Defence Science Journal, Vol. 63, No. 2, March 2013, pp. 210-213, DOI: 10.14429/dsj.63.4266

RESEARCH PAPER

Field-Programmable Gated Array Implementation of Split-Radix Fast Fourier Transform for High Throughput

P.S. Sai Pavan, B. Renuka*, and B.Vinatha

Bharat Electronics Limited, Hyderabad–500 076, India
**E-mail: renukab@bel.co.in*

ABSTRACT

As the signal processing required in electronic warfare (EW) domain is complex and the sample rates to be handled are very high, IP cores which are freely available are not of much use. A study of various fast fourier transform (FFT) algorithms has been carried out and spit-radix FFT has been chosen to be implemented due to fewer multiplications[3]. This algorithm is attractive to be implemented using field-programmable gated array (FPGA). This paper presents split-radix FFT algorithm for implementation of 512-pt FFT on FPGA platform for EW applications. The algorithm is such designed that it can achieve a throughput of up to 1500 MSPS. 512-pt SRFFT is implemented using parallel pipelined architecture in order to maximize processing speed and thus achieve a throughput of 1500 MSPS with area optimization. The pipeline structure is partitioned to balance the input throughput and to optimize the available FPGA resources. The standard Cooley-Tukey radix-2 FFT algorithm requires $N/2 \log_2 N$ (for N=512, 2304 multiplications) multiplications and $N \log_2 N$ additions where as radix-4 FFT requires $N/2 \log_4 N$ multiplications and $N \log_2 N$ additions. The SRFFT presented in this paper has a multiplicative complexity of only about two-thirds that of the radix-2 FFT, and is better than the radix-4 FFT or any higher power-of-two radix as well. The initial latency is less than N clock cycles.

Keywords: Fast fourier transform, split-radix FFT, field-programmable gated array, commutator

1. INTRODUCTION

To perform frequency analysis on a discrete-time signal x[n], it is necessary to convert the time-domain sequence to an equivalent frequency-domain representation[1]. Fourier transform $X(\omega)$ gives the spectrum of the sequence $x[n]$. However, $X(\omega)$ is a continuous function of frequency and therefore is not a computationally convenient representation of the sequence $x[n]$. For computational purposes, we consider the representation of a sequence $x[n]$ by samples of its spectrum $X(\omega)$ Such a frequency domain representation leads to the discrete fourier transform (DFT), which is a powerful computational tool for performing frequency analysis of discrete-time signals[4].

The DFT plays an important role in many applications of digital signal processing including linear filtering, correlation analysis, and spectrum analysis. The number of complex multiplication and addition operations required by simple forms of both the discrete fourier transform (DFT) and inverse discrete fourier transform (IDFT) is of the order of N^2 where N is the number of data points to calculate, each of which requires N complex arithmetic operations. The fast fourier transform (FFT) is another method for calculating the DFT. The FFT decomposes the set of data to be transformed into a series of smaller data sets and decomposes those smaller sets into even smaller sets. There are two different approaches to find DFT of a sequence:

(1) Divide and conquer approach.
(2) Linear filtering approach.

In the former approach, a DFT of size N, where N is a composite number, is reduced to computation of smaller DFTs from which the larger DFT is computed. FFT algorithms (Radix-2, Radix-4 and Split Radix) fall into this category. The latter approach is based on linear filtering operation on the data. Two algorithms, the Goertzel algorithm and the chirp-transform algorithm compute the DFT via linear filtering of the data sequence[2].

In this paper, 512-pt Split Radix FFT (SRFFT) implementation is presented. SRFFT algorithms exploit both radix-2 and radix-4 decomposition in the same FFT algorithm to reduce the number of multiplications. Even numbered samples are implemented using Radix-2, where as odd numbered samples are implemented with Radix-4 FFT algorithms[4].

2. FFT ALGORITHM

SRFFT algorithm for the fast computation of the DFT is developed by Duhamel and Hollmann for data sequences having a length which is integer power of 2 ($N=2^m$). Radix-2 decimation-in-frequency indicates that the even-numbered points of the DFT can be computed independent of the odd-numbered points. So, there is a possibility of using different computational methods for independent parts of the algorithm to reduce the number of computations. Given a sequence x_n of length N (integer power of two), the computation of the coefficients X_k using the SRFFT algorithm with decimation-in-frequency is done by observing that the even coefficients can

