A Possible 100 MSPS Altera FPGA FFT Processor

Grant Hampson

March 12, 2002

Introduction

This document describes a FPGA implementation and simulation of the FFT component of the IIP Radiometer RFI processor described in [1]. The FFT processor will ideally be capable of processing 100% of the input bandwidth (100 MSPS.) The chosen FFT length (1024) is a trade off of FFT bin width (approximately 100kHz) and possible RFI detection.

This document describes a possible implementation in an Altera FPGA, as opposed to currently available ASIC designs [2, 3]. These ASICs are capable of performing FFTs on 100 and 84 MSPS data streams continuously using two ASICS chained together (16W of power.) They come in large Ball Grid Array packages which may propose difficulties in assembly. There are many other FFT processors available [4], however most are significantly slower, or are sold as modules. Investigations are continuing in this area.

This document is broken into three separate sections. The first section describes the Altera FFT Megacore. Secondly, simulation results are presented from the core. Finally, several ideas are proposed for a 100 MSPS FFT processor.

1 Processor

An FFT core is available from Altera [5] for US$500 which can be evaluated for free. The core is fully programmable, with the main features being:

- Data and twiddle widths of 8 to 24 bits precision
- Floating point core with 3 to 8 bits precision
- FFT length from 2^4 to 2^{20}

The FFT Megacore can be connected to internal or external memory. Memory is required for temporary storage as well as ROM storage for the twiddle factors. The size and performance of the core is determined by

- the processing requirements are mainly dependent on the data and twiddle widths,
- the memory requirements are mainly dependent on the FFT length,
- the performance is dependent on the number clock cycles to compute the FFT:

$$\text{clock cycles} = 0.5 \log_2 N \times (14 + N + \log_2 \text{twiddle width}) \quad (1)$$

A test bench for the Altera FFT Megacore function is provided by Altera for benchmarking. Table 1 illustrates the FFT processor sizes for two input data widths and two twiddle data widths. Increasing the size of the twiddle width increases memory and processing requirements. The maximum clock rate is also 20-30% above the available clock rate of 100MHz.

Table 1: Example compilation sizes and speeds of two popular input data widths for 8 and 16 bit twiddle factors. The length of the FFT is 1024 points.

Data	Twiddle	LE	Memory	Speed
8-bit	8-bit	1344	20480	131
	16-bit	1722	20480	128
10-bit	8-bit	1539	24576	125
	16-bit	2195	28672	105

Another compilation example is shown in Figure 1 where the memory requirements of the FFT are shown to be linear with FFT length. The processing requirements (number of Logic Elements (LE)) varies only about 10% over all FFT lengths. The memory resources required for implementation will be a strong trade off against the FFT length. Shorter FFT lengths will be preferred over longer FFTs.

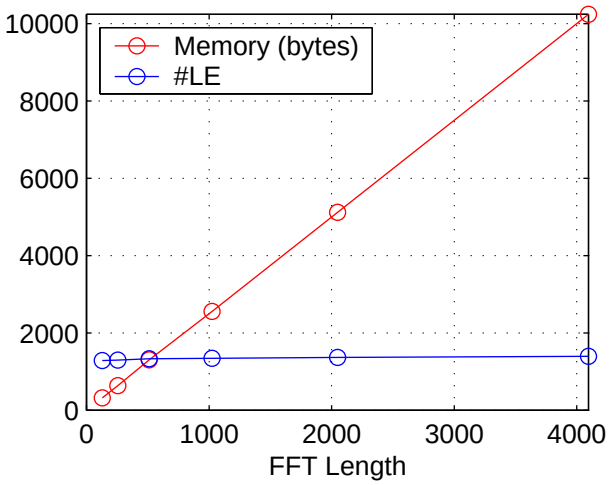


Figure 1: Memory and processing requirements of the FFT core for various lengths of FFTs. The data and twiddle widths used here are both 8-bit.

