‘Oliver wasn’t as altogether happy as the lucky pig that accidentally got locked into the malt house of the local brewery.’

Charles Dickens

Chapter 8

Revised ADSL Simulator Design

Although the words in Charles Dickens’s book Oliver Twist were written over one hundred years ago, they deftly conveyed my feelings when on delivery of the Foundation and Core Generator software I discovered that the FFT Core didn’t actually exist!

This chapter details the revised design of the line simulator using a Xilinx ‘Reference Design’ for a 1024 point FFT in place of the Core design. In addition to new memory interface circuitry between FPGA blocks, a time manipulation FPGA after the IFFT and a new complete AFE at the final design stage from Fujitsu including ADC, DAC and filtering are incorporated into the design.

8.1 Overall Revised Simulator Design

Figure 8.1 shows a block diagram of the overall revised line simulator design incorporating additional time domain manipulation and integrated AFEs. The revised design builds on the transform principles developed originally, with the only major change being the inclusion of the time manipulation block to simulate time domain modeled noise components.

In contrast to the FFT Core which split its real 1024 input data points into two blocks (t(0) to t(511) and then t(512) to t(1023)), the FFT Reference Design splits the real and imaginary components of both input and output vectors into two separate continuous blocks.

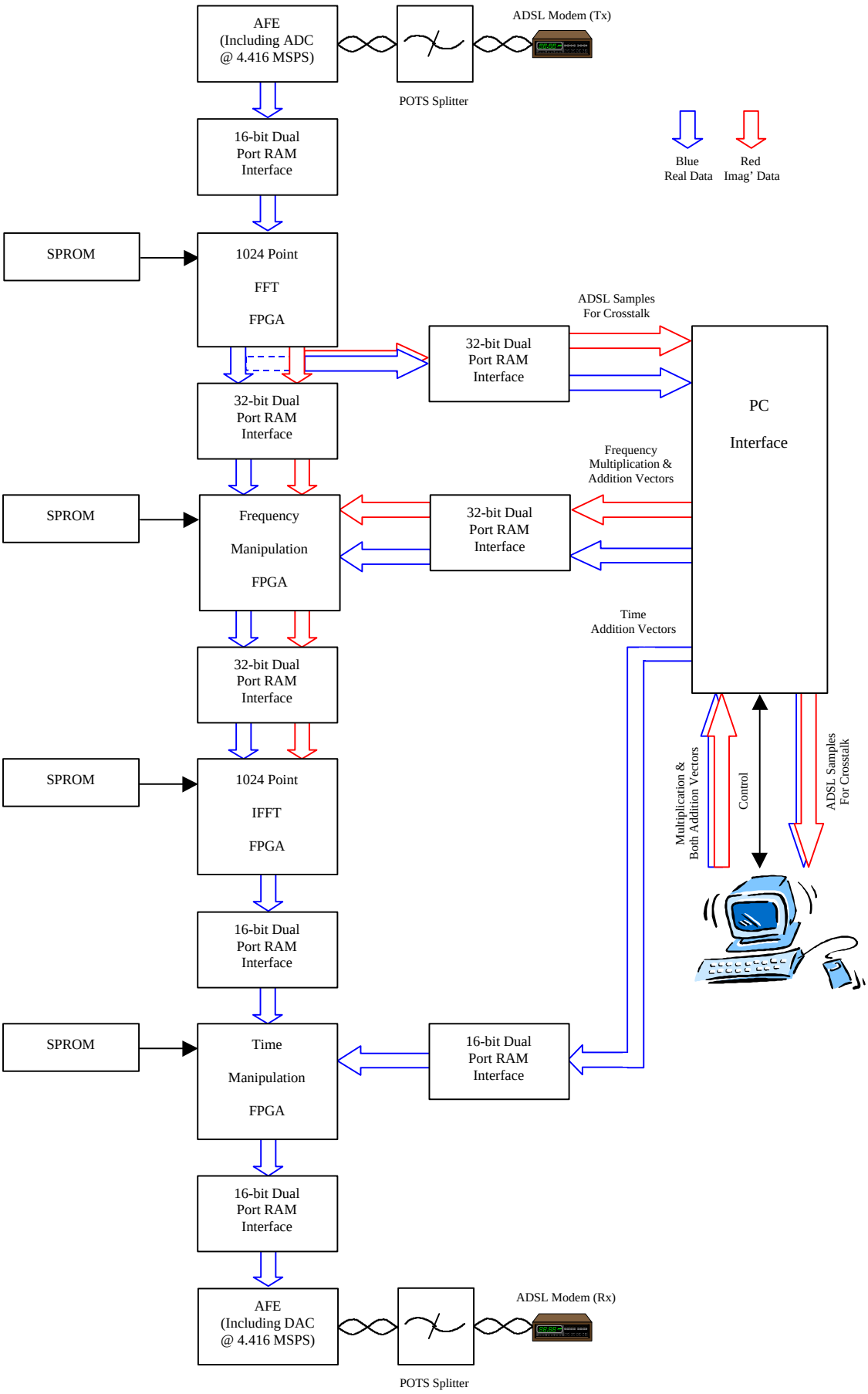

Figure 8.1 Revised ADSL line simulator block diagram

8.2 Xilinx Reference FFT Design

The full data sheet for the 1024 point FFT Reference Design from Xilinx may be viewed on the enclosed CD. The most salient first four pages are included in appendix 6. The most notable differences between the new Reference Design and the original FFT Core are as follows

