# A MEMORY EFFICIENT, LOW POWER FAST FOURIER TRANSFORM ARCHITECTURE

K Babionitakis, K Manolopoulos, K Nakos, D Reisis and N Vlassopoulos
Electronics Laboratory, Department of Physics,
National and Kapodistrian University of Athens, Greece
dreisis@cc.uoa.gr

## Abstract

FFT computations are critical to signal processing and telecommunication systems, targeting real-time processing and demanding area and power reduction. Toward achieving these goals, this paper describes a VLSI FFT architecture that can meet real time requirements, reduce the overall memory area and have considerably low power dissipation. A technique involving three consecutive radix-4 schemes to result in a 64-point FFT engine is presented. The use of a radix-4 scheme simplifies significantly the processing requirements. Further, the cascading of 64-point FFT engines allows the design of architectures that can efficiently accommodate large input data sets in real time, while the memory requirements are reduced to one third compared to the fully unfolded radix-4 architecture. To validate the efficiency of the architecture an example implementation on FPGAs handled a throughput of 4096-point FFT at 22.76 usec.

## 1 Introduction

Today's applications in signal processing and telecommunications require *FFT* implementations that can perform large size, low latency computations and obey power dissipation restrictions [12]. These demanding computational tasks can be accomplished either by using a single processor operating at high clock frequency [5] or by using an Application Specific Integrated Circuit (*ASIC*).

Several *FFT* organizations that perform the computation have been proposed in the literature [2],[3],[7]. The architectures vary with respect to the level of parallelism, the sustained throughput rate, the memory space, utilization of the hardware resources and power dissipation. Fully unfolded *FFTs* [10] can sustain the maximum throughput at lower clock rates while occupying more VLSI area and use larger size memory between successive stages. Cascade *FFT* topologies [2] reduce the memory requirements but occupy more area for computations especially for higher Radix ($> 2$) structures. Higher Radix techniques reduce the number of stages of the *FFT* but increase the VLSI area of each stage. As a rule of thumb, the power dissipation of the *FFT* processors improves with the increase of the level of parallelism. To achieve low power consumption, asynchronous *FFT* designs [11] utilize fully unrolled *FFT* circuits but occupy more *VLSI* area. The afore-mentioned organizations use *FFT* core with either Radix-2 or Radix-4. Radix-2 calculations are straightforward to implement while Radix-4 can reduce the number of multiplications and reduces the number of stages in the *FFT* computation.

This paper presents an efficient *FFT* architecture with respect to the sustained throughput, the memory utilization and power dissipation requirements. The architecture uses Radix-4 techniques and to sustain maximal throughput each Radix-4 stage is mapped onto a distinct Radix-4 circuit. To improve on the memory requirements, especially for large input data sets, the architecture combines three (3) Radix-4 circuits to result in a 64-point FFT engine. The latter engine has still the advantages of simplifying multiplications and furthermore, it requires only the one third of memory as compared to unfolded FFT architectures with radix-4 engines. The 64-point FFT engines can be easily combined to form efficient $N$-point FFT architectures, $N > 2^7$. Maximal achievable parallelism and bit-pipelining techniques result in low power operation and an easily expandable architecture with respect to data and/or coefficient word length. To show the efficiency of combining 64-point FFT engines in the paper we present the details of a 4096-point design.

Moreover, the 4096-point FFT has been realized as an example implementation on a Xilinx Virtex II *FPGA*. The architecture requires only two (2) memory banks, each of $2 \times 4096$ words depth. The implementation has a maximum operating frequency of 180 MHz (sustained throughput 22.76us) and consumes 3.2 Watts for a typical application at 100 MHz.

The paper is organized with the following Section describing the derivation of the Radix-$4^3$ schema from the *FFT* equation and Section 3 presents the *FFT* architecture and the details.

## 2 Analysis

The Discrete Fourier Transform (DFT) of a signal $x[n]$ of length $N$ is given by the series

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

where, the coefficients $W_N = e^{-j\frac{2\pi}{N}}$ are called the twiddle factors. The Fast Fourier Transform algorithm (FFT) [4] exploits the symmetry of the twiddle factors and significantly improves the computational complexity of the DFT. The FFT algorithm recursively decomposes the DFT series into partial sums using index maps. In this section we describe a multi-dimensional index map decomposition similar to that of [7], where the series is decomposed using a three-dimensional index maps. The resulting architecture of [7] is an improved radix-4 butterfly structure, which is called radix-$2^2$.

The architecture presented in this paper is based on a four-dimensional index map and on a radix-4 decomposition of the DFT series. This approach reduces the overall memory required for storing the intermediate results of the FFT and exploits the optimized structure of the radix-4 butterfly.

