A Comparison Between High Throughput Configurable FFT/IFFT Processors

Renan Netto, Pedro Michel and José Luís Güntzel
Embedded Computing Lab. - Department of Informatics and Statistics
Federal University of Santa Catarina
Florianópolis, Brazil
renan77@inf.ufsc.br, pedromichel@grad.ufsc.br, guntzel@inf.ufsc.br

ABSTRACT

This paper presents three configurable Fast Fourier Transform (FFT) processors based upon the well-known Single-Path Delay Feedback (SDF) architecture. The three processors may be configured to compute the FFT/IFFT of 64 to 2048-point sequences and differ by the basic processing block, which is the radix-2, radix-2^2 or radix-4 butterfly. Their architectures were described in Verilog and synthesized for a 90nm commercial standard-cells library by using Synopsys Design Compiler tool. The SQNR values were evaluated by comparing the results provided by the designed FFT/IFFT processors to that obtained from a software implementation of the FFT/IFFT algorithm. Synthesis results show that the radix-4 based processor requires 3% less area and consumes 4.7% more energy than the radix-2 based one and thus may be used when silicon area is more important than energy. On the other hand, the radix-2^2 based architecture requires 4.8% less area than the radix-2 one at a cost of 102% increase in critical delay, resulting in poor energy efficiency.

Categories and Subject Descriptors

D.3.3 [**Integrated Circuits**]: Types and Design Styles – *algorithms implemented in hardware.*

General Terms

Design.

Keywords

FFT; DIF; DIT; Radix-2; Radix-2^2; Radix-4; butterfly.

1. INTRODUCTION

Wireless communication is becoming a must in electronic products. Currently, it is found not only in personal portable devices but also in printers, keyboards, mice, among many others. Since such products have distinct operation features, new wireless communication protocols, more suited for specific applications, are being devised. The modulation/demodulation used in Wireless communication usually requires the application of the Discrete Fourier Transform (DFT) and the Inverse Discrete Fourier Transform (IDFT).

The original DFT algorithm has an $\boldsymbol{O}(n^2)$ complexity and, therefore, is not used in direct hardware realizations [1]. A lower complexity DFT form, referred to as Fast Fourier Transform (FFT) algorithm, is used instead. The FFT algorithm has come into focus through a paper by Cooley and Tukey [2] in 1965. Also, the Inverse Fast Fourier Transform (IFFT) is used to calculate the IDFT.
While aiming current electronic devices, to meet the requirements in a single solution, we present and evaluate three high throughput configurable FFT/IFFT processors to calculate 64/128/256/512/1024/2048-point FFT/IFFT based on the Single-Path Delay Feedback (SDF) architecture [3].

The proposed configurable FFT/IFFT processors were modeled in Verilog HDL and synthesized to a commercial 90nm standard-cells library using Synopsys Design Compiler tool [4]. The processors were validated through the simulation of testbenches in ModelSim environment. The SQNR in FFT/IFFT calculation was also evaluated.

The remaining of this paper is organized as follows. Section 2 presents the chosen decomposition of the FFT/IFFT algorithm. Section 3 highlights the main characteristics of the basic Single-Path Delay Feedback (SDF) architecture, the radix and configurability changes. Section 4 presents the synthesis results and establishes comparisons between each processor. Finally, section 5 draws some conclusions and comments on future works.

2. BASIC FFT/IFFT ALGORITHM

The Discrete Fourier Transform (DFT) for an N-point sequence $x(n)$ is given by (1), where $X(k)$ and $x(n)$ are complex numbers, n is the time index and k is the frequency index [1]. By using this equation the DFT computation requires N^2 complex multiplications and $N(N-1)$ complex additions, leading to a complexity of $\boldsymbol{O}(N^2)$. In [2] Cooley and Tukey proposed (actually they "rediscovered") an algorithm to compute the DFT with complexity $\boldsymbol{O}(N.\log_2 N)$. Such algorithm, known as Fast Fourier Transform (FFT), is widely used in hardware accelerators.

