NATIONAL RADIO ASTRONOMY OBSERVATORY
Green Bank, West Virginia

SPECTRAL PROCESSOR MEMO NO. 24

# MORE DETAILED SPECTRAL PROCESSOR DESIGN

J. Richard Fisher

January 5, 1985

NATIONAL RADIO ASTRONOMY OBSERVATORY
GREEN BANK, WEST VIRGINIA

# MORE DETAILED SPECTRAL PROCESSOR DESIGN

J. Richard Fisher

This note is a sequel to Spectral Processor Memo No. 22 and deals with the next level of detail in the design. The topics included here are the same as the ones covered in the meeting of December 18 with a couple of additional thoughts on the fast adder/memory section and the Faraday rotation corrector.

After the overall block diagram in Figure 1 appeared in Memo No. 22 a number of changes have been made to the specifications and design. The most significant is the request for more IF inputs and hence more output spectra. Observers of the 18 cm OH lines would like to have 8 inputs (4 lines x 2 polarizations), and the low frequency recombination line people would like to have as many as 16 inputs. We probably cannot afford to build more than 4 IF drawers before 1987 since each costs about $5000, but I do propose that we incorporate 8 A/D converters and enough switching electronics in the input buffers to accept a total of 16 A/D converters. There was a little bit of discussion at the meeting about the feasibility of multiplexing 8 A/D's per FFT section, and a final decision about 8 or 16 digital inputs will have to wait for more information on the electronics. Adding more input channels adds only to the number of addressing sequences which must be accommodated at the buffer and accumulator stages. The maximum number of frequency channels per FFT section is still 1024.

One minor change in Figure 1 is to move the FFT input buffer from behind to ahead of the window multipliers. This saves memory by storing 6-bit A/D words instead of 16-bit multiplier output words. Also, because of budget and time pressures the Faraday rotation corrector section (lower left 3 boxes of Figure 1) will not be included in the spectral processor when it first goes into operation. Some design will have to be done on this section before major construction starts to allow it to be easily incorporated later.

Be aware that several different FFT configurations (decimation in time or frequency and bit-reversed input or output) have been implied in various memos and discussions. We have yet to see any advantage to using one over the others so we have not yet been compelled to chose a specific configuration. Memo No. 22 assumed D.I.F., and Ron Weimer's discussion of the butterfly assumed D.I.T. (Memo No. 23). This memo continues the D.I.T. approach and makes the additional assumption of bit-reversed input.

Relatively minor modifications to the design thus far would be necessary to adopt a different combination.

## Input Buffers

Figure 2 shows the data order expected by the input of an 8-point FFT (top diagram). Each datum is a complex number composed of two successive samples of the A/D which fill the real and imaginary components. Each butterfly has two complex inputs (4 A/D samples) which have been labeled A and B. Note that with a bit-reversed input all of the A inputs are filled by the A/D before any of the B inputs are filled. If the FFT is broken in half (2 A/D inputs) the A inputs for both halves would be filled in parallel, then the B inputs would be filled in parallel.

Figure 3 shows a possible sample routing network and buffer memory configuration between the A/D's and the FFT input for 4 inputs to the FFT. The triangles represent tristate gates which are controlled through their third port. Each line to the FFT butterfly is double buffered so that the data can be read out in a completely different order from which it was written to the buffer memory. One of the memories of each pair is read while the other is being written to. The FFT input needs four integer values at once, but these values are put into the buffers one at a time. Delays may have to be introduced into the buffer input lines to synchronize the switching of the double buffers.

Any of the inputs can be connected to any of the memories, and the switching sequences for 1, 2 or 4 inputs are shown in Figures 4-6. Eight inputs could be handled by doubling the number of input gates and implementing the proper switching sequence. If we assume that the buffer memory locations are always read sequentially the writing order will depend on the number of input channels. Note that the data rate into each memory is independent of the number of inputs because the cumulative bandwidth of all channels cannot exceed 40 MHz (80 million samples/second). Note, also, that each gate represents 6 data lines. Each of the 8 memory buffer sizes is 512 x 6 bits.

## Window Multipliers

Before entering the FFT section the data can be weighted by a window or taper function. Figure 7 shows the multiplier array needed for this windowing. No additional buffering is required because each set of four data is multiplied as needed by the FFT, but the window coefficients must be accessed in an order identical to the order in which the data came from the A/D converters. More than one input will require that the coefficients be cycled through more than once for each run of FFT inputs. Because the window samples must be symmetrical about the center of the window, two sets of window samples must be stored; one for a single input channel and another for the even number of input channels. A selection of two or three window functions will be available. Whether the window function is changed by relaoding RAM or selecting a different set of ROM will depend on hardware expediency. The window coefficients and multiplier outputs will be 16 bits wide, and each of the four window coefficient memories will contain about 1024 x 16 bits for each window function.

## Correction for Real Inputs

Because the data from the A/D converters are real and not complex numbers as expected by the FFT, the output of the FFT is not a finished voltage spectrum. The result of loading successive real samples into real and imaginary components of the complex inputs is that the information for a particular frequency channel is split between its proper position and its mirror image position (e.g., outputs 1 and N-1) in the FFT output. The proper vector combinations of these two outputs will yield the independent information for the corresponding frequency channels. The mathematical form and functional implementation of this vector combination is given in Memo No. 23.

Figure 8 shows the combining sequence for the correction of the FFT output. The correction operation is very similar to an FFT butterfly operation so the same diagram symbol is used. The order in which the last FFT stage stores data in the output buffer is shown at the left of the diagram. In the case of 1 input channel the buffer must be completely filled before any correction operations can begin; hence, the FFT output must be double buffered.

The real correction sequence assumes that the buffer contains one more complex data value than comes out of the FFT; this extra value is just a duplicate of the zeroth value in the Nth position. As a result there is one more correction operation than there are FFT butterfly operations which will cause timing problems unless we take advantage of the simplified operations necessary on the end and center points. The zeroth data value need not be duplicated in the buffer, and I think that we need not calculate the value for frequency channel N since it is well down on the skirt of the anti-aliasing filter. Channel zero is the DC channel and contains possibly useful information on A/D offsets and spectrometer errors so I suggest that we retain it.

The real component of channel zero is the sum of the real and imaginary components of the zeroth FFT output and the imaginary component of channel zero is always zero. The complex value of frequency channel N/2 is equal to the N/2 output of the FFT except that the imaginary part changes sign. Channels zero and N/2 could be generated in the same correction operation by bypassing all but one hardware adder for each channel.

Figure 9 shows the signal paths and buffer memory in a little more detail. Each data line corresponds to a 16-bit word, and each of the eight buffer memory banks contains 1024 x 16 bits. The real correction butterfly contains ten adders and four multipliers. Hidden in the butterfly box is a 512 x 16-bit table of vector rotation coefficients very similar to the rotation coefficients in the FFT stages.

## Self/Cross Multipliers

The next elements in the spectral processor signal path are the self/cross multipliers. In the spectrometer mode of operation these multipliers turn the complex voltages into power values, and in the polarimetry mode they also produce cross products which are used to generate Stokes parameters. Since the inputs of each self/cross multiplier come from only one frequency channel at a time the channels can be processed in any order, and, hence, no buffering is needed between the real corrector and the multipliers.

Figure 10 shows the four basic configurations of the spectral processor. The top three use only self multipliers and do not require any data to be shared between the two FFT sections. The higher time resolution in configuration c is obtained by reducing the number of channels processed so that the effective data rate is the same. The polarimeter mode in Figure 10d requires twice as many multiplications per frequency channels so the number of channels processed is again reduced by a factor of two to keep the output data rate within the maximum allowed by the hardware. Note that the output data notation (A,B,C,D) comes from an earlier memo and is different from that in the rest of this report. The new notation is shown at the bottom right-hand corner of Figure 11.

Figure 11 shows the multiplier and data routing in slightly more detail. In the polarimeter mode each multiplier pair is used for both self and cross products. This is done by holding the real correction output for twice as long as normal and performing the self and cross products in alternation. Only half of the FFT output buffer is processed this way, but that is the price which must be paid for producing twice as many products in the same amount of time. (Another processing sequence would be to cycle through half of the FFT output buffer twice producing all of the self products first, then all of the cross products, but when the Faraday rotation correction is implemented it needs the two sets of products for each channel simultaneously so the first method is preferable.) The maximum data rate on each input or output line is 50 ns, and each line represents 16 bits. With bandwidths narrower than 20 MHz we could handle more channels in the polarimeter mode by running the self/cross multipliers faster than the FFT butterfly multipliers.