The FFT Megacore can be simulated with any type of data desired. A Matlab script generates the required input simulation files for Quartus and the resulting simulation wave forms can be saved and analyzed in Matlab. Figure 2 illustrates the simulation of a 1024 point FFT. There are three main sections to the simulation. Firstly, input data is written into the FFT data memory. A 'go' signal is activated and the FFT processes the data. When the FFT finishes a 'done' signal is activated and the result can be read from the data memory.

Since data is being sampled continuously, the writing of data to RAM doesn't contribute to the overall processing time. The processing time is however, the time it takes to

process the data, as well as, the time it takes to read data from the FIFO. Specifications of the core do not include this read time - just the FFT time.

In this simulation the clock rate was set to the input data rate of 100MHz. However, as Table 1 suggests there could be a gain of 25% by increasing the processing clock rate to its maximum. Problems may however arise in the different clock rates. Once again the specifications used the maximum clock rate. The following sections measure the throughput of the FFT Megacore using a data and processing rate of 100MHz.

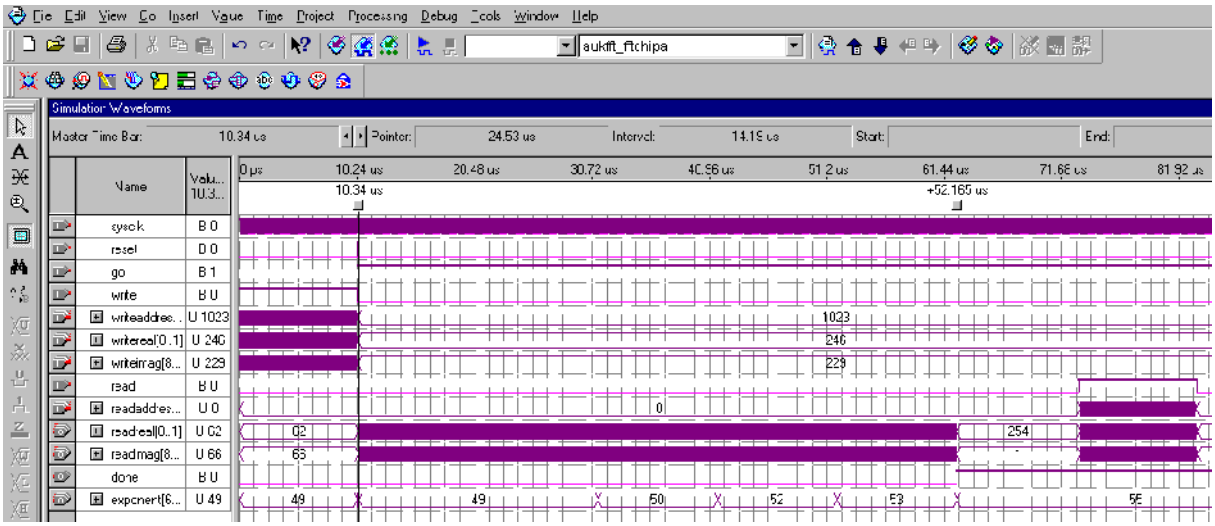


Figure 2: Example simulation of a FFT implementation using the Altera FFT Megacore. Three distinct sections can be seen in the simulation. The first is loading input data into the memory, secondly is the FFT processing time, and thirdly reading of the results.

2 FFT Processor Results

Simulations of the FFT core were done to verify correction operation of the core and to see the effects of data and twiddle widths. Figure 3 illustrates one such result where 16-bit twiddle factors were used to compute the FFT of a sinusoid plus noise. Small differences exist between the Matlab computed FFT and the Megacore output - most of which seem to be effected by dynamic range of the output. A simulation of 8 and 16 bit twiddle factors showed little difference between outputs - the average error was almost identical. Increasing the number of output data bits would most likely have a greater effect. The FFT Megacore however fixes the number of output bits to be identical to the number of input bits. In order to increase the number of output bits the input width has to be increased, thus increasing the processing requirements.