### 2.1 Index Maps

The implemented architecture was based on the four-dimensional index map

$$\begin{array}{rcl} n & = & n_1 + \frac{N}{64}n_2 + \frac{N}{16}n_3 + \frac{N}{4}n_4 \\ k & = & 64k_1 + 16k_2 + 4k_3 + k_4 \end{array} \qquad (1)$$

Applying 1 to the twiddle factors $W_N^{kn}$ yields

$$\begin{array}{rcl} W_N^{kn} & = & W_N^{\left[n_1+\frac{N}{64}n_2+\frac{N}{16}n_3+\frac{N}{4}n_4\right][64k_1+16k_2+4k_3+k_4]} \Rightarrow \\ W_N^{kn} & = & W_4^{n_4k_4} W_{16}^{n_3k_4} W_4^{n_3k_3} W_{64}^{n_2(4k_3+k_4)} \times \\ & & \times W_4^{n_2k_2} W_N^{n_1[k]} \end{array} \qquad (2)$$

Applying equation 2 to the DFT equation yields

$$\begin{array}{rcl} X[k] & = & \sum_{n_1=0}^{\frac{N}{64}-1} W_N^{n_1[k]} \sum_{n_2=0}^{3} W_4^{n_2k_2} \times \\ & \times & \left[ \sum_{n_3=0}^{3} W_4^{n_3k_3} \left( \sum_{n_4=0}^{3} x\left[n\right] W_4^{n_4k_4} \right) W_{16}^{n_3k_4} \right] \times \\ & \times & W_{64}^{n_2(4k_3+k_4)} \end{array} \qquad (3)$$

Equation 3 describes a radix-64 based FFT. Further, the equation describes the internal structure of the radix-64 butterfly, which is based on three radix-4 butterflies. To justify the above, we will refer to the radix-64 butterfly as radix-$4^3$ ($R4^3$). The radix-$4^3$ butterfly is a 64-point FFT and is described by the following equation

$$\begin{array}{rcl} R4^3 & = & \sum_{n_2=0}^{3} W_4^{n_2k_2} \times \\ & \times & \left[ \sum_{n_3=0}^{3} W_4^{n_3k_3} \left( \sum_{n_4=0}^{3} W_4^{n_4k_4} x[n] \right) W_{16}^{n_3k_4} \right] \times \\ & \times & W_{64}^{n_2(4k_3+k_4)} \end{array} \qquad (4)$$

As an example, to implement a 4096-point FFT using ($R4^3$) butterflies we use a two dimensional index map based on radix-64. This index map decomposes the 4096-point series into two sums using a radix-64 butterfly. Finally, we replace the radix-64 butterflies with the $R4^3$ to obtain the radix-$4^3$ based transform.

## 3 Architecture

This section presents the design and the details of the FFT architecture realizing a 4096-point FFT. The overall architecture of the 4096-point FFT is depicted in figure 1. The FFT processor consists of two ($R4^3$) processing cores, two 4096-word dual bank memory elements, a 4096 point read-only memory that stores the $W_{4096}$ twiddle factors, a complex multiplier and a control unit that synchronizes the individual elements. The following subsections describe in detail the most important modules of the architecture.

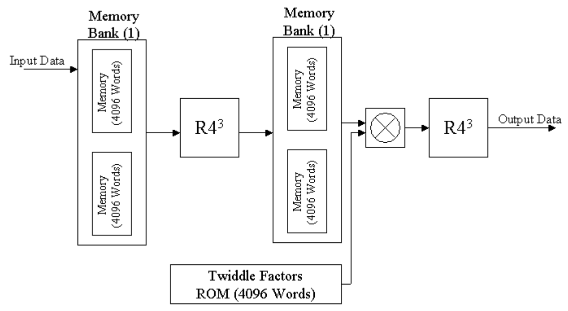

Figure 1: Overall FFT Architecture

### 3.1 $R4^3$ Engine

The internal structure of the $R4^3$ engines is depicted in figure 2. Each engine consists of three radix-4 butterflies, two complex multipliers, two double bank memory elements (one consisting of 2x16 words and one consisting of 2x64 words) and two Read Only Memories where the $W_{16}$ and $W_{64}$ twiddles are stored. Finally, the $R4^3$ *control unit* generates the signals that synchronize the individual modules. Each $R4^3$ engine operates as a stand-alone 64-point FFT.