## Faraday Rotation Corrector

Although the Faraday rotation corrector will not be included in the spectral processor when it first goes on line its design must be kept in mind.

Two methods for doing the Faraday rotation correction come to mind. The first which was introduced in Memo No. 22 would accumulate the outputs of the self/cross multipliers for a sufficient amount of time (100 μs or more) so that one hardware multiplier could perform the seven different necessary multiplications at the reduced data rate. This would require a moderate amount of buffer memory (2 x 1024 x 16-bit words) and a fair number of data routing gates and would limit the time resolution to 100 μs or longer.

The second method would avoid accumulation by processing each frequency channel as it arrives by having eight hardware multipliers to handle the full 12.8 μs time resolution with 256 frequency channels. Upon second thought it appears that the second method may not cost much more than the first and may actually be less complex, so let us outline the Faraday rotation corrector without pre-accumulation.

If we shorten the notation of the self/cross products of Figure 11 we get

$$E = A_R^2(1) + A_I^2(1) \quad \text{or} \quad B_R^2(1) + B_I^2(1)$$

$$F = A_R^2(2) + A_I^2(2) \quad \text{or} \quad B_R^2(2) + B_I^2(2)$$

$$G = A_R(1) \cdot A_R(2) + A_I(1) . A_I(2) \quad \text{or} \quad B_R(1) . B_R(2) + B_I(2) \cdot B_I(2)$$

$$H = A_R(1) \cdot A_I(2) - A_I(1) \cdot A_R(2) \quad \text{or} \quad B_R(1) \cdot B_I(2) - B_I(1) \cdot B_R(2)$$

The first correction to be made is for the difference in gain between the two linear polarization channels, then the products for each frequency channel can be combined to form Stokes parameters:

$$I = E + gF$$

$$Q = E - gF$$

$$U = \sqrt{g}G$$

$$V = \sqrt{g}H$$

where the relative gain, g, may be different for each frequency channel but the same for all four Stokes parameters. The last correction is for the relative Faraday rotation angle of the linear polarization vector in each channel. Parameters I and V are unchanged by this correction, but Q and U require two multiplications and one addition each.

$$I' = I$$

$$Q' = Q \cos 2X + U \sin 2X$$

$$U' = U \cos 2X - Q \sin 2X$$

$$V' = V$$

where X is the relative vector rotation to be performed on a particular channel. Parameters g and X are functions of channel number and need to be changed as the receiver gain varies and new sources are observed.

Figures 12 and 13 show one possible implementation of the equations given above. As in most of the system each signal line represents a 16-bit word. The tables used by the two sets of multipliers must be RAM whose contents may be easily changed by the central processor. Most of the switches shown change position every 50 ns at the highest data rate to accommodate the data multiplexing imposed by the cross multipliers. The adders at the output of Figure 13 run at half this speed so we could get away with two instead of four adders, but the additional multiplexing circuitry probably offsets the savings in adder chips.

## Fast Adder/Memory

With or without the Faraday rotation corrector the next stage in the signal paths is the fast adder/memory. The primary function of this stage is to accumulate the spectral processor output so that the data rate is slowed to a point where it can be comfortably handled by the central processing computer. The accumulation sequence must be very flexible so that a wide variety of averaging schemes such as dedispersion, multiple spectra averaging, and synchronous pulse averaging may be accomplished.

Figure 14 is a simplified schematic of half of the fast adder/memory. The basic accumulation element is an adder-memory pair. The inputs to the adder are the new data word and the current contents of the memory location to be updated. One accumulation cycle takes between 75 and 100 ns while the input data period is 50 ns so each data stream is divided between two identical accumulators whose contents may be summed at dump time, if necessary. Some averaging schemes will require separate access to the two memories in the split data stream so two selectable multiplexing cycles to the computer bus are shown.

Not shown in Figure 14 are the address generators for the accumulator memories. These will be RAM lookup tables which can be loaded from the central processor. The read and write operations of one accumulation cycle will be to the same address. Each of the four memory banks in Figure 14 will have 2048 data words with a minimum width of 24 bits. A first guess at the size of each address lookup table is 2048 x 12 bits, but there probably is a more clever combination of smaller memories.

The accumulators will provide for the rejection of contaminated data either by suppressing accumulation cycles or by dumping the bad data into a separate part of the accumulation memory. The missing data will be accounted for in a memory bank whose addressing sequence is an image of the one used in the accumulators. By reading the rejected data counters the computer can correct the amplitudes in locations with missing data.

Since the memories can be read during the add phase of an accumulation cycle it will sometimes be possible to dump the accumulators without stopping accumulation. However, some time out for dumping will be necessary when all of the memory is used for averaging at the fastest data rate or when the memory fills up faster than the computer can read it.

The data word size in the accumulator memory depends on the maximum number of accumulations expected in each location and the average signal strength at that location. Every data word into the accumulators can be positive so we must assume a linear growth of amplitude with time in the memory location.

Not all bits of the 16-bit words coming out of the self/cross multipliers need to be presented to the accumulation adder since some of the low order bits contain unneeded noise and some of the high order bits may be assumed to contain nothing or data that would have to be rejected anyway.

Figure 15 shows the relative bit positions of various levels expected in the spectral processor output words. For the time being we will not worry about which bits of the 16-bit self/cross multiplier output word these correspond to. The reference or zeroth bit in Figure 15 is the rms levels of a single spectral sample when the input to the A/D converter is set such that the quantization interval is equal to the voltage rms levels. Normally, one would not run the A/D input at such a low level because of reduced sensitivity.

The highest input level to the A/D will probably be such that there will be about seven quantization levels per voltage rms which would put the highest sample level at about 4.5 rms and properly sample a CW signal which is about 5 dB over the system noise power. The resulting noise power in the output word would be about five bits above the reference as shown by the second vertical arrow from the right in Figure 15. The third arrow shows the word size before accumulation in one of 1000 channels containing a CW signal whose strength is 10 dB above the wideband input power to the spectrometer (A/D input level set to 7 levels/rms). The fourth arrow from the right shows the width to which an accumulated word will grow in one second if the input bandwidth is 40 MHz divided among 1000 channels and the signal is pure noise sampled at 7 levels/rms. The same integration and bandwidth would send the CW signal into the 35th bit as shown by the left most arrow.

Overflow of a CW signal in the accumulators is not a particular problem for the intended uses of the spectral processor since the overflowed channels would normally be rejected anyway. For a one second integration of 1000 channels of a 40 MHz bandwidth the largest CW signal which would not overflow the 24th bit in Figure 15 would have a signal to noise ratio of about 45 dB because the ratio of noise to DC level of the integrated noise is $1.6 \times 10^{-4}$.

The minimum word size in the accumulators is 24 bits placed roughly in the positions 0-23 in Figure 15. If this could be extended to 28 bits with little more than the cost of the extra memory, the reduced processing load on the central processor by allowing dump periods much greater than one second may be worth the expense. Some more thought and experimentation should be done on the word size problem before making a final decision. At first thought it appeared that shifting the accumulator input word to favor either more noise in the low order bits or more room for large signals would be a useful option, but roughly the same effect can be realized by changing the input level to the A/D converter so this option is probably not worth the effort.