The next, and probably most important tests to be conducted were to measure the processing capabilities of the Megacore. Figure 4 illustrates the results of six simulations with different FFT lengths (from 128 to 4096.) The time was recorded for the FFT processing time and the time to read the data from the SRAM. From the total time it is possible to calculate the throughput. For shorter FFTs the throughput is as high as 18 MSPS and for longer FFTs it is as slow as 14 MSPS. For 1k FFTs the throughput is 16.4 MSPS which is far from the desired 100 MSPS (1/7th the desired rate.)

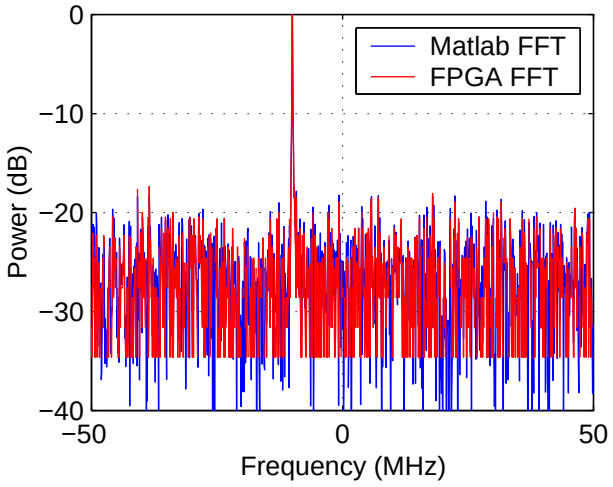
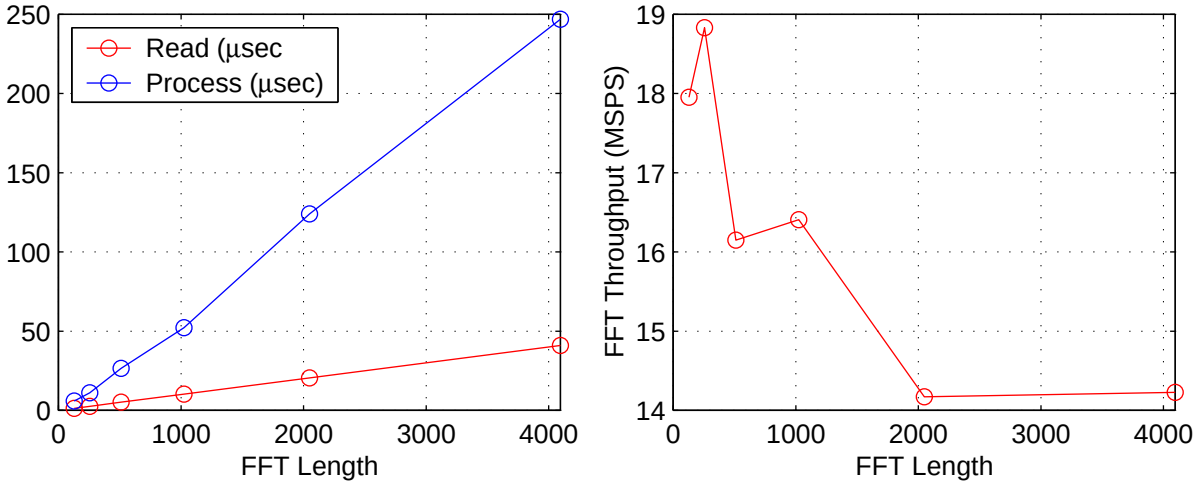


Figure 3: Example simulation of a FFT implementation using the Altera FFT Megacore with 8-bit inputs and 16-bit twiddle factors. There is little difference between the Matlab FFT and FPGA FFT. The greatest effect is the limited dynamic range of the output (8-bits) reducing the effect of 16-bit twiddle factors. 8-bit twiddle factors produced almost identical errors.