- Radix 4 operation.
- Real and imaginary input and output data vector separation.
- The design requires two banks of 1 k-byte 32-bit external scratch pad RAM.
- Increased processing speed, reducing block processing to within 100 µs.
- Considerably increased CLB count, targeting XC4062 or larger devices.

Figure 8.2 shows the FFT Reference Design pin-out diagram.

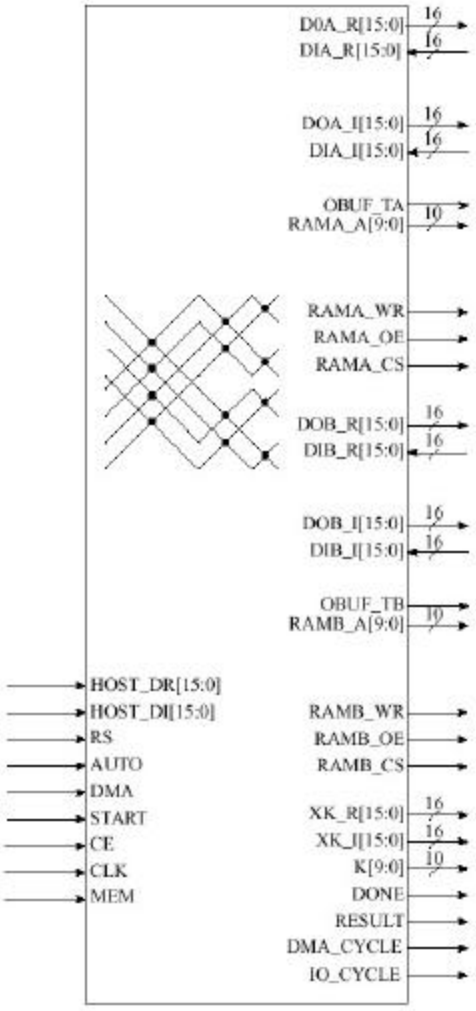

Figure 8.2 FFT Reference Design pin-out diagram

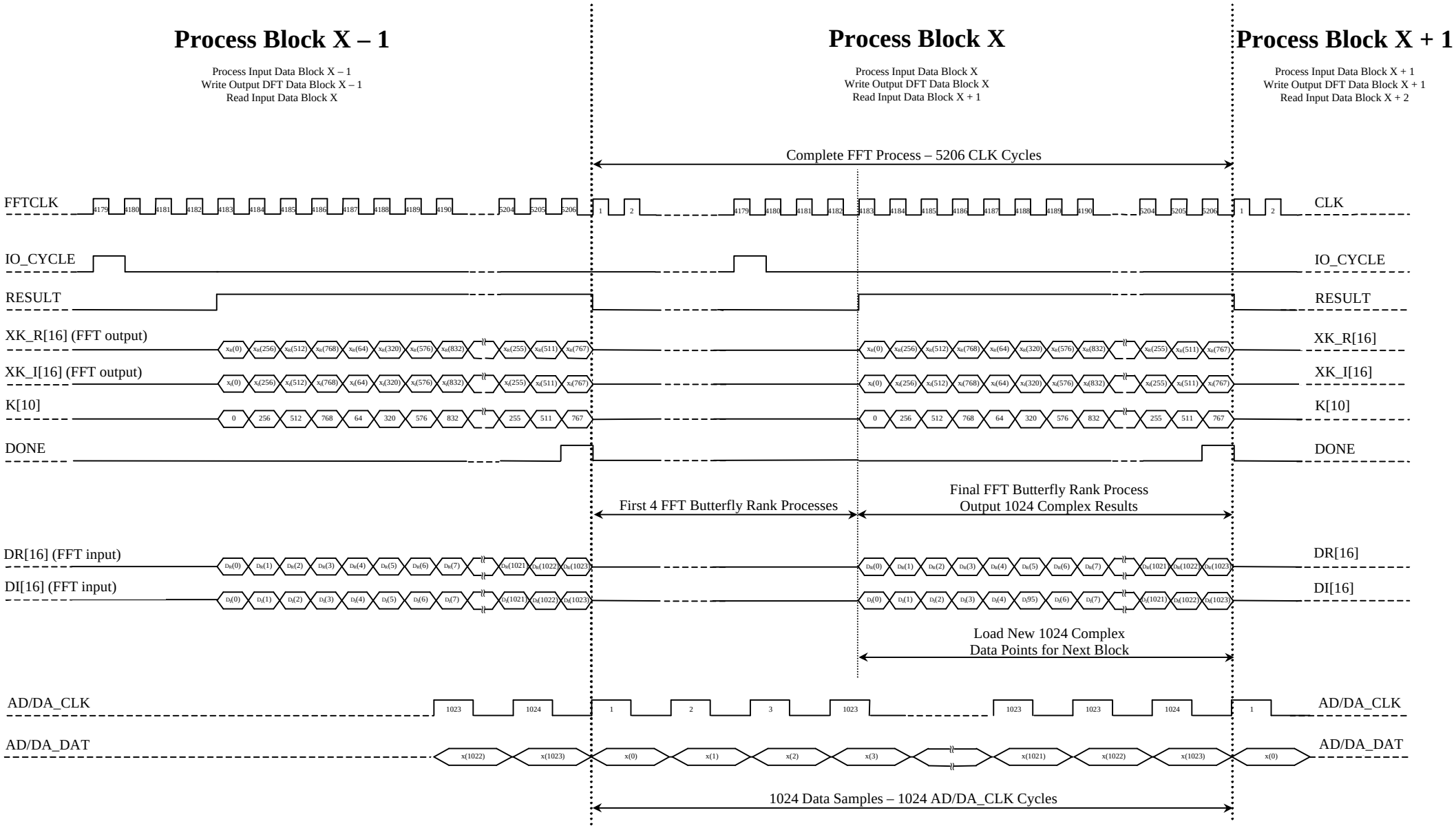

Figure 8.3 Timing diagram for FPGA FFT I/O, ADC and DACs

8.2.1 FFT Reference Design I/O and Control Timing

Figure 8.3 summarises the published I/O and control timing for the FFT Reference Design. The main points impacting the supporting circuitry are

- The FFT process consists of five Butterfly rank operations.
- The total FFT process requires 5206 FFTCLKS: 5 x 1024 Butterfly rank clock cycles, 17 clock cycles before each rank and one more at the start of each FFT operation.
- During the final rank process, the result is output whilst at the same time, data for the next FFT block operation is read into one of the scratch pad RAM blocks, alternating between blocks A and B on subsequent FFT transforms due to the odd number of Butterfly ranks.
- The real and imaginary components of each of the 1024 samples are accessed or output at the same time, thus requiring only one address bus for each block of scratchpad RAM and output data buses.