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{\frac{-j2\pi kn}{N}} \qquad (1)$$

Direct hardware implementations of the FFT algorithm either use the Decimation in Time (DIT) decomposition or the Decimation in Frequency (DIF) decomposition [1], the latter used in this work. In the DIF approach an N-point DFT is decomposed into two $N/2$-point DFTs, one for the even-indexed frequency outputs and another for the odd-indexed frequency outputs. The inputs to the even-indexed outputs $N/2$-point DFT are sums between first half inputs and second half inputs. The inputs to the odd-indexed outputs $N/2$-point DFT are differences between first half inputs and second half inputs, multiplied by constants that are referred to as "twiddle factors".

Each $N/2$-point DFT may be further decomposed into two $N/4$-point DFTs. Such decomposition may be applied recursively until reaching 2-point DFTs, which can be assumed as basic computation elements. Figure 1 shows in detail the signal flow chart (SFG) for an 8-point DIF FFT.

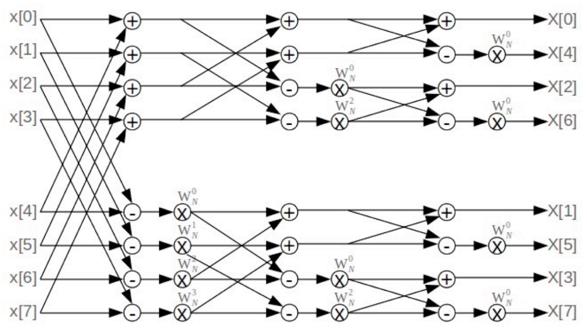

Figure 1. Signal flow graph (SFG) for 8-point DIF FFT.

A 2-point FFT processing element, normally referred to as "butterfly", is composed by one complex multiplication, one complex addition and one complex subtraction. Therefore, an *N*-point FFT has $\log_2 N$ stages where each stage has $N/2$ complex multiplications, $N/2$ complex additions and $N/2$ complex subtractions, resulting in $\boldsymbol{O}(N.\log_2 N)$ complexity, which is significantly lower than that of a direct implementation of (1)

The Inverse Discrete Fourier Transform (IDFT) is given by (2). As it can be seen, the same algorithm used to calculate the DFT can be used to calculate the IDFT, with minor changes (the sign of the imaginary part of the coefficients in the multiplication and a division by *N* of the final results).

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\cdot e^{\frac{j2\pi kn}{N}} \qquad (2)$$

3. R2SDF-BASED FFT/IFFT CONFIGURABLE PROCESSOR

We have designed three dedicated processors that can be configured to compute the FFT/IFFT for sequences with 64, 128, 256, 512, 1024 or 2048 points. The processors' architectures are based upon the well-known Single-Path Delay Feedback (SDF) architecture [3], which was carefully modified to allow the desired configurability. Also, three versions of the same architecture were designed: the first one using Radix-2 butterflies on the whole processor, the second one using Radix-4 butterflies on the last stages of the processor and the third one using Radix-2^2 butterflies on the last stages of the computation. Hereafter, these three processors will be called, respectively, CR2SDF, CR4SDF and CR2^2SDF.

The original R2SDF architecture datapath is built up from the basic stage, shown in Figure 2, which is composed by one radix-2 butterfly (either DIF or DIT), a ROM memory (to hold the stage twiddle factors), two multiplexers and a delay element ("D"). Both ROM memory and delay element were implemented using registers. The delay element is organized as a FIFO (first-in first-out) structure, being able to store $N/2^i$ inputs, where i=1 in the first stage and $i=\log_2 N$ in the last stage (*N* is the total number of points in a sequence i.e., the inputs applied to the first stage). At the beginning, the first $N/2^i$ inputs to the stage are fed into the FIFO, one after another. As a second step, each of the second $N/2^i$ inputs is directly applied to the butterfly along with each of the inputs previously stored in the FIFO. Then, the resulting lower butterfly output is fed back to the FIFO while the resulting upper butterfly output is fed to stage i+1. Considering the described behavior, one can notice that the R2SDF architecture is used only during 50% of the time to operate on a new pair of inputs, thus staying idle during other 50%

Figure 3 shows the simplified datapath block diagram of the designed CR2SDF FFT/IFFT processor. It is composed by 11 stages, being the first 10 stages similar to the one shown in Figure 2. The last stage (i=11) was simplified, since it needs only one twiddle factor. Multiplexers were added to some stages to provide the desired configurability according to the desired number of points: 64, 128,256, 512, 1024 or 2048. To allow the configurability between FFT and IFFT the complex multiplier inside each butterfly was appropriately modified. As long as all of the possible numbers of points are powers of 2, the division by *N* in the IFFT can easily be done by shifting the result to the right. Besides the datapath of Figure 3, the CR2SDF processor has a control block that generates the addresses used to access the twiddle factors in the corresponding ROMs. In order to implement the control block we have chosen the DIF decomposition, since it allows for a less complex circuitry.