(a) FFT processing and reading times

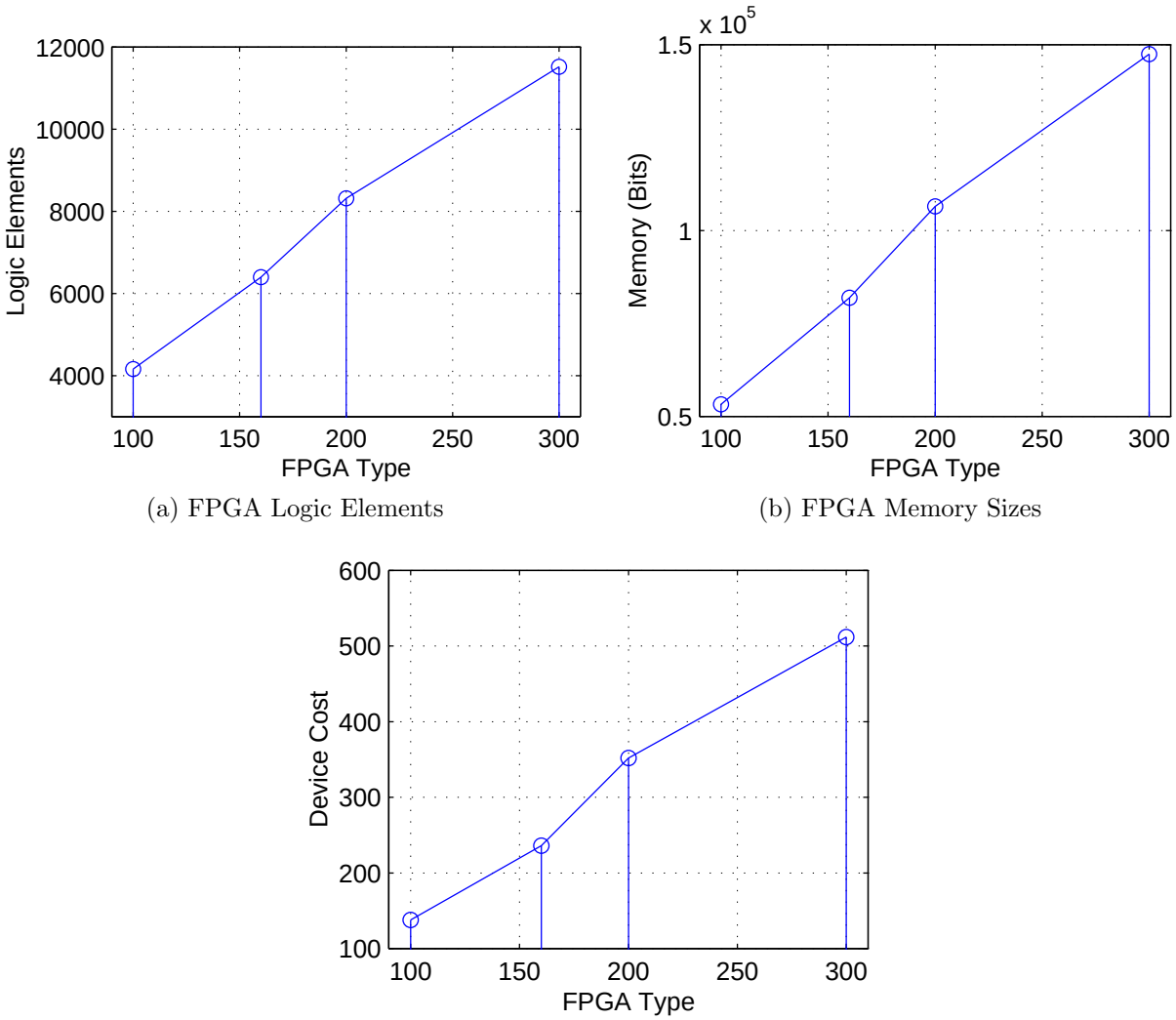
(b) Single FFT processor throughput

Figure 4: Results for various length FFTs for 8-bit inputs and twiddle factors. (a) The recorded processing and read times for the Altera FFT Megacore. (b) The calculated throughput based on the total processing time and read time.

3 A Possible 100 MSPS FFT Processor

A brute force technique of increasing the throughput of the FFT Megacore is to do parallel processing. Instead of having only one core operating it is possible to have multiple FFT cores. Several factors then become quite important - FFT memory and processing requirements - since everything is now multiplied by M (the number of cores.)

