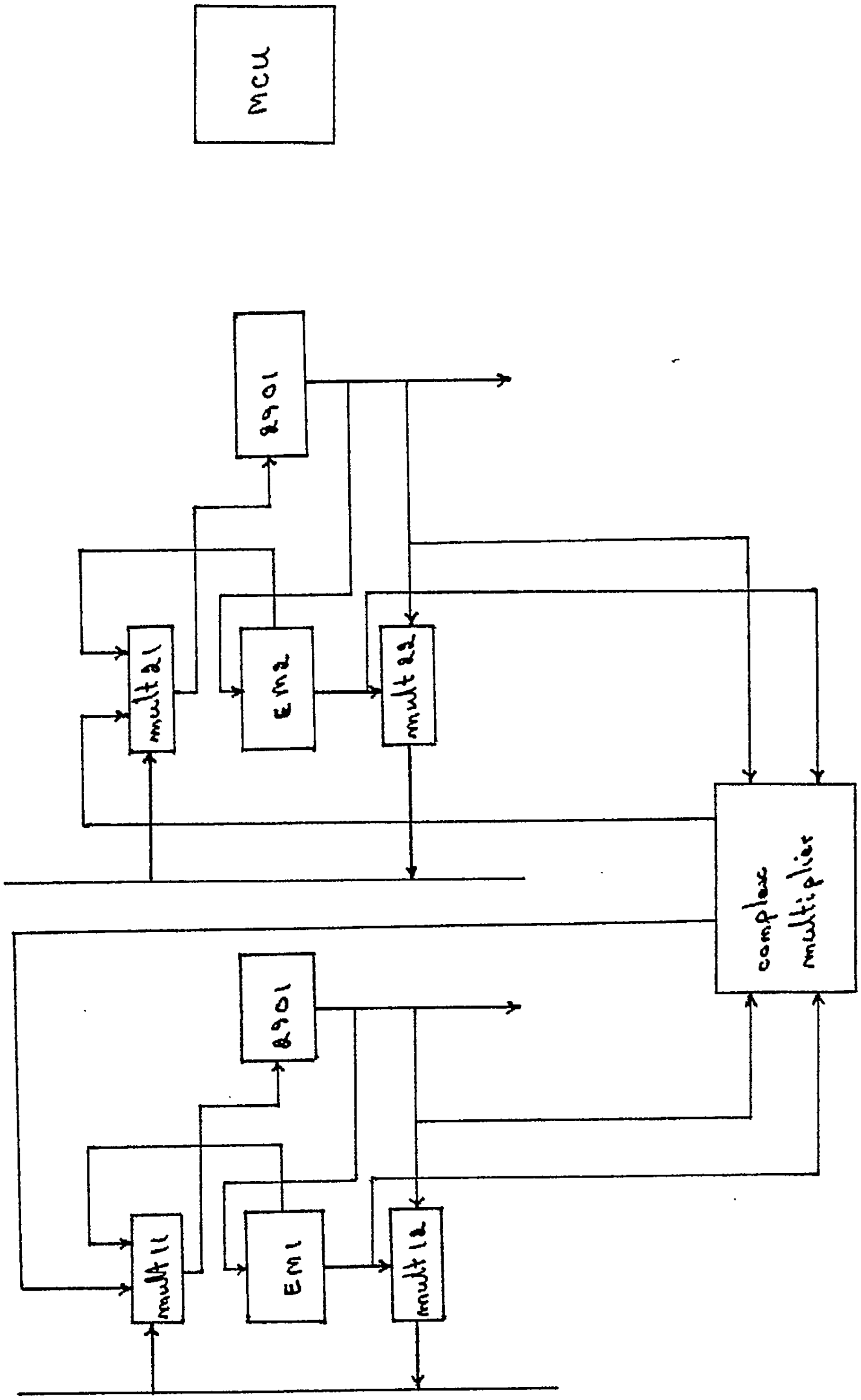


Fig.8. Complex Data Processing Element Structure

5. PARTIAL PLOT CORRELATION

5.1 Processing Problem

A calculation based on the centre of gravity was provided in [5]. This was a quite straightforward problem.

