Implementations and Optimizations of Pipeline FFTs on Xilinx FPGAs

Bin Zhou[1, 2], David Hwang[2]
[1]*Dept. of EE, Tsinghua University, China* [2]*Dept. of ECE, George Mason University, USA*
{bzhou,dhwang}@gmu.edu

Abstract

This paper presents optimized implementations of two different pipeline FFT processors on Xilinx Spartan-3 and Virtex-E FPGAs. Different optimization techniques and rounding schemes were explored. The implementation results achieved better performance with lower resource usage than prior art. The 16-bit 1024-point FFT with the R2^{2}SDF architecture had a maximum clock frequency of 95.2 MHz and used 2802 slices on the Spartan-3, a throughput per area ratio of 0.034 Msamples/s/slice. The R4SDC architecture ran at 123.8 MHz and used 4409 slices on the Spartan-3, a throughput per area ratio of 0.028 Msamples/s/slice. The R2^{2}SDF was more efficient than the R4SDC in terms of throughput per area due to a simpler controller and an easier balanced rounding scheme. This paper also shows that balanced stage rounding is an appropriate rounding scheme for pipeline FFT processors.

1. Introduction

The Fast Fourier Transform (FFT), as an efficient algorithm to compute the Discrete Fourier Transform (DFT), is one of the most important operations in modern digital signal processing and communication systems. The pipeline FFT is a special class of FFT algorithms which can compute the FFT in a sequential manner; it achieves real-time behavior with non-stop processing when data is continually fed through the processor. Pipeline FFT architectures have been studied since the 1970's when real-time large scale signal processing requirements became prevalent. Several different architectures have been proposed, based on different decomposition methods, such as the Radix-2 Multi-path Delay Commutator (R2MDC) [1], Radix-2 Single-Path Delay Feedback (R2SDF) [2], Radix-4 Single-Path Delay Commutator (R4SDC) [3], and Radix-2^2 Single-Path Delay Feedback (R2^{2}SDF) [4]. More recently, Radix-2^2 to Radix-2^4 SDF FFTs were studied and compared in [5], and in [6] an R2^{3}SDF was implemented and shown to be area efficient for 2 or 3 multi-path channels. Each of these architectures can be classified as multi-path or single-path. Multi-path approaches can process *M* data inputs simultaneously, though they have limitations on the number of parallel data-paths, FFT points, and radix. In this paper we focus on single-path architectures.

From the hardware perspective, Field Programmable Gate Array (FPGA) devices are increasingly being used for hardware implementations in communications applications. FPGAs at advanced technology nodes can achieve compatible performance with ASICs, while having more flexibility, faster design time, and lower cost. As such, FPGAs are becoming more efficient and attractive for FFT processing and are the target platform of this paper.

The primary goal of this research is to optimize pipeline FFT processors to achieve better performance and lower cost than prior art implementations. In this paper, we present two comparative implementations (R4SDC and R2^{2}SDF) of pipeline FFT processors targeted towards Xilinx Spartan-3 and Virtex-E FPGAs. Different parameters such as throughput, area, and SQNR are compared.

The rest of the paper is organized as follows. Section 2 discusses the methodology used to select the two architectures. Section 3 describes the implementation tools and optimization methods used to improve performance and reduce resource utilization. Section 4 explains the balanced rounding schemes that were implemented and their impact on the signal-to-quantization noise ratio (SQNR). Section 5 presents the results and Section 6 presents some brief conclusions.

2. Pipeline FFT Architectures

2.1. Architecture Selection

The major characteristics and resource requirements of several pipeline FFT architectures are listed in Table

978-0-7695-3474-9/08 $25.00 © 2008 IEEE
DOI 10.1109/ReConFig.2008.56

Table 1. Hardware resource requirements comparison of pipeline FFT architectures (based on [4]).