Firstly, it is important to consider what FPGA sizes currently available on the market. Figure 5 illustrates the characteristics of four available FPGAs [6]. These FPGA sizes are all available in Quad Flat Packs (QFP) which are relatively easy to assemble. Larger FPGAs are available in Ball Grid Array (BGA) packages.



(a) FPGA Logic Elements

(b) FPGA Memory Sizes

(c) FPGA Cost

Figure 5: Altera APEX series of FPGAs come in 10 different sizes (with many other possible mutations.) Four sizes suitable for implementation of the multi-FFT core are shown here; the EP20K100 (144-pins), EP20K160 (144-pins), EP20K200 (208-pins) and EP20K300 (240-pins). (a) illustrates the number of logic elements within the FPGA, (b) shows the available memory resources, and (c) shows the device cost. The cost function seems directly proportional to the FPGA hardware.

Next consider the implementation of a multi-FFT core processor. For a 10-bit input data width and using 8-bit twiddle factors, requires 1539 LE and 24576 bits of memory. Multiplying this by 8 to achieved the desired throughput requires 12312 LE and 196608 bits of memory. Given that a safe fill factor of FPGAs is approximately 75% of their resources - none of the available FPGAs are suitable. A technique to overcome this would be to use two FPGAs, as shown in Figure 6. Each FPGA would contain four FFT cores, and output multiplexer. The master FPGA would contain a controller for both FPGAs. The master controller also generates read and write addresses (all identical) which are not shown to simplify the drawing.