As the lowest number of points that will be calculated is 64, only the last stages of the CR2SDF processor can be rearranged using higher radix butterflies.

The CR4SDF processor datapath is built in a similar way than the CR2SDF one, with a few differences on its basic stage. The basic stage of the C4RSDF processor is shown in Figure 4 and is composed by one radix-4 butterfly, three ROM memories (to hold the stage twiddle factors), six multiplexers, with two of them being a three-input multiplexer and three delay elements. Both ROM memory and delay elements were implemented the same way than the CR2SDF processor. Each delay element is able to store $N/4^i$ inputs, where i is the stage. At the beginning, the first $N/4^i$ inputs to the stage are fed into the first FIFO, one after another. The second and third steps are similar, except that the next $N/4^i$ inputs are fed to the second and third FIFO, respectively. The last $N/4^i$ inputs are directly applied to the butterfly along with each of the inputs, previously stored in the FIFOs. Then, the resulting three lower butterfly outputs are fed back to the FIFOs while the resulting upper butterfly output is fed to stage i+1. Unlike in the CR2SDF processor, in the CR4SDF processor the butterfly is used only 25% of the time to operate on a new group of inputs, therefore staying idle during the other 75%.

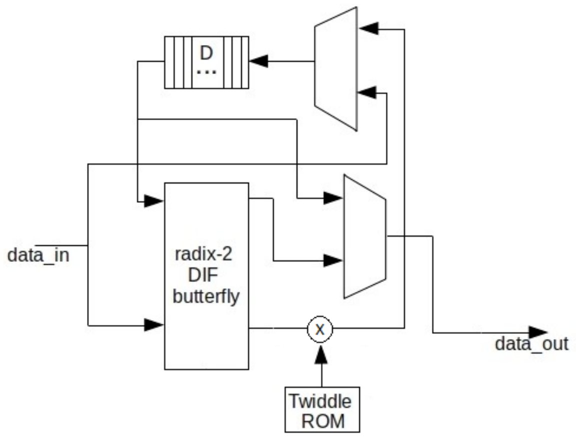

Figure 2. The i-th stage in the R2SDF architecture datapath.

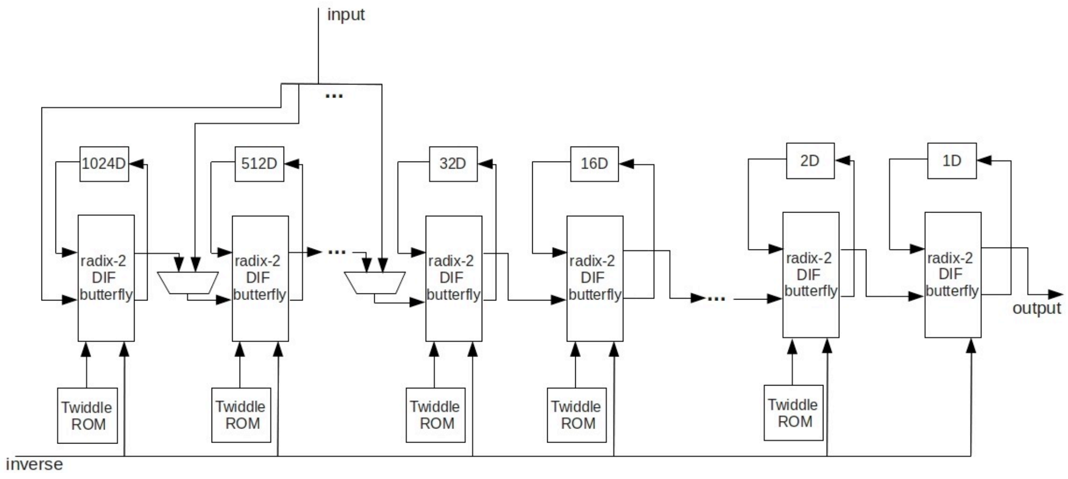

Figure 3. Datapath block diagram for the CR2SDF processor.