	Complex Multipliers	Complex Adders	Memory Size	Control Logic	Comp. Efficiency	
					add/sub	Multiplier
R2SDF	$\log_2$N-2	2$\log_2$N	N-1	simple	50%	50%
R4SDF	$\log_4$N-1	8$\log_4$N	N-1	medium	25%	75%
R4SDC	$\log_4$N-1	3$\log_4$N	2N-2	complex	100%	75%
R2^2SDF	$\log_4$N-1	4$\log_4$N	N-1	simple	75%	75%
R2MDC	$\log_2$N-2	2$\log_2$N	3N/2-2	simple	50%	50%
R4MDC	3($\log_4$N-1)	8$\log_4$N	5N/2-4	medium	25%	25%

1. As shown in the table, the radix-4 Single-Path Delay-Commutator (R4SDC) and radix-2^2 Single Path Delay Feedback (R2^2SDF) architecture provide the highest computational efficiency and were selected for implementation. The R4SDC architecture is appealing due to its computational efficiency of addition; however the controller design is complex. The R2^2SDF architecture has a simple controller but less efficient addition scheme. These designs are both radix-4 and scalable to an arbitrary FFT size N (N is a power of 4).

2.2. R4SDC Architecture

R4SDC Algorithms: The R4SDC was proposed by Bi and Jones [3] and uses an iterative architecture to calculate the radix-4 FFT. The key to the algorithm is splitting the FFT into different stages by using different radices. In our design, the radix is always 4.

The derivation starts from the fundamental DFT equation for an N-point FFT:

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{\ nk}\ \ (k=0,1...N-1\ ;\ W_N=e^{-j(2\pi/N)}) \quad (1)$$

We represent N as composite of v numbers $N=r_1r_2...r_v$ and define

$$N_t=N/(r_1r_2....r_t),1\le t\le v-1 \quad (2)$$

where t is the stage and r_t is the stage radix. After putting eqn. (2) into eqn. (1) and applying the relationship $W_{N_iN_j}^{N_jk}=W_{N_i}^{\ k}$, eqn. (1) becomes:

$$X(k)=\sum_{q_1=0}^{N_1-1}W_N^{\ q_1k}\sum_{p=0}^{r_1-1}x(N_1p+q_1)W_{r_1}^{\ pk} \quad (3)$$

Indexes k_1 and m_1 can be defined by $k=r_1k_1+m_1$, where $0\le k_1\le N_1-1, 0\le m_1\le r_1-1$. Eqn. (3) becomes:

$$X(r_1k_1+m_1)=\sum_{q_1=0}^{N_1-1}x_1(q_1,m_1)W_{N_1}^{\ q_1k_1} \quad (4)$$

$$x_1(q_1,m_1)=W_N^{\ q_1m_1}\sum_{p=0}^{r_1-1}x(N_1p+q_1)W_{r_1}^{\ pm_1} \quad (5)$$

Therefore the complete N-point DFT can be written as v-1 different stages with intermediate stages in a recursive equation:

$$x_t(q_t,m_t)=W_{N_{t-1}}^{q_tm_t}\sum_{p=0}^{r_t-1}x_{t-1}(N_tp+q_t,m_{t-1})W_{r_t}^{\ pm_t} \quad (6)$$

$W_{N_{t-1}}^{q_tm_t}$ is the twiddle factor. For radix-4, the equations become:

$$\begin{aligned}X(4k_1+m_1)&=\sum_{q_1=0}^{N/4-1}x_1(q_1,m_1)W_{N/4}^{\ q_1k_1}\\ x_t(q_t,m_t)&=W_{N_{t-1}}^{q_tm_t}\sum_{p=0}^{3}x_{t-1}(N_tp+q_t,m_{t-1})W_4^{\ pm_t}\end{aligned} \quad (7)$$

The R4SDC architecture is presented in Fig. 1-3. An N-point radix-4 pipeline FFT is decomposed to $\log_4N$ stages. Each stage consists of a commutator, a butterfly, and a complex multiplier. Fig. 1 outlines the commutator for the R4SDC. Its six shift registers provide N_t delays. The control signals are generated by logic functions. The butterfly element, shown in Fig. 2, performs the summation, where trivial multiplication is replaced by add/sub and imaginary/real part swapping. Fig. 3 shows the overall architecture.