Now 8 or 16 inputs

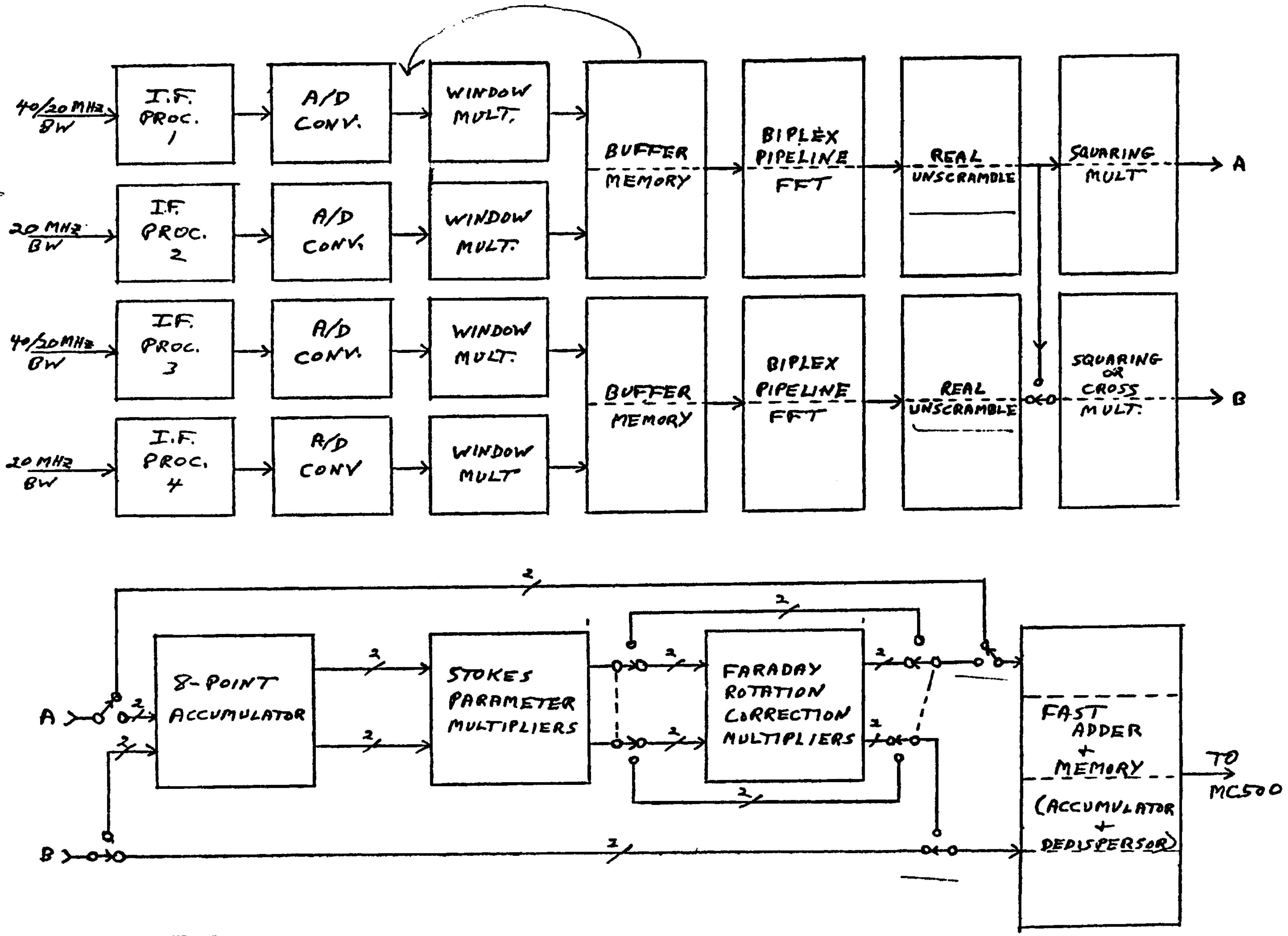


Figure 1

$\xrightarrow{2}$ ≡ 2 quantities/channel

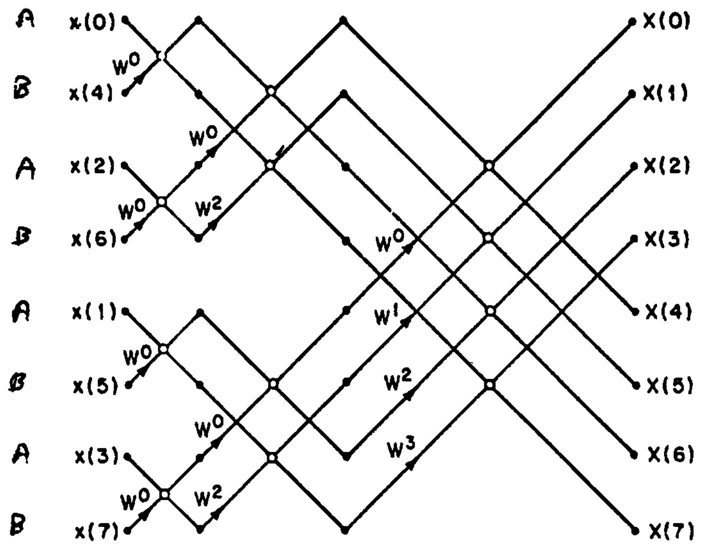


Fig. 6.3 Eight-point FFT obtained by successive splitting into two's.

described later in this section. First, however, it is both interesting and informative to discuss some of the general properties of methods for performing fast Fourier transforms.

The basic operation of the decimation-in-time algorithm is the so-called butterfly in which two inputs $A$ and $B$ are combined to give the two outputs $X$ and $Y$ via the operation

$$\begin{aligned} X &= A + W_N^k B \\ Y &= A - W_N^k B \end{aligned} \qquad (6.14)$$

Figure 6.4 shows a flow graph for the butterfly of Eq. (6.14). Careful examination of the flow graph of Fig. 6.3 shows that at each stage there are ($N/2$) butterflies. For the case where $W_N^k$ is a nontrivial multiplier, only one multiplication is required per butterfly since the quantity $B \cdot W_N^k$ can be computed and saved. The importance of the butterfly structure to the FFT flow graph is that only one additional memory location is required to transform an $N$-point sequence stored in memory. Thus one can compute

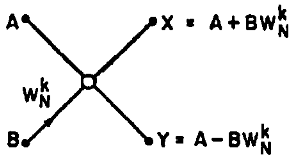
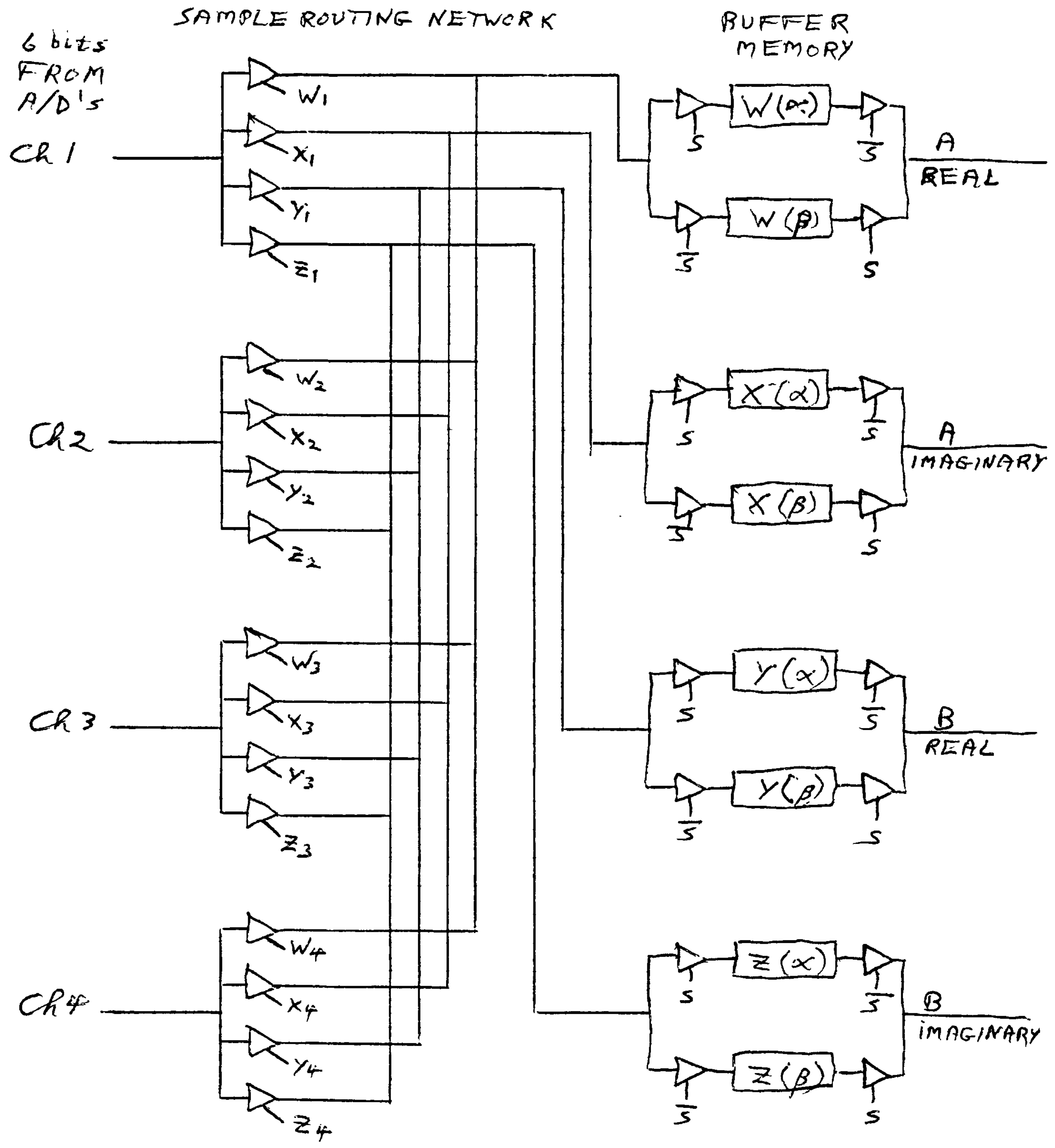


Fig. 6.4 FFT Butterfly.