5.2 Implementation

The major requirements for this problem are to perform a series of multiplications and follow this with a division by $\sum_{i=1}^{k} S_i$. The approach proposed in the interim report was examined and discarded in favour of the algorithm given in Figure 9. An example of the required microcode is given in Figure 10.

5.3 Evaluation

This implementation showed the feasibility of performing a division by the "software method" similar to the multiplication routine employed for a two-pole filter. There is no necessity to speed up this division due to the length of time calculating $\sum_{i=1}^{k} S_i r_i$ and $\sum_{i=1}^{k} S_i \theta_i$ and the time necessary to accumulate $\sum_{i=1}^{k} S_i$.

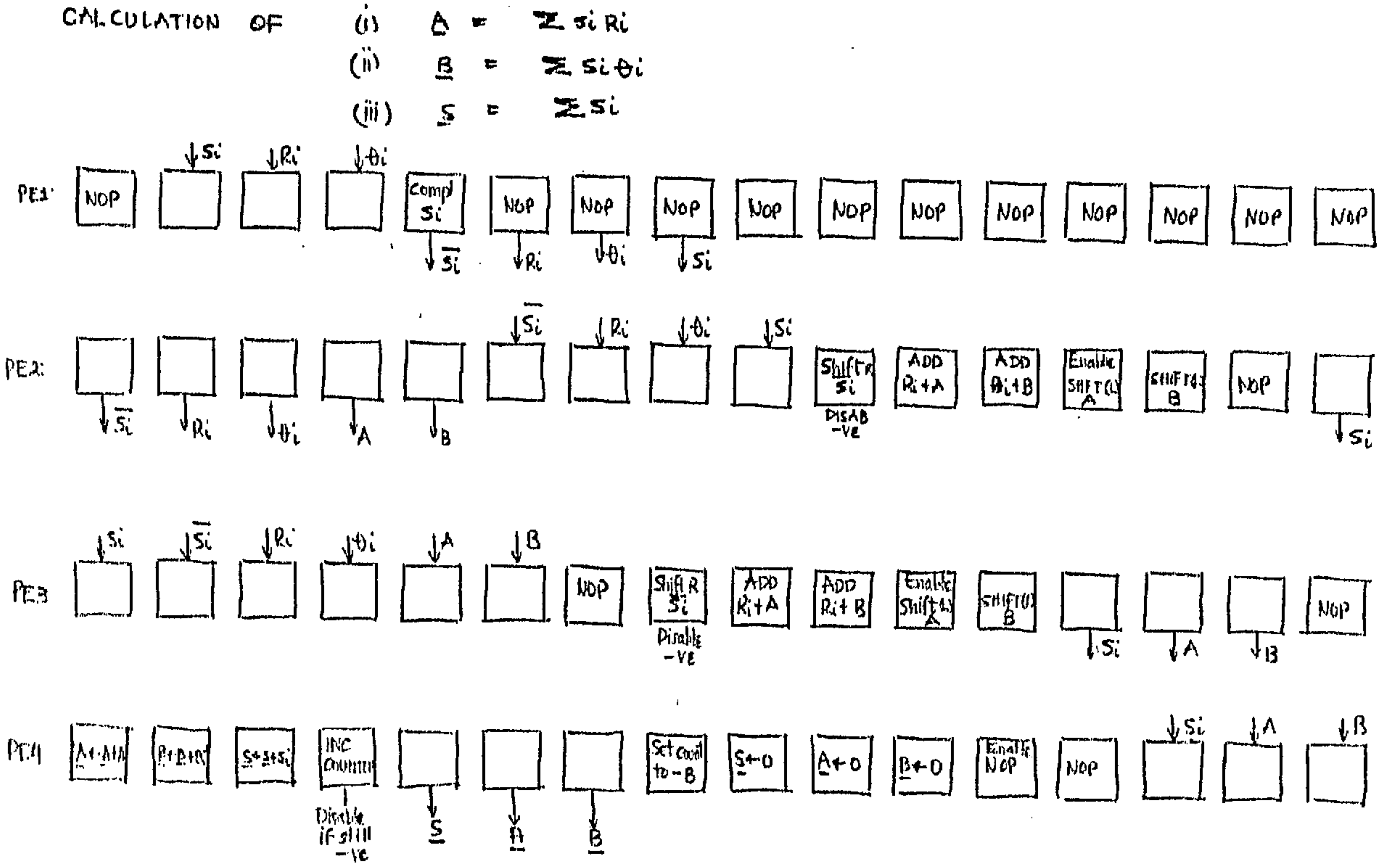
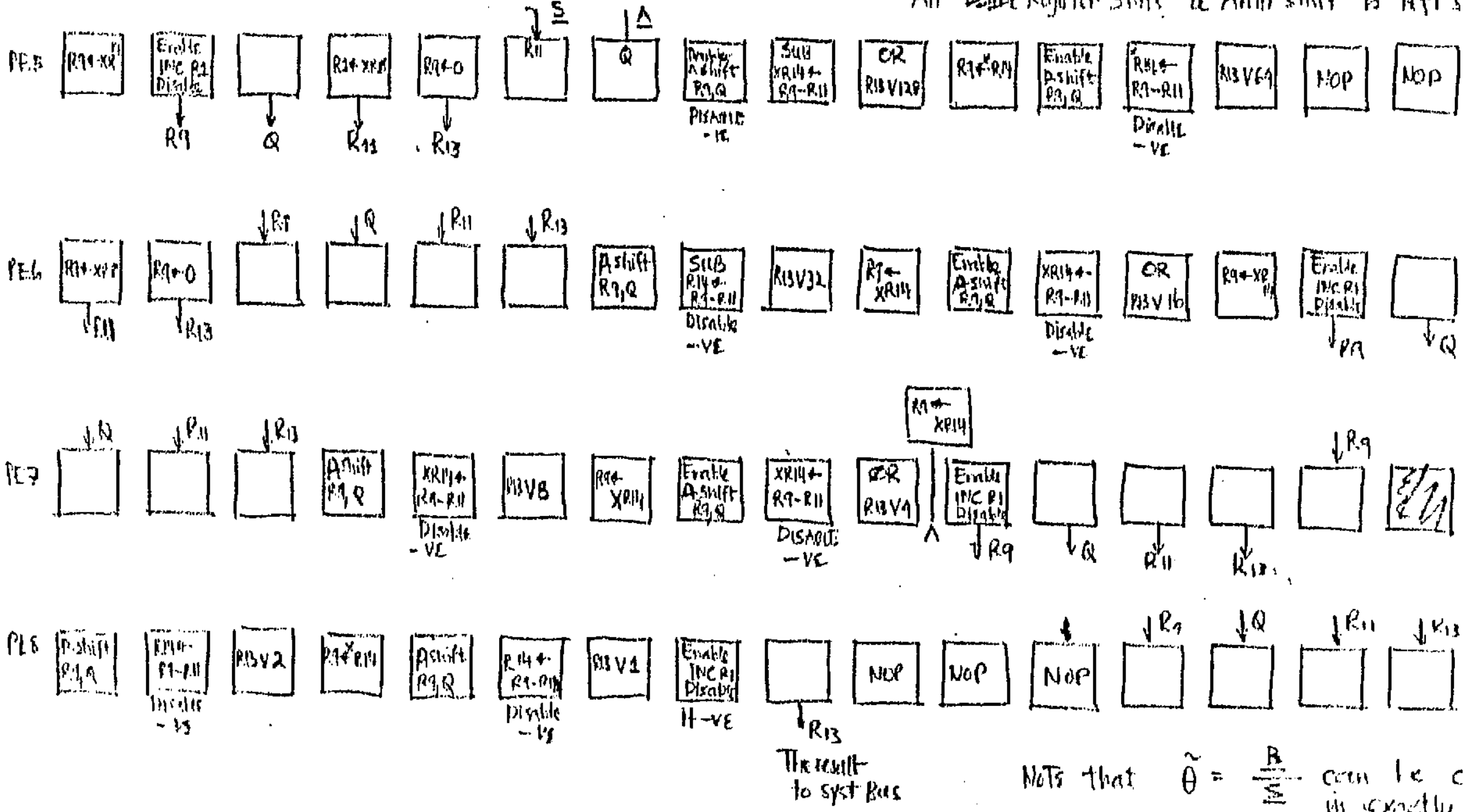