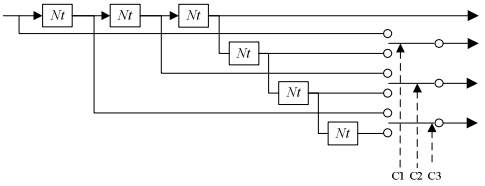


Figure 1. R4SDC commutator of stage *t*.

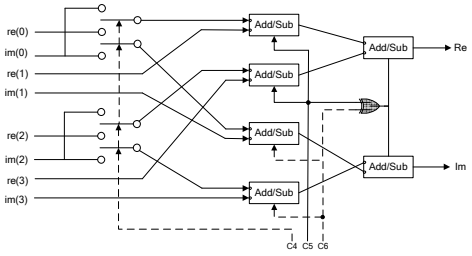


Figure 2. R4SDC butterfly element of stage *t*.

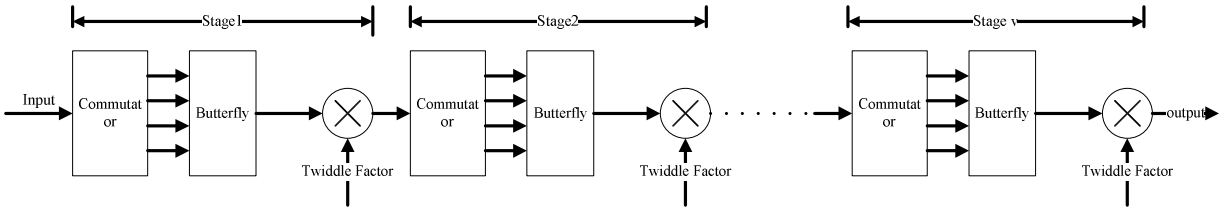


Figure 3. *N*-point R4SDC pipeline FFT processor architecture.

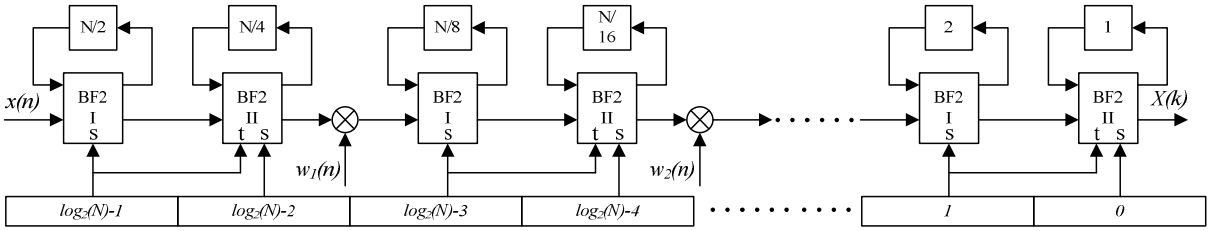


Figure 4. *N*-point R2^{2}SDF pipeline FFT processor architecture.

2.3. R2^{2}SDF Architecture

The R2^{2}SDF architecture was proposed by He and Torkelson [4] and also begins from eqn. (1). He applies a 3-dimensional index map:

$$n = < \frac{N}{2} n_1 + \frac{N}{4} n_2 + n_3 >_N ; \quad k = < k_1 + 2k_2 + 4k_3 >_N \qquad (8)$$

Using the Common Factor Algorithm (CFA) to decompose the twiddle factor, the FFT can be reconstructed as a set of 4 DFTs of length *N*/4:

$$X(k_1 + 2k_2 + 4k_3) = \sum_{n_3=0}^{N/4-1} [H(k_1, k_2, n_3) W_N^{n_3(k_1+2k_2)}] W_{\frac{N}{4}}^{n_3 k_3} \qquad (9)$$