- The FFT doesn't address the input RAM, the latter is required to produce its data in chronological order and to the specified timing during the final Butterfly rank process, whereas the FFT actively addresses the data components of the output results through the use of a de-scrambling index bus, K.
- Periphery devices are notified of the FFT's need for new input data or its intention to write new results through the control strobes: DMA_CYCLE, IO_CYCLE, RESULT and DONE.
- The FFTCLK theoretically operates at upto 68 MHz, giving a clock period of 15 ns.

8.3 Internal Memory Interfaces

In the initial simulator design, the difference between time sample ordering and generation rate at the DAC's output and the required ordering and input rate to the FFT Core gave rise to the need for memory buffering between the two, operating on a pipelined block processing approach. Implementation using paged dual port RAM interfaces was developed. A similar situation occurs using the Reference Design. Conceptually, the full ordering and rate conversion scheme is shown in figure 8.4. Each page of dual port RAM contains 1024 16-bit locations.

8.3.1 Memory Interfaces - Justification

Fast dual port RAM is expensive, so should only be used where necessary. The following five sub-sections explain why dual port memory buffering is necessary between the ADC, FFT, frequency manipulation FPGA, IFFT, time manipulation FPGA and DAC. Reference to the top of figure 8.3 should be made for the control logic timing for the FFT Reference Design and the bottom of the same figure for general parallel flash A/D and D/A conversion processes when the FFT processing time and time sampling window are exactly the same.

