High Performance 16K, 64K, 256K complex points VLSI Systolic FFT Architectures

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

V.A. Chouliaras
Department of Electronic and Electrical Engineering,
Loughborough University, Loughborough,
LEICS, LE11 3TU, UK
Email: V.A.Chouliaras@lboro.ac.uk

Abstract—Targeting to improving the efficiency of real-time Fourier Transform computations with large input data sets, this paper presents the design and the VLSI implementation of 16K, 64K and 256K complex points Fast Fourier Transform (FFT) systolic architectures. These organizations are deeply pipelined to maximize the operating frequency and follow the approach of decomposing the transforms into 64-point FFT computations to minimize the buffer size between consecutive stages. The resulting organizations achieve real time performance on testing and observation applications. They include simple processing elements and they are scalable with respect to the operating frequency and data width. Validation on FPGA showed operation at 250MHz and 125MHz for the 16K and the 64K architectures with throughput 1Gs/s and 500Ms/s respectively. The VLSI implementations of the proposed 16K, 64K and 256K architectures achieve post-route clock frequencies of 352, 256.5, and 188 MHz respectively and they can sustain throughputs of 1.4Gs/s, 1Gs/s and 188Ms/s.

I. INTRODUCTION

A large number of signal processing applications, especially in the areas of testing and observation requires the real time computation of the Fourier transform involving large dataset inputs. These applications envisage problems of input size greater than $8K$, they request high throughput rates and they demand for lessening the memory size. Such problems depend on Application Specific Integrated Circuits (ASIC) for their solution [1]-[3]. Most of the techniques and FFT architectures that have been proposed to accommodate these needs [4]-[14] vary with respect to the level of parallelism, the throughput rate and the organization. The most common of the application specific architectures include the fully unfolded FFT realizations [15]. Such implementations occupy more VLSI area and utilize larger memory arrays between their successive stages compared to the latency and memory efficient cascade FFT topologies [4], [6], [16]. Cascade designs though, become complicated due to the different memory requirements on each stage. Also, in the cascade architectures the size of the memory and the area of the control circuits within each butterfly are proportional to the pipeline depth and they have to be designed according to a specific performance. Higher Radix techniques reduce the number of stages of the FFT at an increased cost in terms of VLSI area for each stage. An interesting alternative approach utilizes asynchronous FFT architectures [17] (fully unrolled FFT). To achieve a significant improvement of the throughput, current organizations include smaller size FFT computations that are executed in parallel systolic data paths [1]-[3] and they combine their outputs to result in the required FFT size. Organizing parallel paths multiplies the throughput by the number of these paths.

Addressing the problem of computing in real time FFT of input size greater than $8K$, this paper presents efficient, deeply-pipelined FFT architectures for the inputs of $16K$, $64K$ and $256K$ complex points. These designs maximize the throughput, while they keep the control and the memory organizations simple compared to the single and multiple path unfolded and the cascade FFT architectures. Moreover, they are highly efficient by offering advanced scaling of the maximum operating frequency and consequently, in terms of power consumption, pipeline depth and data and/or twiddle bit-widths, comparing to the previously mentioned architectures. In order to improve the memory requirements -particularly for very large input data sets (i.e. $64K$ and $256K$)- the proposed architectures use a 64-point FFT engine based on three radix-4 (R-4) circuits [18]. The R-4 schema enhances the designs because of its simpler computations compared to the other radices. It also reduces latency compared to the radix-2 (R-2).

The proposed 16K, 64K, 256K complex point architectures have been implemented on a high performance 0.13µm, 1Poly-8Copper layer standard cell technology from TSMC. The 16K FFT achieves a worst-case (0.9 V, 125 C) post-route clock frequency of 352 MHz and a sustained throughput of 1.4Gs/s (16384 complex points/11.6 usec) while consuming 3.45 Watt. The 64K FFT achieves a worst-case (0.9 V, 125 C) post-route clock frequency of 256.5 MHz and a sustained throughput of 1Gs/s (65536 complex points/16 usec) while consuming 9.8 Watt. Finally, the 256K FFT achieves a worst-case (0.9 V, 125 C) post-route clock frequency of 188 MHz and a sustained throughput of 188Ms/s (262144 complex points/1.39 msec) while consuming 35.75 Watt. Validation on FPGA showed operation at 125MHz and 250MHz for the $64K$ and the $16K$ architectures with throughput 500 Ms/s and 1 Gs/s respectively.

