FAST DISCRETE FOURIER TRANSFORM IN FPGA BY USING THE METHOD OF FAST PAIRED TRANSFORM

Tilak Nagaraju and Artyom M. Grigoryan
Department of Electrical and Computer Engineering
University of Texas at San Antonio, San Antonio, Texas 78249, USA
E-mail: {tilak.nagaraju, amgrigoryan}@utsa.edu

Abstract

In this paper an effective design and implementation of the fast Fourier transform (FFT) by the paired transforms is presented and compared with the existent radix-2 algorithm. A block level design of the fast transform methods is implemented and tested in this research work. Discussed is a possibility of reducing the arithmetic computations, hardware utilization and the number of clock cycles in the FFT process that ultimately results in an optimized FFT design and thereby increases the overall speed-up. It is shown that the method of the fast paired-transform utilizes less hardware for higher sampling rates and is best suited for FFT designs in FPGA where speed, area, and cost are the major factors. An in-depth count of arithmetic operators involved in the 4-point up to the 64-point FFT is tabulated in this paper, which gives a clear comparison in choosing the best fast-transform methods. The signal-flow graph for the 16-point FFT is considered for detailed explanation and analysis of the FFT methods. As the development of fast digital signal processing (DSP) algorithms and their implementation in FPGA is a field of great interest, all the designs discussed in this paper are targeted for high performance FFT implementations in FPGA.

Introduction

Verilog is often used to design hardware that implements complicated and numerically-intensive algorithms. Examples include matrix transformations used in graphics, the FFTs, the hidden Markov model (HMM) in speech recognition, the back propagation (BP) algorithm in the Neural Nets (NN) and the discrete cosine transform (DCT) in image compression [11]. The ground level description and programming of the designs presented in this paper work is in VerilogHDL. The 4-, 8-, 16-, 32, 64-point FFTs are compared for both radix-2 and paired transform algorithms. Each of the designs is analyzed for computational time and hardware utilization that contributes mainly towards the performance of the design. The paper discusses the count of each of the arithmetic operators involved in the fast transform methods that gives a clear idea of the operational blocks involved in the designs. Compared to the implementation technique done in the past [13] for the paired transform methods, this paper elaborates the paired transform methods for the lower N-point FFTs. The twiddle factor (TF) values required for the FFT process are supplied through the pre-declared constants. These values have been previously calculated in MATLAB and, then, stored in registers or ROM blocks. The modeling of the N-point FFT units is done by using MATLAB-Simulink blockset and Xilinx-Simulink blocksets. Xilinx system generator software tool is used to test the designs. Different styles of implementing the FFT processor designs like the parallel/pipelining technique [1] are not compared in this paper as the designs were implemented using the block level
features in the Xilinx Sys-Gen software. These optimization techniques for high speed FFTs can also be applied and tested for the paired transform methods with a careful consideration of the main algorithm that will be a new challenge to the FFT design engineers. Furthermore, the simulation results for the method of fast paired transform show that there is a considerable overall performance improvement when implementing in FPGA hardware.

**Fast Fourier Transforms**

The discrete Fourier transform (DFT) is the most widely used tool for transforming the signals from the time domain to their representation in the frequency domain. Processing these signals in today’s DSP kits requires the signals to be in discrete form. The N-point DFT, \( F(k) \), of the signal, or sequence \( f(n) \) of length \( N \) is calculated by

\[
F(k) = (F_N \circ f)(k) = \sum_{n=0}^{N-1} f(n)W_N^{kn},
\]

where \( W_N = e^{-j2\pi/N} \), and \( j^2 = -1 \). In general, the data sequence \( f(n) \) is assumed to be a complex value. The inverse DFT (IDFT) is calculated by

\[
f(n) = \frac{1}{N} \sum_{k=0}^{N-1} F(k)W_N^{-nk}, \quad n = 0,1,2,...,(N-1).
\]

Since DFT and IDFT involve similar type of butterfly computations, the discussions in this paper can be extended to both forward and backward DFT processes. In the radix-2 method, the signal-flow graph is symmetric and all results for the DFT process is calculated in the last stage of the algorithm, whereas in the paired transform method the signal-flow graph is completely different from the radix-2 method and the results are calculated in much earlier stages of the algorithm. If the signal \( f(n) \) is real, the calculations can be reduced and the graph can be simplified in the last stage, if consider the property of the complex conjugate, \( F(N-k) = F^*(k) \), for \( k = 1, 2, ..., N/2 - 1 \).