The CR2²SDF processor datapath is also built up in a similar way than the CR2SDF, with the basic stage shown of Figure 5. Its basic stage is composed by two CR2SDF basic stages interconnected. However, there is no multiplier between them. This occur because the multiplication between these two basic stages can be done only switching the real and imaginary parts of the input and inverting the operations of the second CR2SDF stage. The others characteristics of the CR2²SDF are the same than the CR2SDF one, except that the twiddle factors in each stage are different and, unlike the CR2SDF processor, the multiplier cannot be removed from the critical path. Once the behavior of the CR2²SDF processor is similar to that of the CR2SDF processor, it can be inferred that the butterfly of the CR2²SDF processor stays idle during the same amount of time than the CR2SDF's butterfly.

It is important to notice that, in all three processors, the butterfly upper output of a given stage i is directly fed to the butterfly input in stage i+1. This way, there is a long combinational path that begins at the lower input of the first stage butterfly, traverses all stages and ends at the upper output of the last stage butterfly. However, at every clock edge, a new point is applied to the first stage input, while the contents of the first stage FIFO is shifted. This assures high throughput because the data is forced to move forwards along this combinational path at every clock edge, making the datapath to behave as a "wave pipeline".

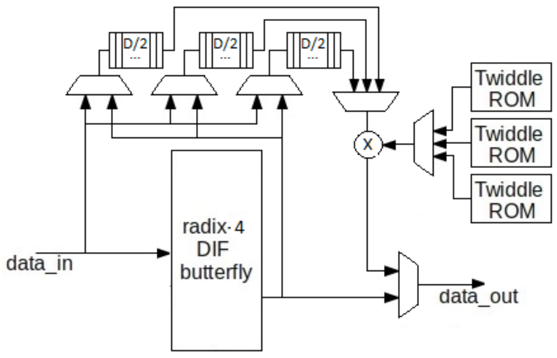

Figure 4. Radix-4 Butterfly.

4. SYNTHESIS RESULTS

The three processors were modeled in Verilog HDL and synthesized to a commercial 90nm standard-cells library using Synopsys Design Compiler tool [4]. The synthesis results, summarized in Table 1, can be used to compare CR2^2SDF and CR4SDF to CR2SDF.

This table shows that the CR2²SDF processor requires 4.8% less area than the CR2SDF processor and dissipates 3% less power. On the other hand, its critical delay is 2.02 times greater than that of the CR2SDF processor, resulting in severe performance degradation since it has the same throughput (in cycles) than the CR2SDF.

Synthesis results also show that the CR4SDF processor requires 3.5% less area than the CR2SDF processor and dissipates 1% less power at a cost of 4% increase in critical delay.

Table 1. Synthesis results of the three processors

	CR2SDF	CR2²SDF	CR4SDF
Total Area ($\mu m^{2)}$	2,614,281	2,488,299	2,522,842
Total Power (mW)	69.32	67.05	68.96
Worst Timing Path (ns)	7.08	14.28	7.41

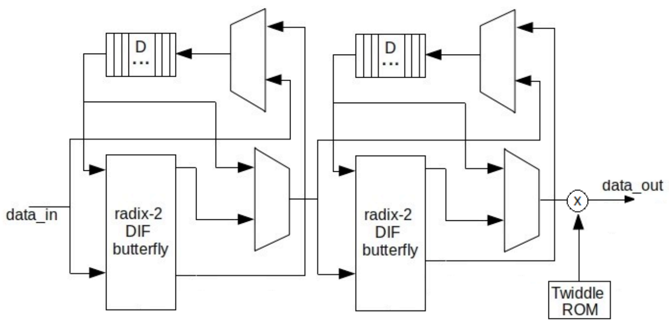

Figure 5. Radix-22 Butterfly.