This paper includes 3 more sections.The following section describes the architecture of the $16K$ complex points FFT
architecture. Section III presents the 64\(K\) point FFT and the 256\(K\) point FFT architectures. Section IV gives the VLSI implementation details and the performance of the architectures.

II. 16\(K\) FFT Architecture

The FFT architectures proposed in this work use a radix-64 (R-64) engine realized by the radix-\(4^3\) (\(R - 4^3\)) algorithm [18], in which the linear index mapping transforms into a four dimensional index map as follows:

\[
\begin{align*}
    n &= n_1 + \frac{N}{64}n_2 + \frac{N}{64^2}n_3 + \frac{N}{64^3}n_4 \\
    k &= 64k_1 + 16k_2 + 4k_3 + k_4
\end{align*}
\]

(1)

Applying this equation into the DFT equation and expanding the summation with index \(n_4\), \(n_3\) and \(n_2\) yields a set of 64 DFTs of length \(N/64\).

A. 16\(K\) FFT Computations

The DFT equation for a 16\(K\) point FFT takes the form

\[
X[k] = \sum_{n=0}^{16383} x[n]W_{16384}^{nk} = \sum_{n=0}^{4096-1} x[n]W_{4096}^{nk}
\]

(2)

Setting \(n = n_1 + \frac{N}{64}n_2\) and \(k = k_14096 + k_2::

\[
k \cdot n = \left(n_1 + \frac{N}{4096}n_2\right) (4096k_1 + k_2) \Rightarrow \\
k \cdot n = 4096n_1k_1 + n_1k_2 + Nn_2k_1 + \frac{N}{4096}k_2 \Rightarrow \\
W_{4096}^{kn} = W_4^{n_1k_1}W_{4096}^{n_2k_2}
\]

Therefore, the transform becomes

\[
X[k] = \sum_{n_1=0}^{3} W_4^{n_1k_1} \left( \sum_{n_2=0}^{4095} x[n]W_{4096}^{n_2k_2} \right) W_4^{n_1k_2}
\]

(3)

The above decomposition leads to the 16\(K\) FFT architecture. According to equation 3, the 4K points FFT can be extended to 16K points. This is accomplished by first performing four consecutive 4K point FFT transforms. Next, the data is multiplied by the twiddle factors that correspond to a 16K points FFT and finally, a R-4 stage completes the 16K point FFT computation. The decomposition results in two architectures constituting straightforward applications of equation 3: The first, called sequential/parallel, uses a single “radix-4096” processor and a R-4 stage. The second, called parallel/parallel, involves four “radix-4096” processors in parallel and a R-4 stage. The following subsections describe the two architectures.

B. 16\(K\) FFT Sequential/Parallel Architecture

The sequential/parallel architecture uses a 4K FFT block (Figure 1). The 4K points FFT consists of two successive \(R - 4^3\) stages [18]. The 4K FFT block performs four consecutive transforms generating results stored to four 4K word memories. Then, the data is multiplied with the corresponding twiddle factors and processed by the parallel R-4 engine. To avoid the use of double buffers at the final stage of the algorithm we use an addressing scheme that replaces the existing elements in the memory banks with those produced by the R4 processors. The memory banks are arranged to use a read-before-write configuration and the same address sequence is used for both storing the outputs of the processors and reading the results of the previous stage. This technique complicates the digit reversal at the output of the 16K transform but it reduces the overall memory size. In an alternative mode, the data storage takes place in its respective digit reversed position at each memory, and performs the R-4 computations in the same digit-reversed order. By utilizing these techniques, a 16K point FFT transform uses a total of \(2 \times 2 \times 4K + 16K\) memory words, exceeding only by 16K more the memory required by the 4K FFT transform.