Fig.9. Partial Plot Correlation Microprogram (block diagram)

CALCULATION OF $\tilde{R} = \frac{\Sigma S_i R_i}{\Sigma S_i} = \frac{\underline{A}}{\underline{S}}$

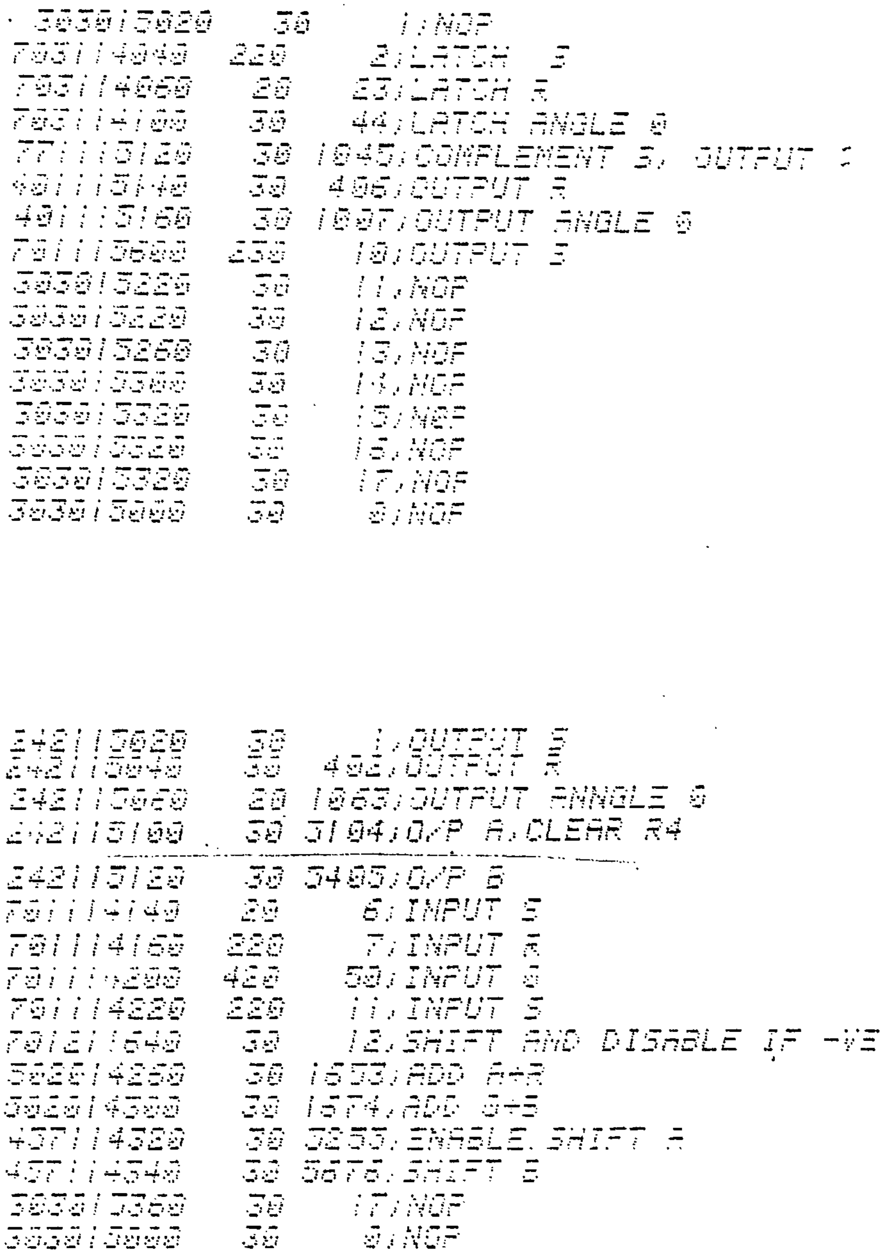
- RB is the Result Register
- R1 is a counter — initially (−8)
- All ~~Double~~ Register shift ie Arith shift is left shift.