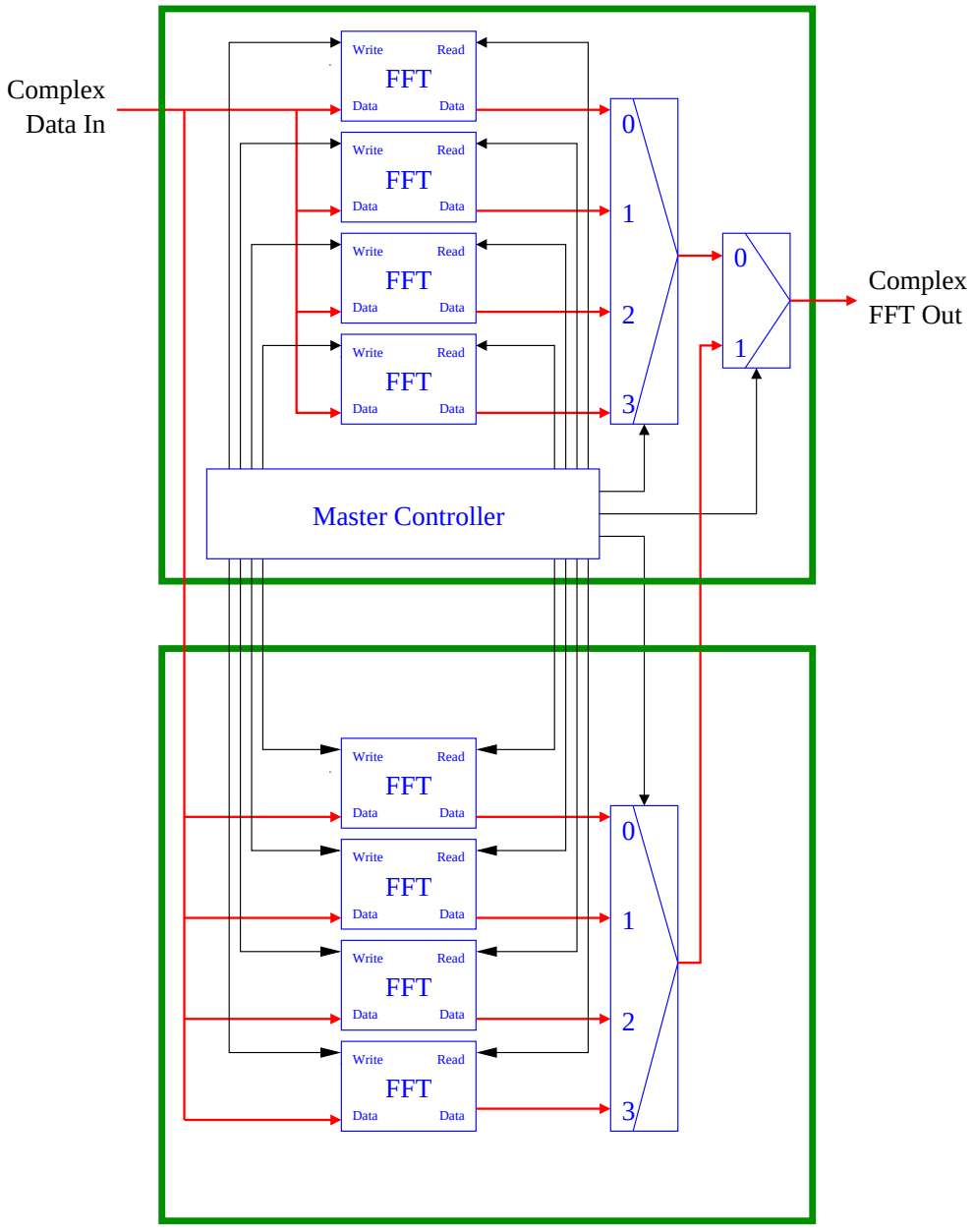


Figure 6: Example implementation of a multi-FFT core processor evenly distributed over two FPGAs. If the FFTs used are 10-bit data, 8-bit twiddle and length 1024, 53% of LE and 67% of the memory are used in a EP20K300 device.

Given that there are now four FFT cores per FPGA, this would load an EP20K300 with 53% of the Logic Elements and 67% of the memory. This provides adequate room for the controller and multiplexers. Additional hardware may also be required for the implementation of the Bartlett window and power integration described in [1].

Next, consider the timing of the multi-FFT core design. An example timing diagram is shown in Figure 7. Here a length 1k FFT is implemented using 10-bit inputs and 8-bit twiddle factors. The number of clock cycles required to compute the FFT can be calculated from Equation 1, which equals 5205 clock cycles. The timing diagram starts by writing data to RAM-0 which takes 1024 clock cycles. After 5205 clock cycles the FFT processing is finished. During this processing each of the FFT RAM blocks are written to in succession. There is a small delay of 939 clock cycles between the FFT finishing and the RAM being read. After reading the whole cycle starts again.

The controllers for the multi-FFT core processor might be implemented as a master controller initializing the start time of 8 smaller identical controllers. In this way the smaller controllers should have no speed limitations. Each smaller controller will simply start writing at $n = 0$, stop writing at $n = 1024$, start reading at $n = 7168$, stop writing at $n = 8192$. The timing diagram will scale linearly if the FFT length changes.