Based on the different worst timing paths of the processors, their latencies and execution times were calculated for each possible number of points. These results are shown on Table 2 and Table 3. From table 2 it becomes evident the penalty in both execution time and latency resulted from the longer critical delay of the CR2^2SDF, which is not compensated by the power reduction nor by the area savings. In the case of the CR4SDF processor, the 4% increase in critical delay (and the resulting performance degradation) is compensated by the small area savings.

Table 2. Latency (ns)

# of points	CR2SDF	CR2^2SDF	CR4SDF
64	1798.32	3627.12	1882.14
128	3610.80	7282.80	3779.10
256	7235.76	14594.16	7573.02
512	14485.68	29216.88	15160.86
1024	28985.52	58462.32	30336.54
2048	57985.20	116953.20	60687.90

Table 3. Execution Time (ns/sequence)

# of points	CR2SDF	CR2^2SDF	CR4SDF
64	906.24	1827.84	948.48
128	1812.48	3655.68	1896.96
256	3624.96	7311.36	3793.92
512	7249.92	14622.72	7587.84
1024	14499.84	29245.44	15175.68
2048	28999.68	58490.88	30351.36

The energy consumption of the processors was also calculated for each possible number of points and considering power and critical delay estimated in synthesis. Table 4 shows these results. As it can be seen, when processing the same number of points, CR2^2SDF and CR4SDF consume more energy than CR2SDF.

Table 4. Energy (mJ)

# of points	CR2SDF	CR2^2SDF	CR4SDF
64	62.83	122.55	65.40
128	125.66	245.10	130.81
256	251.31	490.20	261.61
512	502.62	980.40	523.23
1024	1005.24	1960.80	1046.46
2048	2010.49	3921.59	2092.91

To verify the designed processor, a script in Python language was written to generate appropriate random inputs, which were used only to verify the functionality by calculating an FFT/IFFT using the three processors and a software solution based on the FFTW library [5], written in C language. After that, the signal-to-quantization-noise ratio (SQNR) values of the designed processor were calculated. The results showed that the processors achieved a good precision, the lowest SQNR value was 62.4204 dB.

5. CONCLUSIONS

This paper presented three FFT/IFFT processors that can be configured to process 64/128/256/512/1024/2048-point sequences. These FFT processors were modeled with Verilog HDL and synthesized into a 90nm commercial standard-cells library. Synthesis results were compared to CR2SDF.

All three FFT processors presented a good precision compared to an FFT software implementation: the lowers SQNR value was 62.4204dB.

Synthesis results show that, for the considered SDF architecture, using registers to implement the delay elements and twiddle ROMs, the CR4SDF processor requires 3.5% less area and consumes 4.7% more energy than the CR2SDF for the same number of points. Although there is no significant difference between CR4SDF and CR2SDF architectures, designers can tradeoff between them considering the application requirements.

Although the area reduction (4.8%) in relation to CR2SDF architecture, the longer critical delay of the CR2^2SDF makes it the worst choice among the three considered FFT/IFFT architectures.

As future works we intend to devise several architectural modifications in the CR2SDF datapath to reduce hardware cost and to increase energy efficiency as well. Also, we intend to synthesize two new versions, one using a 64-points basic stage and other fully combinational.

6. ACKNOWLEDGMENTS

This work was partially supported by the Brazilian Council for the Scientific and Technological Development (CNPq), through the INCT-Namitec Project (under contract 573738/2008-4) and PIBIC Program.

7. REFERENCES

[1] Oppenheim, A. V., and Schafe, R. W. 1989. Discrete-time signal processing. Prentice Hall, Englewood Cliffs.

[2] Cooley, J. W., and Tukey, J. W. 1965. An algorithm for the machine calculation of complex Fourier series. *Math. Comput*, 19 (Apr. 1965), 297-301.

[3] Groginsky, H. L., and Works, G.A. 1970. A Pipeline Fast Fourier Transform. *IEEE Transactions on Computers*, C-19 (Nov. 1970), 1015-1019.

[4] Synopsys, Inc, <http://www.synopsys.com/home.aspx> Accessed in February 2012.

[5] FFTW, <http://www.fftw.org/>. Accessed in January 2012.