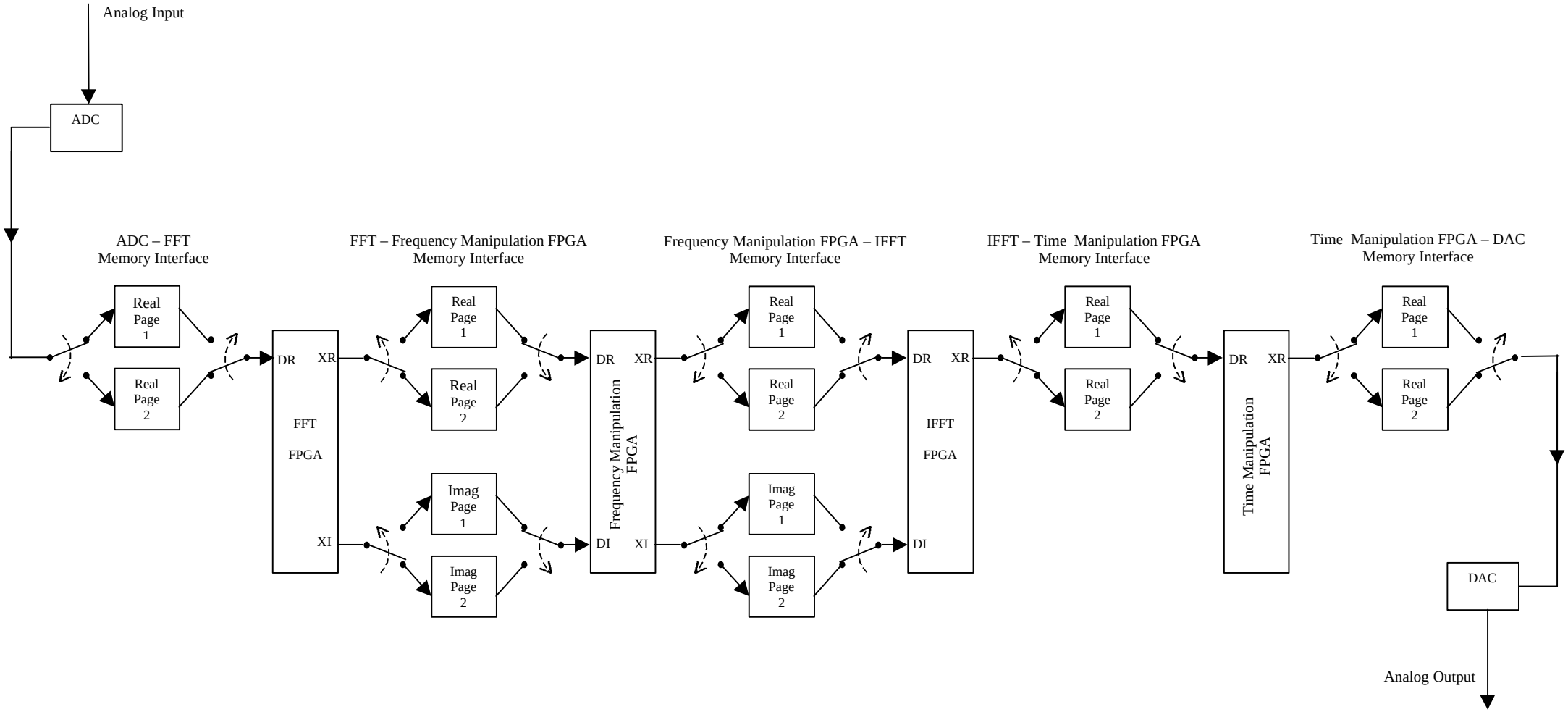

Figure 8.4 Conceptual memory page structure for ADC, DAC, FFT, IFFT and manipulation FPGAs
(All data lines 16-bit wide)

8.3.1.1 ADC – FFT FPGA Interface

From the ADC, time samples are output in chronological order (x(0), x(1),…,x(1023)) and generated at a rate of 1 sample every ADCLK (4.416 MHz), bottom of figure 8.3. The FFT Reference Design requires its 1024 point input data vector split into real and imaginary components then also reads them in chronological order, one sample every FFTCLK (≈50 MHz), top of figure 8.3. The real – imaginary split isn't a problem between the ADC and FFT FPGA as the time samples are purely real, but the difference in ADC write and FFT read rates requires sample rate conversion through a memory interface with pipelined block processing.

8.3.1.2 FFT – Frequency Manipulation FPGA Interface

Unlike common radix 2 bit reversed output ordering, the real and imaginary radix 4 FFT output frequency samples are in the digit reversed order[1] of the form

$$A[10] = [a_1, a_0, a_3, a_2, a_5, a_4, a_7, a_6, a_9, a_8] \qquad a_i \in \{1, 0\}, \quad i = 0, 1, ..., 9$$

This order is shown in figure 8.3. These samples are read by the processing FPGA and, if not for the mismatch in inter-sample FFT write time and manipulation processing delay, could be manipulated in the order they are produced by the FFT FPGA. The low level designs of chapter 9 will show that 16-bit multiplication and addition takes approximately 114 ns using Core multipliers and adders (8 PRIMARY_CLKs). However, frequency samples from the FFT FPGA are written once every 29 ns (1 35 MHz FFTCLK) during the last 1024 FFTCLKs of the complete 5206 cycle FFT operation, figure 8.3. Therefore, the frequency samples cannot be manipulated at the rate the FFT FPGA writes them, so memory buffering for pipelined operation is also required between the FFT and manipulation FPGAs.