C. 16\(K\) FFT Parallel/Parallel Architecture

The parallel/parallel architecture is presented in Figure 2. In this case, there are four 4K FFT blocks operating in parallel. A parallel R-4 engine follows these blocks and operates on the results of the 4K FFTs. The main advantage of this approach is that it requires almost the same time to complete a 16K FFT as the 4K FFT. However, this improvement comes at an increase of the VLSI area compared to the serial/parallel architecture. The parallel/parallel architecture has been implemented in VLSI (and FPGA) as described in section IV.
III. 64K AND 256K FFT ARCHITECTURES

A. 64K FFT Architecture

The DFT equation for a 64K point FFT takes the form

\[
X[k] = \sum_{n=0}^{65535} x[n]W_n^{nk} = \sum_{n=0}^{16384-1} x[n]W_n^{nk} \tag{4}
\]

Setting \( n = n_1 + \frac{N}{16384} n_2 \) and \( k = k_116384 + k_2 \):

\[
k \cdot n = 16384 n_1 k_1 + n_1 k_2 + N n_2 k_1 + \frac{N}{16384} k_2 = 4 \cdot 16384 k_2
\]

Therefore, the transform becomes

\[
X[k] = 3 \sum_{n_1=0}^{1} \left( \sum_{n_2=0}^{16383} x[n] \right) W^{n_2 k_2} \tag{5}
\]

According to equation 5, the 16K points FFT can be extended to 64K points. This is accomplished by first performing a 16K point FFT transform (section II). Next, the data is multiplied by the twiddle factors that correspond to a 64K point FFT transform. Finally, a R-4 stage completes the 64K point FFT computation, as shown in Figure 3. The 64K FFT has been VLSI (and FPGA) implemented by using the 16K parallel/parallel computation.

B. 256k FFT Architecture

Figure 4 depicts the 256K FFT architecture, which is a straightforward application of the \( R - 4^3 \) algorithm. Three consecutive \( R - 4^3 \) engines are used to accomplish the task.

IV. VLSI IMPLEMENTATION AND PERFORMANCE EVALUATION

The 16K parallel/parallel, the 64K and the 256K FFT architectures were implemented in a high performance, 0.13 um, 1Poly-8Copper layer standard cell technology from TSMC. In the case of the 16K FFT and 64K FFT, a flat back-end flow was used in which the design was first synthesized to gates using Synopsys Design Compiler and optimized for a frequency of 300 MHz. The uniquified netlists were then read into Cadence SoC Encounter where floorplanning, power planning, clock-tree-synthesis, placement, routing and IPO routing took place. In the case of the very large 256K FFT, tool capacity mandated the use of a hierarchical flow. Following the same front-end synthesis process (again optimized for 300 MHz), the optimized netlist was read into Cadence SoC encounter where partitioning was first performed. This process created six instances of the 256K memory block (for a total of 1.5 MB of on-chip SRAM) which were individually placed and routed. The same process was performed for the \( R - 4^3 \) engines and the twiddle ROM block. Note that to complete the entire FFT computation it requires a 3-frame latency (786792 cycles) and the computation latency within the three \( R - 4^3 \) (360 cycles), which is 4.1 msec. Finally, all designs were brought back into the top-level for final top-level placement, routing and timing analysis. Figures 5, 6 and 7 depict the three VLSI Cells for the 16K, 64K and 256K FFT designs respectively.
TABLE I
VLSI IMPLEMENTATION ROUTING RESULTS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>FFT 16K</th>
<th>FFT 64K</th>
<th>FFT 256K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total μm²</td>
<td>1.6561e + 07μm²</td>
<td>3.4894e + 08μm²</td>
<td>2.7647e + 08μm²</td>
</tr>
<tr>
<td>Std Cell Rows</td>
<td>1102</td>
<td>1600</td>
<td>4500</td>
</tr>
<tr>
<td>Number of Cells</td>
<td>885456</td>
<td>896148</td>
<td>735945</td>
</tr>
<tr>
<td>Number of RAMs</td>
<td>64</td>
<td>192</td>
<td>384</td>
</tr>
<tr>
<td>Statistical Power</td>
<td>3.45 W</td>
<td>9.8 W</td>
<td>35.75 W</td>
</tr>
<tr>
<td>Fmax</td>
<td>352 MHz</td>
<td>256.5 MHz</td>
<td>188 MHz</td>
</tr>
</tbody>
</table>

frequency of 250MHz and it uses 12264 slices. The architecture has a four data parallel input and output and sustains a throughput of 1G/s (28 Gbits/sec).

The 64K (using the 16K parallel/parallel computation) has been implemented on the Xilinx Virtex 5 (-2) achieving an operating frequency of 125MHz and it uses 13461 slices. The architecture has a four data parallel input and output and sustains a throughput of 500Ms/s (14 Gbits/sec).

REFERENCES