Figure 2



$S$ + $\bar{S}$ exchange states after each full window interval to allow samples to be passed to FFT while collecting a new sample set

Figure 3

1 CHANNEL (40MHz B.W.)

25m

| SAMPLE | $W_1$ | $X_1$ | $Y_1$ | $Z_1$ | $W_2$ | $X_2$ | $Y_2$ | $Z_2$ | $W_3$ | $X_3$ | $Y_3$ | $Z_3$ | $W_4$ | $X_4$ | $Y_4$ | $Z_4$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | INPUTS | | | | | | | | | | | | | | | |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ⋮ | | | | | | | | | | | | | | | | |
| B | INPUTS | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ⋮ | | | | | | | | | | | | | | | | |

Figure 4.

2 CHANNELS (20MHz EACH)

| SAMPLE | $W_1$ | $X_1$ | $Y_1$ | $Z_1$ | $W_2$ | $X_2$ | $Y_2$ | $Z_2$ | $W_3$ | $X_3$ | $Y_3$ | $Z_3$ | $W_4$ | $X_4$ | $Y_4$ | $Z_4$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A INPUTS | | | | | | | | | | | | | | | | |
| 25 n [0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | (1) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ⋮ | | | | | | | | | | | | | | | | |
| B INPUTS | | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 1 | 0 | 0 | (1) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ⋮ | | | | | | | | | | | | | | | | |

Figure 5

4 CHANNELS (10 MHZ EACH)

50 ns

| SAMPLE | $W_1$ | $X_1$ | $Y_1$ | $Z_1$ | $W_2$ | $X_2$ | $Y_2$ | $Z_2$ | $W_3$ | $X_3$ | $Y_3$ | $Z_3$ | $W_4$ | $X_4$ | $Y_4$ | $Z_4$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A INPUTS | | | | | | | | | | | | | | | | |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | (1) | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | (1) |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 4 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ⋮ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| B INPUTS | | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | (1) | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | (1) | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 3 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 5 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| ⋮ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