The fast paired transform based FFT [4]-[6] is based on the orthogonal discrete paired transform which transfers the signal \( f(n) \) of length \( N = 2^r \), \( r > 1 \), to the set of \( (r+1) \) separate splitting-signals of length \( N/2, N/4, ..., 2, 1 \). The paired transform is binary and allows for calculating the \( N \)-point DFT with the minimum number of operations of multiplication by twiddle factors, which equals \( N/2(\log_2(N) - 3) + 2 \). The paired algorithm is effective and can also be used for calculation of the Hadamard, Hartley, and cosine transforms [7]-[9]. The signal-flow graph of the paired algorithm for calculating the 16-point DFT is given in Figure 1. In the Cooley-Tukey FFT, the splitting of the \( N \)-point DFT is performed by equal parts, i.e. on the first step; the transform is split as \( F_N \sim \{F_{N/2}, F_{N/2}\} \). Then each \( N/2 \)-point DFT is split in the same way, \( F_{N/2} \sim \{F_{N/4}, F_{N/4}\} \), and this process is continued \( \log_2(N) - 2 \) more times. In the fast paired transform algorithm [6], the splitting of the \( N \)-point DFT is performed differently, as shown in Figure 2 for the \( N = 16 \) case. On the first step, the transform is split as

\[
F_N \sim \{F_{N/2}, F_{N/4}, F_{N/8}, ..., F_{N/2}, 1, 1\}
\]
Figure 1. Signal-flow graph for the 16-point FFT by the paired transform

Then each short $N/2^k$-point DFT, where $k = 1: \log_2(N)−2$, is split in the same way. The process of splitting is faster than in the Cooley-Tukey method and is performed by the paired transform. The $N$-point paired transform is fast and requires only $2N − 2$ operations of addition/subtraction [20].

Figure 2. Signal-flow graph for the 16-point FFT by the fast paired transform
Circuit Implementation

Consider the single basic Cooley-Tukey butterfly \((a, b) \rightarrow (a + bW^k, a - bW^k), k \in \{1, 2, \ldots, N/2 - 1\}\) which is used in designing the FFT butterfly in FPGA, MATLAB-Simulink etc [12]-[16]. The butterfly operation is the main unit on which the speed of the whole process of the FFT depends. The faster the butterfly operation, and faster is the FFT process [13]. The adders and subtractors are implemented using the LUTs (distributed arithmetic) [14]. Different implementation techniques have been employed in the past research papers [3], [13], [15] and an effort is made to design an optimized FFT architecture. In relating to these research works, it can be shown that the paired transform based FFT will be a suitable method to address the issue of lower latency in completing the DFT process much earlier in time.

Figure 3. 4-point FFT design by radix-2 transform in Xilinx Sys-Gen

FFT models are designed using the Xilinx-Simulink block sets and implemented using an automated synthesis. The number of clock cycles for each design is calculated through the timing report which is generated during the synthesis from the timing analyzer. In our approach, the unnecessary re-declaration of registers and multipliers in the initial stages of the FFT designs is avoided. As the multiplication of the twiddle factor coefficient value \(-j\) (in 4-point FFT block) to a complex value \(A = (a + bj)\) results in \(A = (a + bj) * (-j) = (b - aj)\), which is \(a\) and \(b\) interchanged with sign change on \(a\). This can be hardwired by cross feeding the input signals with a sign change on one of them avoiding the use of a complete complex multiplier module. As the 4-point FFT blocks can be used in the hierarchy of the higher N-point FFTs, this method of designing would reduce the number of actual multipliers. Figure 5 shows the block diagram for calculating the 16-point FFT by the paired transform. This model gives a complete overview of the architectural design which is used to model the paired-transform based FFTs in Xilinx Sys-Gen. It shows how the lower N-point models are used in the design. Figure 4 shows the 4-point FFT design by radix-2 method implemented in Xilinx Sys-Gen.
From the post route synthesis report, we calculate the hardware utilization percentage in FPGA [2]. The automated synthesis also generates Verilog netlist which can be used in Cadence-Encounter tool for drawing the layout and finally implementing the FFT on chip. Figure 4 shows the implementation of the paired-transform. The higher cases of N are implemented and synthesized in the similar way. The higher N-point FFT make use of lower 4-, 8-, 16-point models and so on, as a recurring block in the main design. As the graphic size of the figures concerning the higher N-point designs are beyond the scope of this page, shown is only the lower N-point model for clarity purposes.
Tools and Methodology

The advantage of Xilinx System-Generator tool is that it provides a bridge between the MATLAB and Verilog designs. With Simulink blockset and the Xilinx Verilog blocksets it becomes easier to model the DSP related algorithms [2]. The blockset (adders, multipliers, and subtractors) are configured for signed (2 comp) 32 bit and 30-bit binary point integers. For the arithmetic units of quantization error the truncate option is used, and for the overflow the saturate option is used. Had the outputs of the arithmetic units considered as full precision, the output width would have grown exponentially and would have required huge memory. When the user-defined precision is selected, errors may result from the overflow or quantization. The system generator blockset provides various means to handle such situation. The truncate option does not cost any additional hardware whereas the overflow operation utilizes additional hardware (an adder). The block-set consists of gateway blocks which converts inputs of type SIMULINK integer, double and fixed point to Xilinx fixed point type. On hardware these gateway blocks become the top level input ports for the complete design. Fixed point integer format is used in this design methodology and the constants and integers are limited to ±15.