8.3.1.3 Frequency Manipulation – IFFT FPGA Interface

The (I)FFT Reference Design requires chronologically ordered input vectors, (X'(0), X'(1),…,X'(1023)) and reads one sample per FFTCLK during the final 1024 IFFT cycles, figure 8.3. Because the rate at which the manipulation FPGA can write modified frequency samples is limited by the total manipulation delay of 114 ns, pipelined block processing is required for the manipulation process and IFFT, with a similar paged dual port RAM solution between the two FPGAs.

8.3.1.4 IFFT – Time Manipulation FPGA Interface

The output from the IFFT will be purely real (unless multiplication of the positive and negative frequency samples is by non complex conjugate pairs which would result in complex time samples which can't be converted to a physical time signal by the DAC anyway) and in the radix 4 order. The output from the IFFT includes a sample index bus (K in figure 8.3) specifically to de-scramble the

IFFT output vector, so reordering should occur immediately after the IFFT, justifying the IFFT time manipulation memory interface.

8.3.1.5 Time Manipulation FPGA – DAC Interface

The DAC requires chronologically ordered samples, spaced by 1 DACLK (equal to the ADCLK). Re-ordering has already been accomplished and sample output rate conditioning could also be done through careful timing via the previous memory interface, but without a memory buffer implementing the block processing approach between the time manipulation FPGA and DAC, the length of time that samples can spend being processed in the FPGA will be constrained by the DAC data input requirement of evenly spaced samples at the rate of one per DACLK. In future design developments, depending on what new operations are configured for the time manipulation FPGA, the time processing delay may increase from that of simple addition. In order to allow maximum flexibility, a memory buffer is included between the time manipulation FPGA and DAC, allowing the whole 232 μs block processing window to be available for manipulation operations on the 1024 real time samples.

8.3.2 Practical Memory Implementation

Practically, the page structure for each of the five interfaces can be implemented in a variety of ways. Since the real and imaginary components of each vector are always read or written in the same order and at the same time by any of the four FPGAs (e.g. for the dc frequency component, the FFT FPGA writes $X_R(0)$ and $X_I(0)$ at the same time, figure 8.3), only one address bus for each side of each memory interface is required for both real and imaginary memory blocks. One complex component can be stored in the least significant 16-bits and the other in the most significant 16-bits of each 32-bit memory location. Table 8.1 shows some possible RAM implementations. Figure 8.5 shows the page structure for complex data vectors using a single 2048 location, 32-bit dual port memory chip and figure 8.6 for a single 2048 location, 16-bit memory chip for the real valued data vectors.

ADC – FFT Interface **IFFT – Time Manipulation Interface** **Time Manipulation – DAC Interface** **(real data only)**	**FFT – Frequency Manipulation Interface** **Frequency Manipulation – IFFT Interface** **(real and imaginary data)**
2 x 1024 locations, 16-bit	2 x 1024 locations, 16-bit (for real data) 2 x 1024 locations, 16-bit (for imag data)
	2 x 1024 locations, 32-bit (16 MSBs for real data) / (16 LSBs for imag data)
1 x 2048 locations, 16-bit	1 x 2048 locations, 32-bit (16 MSBs for real data) / (16 LSBs for imag data)

Table 8.1 Memory interface physical RAM implementations

2048 x 32-bit Dual Port RAM Chip

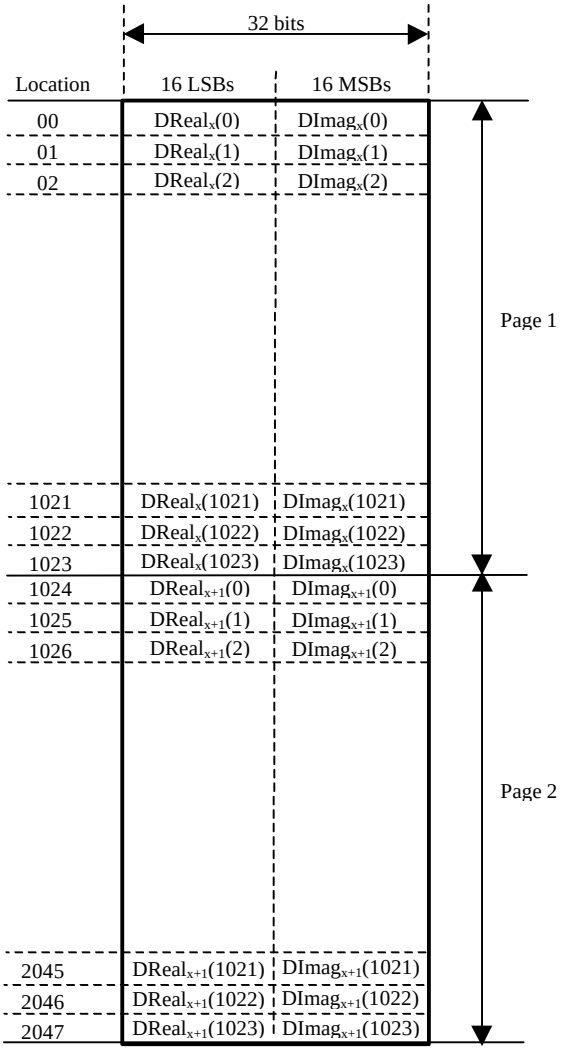

Figure 8.5 32-bit memory interface page structure for real and imaginary data vectors