$H(k_1, k_2, n_3)$ can be expressed as:

$$H(k_1,k_2,n_3) = \overbrace{\underbrace{[x(n_3) + (-1)^{k_1} x(n_3 + \tfrac{N}{2})]}_{BF2\ I} + (-j)^{(k_1+2k_2)} \underbrace{[x(n_3 + \tfrac{N}{4}) + (-1)^{k_1} x(n_3 + \tfrac{3}{4}N)]}_{BF2\ I}}^{BF2\ II} \qquad (10)$$

The R2^{2}SDF algorithm can be mapped to the architecture shown in Fig. 4-6. The number of stages is $\log_4 N$. Every stage contains two butterfly elements, each associated by an N_t feedback shift register. A simple counter creates the control signals. Pipeline registers can be added between butterfly elements and between stages. In our implementation, they are also added inside the complex multipliers to reduce the critical path through the summation to the multiplier. Hence there is a total latency of N+4($\log_4 N$-1) cycles.

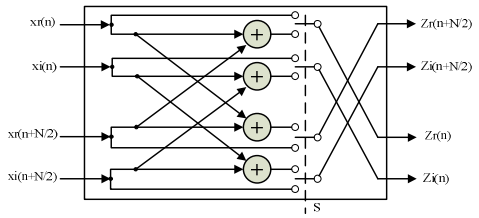


Figure 5. R2^{2}SDF BF2 I structure

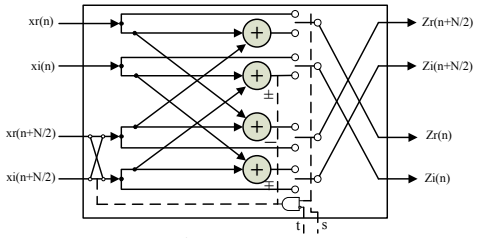


Figure 6. R2^{2}SDF BF2 II structure.

3. FPGA Based Implementations and Optimizations

3.1 Specifications, Tool Flow, and Verification

All the designs were implemented with generic synthesizable VHDL code and verified with simulation against Matlab scripts using Modelsim. Synplify or XST was used to perform the synthesis and ISE was used for place and route and implementation. The architectures were optimized to achieve maximum throughput with minimal area (slices).

Table 2. Implementation tools.

Design Step	Tool
VHDL Simulation	ModelSim SE 6.2b
FPGA Synthesis	Synplicity Synplify Pro Xilinx XST
FPGA Implementation	Xilinx ISE 9.1
Target FPGA	Spartan-3 Family Virtex-E Family
Verification	Matlab R2006a

3.2 General Optimization Methods

Some general optimization measures were performed, including FSM encoding, retiming and CAD-related optimizations. Since the FFT processors were targeted to Xilinx Spartan-3 and Virtex-E FPGAs, the SRL16 component, which can implement a 16-bit shift register within a single LUT, was inferred as much as possible to preserve LUTs. This particularly helped the R2^{2}SDF architecture because of the large number of shift registers. R4SDC also benefited from SRL16 components in its commutator registers. Block RAMs were used to store twiddle factors, which dramatically reduced the combinational logic utilization.

3.3 Architecture-Specific Optimization

A number of architecture-specific optimizations were used. For both architectures, a complex multiplication technique was used. Usually, a complex multiplication is computed as:

$$(a+bi)\times(c+di) = a\times c - b\times d + (a\times d + b\times c)i$$

This requires 4 multiplications and 2 add/sub operations. As is well-known, we simplified the equation to save one multiplier:

$$(a+bi)\times(c+di) = [a\times(c+d)-(a+b)\times d]+[a\times(c+d)+(b-a)\times c]i$$

This requires only 3 multiplications and 5 add/sub operations.

R4SDC: The R4SDC has a complex controller, which creates a long critical path. By observing that all stages have the same control bits but have different sequences, using a ROM with an incremental address was a simpler solution than using a complex FSM. Pipeline registers were also added to the butterfly elements, multipliers, and between stages.