Simulation Results

The 4-point model is a basic building block for the higher N-point models. Once this model is designed, it can further be utilized in modeling the 8-point FFT, the 16-point FFT, and so on. The theoretical calculation of arithmetical operators for 4-, 8-, 16-, 32- and 64-point designs by radix-2 and paired transforms is shown in Table 1.

Table 1. Arithmetic operators, multiplication (M), addition (A), and subtraction (S) used in the FFT process.

<table>
<thead>
<tr>
<th>FAST TRANSFORMS</th>
<th>M</th>
<th>A</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-point radix-2 FFT</td>
<td>0</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>4-point paired FFT</td>
<td>0</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>4-point fast paired FFT</td>
<td>0</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>8-point radix-2 FFT</td>
<td>6</td>
<td>28</td>
<td>30</td>
</tr>
<tr>
<td>8-point paired FFT</td>
<td>6</td>
<td>38</td>
<td>44</td>
</tr>
<tr>
<td>8-point fast paired FFT</td>
<td>6</td>
<td>28</td>
<td>30</td>
</tr>
<tr>
<td>16-point radix-2 FFT</td>
<td>36</td>
<td>88</td>
<td>100</td>
</tr>
<tr>
<td>16-point paired FFT</td>
<td>30</td>
<td>134</td>
<td>132</td>
</tr>
<tr>
<td>16-point fast paired FFT</td>
<td>30</td>
<td>84</td>
<td>94</td>
</tr>
<tr>
<td>32-point radix-2 FFT</td>
<td>120</td>
<td>240</td>
<td>380</td>
</tr>
<tr>
<td>32-point paired FFT</td>
<td>102</td>
<td>344</td>
<td>370</td>
</tr>
<tr>
<td>32-point fast paired FFT</td>
<td>102</td>
<td>228</td>
<td>262</td>
</tr>
<tr>
<td>64-point radix-2 FFT</td>
<td>336</td>
<td>608</td>
<td>720</td>
</tr>
<tr>
<td>64-point paired FFT</td>
<td>291</td>
<td>882</td>
<td>963</td>
</tr>
<tr>
<td>64-point fast paired FFT</td>
<td>291</td>
<td>544</td>
<td>641</td>
</tr>
</tbody>
</table>

From this table one can see that, in the paired transform method of FFT, the number multipliers reduce as N increases when compared with the radix-2 FFT designs. Whereas the fast paired
transform [5]-[10] shows a considerable reduction in the number of additions, subtractions, and multiplications in the FFT design, that makes it suitable for higher $N$-point implementation purposes. Figure 6 show the hardware utilization and the clock cycles count for the FFT designs when implemented in FPGA. All designs are simulated at 80MHz frequency. It is evident from the above set of data that as the sampling rate increases the fast paired transform shows the better results and optimized performance. The paired transform method of FFT has a lower latency and it can complete the DFT process much earlier than radix-2 FFT. Considering a different implementation technique was shown in the research paper [13] the paired transform based algorithm FFT is better applicable to higher frequency signals than the radix-2 FFT. The current implementation can further be improved while implementing on the DSP if the MAC engines are used explicitly, then there may be a possibility of better comparison between the algorithms. The hardware costs in designing memory and complex multipliers in the fast paired transform method can be saved by means of delay feedback and data scheduling approaches.

![Hardware Utilization](image)

![Performance Comparison](image)

Figure 6. Fast Transforms Performance Comparisons

**Conclusion**

The FFT evaluation discussed in this paper shows that the fast paired transform is an efficient butterfly implementation and can be used for improving the FFT performance in FPGA. With the distributed multipliers and fully pipelined stages for higher sampling rates, the fast paired transform could be the best suited technique for FFT implementation in FPGAs as it will definitely offer higher throughput. Also for higher sampling rates like 128-, 256-,512-,1024-point FFTs, one can specifically design a split architecture based FFT processor with distributed multipliers and fully pipelined stages from performance and resource point of view using the lower N-point FFT modules. As these days the FPGA is becoming a key interest in VLSI field, such an efficient method is well suited for optimized and cost effective design implementations on hardware. In military applications only some of the DFT result coefficients are needed earlier and this can be achieved by the fast paired transform based algorithm FFT as it calculates many of the resultant coefficients earlier than the total result is finalized (when compared with the radix-2 FFT).
References