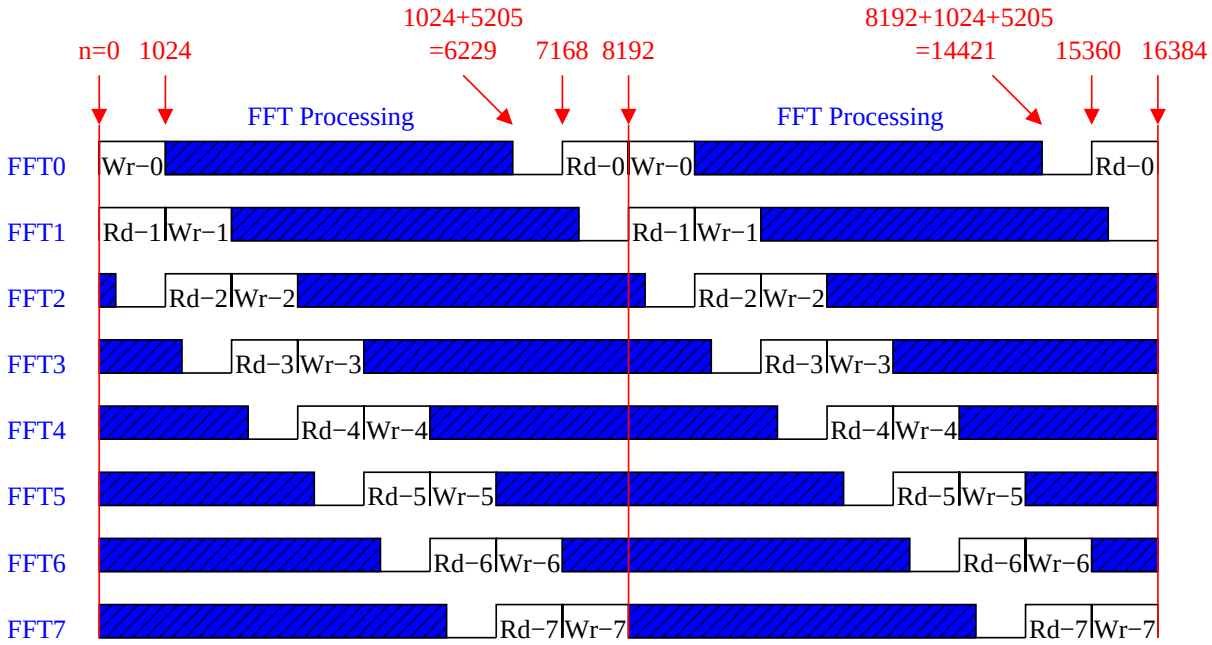


Figure 7: Example timing diagram of an eight FFT core processor.

4 The Floating Point Nature of the FFT Megacore

It was noted in Section 1 that the FFT Megacore uses a floating point mechanism to improve resolution of the result. Additional to the data outputs is a bus which contains the exponent (3 to 8-bits wide.) As a result, the output data needs to be scaled by this exponent in order to have any relative sense. This could be implemented using a large multiplexer. Alternatively, the floating point output could however be useful in a floating point multiplier (such as the power estimator.) Further work is required here.

Summary and Conclusions

This report has presented a possible 100 MSPS Altera FPGA FFT processor implementation. The Altera FFT Megacore is available for $500. The Megacore was implemented in various sizes to evaluate the processing and memory requirements, as well as the operating performance.

The FFT Megacore was found to have a fraction of the throughput required. Consequently, a design which contains eight FFT processors operating in parallel was presented. The proposed design could be implemented in two EP20K300EQC240-1 Altera FPGAs. This FPGA has a 240-pin QFP foot print and each FPGA costs $512 (`www.arrow.com`). This brings the total cost of the 100 MSPS FFT to $1524.

The proposed FFT processor has the limitation that the FFT length cannot exceed 1024 - unless the throughput is sacrificed. The main limitation of larger FFT lengths is the required memory. The FFT length can be shorter than 1024 with no drop in performance.

Using the proposed implementation of a 1024-point FFT with 10-bit inputs and 8-bit twiddle factors will leave ample room for extra processing such as windowing and power estimation in the same FPGAs.

References

[1] S. W. Ellingson, "Design Concept for the IIP Radiometer RFI Processor," January 23 2002. http://esl.eng.ohio-state.edu/rfse/iip/rfiproc1.pdf.

[2] *1K complex full-floating point FFT in 10 us*, Double BW, 2000. http://www.doublebw.com/FFTEngine.htm.

[3] *RDA 108 - Single Chip FFT*, Radix Technologies. http://www.radixtek.com/rda108.htm.

[4] B. M. Baas, "FFT Information Page," October 25 2001. http://www-star.stanford.edu/ bbaas/fftinfo.html.

[5] *FFT MegaCore Function User Guide*, Altera Corporation, March 2001. http://www.altera.com/literature/ug/fft_ug.pdf.

[6] *APEX 20K: Programmable Logic Device Family*, Altera Corporation, December 2001. http://www.altera.com/literature/ds/apex.pdf.