R2^{2}SDF: Due to its simple control requirements, a counter was sufficient as a controller for the R2^{2}SDF; thus a fast adder could potentially be faster than a simple ripple-carry adder. However, due to the small number of stages ($\log_4 N$), no substantial savings were found for a fast adder. Pipeline registers were added between the elements and also between stages. Note that the R2^{2}SDF is not suited to adding pipeline registers within individual butterfly elements, because this would break the timing for the data feedback path.

4. Rounding and SQNR

Due to finite wordlength effects, the implemented FFTs always scaled by $1/N$ at the output of the design. This scaling factor was distributed as divide by two operations throughout each stage to reduce error propagation. As is well known, truncation or conventional rounding (which we denote as round-half-up) will bring a notable quantization error bias in divide by two operations, and this bias will accumulate throughout the processing chain [5]. To alleviate the bias, we investigated three unbiased rounding methods for division by two.

Sign Bit Based Rounding: In this scenario, if the MSB of the number to be divided is 0 (i.e. positive numbers) it is rounded-half-up. This will have a positive bias. On the other hand, if the MSB is 1 (i.e. negative numbers) it is truncated, leaving a negative bias. If we assume the positive and negative numbers are uniformly distributed, this approach will lead to unbiased results. However, selecting the bias based on the MSB implies that these two rounding methods coexist in a single rounding position, which requires extra hardware. This may increase the critical path, harming the performance.

Randomized [7]: In this scenario, if the bit to be rounded is 1, a random up or down rounding is performed. If it is 0, the same rounding scheme as done previously is performed. From the statistical point of view no bias exists, but this method requires a random bit generator and a long accumulation time.

Balanced Stages Rounding [8]: This rounding method explores balancing between stages. Round-half-up and truncation are used in an interlaced way, as shown in Fig. 7.

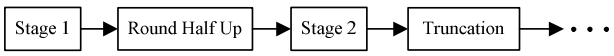


Figure 7. Balanced stages rounding.

For an even number of stages, this will achieve the same results as the randomized approach, while having a smaller resource usage and simpler control. This scheme fits the R2^{2}SDF architecture particularly well, because the two butterfly elements within same stage of R2^{2}SDF can be naturally balanced. This method was chosen for the designs presented in the paper.

In order to compute the signal-to-quantization noise ratio (SQNR), random generated noise was used as the input to the pipeline FFT. A Matlab script generated double precision floating point FFT results, which were used as the true values. Fig. 8 shows how they are compared. Random experiments were run several times and averaged to get a better error approximation.

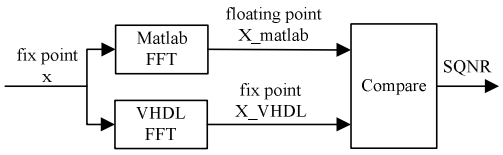


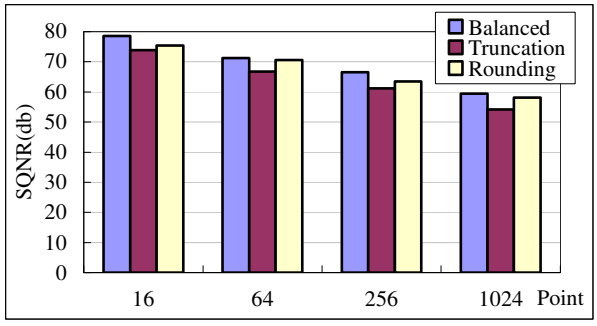
Figure 8. SQNR calculation.

5. Results and Analysis

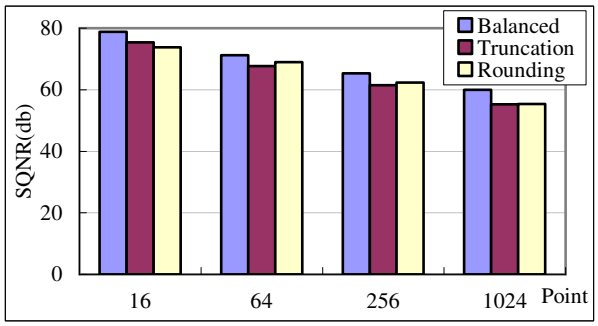
5.1. SQNR Results

Fig. 9 shows the SQNR results with different rounding schemes (truncation, round-half-up, balanced stages), for R4SDC and R2^{2}SDF respectively for a 16-bit data width. The balanced stage rounding typically improved the SQNR by 1-2 dB.