Figure 6

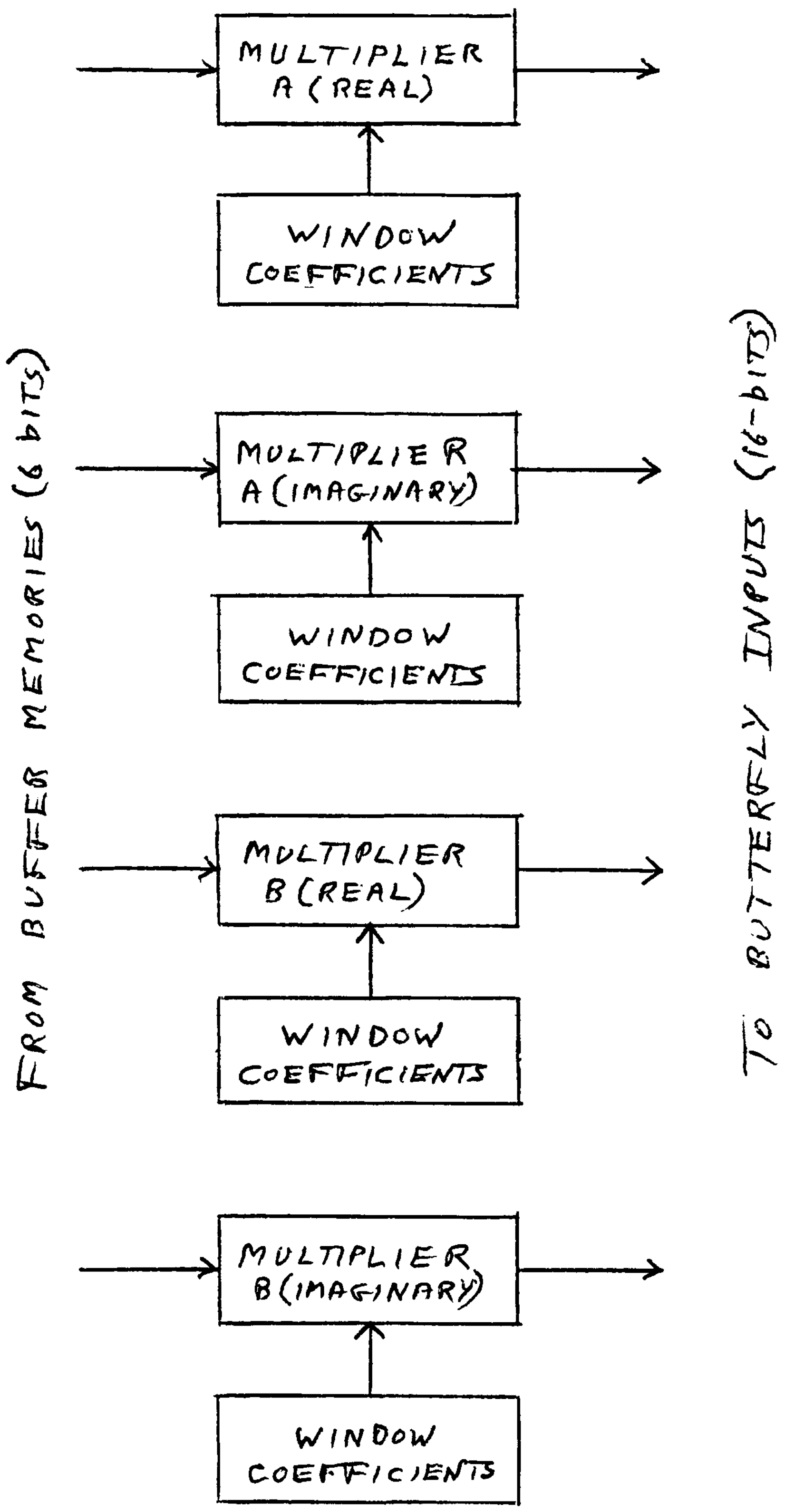
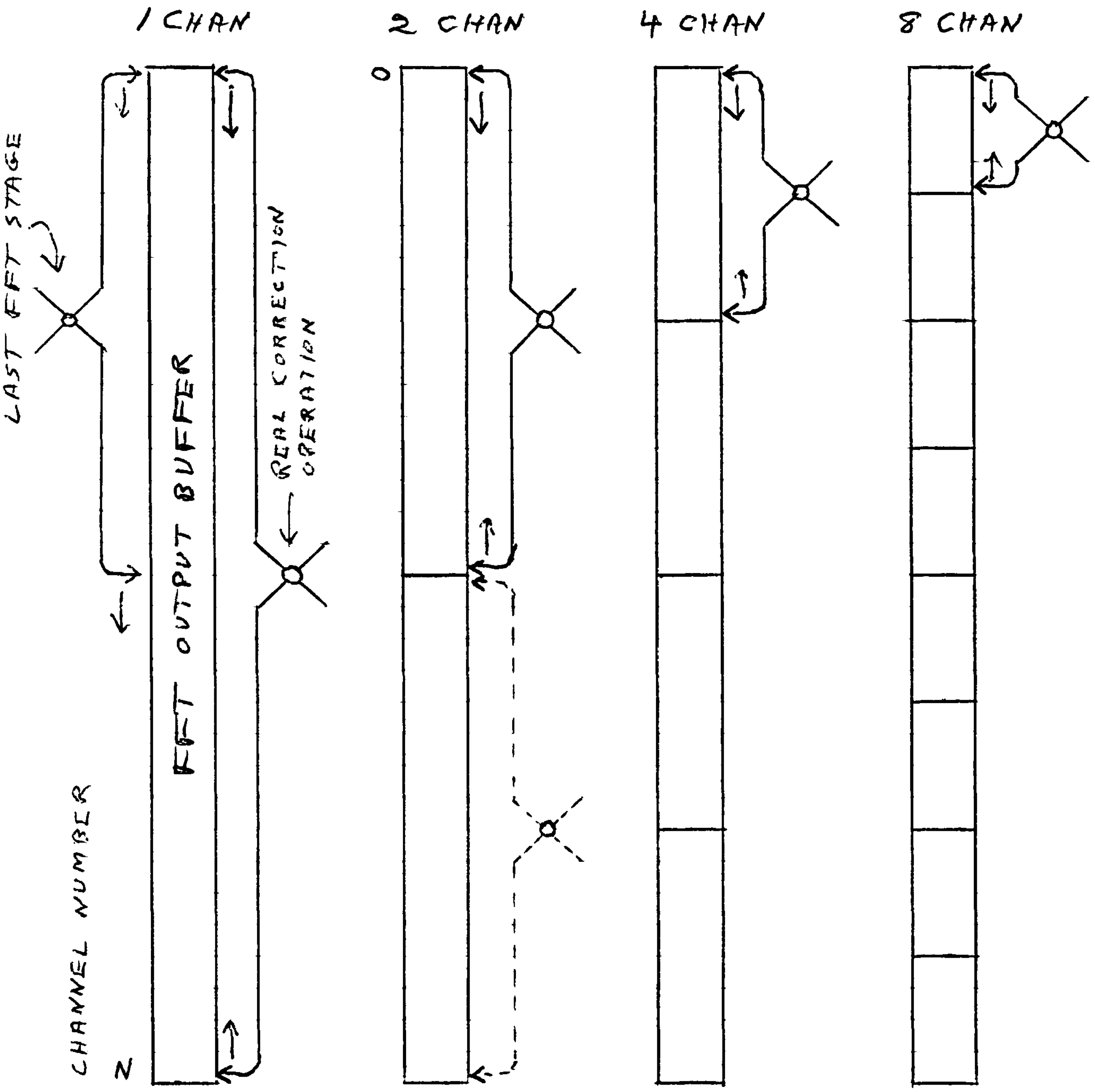
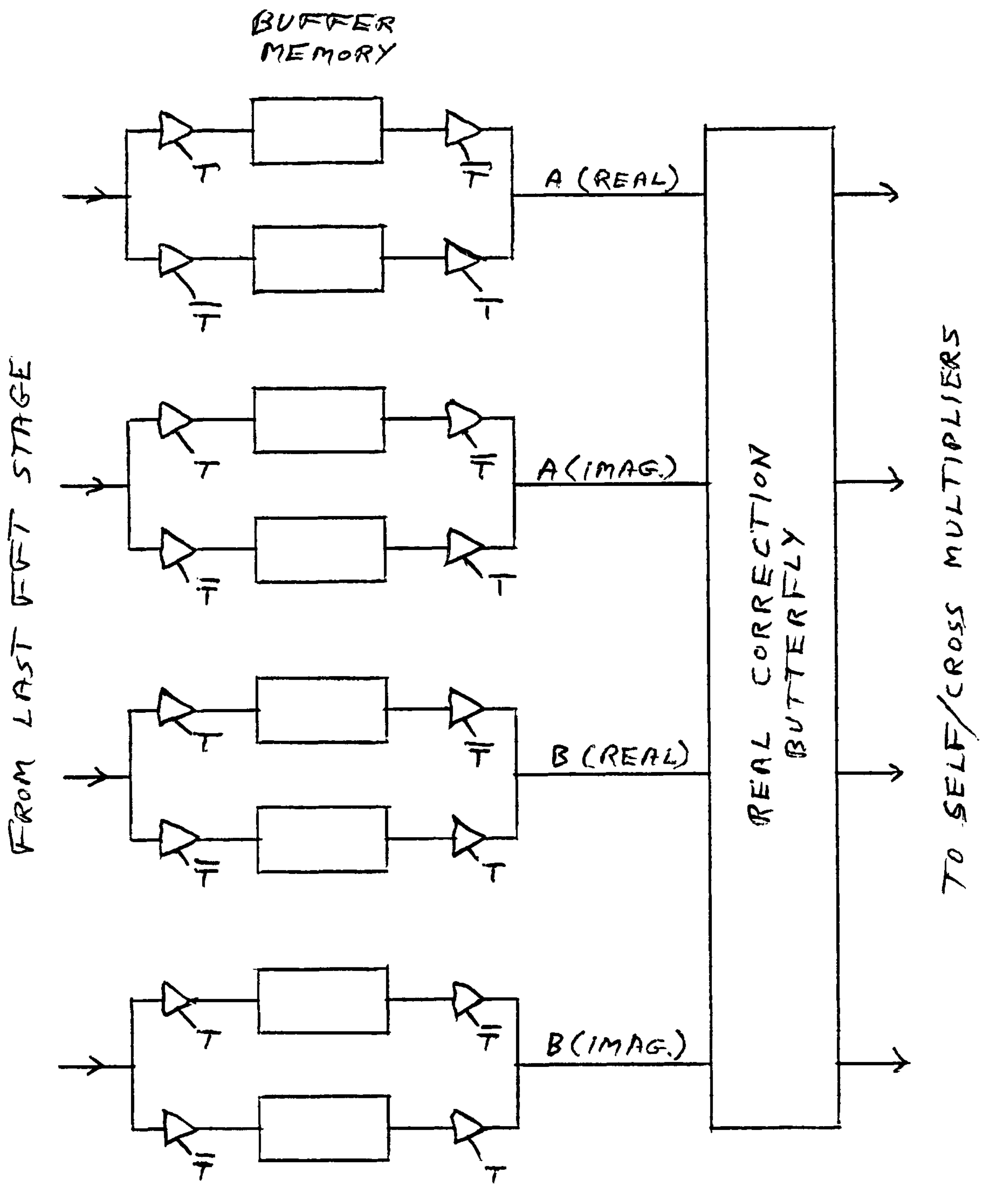

MULTIPLIER A (REAL)
WINDOW COEFFICIENTS
MULTIPLIER A (IMAGINARY)
WINDOW COEFFICIENTS
MULTIPLIER B (REAL)
WINDOW COEFFICIENTS
MULTIPLIER B (IMAGINARY)
WINDOW COEFFICIENTS
FROM BUFFER MEMORIES (6 bITS)
TO BUTTERFLY INPUTS (16-bITS)

Figure 7



REAL CORRECTION "BUTTERFLY" READING ORDER

Figure 8

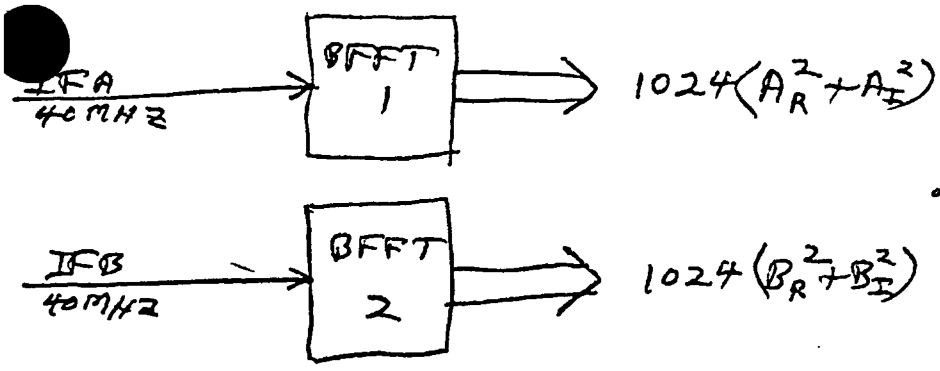


Correction stage to compensate for the input of real instead of complex data.