DETAILED BLOCK DIAGRAM



NOTE that $\tilde{\theta} = \frac{\underline{B}}{\underline{S}}$ can be calculated in exactly the same procedure.

Fig.9. Partial Plot Correlation Microprogram (block diagram)



```
303015020    30      1;NOP
703114040   220      2;LATCH  B
703114060    20     23;LATCH A
703114100    30     44;LATCH ANGLE θ
771115120    30   1045;COMPLEMENT B, OUTPUT C
401115140    30    406;OUTPUT A
401115160    30   1007;OUTPUT ANGLE θ
701115600   250     10;OUTPUT B
303015220    30     11,NOP
303015220    30     12,NOP
303015260    30     13,NOP
303015300    30     14,NOP
303015320    30     15;NOP
303015320    30     16,NOP
303015320    30     17,NOP
303015000    30      0;NOP
```

```
242115020    30      1,OUTPUT B
242115040    30    402,OUTPUT A
242115060    20   1063;OUTPUT ANNGLE θ
242115100    30   3104;O/P A,CLEAR R4
242115120    30   3405;O/P B
701114140    20      6;INPUT B
701114160   220      7;INPUT A
701114200   420     50;INPUT θ
701114220   220     11,INPUT B
701211640    30     12,SHIFT AND DISABLE IF -VE
502614260    30   1653;ADD A+B
302014300    30   1674,ADD θ+B
437114320    30   3255,ENABLE,SHIFT A
437114340    30   5676,SHIFT B
303015360    30     17;NOP
303015000    30      0;NOP
```

Fig.10. Partial Plot Correlation Microprogram (extract)

The values S_i are presumed to be binary (0 or 1) or to be small in magnitude - 0 to 15, say.

6. MTI PROCESSING

6.1 Processing Problem

The procedure described in [6] was noted. Here, a radar arithmetic processing element is described which is specifically designed to handle both a second order filter and an FFT butterfly.

The required second order filter can be given by the following calculations:

$$Y(k) = X(k) + \alpha_1 W_1(k) + \alpha_2 W_2(k)$$

$$W_0(k) = X(k) + \beta_1 W_1(k) + \beta_2 W_2(k)$$

where $X(k)$ = the input value

$Y(k)$ = the output value

$\alpha_1, \alpha_2, \beta_1, \beta_2$ = constant coefficients

and $W_0(k), W_1(k), W_2(k)$ = intermediate values

After each such calculation, the values of $W_1(k+1)$ and $W_2(k+1)$ are generated as follows:

$$W_1(k+1) = W_0(k)$$

$$W_2(k+1) = W_1(k) = W_0(k-1)$$

Use of "intelligent" input, output and working memory buffers are required in order to provide X(k), $W_1(k)$ and $W_2(k)$ for these calculations. The X(k) values are provided by the input buffer and the $W_1(k)$ and $W_2(k)$ values are provided by the working memory buffer.

6.2 Implementation

This implementation is again very straightforward and makes use of the hardware multiplier. The working memory buffer does not have to "juggle" the intermediate values $W_1(k)$ and $W_2(k)$. They are simply output on the system bus and replaced by new values generated by the PE array and placed on the system bus. Figure 11 provides a block description of the required processing.

6.3 Evaluation

There are many ways in which this sort of problem can be microcoded. Indeed, it is easy to see how this sort of microprogram could be expanded to support more complicated filters. The PE array seems admirably suited to such computational problems.