2048 x 16-bit Dual Port RAM Chip

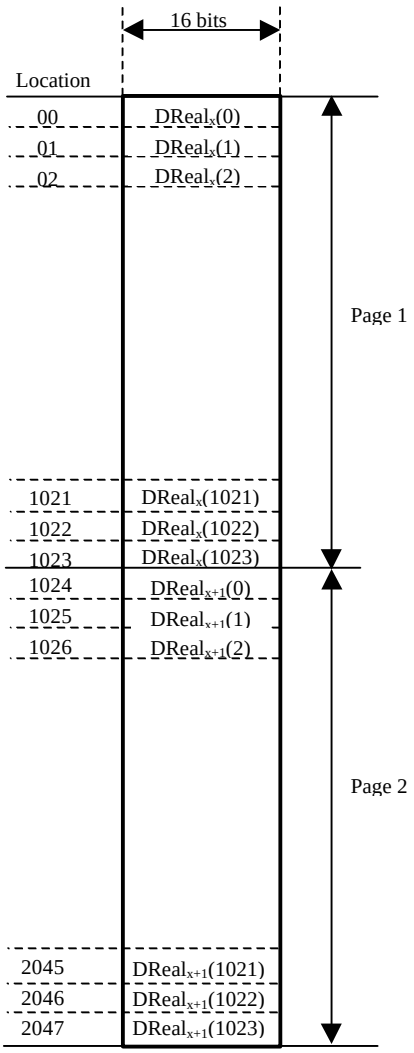

Figure 8.6 16-bit memory interface page structure for real valued data vectors

8.3.3 Memory Interfaces for PC I/O

The I/O requirements for PC control are similar to the initial design, with an extra set of 1024 point real addition vectors for the time domain manipulation required from the PC every 232 µs sampling period window. In order to allow data to be read by the FPGAs during the current processing time window whilst the PC downloads values for the next, dual port RAM buffering offering paged operation similar to that previously described can be used. The memory interfaces for PC I/O also provide sample rate transfer control to both manipulation FPGAs, simplifying the download timing demands on the PC. With paged memory buffering, the PC only has to download the full set of 512 complex conjugate multiplication coefficients, 512 complex conjugate frequency addition components and 1024 real time addition components within each 232 µs sampling window. The PC doesn't have to download the data

to satisfy the specific timing requirements of each of the manipulation blocks, nor does it have to upload the crosstalk samples from the FFT block according to their rate of production.

The great advantage of incorporating these PC I/O memory interfaces is to allow future work to define the timing for the PC interface independently of the simulator's internal processing operations. Figure 8.1 also shows all the necessary components for PC I/O, which can be practically implemented as shown in figure 8.5 and 8.6 for the complex and real data vectors.

8.3.4 Page Addressing for Memory Interfaces

In chapter 7, memory page selection was accomplished very neatly through simply deriving a PAGESELECT signal from the ADC clock. For two 1024 location memory pages (2048 location RAM chip), the ADCLK should be divided by 2048 and this signal used to drive the most significant address pin of one side of the 2048 location RAM chip with the other side's most significant address pin being driven by its logical inverse, shown below in figure 8.7 and 8.8. With this configuration, the opposite sides of the dual port RAM are always being written to, or read from, different pages in the paged storage space. Chapter 9 will present a low level design capable providing synchronous ADC, DAC, FFT, IFFT and PAGESELECT clocks from a single master oscillator.