Table 3 presents the SQNR results varying with FFT size. The larger the FFT, the worse the SQNR is due to the longer processing chain. Both architectures gave compatible results. It is clear that larger data widths will also give better SQNR, but will increase area and critical path. A 16-bit bit width is a sufficient choice for many signal processing applications.



(a) R4SDC



(b) R2^{2}SDF

Figure 9. Rounding effects on SQNR.

5.2. Implementation Results

Table 4 gives the performance results of both architectures with different FFT sizes on Spartan-3 FPGAs (90 nm). The R2^{2}SDF achieved a smaller area and better throughput per area than the R4SDC. Due to the pipeline design, the maximum clock frequency did not change drastically with FFT size for either design. As expected the throughput per area decreases for larger FFT sizes, which require more stages and area.

Table 3. SQNR with different FFT sizes.

	FFT Size	Input Data Width	Twiddle Factor Width	Stage Num	SQNR (dB)
R4SDC	16	16	16	2	78.53
	64	16	16	3	71.24
	256	16	16	4	66.52
	1024	16	16	5	59.39
R2^{2}SDF	16	16	16	2	78.84
	64	16	16	3	71.23
	256	16	16	4	65.34
	1024	16	16	5	59.95

Comparisons with prior art are shown in Table 5, which shows publicly available pipeline FFT implementations on FPGAs from the literature. The best performance for the R2^{2}SDF method for a 1024-point FFT was published by Sukhsawas and Benkrid in [9]. They used Handel-C as a rapid prototype language and implemented the design on Virtex-E FPGAs (180 nm). They achieved 82 MHz maximum clock frequency and 7365 slices, giving a throughput per area ratio of 0.011 Msamples/s/slice.

For comparison we also implemented our design on the Virtex-E FPGA. Our R2^{2}SDF achieved better performance of 95 MHz and a smaller area of 5008 slices, giving a superior throughput per area ratio of 0.019 Msamples/s/slice. Our R4SDC architecture was also superior to prior art, running at 94.2 MHz and using 7052 slices, a throughput per area ratio of 0.013 Msamples/s/slice.

Another point of reference is the Xilinx FFT IP core. For comparison sake, the IP core for Virtex-E is shown in the table. The Virtex-E core shows four times the latency (4096) in cycles due to its internal architecture. Its throughput per area ratio is also only 0.011 Msamples/s/slice. Note that all comparisons for throughput per area do not take into account block RAMs, though each of the designs had a similar number of required block RAMs.

6. Conclusions and Future Work

In this paper, we presented optimized implementations of R4SDC and R2^{2}SDF pipeline FFT processors on Spartan-3 and Virtex-E FPGAs. The 16-bit 1024-point FFT with the R2^{2}SDF architecture had a maximum clock frequency of 95.2 MHz and used 2802 slices on the Spartan-3. The R4SDC ran at 123.8 MHz and used 4409 slices on the Spartan-3. Different rounding schemes were analyzed and compared. SQNR

Table 4. Implementation results on Spartan-3 devices.

	Point Size	Input Data Width	Twiddle Factor width	Slices	Block RAM	Max. Speed (MHz)	Latency (cycles)	Transform Time		Throughput (MS/s)	Throughput/ Area (MS/s/slice)
								Cycles	Time (us)		
R4SDC	16	16	16	468	2	108.20	21	16	0.15	108.20	0.231
	64	16	16	952	2	107.23	73	64	0.60	107.23	0.113
	256	16	16	1990	3	111.98	269	256	2.76	111.98	0.056
	1024	16	16	4409	8	123.84	1041	1024	8.27	123.84	0.028
$R2^2SDF$	16	16	16	427	2	121.24	22	16	0.13	121.24	0.284
	64	16	16	810	2	98.14	74	64	0.65	98.14	0.121
	256	16	16	1303	3	98.73	270	256	2.59	98.73	0.076
	1024	16	16	2802	8	95.25	1042	1024	10.75	95.25	0.034

Table 5. Performance comparison versus prior art on Virtex-E devices.