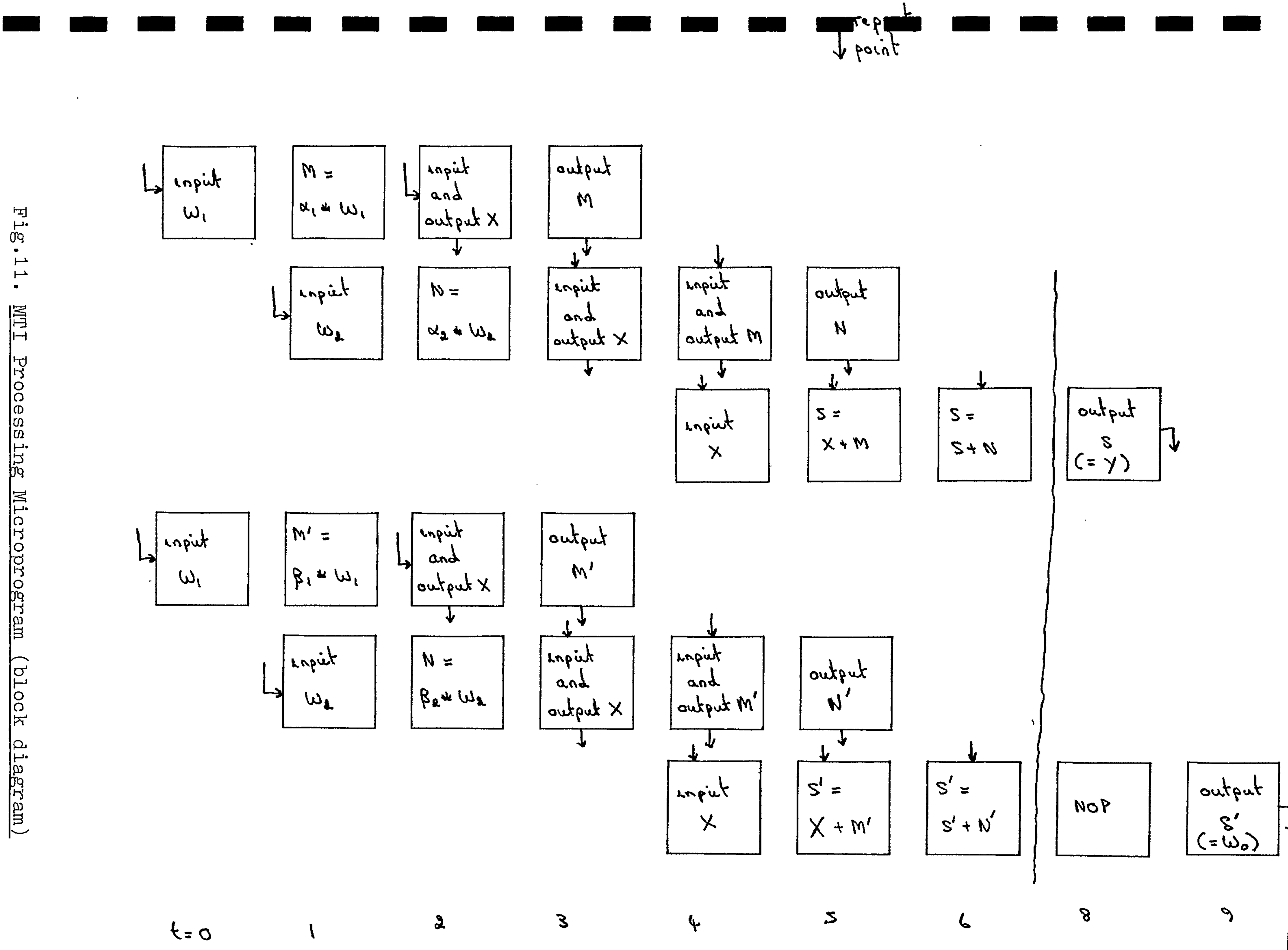


Fig.11. MTI Processing Microprogram (block diagram)

7. CONCLUSION

From this research, it seems that the MACS system meets many of the requirements of digital signal processors. With suitable modifications, it can support the two major activities of all digital processing, namely the FFT and digital filter processing.