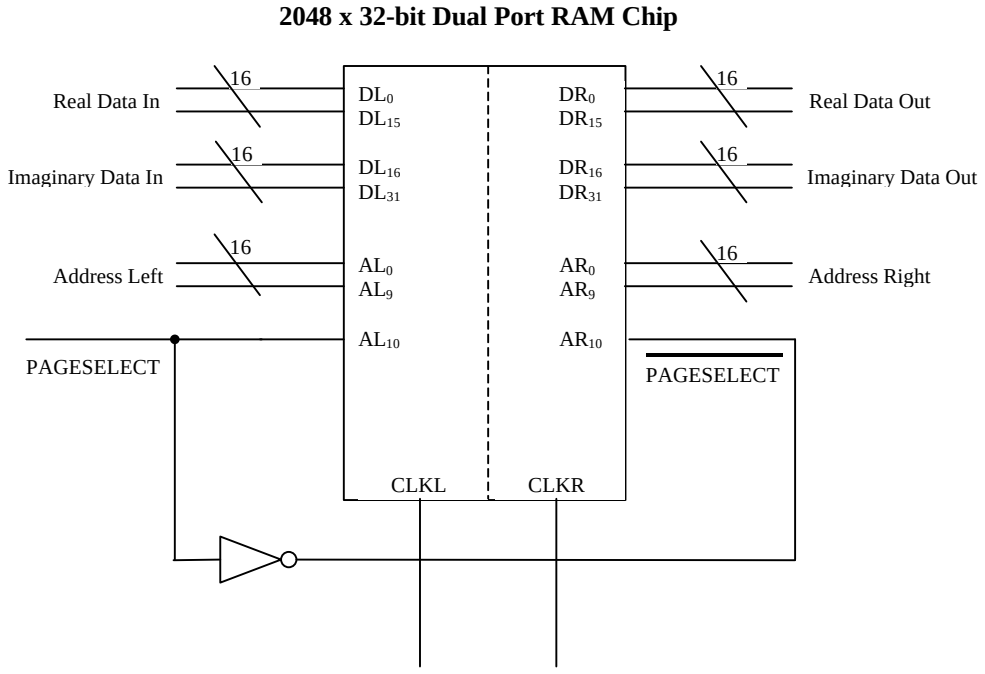

Figure 8.7 Page addressing for 2 k-word dual port RAM

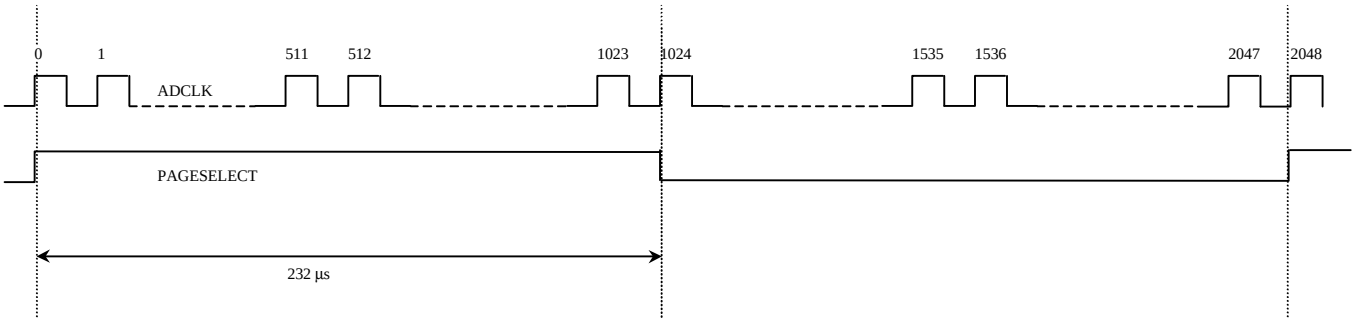

Figure 8.8 Digital timing diagram for memory page selection from the ADC clock

8.4 Processing Operations' Timing

Each of the six processing operations (A/D conversion, FFT, frequency manipulation, IFFT, time manipulation, D/A conversion) will be driven by a different frequency clock and will each require a different number of cycles of that clock to complete. That is, the FFT and IFFT operations will be driven by a clock at around 50 MHz and require 5206 cycles, the A/D and D/A conversion operations will be driven by a 4.416 MHz clock and require 1024 cycles whereas the manipulation processes will be driven by clocks at around 100 MHz and require a different number of cycles to complete depending on the internal logic design. Once memory interfaces are placed between each of the six processing blocks, the blocks can operate separately within each time sampling window period. The only constraint on each operation is that it must be completed before the start of the next time sampling window. This greatly eases logic design as each block can be designed to function independently of the others' internal timing. Figure 8.9 shows how each operation can take different periods to complete, but all within the fixed 232 µs time sampling window and also shows the passage of blocks of time samples through the simulator's six functional operations. Clearly, a time sample taken by the ADC at time T_0 will be leave the simulator from the DAC 5 sampling windows later.

As mentioned, the operational isolation by memory interfaces allows each functional block to be driven at its own internal rate, allowing future development of the manipulation blocks without impacting on the conversion and transform blocks. A single external high frequency master clock can be divided to drive the AFEs at 17.664MHz, the FFT and IFFTs at around 50 MHz (depending on final logic delays within the placed FPGA). And the manipulation blocks' clocks at around 100 MHz, again depending on the net delays within the FPGAs they are implemented in. All clocks should be chosen to be multiple of the AFE's 17.664 MHz clock to allow simple generation by division of the master.