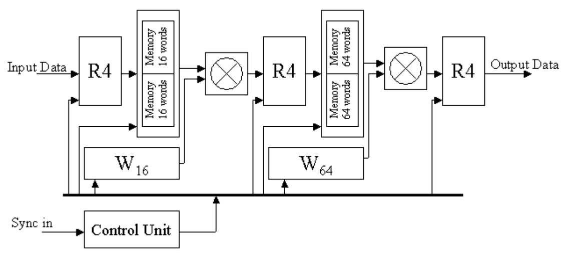

Figure 2: $R4^3$ Butterfly Architecture

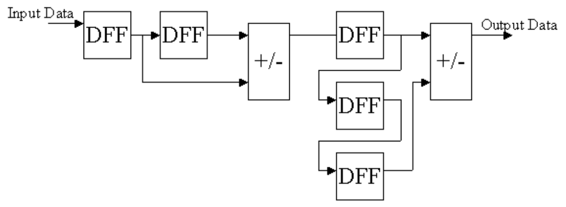

Figure 4: Accumulator architecture

## 3.2 radix-4 engine architecture

In this section we describe in detail the architecture of the radix-4 engine. Each radix-4 butterfly (Figure 3) consists of 4 distinct accumulators and four processing elements that swap the real and imaginary parts of each input value ("swap" elements). The input data are processed in parallel by the four accumulators and are output as a serial stream. The control unit synchronizes the operations of the swap elements, the accumulators and controls the output multiplexer.

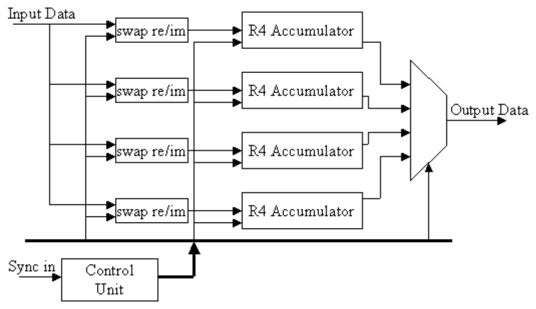

Figure 3: radix-4 Butterfly Architecture

The structure of the accumulator units is depicted in figure 4.To avoid single clock feedback elements, the accumulators use an "unrolled" architecture. The implemented architecture has the following advantages: First, the unrolled structure of the accumulator can operate in higher frequencies compared to an accumulator using feedback paths. Second, the radix-4 engine is easily configurable and can be expanded with respect to word-length.

## References

[1] A. Oppenheim, R. Schafer *"Digital Signal Processing"*, Prentice Hall 1975.

[2] Clark D. Thompson *"Fourier Transform in VLSI"*, IEEE Transactions on Computers, 1973.

[3] E.H. Wold and A.M. Despain *"Pipeline and Parallel FFT Processors for VLSI Implementations"*, IEEE Transactions on Computers, vol. C-33, 1984.

[4] J.W. Cooley and J.W. Tukey *"An algorithm for the machine calculation of complex Fourier series"*.

[5] J. Lee, J. Lee, M. H.Sunwoo, S. Moh and S. Oh *"A DSP Architecture for High-Speed FFT in OFDM Systems"*, ETRI Journal, 2002.

[6] S. He and M. Torkelson *"Design and Implementation of a 1024-point Pipeline FFT Processor"*, IEEE 1998 Custom Integrated Circuits.

[7] S. He and M. Torkelson *"A New Approach to Pipeline FFT Processor."*, Proceedings of the IPPS, 1996.

[8] G. Bi and E.V. Jones *"A pipelined FFT processor for word-sequential data"*, IEEE Trans. Acoust, Speech, Signal Processing, 37(12):1982-1985, Dec. 1989.

[9] S. Choi, G. Govindu, J.-W. Jang, V. K. Prasanna *"Energy-Efficient and Parameterized Designs of Fast Fourier Transforms on FPGAs"*, The 28th International Conference on Acoustics, Speech, and Signal Processing (ICASSP), April 2003.

[10] L. R. Rabiner and B. Gold *"Theory and Application of Digital Signal Processing"*, Prentice-Hall

[11] B. Suter and K. S. Stevens *"A Low Power, High Performance approach for Time-Frequency / Time-Scale Computations."*, Proceedings SPIE98 Conference on Advanced Signal Processing Algorithms, Architectures and Implementations VIII. Vol. 3461, pp. 86–90, July 1998.

[12] I. Saarinen, G. Coppola, A. Polydoros, J.L. Garcia, M. Lobeira, P. Dallas, M. Gertou, R. Cusani and G. Razzano *"High Bit Rate Adaptive WIND-FLEX Modem Architectures for Wireless Ad-Hoc Networking in Indoor Environments"*.