FFT Design	Point Size	Input Data Width	Twiddle Factor width	Slices	Block RAM	Max. Speed (MHz)	Latency (Cycle)	Transform Time		Throughput (MS/s)	Throughput/ Area (MS/s/slice)
								Cycles	Time (us)		
Amphion [9]	1024	13	13	1639	9	57	5097	4096	71.86	14.25	0.009
Xilinx [9][10]	1024	16	16	1968	24	83	4096	4096	49.35	20.75	0.011
Sundance [11]	1024	16	10	8031	20	49	1320	1320	27.00	49.00	0.006
Suksawas $R2^2SDF$ [9]	1024	16	16	7365	28	82	1099	1024	12.49	82.00	0.011
Our $R2^2SDF$	1024	16	16	5008	32	95.0	1042	1024	10.78	95.00	0.019
Our R4SDC	1024	16	16	7052	32	94.2	1041	1024	10.87	94.20	0.013

analysis showed the balanced stages rounding scheme gave high SQNR with small overhead.

The $R2^2SDF$ architecture outperformed the R4SDC architecture in terms of throughput per area, a measure of efficiency, for the 1024-point FFT. This is due to its simpler controller and compatibility with pipelining insertion. Both architectures have comparable maximum clock frequency and SQNR with the balanced stages rounding scheme.

Future work includes twiddle factor compression optimization, SQNR analysis of different data widths, and implementing DSP48 hardwired multipliers for slice optimization on advanced Xilinx devices.

7. References

[1] L.R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*, Prentice-Hall, Inc., 1975.

[2] E.H. Wold and A.M. Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementation," *IEEE Trans. Computers.*, vol. C-33, no. 5, 1984, pp. 414-426.

[3] G. Bi, and E.V. Jones, "A pipelined FFT processor for word-sequential data," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 12, 1989, pp. 1982-1985.

[4] S. He and M. Torkelson, "A new approach to pipeline FFT processor," *10th International Parallel Processing Symposium (IPPS '96)*, Apr. 1996, pp. 766-770.

[5] J.-Y. OH and M.-S. LIM, "New Radix-2 to the 4th power pipeline FFT processor," *IEICE Transactions on Electronics* 2005, pp. 1740-1746.

[6] T. Sansaloni, A. Pe´rez-Pascual, V. Torres and J. Valls, "Efficient pipeline FFT processors for WLAN MIMO-OFDM systems, " *Electronics Letters*, vol. 41, Sep. 2005, pp. 1043-1044.

[7] S. Johansson, S. He, and P. Nilsson, "Wordlength optimization of a pipelined FFT processor," *42nd Midwest Symposium on Circuits and Systems*, 1999, pp. 501-503.

[8] P. Kabal and B. Sayar, "Performance of fixed-point FFT's: rounding and scaling considerations," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '86)*, 1986, pp. 221- 224.

[9] S. Sukhsawas and K. Benkrid, "A high-level implementation of a high performance pipeline FFT on Virtex-E FPGAs," *IEEE Computer Society Annual Symposium on VLSI (ISVLSI '04)*, Feb. 2004, pp. 229-232.

[10] Xilinx, Inc., High-Performance 1024-Point Complex FFT/IFFT V2.0. San Jose, CA, July 2000. http://www. xilinx.com/ipcenter/.

[11] Sundance Multiprocessor Technology Ltd., 1024-Point Fixed Point FFT Processor, July 2008, http://www.sundance.com/web/files/productpage.asp?STRFilter=FC200.