Figure 9

# 4 FFT configurations

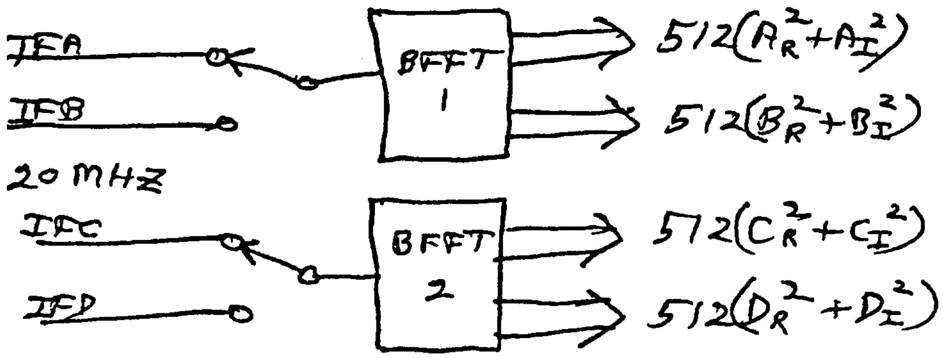
(a)



S.L. only

25.6 μs time res
2 mult/pt

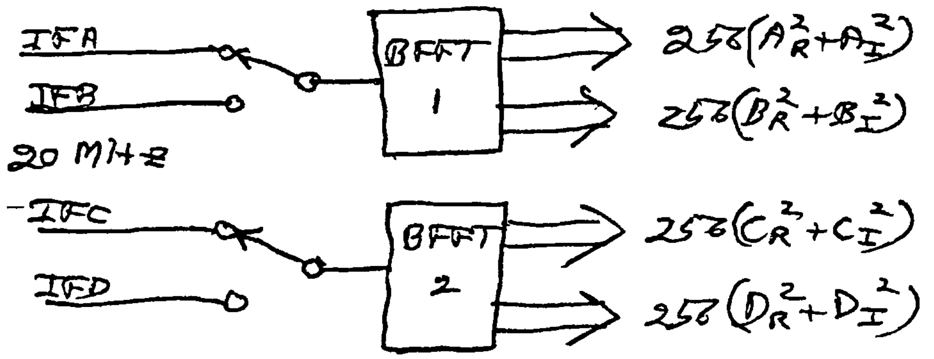
(b)



S.L. only

25.6 μs time res.
2 mult/pt

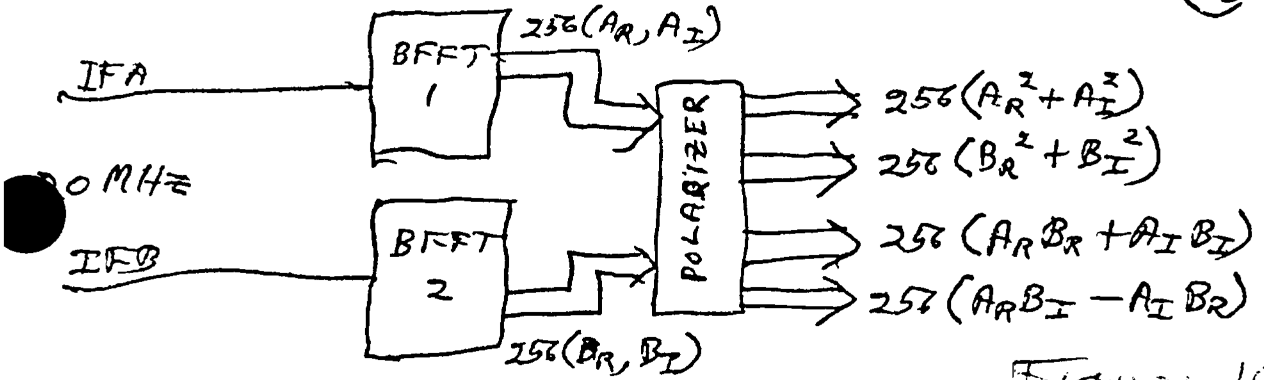
(c)



Pulsar only

12.6 μs time res
2 mult/pt

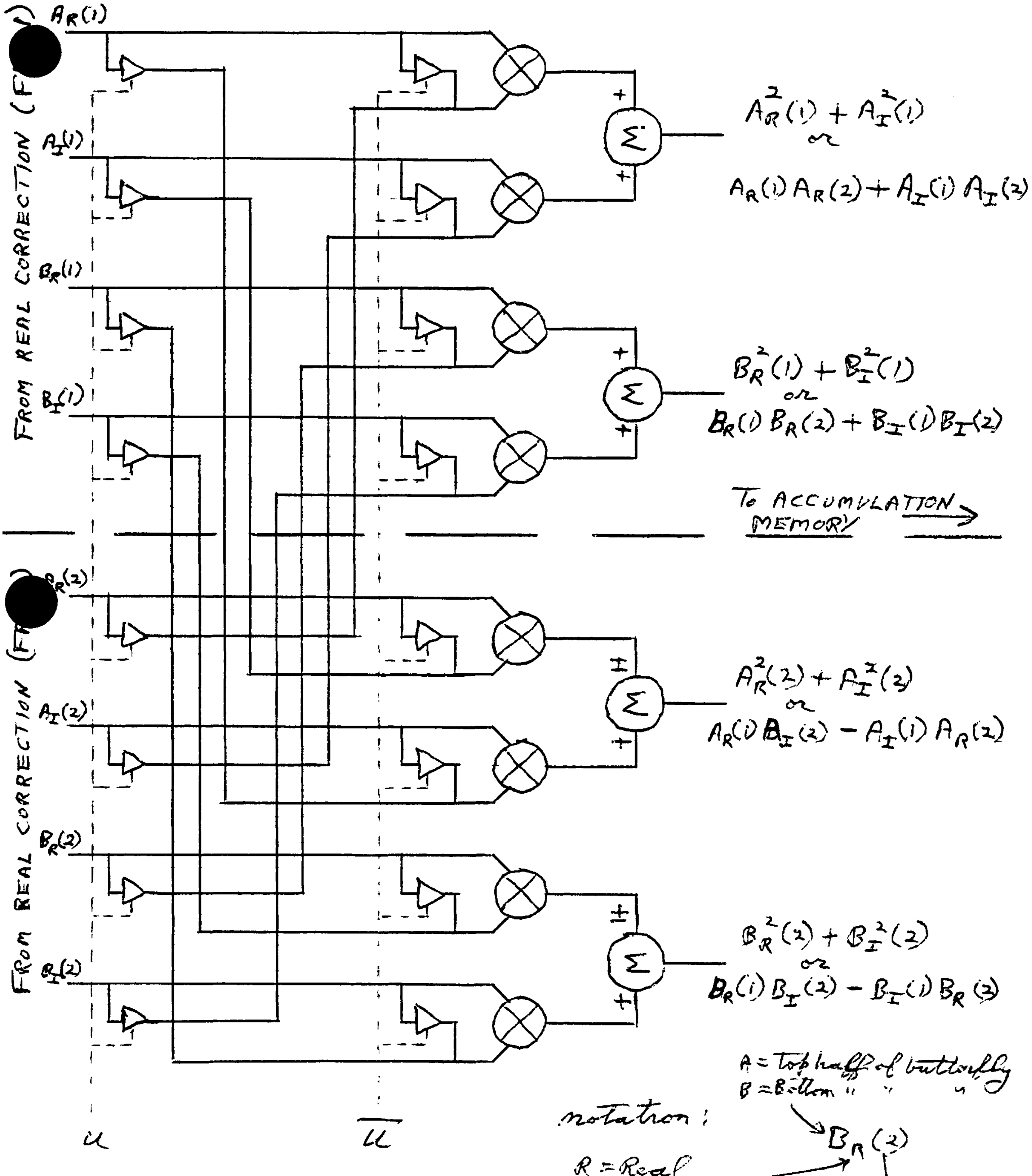
(d)



Pulsar only
12.6 μs time res
2 mult/pt

Figure 16

SQUARE OR CROSS MULTIPLIERS



notation:

A = Top half of butterfly
B = Bottom " " "