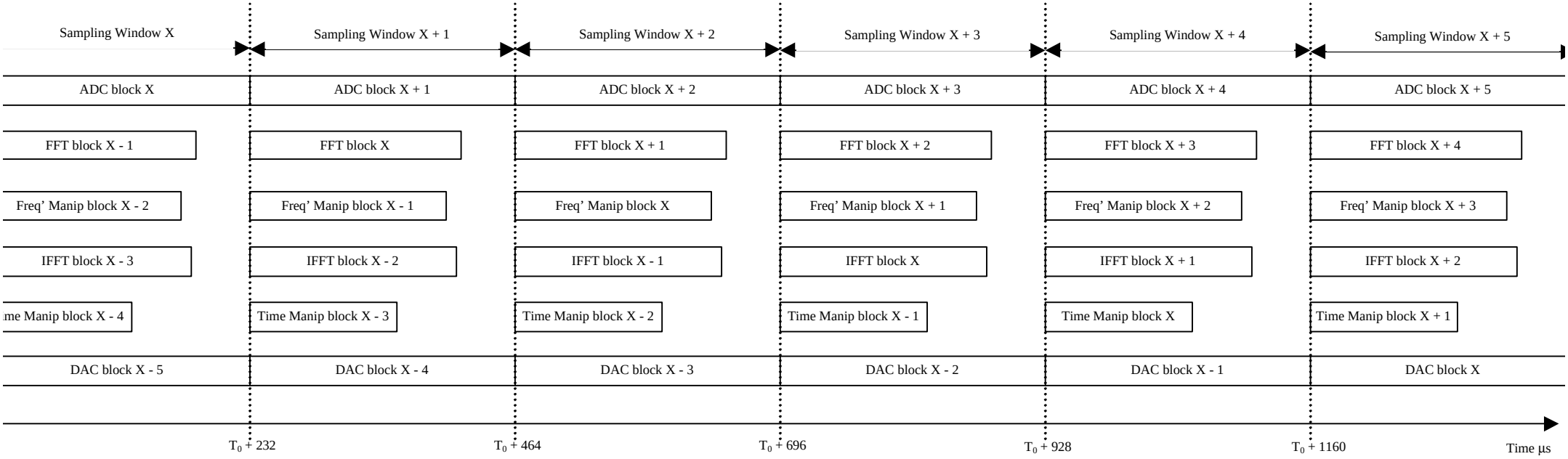

Figure 8.9 Independent operation timing for all six operations over a period of 6 time sampling windows

8.5 Analogue Front Ends

Shown in appendix 3 is the advance product preview of the Fujitsu KeyWave AFE. At present the full data sheet is commercially confidential, but the device is known to be suitable for use in the line simulator. The pin-out is shown below in figure 8.10.

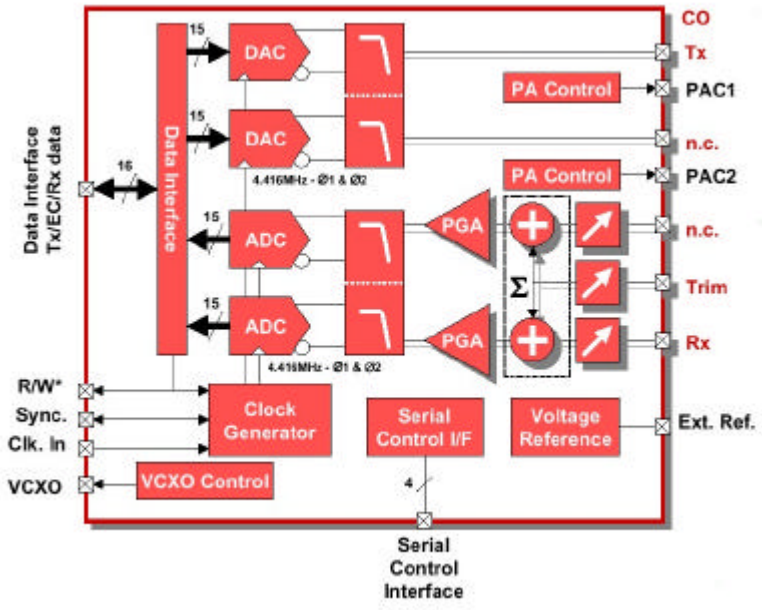

Figure 8.10 KeyWave pinout diagram

From the product preview, it is apparent that the IC:

- Is designed as a complete solution for both receivers and transmitters.
- Contains programmable low pass filters with cutoff frequencies upto 1.2 MHz.
- Has dual internal 16 bit 4.416 MSPS ADCs and DACs which can be configured to sample at 4.416 or 8.832 MSPS.
- Has an active rising edge read – write strobe for writing and reading to and from external memory
- Requires a 17.664 MHz external clock signal for timing.
- Only requires an external power line driver and hybrid transformer for connection to a twisted copper pair.

The Keywave AFE is ideal for both the receiver and transmitter front ends of the line simulator.

References

[1] E. Oran Brigham, “The Fast Fourier Transform and Its Applications”, Prentice Hall, 1988, p140.