A number of deficiencies have become apparent in this design. There is a basic limitation imposed by using the 2901 microprocessor architecture although there are no suitable alternatives. This microprocessor does not provide a fast means of performing multiplication and the amount of storage available is quite limited. Many signal processing applications require frequent multiplications and more storage than is available here.

A number of recommendations have come from a study of these chosen applications:

(1) The implementation of a hardware multiplier.

(2) The implementation of a hybrid multiplier which reduces the amount of software required by introducing some possible manipulation of the operation codes.

(3) A variation of the data structure to allow complex operations on complex data.

(4) An increase in the size of the external memory to 64 words, with a simple addressing mechanism to allow implicit shifting. (This mechanism need only be a counter which could be selected for use in addressing either of the two ports. This would be in addition to the present scheme).

(5) A lookup table command for use by the working memory.

(6) A double-length shift operation making use of the Q register and any one of the 16 internal registers.

(7) Greater flexibility with respect to test and disable operations.

This final point is not so important for numeric computation but seems important for more spatial and logical processing involved in such applications as image processing. Mr. F. Gougeon will be studying the applicability of the MACS processor to image processing in the following months and will note further recommended changes.

The conditional disabling could be modified in three different ways:

(1) More conditions could be made available for testing. A four-bit field could determine what conditions could be tested.

(2) Disabling and enabling of a PE could be associated with an "activity bit" within the PEs and these activity bits could be stacked to allow more sophisticated multiway decisions to be made.

(3) Such activity bits could be toggled, switching "off" one set of PEs whilst another was switched "on".

One area which seems fruitful for further research is in the area of complier design. This PE structure is quite unique with a number of major constraints such as the amount of processing allowable to a PE and the amount of data available for this processing. There sometimes is considerable difficulty determining how to devise an algorithm and how to "fit it into the PE array". Some method of describing the data dependencies of a problem with this structure particularly in mind could lead to software which could greatly assist the microprogrammer. A clear description may possibly lead to the feasibility of a compiler specifically designed for generating microcode for this PE array.

Acknowledgements

I would like to thank Francois Gougeon who was my Research Associate throughout the term of this research contract. He both designed and programmed the MACS simulator and made the various modifications that were necessary as this research progressed. In addition, he provided the microprograms for the plot-to-track correlation problem and suggested various modifications that were necessary to accommodate these microprograms. The material in Section 2 of this Final Report is drawn from the MACS simulator user manual which he has written.

I would also like to thank Hon Wong Wang, Rama Mwikalo and P. Rajani Kanth, who were Research Assistants on this project. Hon Wong Wang provided much of the input and the microprogram for the FFT processing algorithm. Rama Mwikalo provided the work on the partial plot correlation algorithm and microprogram and P. Rajani Kanth provided some test microprograms for the MACS simulator and also investigated the Kalman filter processing requirements.

References

(in order of citation)

(1) Research on Microprocessor Arrays for Signal Processing at High Data Rates. C.V.W. Armstrong, Research Report No.2, MACS Project, November 15, 1978, Supply and Services Canada, Contract No. OSU-00099, Interim Report.

(2) MACS Simulator User Manual. F. Gougeon, Research Report No.4, MACS Project, March 31, 1979, Supply and Services Canada, Contract No. OSU-78-00099.

(3) An Optimal Data Association Problem in Surveillance Theory. R.W. Sittler, IEEE Trans. Military Electronics, April 1964, pp.125-139.

(4) A Pipeline Fast Fourier Transform. H.L. Groginsky and G.A. Works, IEEE Trans. Computers, Vol.c-19, No.11, pp.1015-1019, 1970.

(5) On the Generation of Plots. G. Painchaud, C.R.C. Technical Memorandum, July 18, 1978.

(6) The Radar Arithmetic Processing Element as an MTI Filter. B.P. Shay, from "Digital Signal Computers and Processors" edited by A.C. Salazar, IEEE Press, New York, N.Y., 1977, pp.310-317.