$B_R(2)$

R = Real
I = Imaginary

FFT 1 or
FFT 2

Figure 11

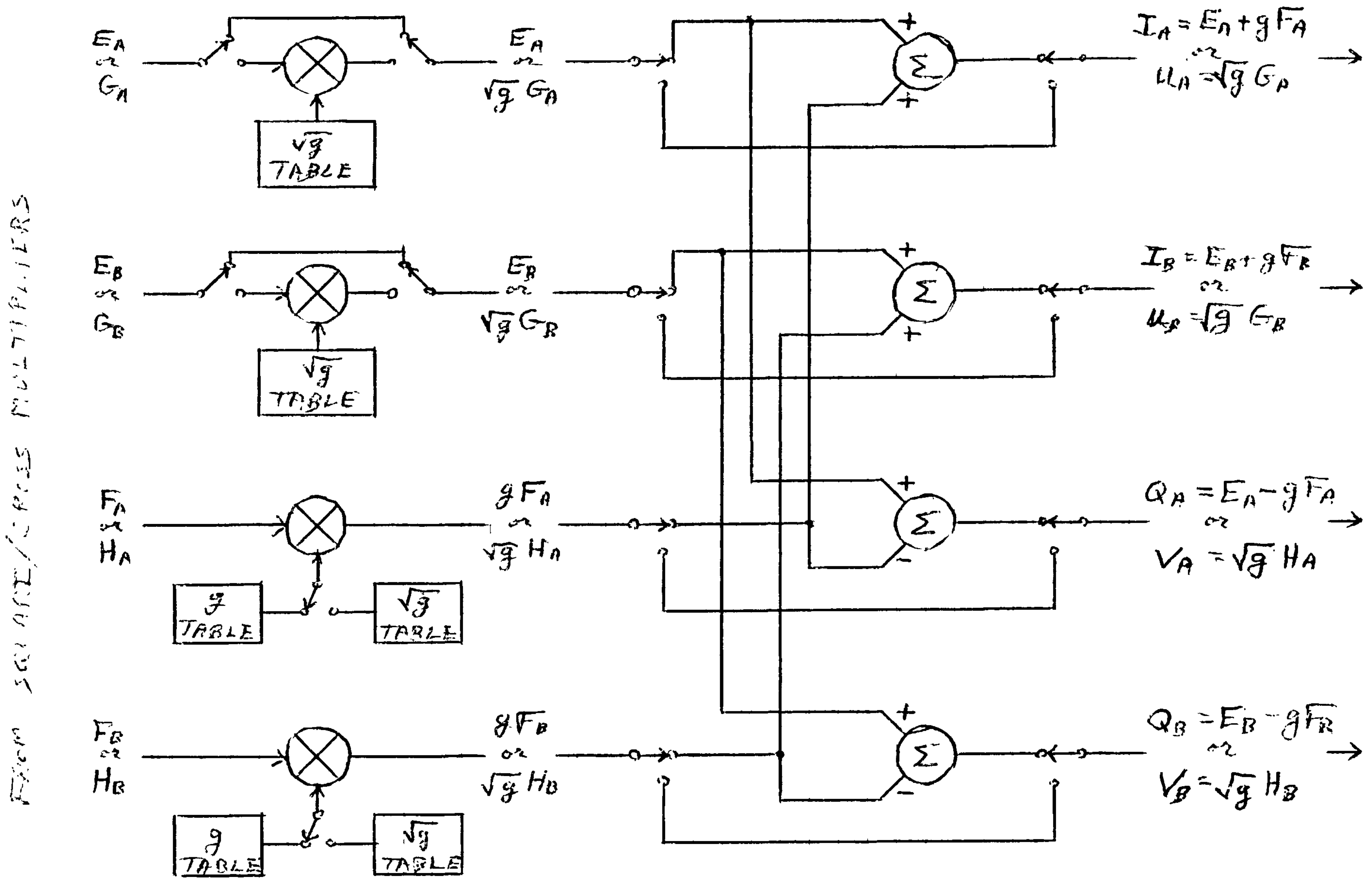
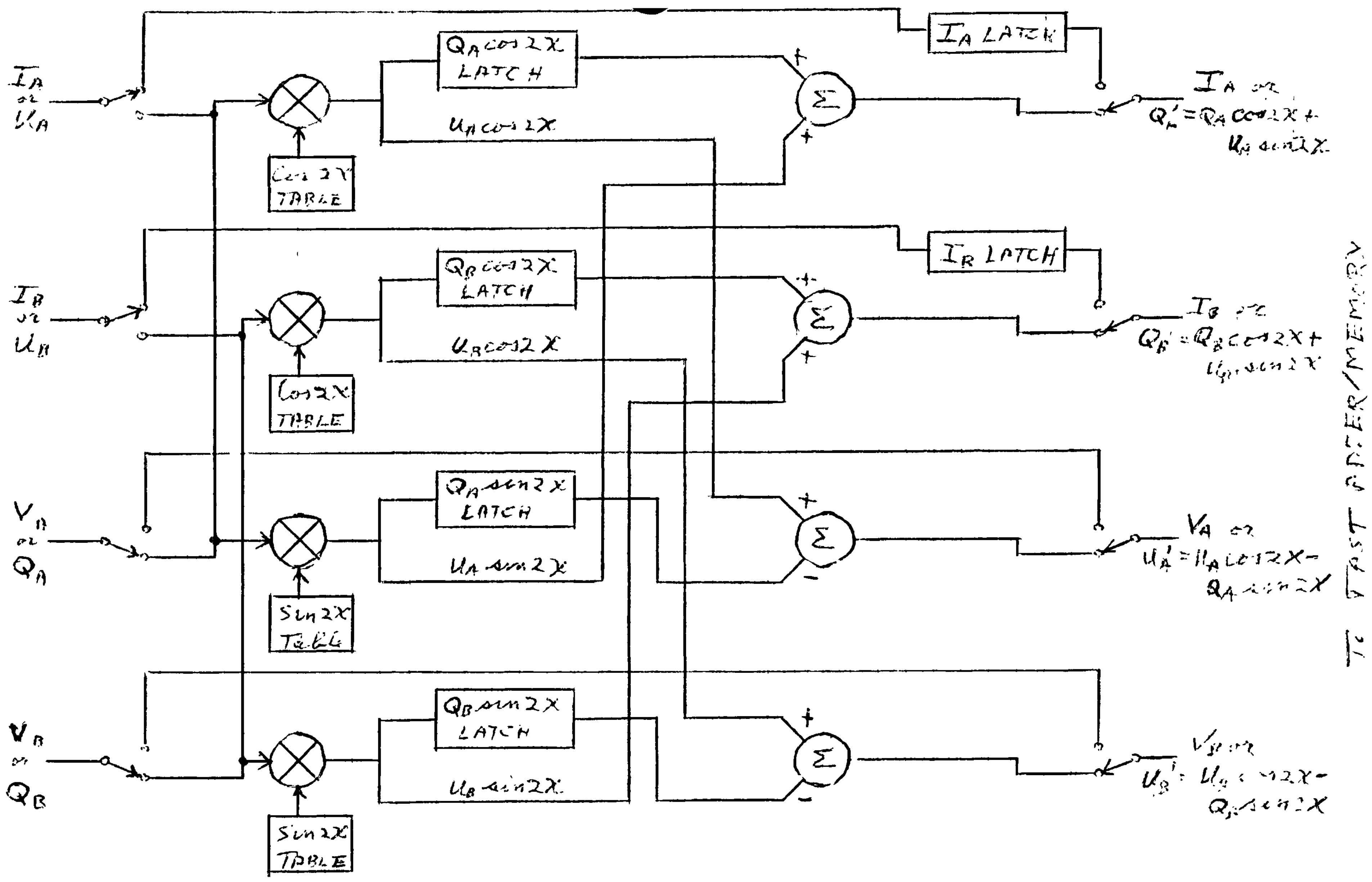

FROM SQUARE/CROSS MULTIPLIERS
$E_A$ or $G_A$
$E_B$ or $G_B$
$F_A$ or $H_A$
$F_B$ or $H_B$
$\sqrt{g}$ TABLE
$\sqrt{g}$ TABLE
$g$ TABLE
$\sqrt{g}$ TABLE
$g$ TABLE
$\sqrt{g}$ TABLE
$E_A$ or $\sqrt{g}\,G_A$
$E_B$ or $\sqrt{g}\,G_B$
$gF_A$ or $\sqrt{g}\,H_A$
$gF_B$ or $\sqrt{g}\,H_B$
$\Sigma$
$I_A = E_A + gF_A$ or $U_A = \sqrt{g}\,G_A$
$I_B = E_B + gF_B$ or $U_B = \sqrt{g}\,G_B$
$Q_A = E_A - gF_A$ or $V_A = \sqrt{g}\,H_A$
$Q_B = E_B - gF_B$ or $V_B = \sqrt{g}\,H_B$

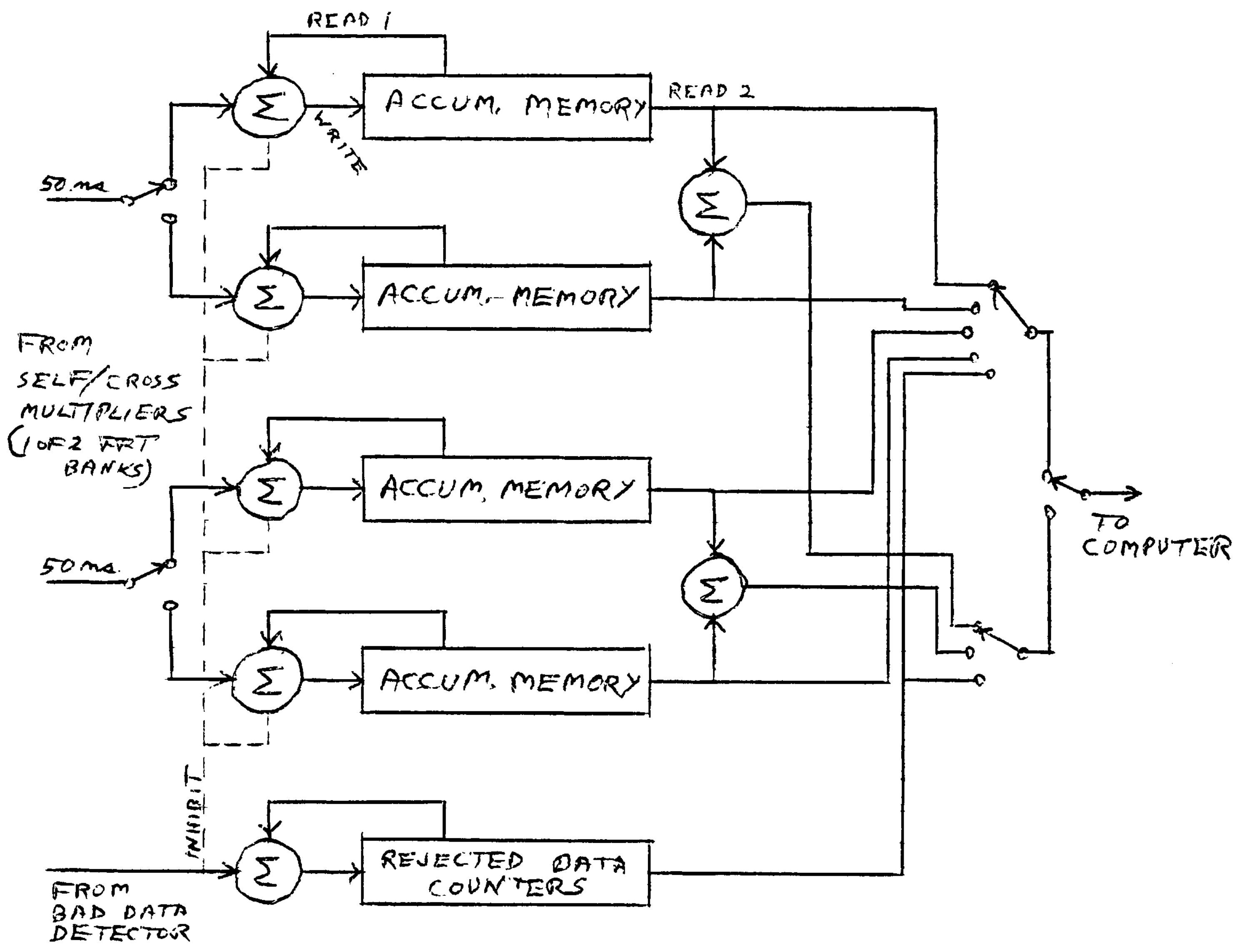
Figure 12



$I_A$ or $U_A$
$I_B$ or $U_B$
$V_A$ or $Q_A$
$V_B$ or $Q_B$
$\cos 2X$ TABLE
$\cos 2X$ TABLE
$\sin 2X$ TABLE
$\sin 2X$ TABLE
$Q_A \cos 2X$ LATCH
$U_A \cos 2X$
$Q_B \cos 2X$ LATCH
$U_B \cos 2X$
$Q_A \sin 2X$ LATCH
$U_A \sin 2X$
$Q_B \sin 2X$ LATCH
$U_B \sin 2X$
$I_A$ LATCH
$I_B$ LATCH
$I_A$ or $Q_A' = Q_A \cos 2X + U_A \sin 2X$
$I_B$ or $Q_B' = Q_B \cos 2X + U_B \sin 2X$
$V_A$ or $U_A' = U_A \cos 2X - Q_A \sin 2X$
$V_B$ or $U_B' = U_B \cos 2X - Q_B \sin 2X$
TO FAST FILTER/MEMORY

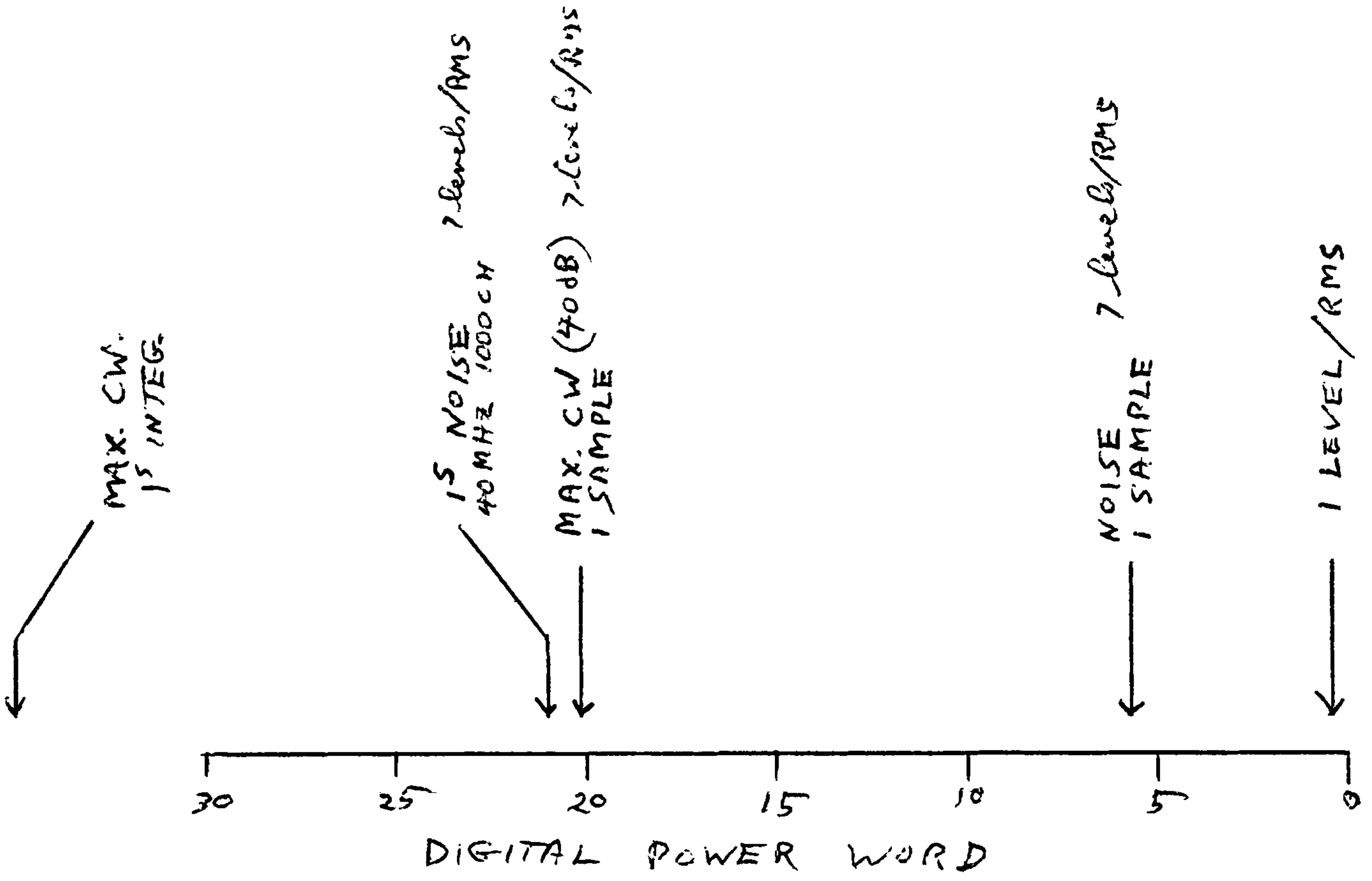
Figure 13

# FAST ADDER/MEMORY



| | |
|---|---|
| READ 1 | 25 ms |
| ADD + READ 2 | 50 ms |
| WRITE | 25 ms |

Figure 14



Maximum word sizes relative to power resulting from noise at 1 A/D level/rms power level

Figure 15