Received 07 September 2012, revised 08 February 2013, online published 23 March 2013

be expressed by the following equation[3]:

$$x_{2k} = \sum_{n=0}^{N/2-1} (x_n + x_{n+N/2}) W_N^{2nk}, \quad k = 0, 1, 2.....N/2-1 \quad (1)$$

$$x_k = \sum_{n=0}^{N-1} x_n W N^{nk}, \quad k = 0, 1, 2.....N-1 \quad (2)$$

and odd coefficients are expressed by the following equation:

$$x_{4k+1} = \sum_{n=0}^{N/4-1} \left[\left(x_n - x_{n+\frac{N}{2}} \right) - j \left(x_{n+\frac{N}{4}} - x_{n+\frac{3N}{4}} \right) \right] W_N^{n} W_N^{4nk}, \quad k = 0, 1, ...N/4-1 \quad (3)$$

$$x_{4k+3} = \sum_{n=0}^{N/4-1} \left[\left(x_n - x_{n+\frac{N}{2}} \right) - j \left(x_{n+\frac{N}{4}} - x_{n+\frac{3N}{4}} \right) \right] W_N^{3n} W_N^{4nk}, \quad k = 0, 1, ...N/4-1 \quad (4)$$

Figure 1 explains the operations required to implement Eqns. (1) to (4). $X(2k)$ and $X(2K+2)$ are calculated by Radix-2 Eqn. (2), which require only complex additions. $X(4k+1)$ and X(4k+3) are calculated using Radix-4 Eqns. (3) and (4). {x(n), x(n+N/2)} and {x(n+N/4), x(n+3N/4)} are subtracted and then multiplied with -j & j respectively. Addition is applied and the resulting data is multiplied with the twiddle factors, which are taken from ROM.

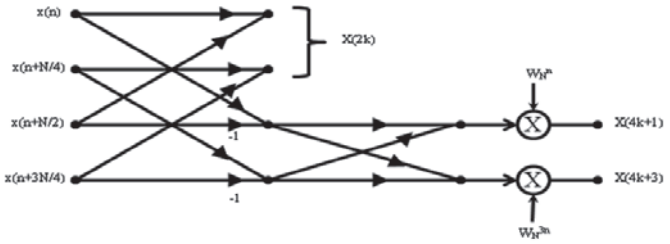

Figure 1. L – Shaped dragonfly.

Each L-shaped dragonfly contains one Radix-2, and one Radix-4 butterflies. In Fig. 2, each column represents one stage. The algorithm has a recursive nature. This means that the algorithm is first performed as an N-point L-shaped dragonfly, and then the results are fed to one N/2-point and two N/4point stages, that themselves can be performed as an L-shaped dragonfly. This continues until all the points have been fully transformed[3,4]. Near the end of the process, 8-point

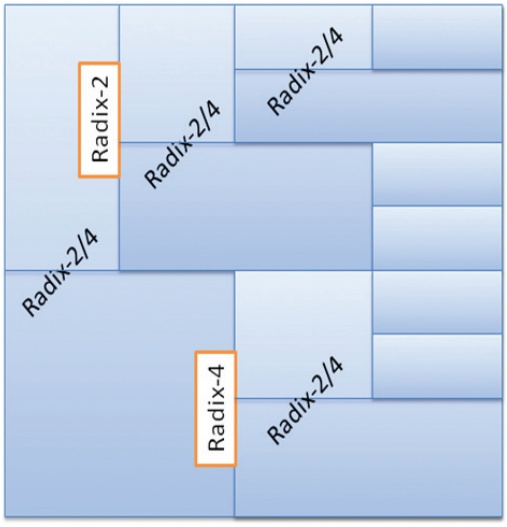

Figure 2. L-shaped dragonfly structure of SRFFT.

and 4-point transforms using the L-shaped dragonfly yield 2-point outputs. For 2-pt transforms, simple Radix-2 FFT butterflies are performed.

3. FPGA IMPLEMENTATION OF SRFFT

Figure 3 shows the block diagram of the 512-pt SRFFT implemented in FPGA. The classic SRFFT algorithm proposed by Duhamel and Hollmann for 512-pt requires 3076 real multiplications and 12292 real add-sub operations[1], performed in 9 stages. This needs 9222 DSP48Es for implementation in FPGA.

Contrasted with the above, our algorithm for 512-pt SRFFT is implemented using only three building blocks-dragonfly computing block, commutator block and ROM, with their connections between adjacent blocks. The implementation is achieved using seven stages i.e. seven dragonfly computing blocks, seven commutator blocks and six ROMs. Only 336 DSP48Es have been used to implement the building blocks and achieve a 1500 MSPS throughput.

The incoming data is multiplexed on to four parallel buses by retiming with a suitable lower clock and stored in four FIFOs for processing. To handle the throughput of 1500MSPS, stage-1 needs to be completed within 64 clock cycles, which is accomplished by reading the samples simultaneously from all the four FIFOs and sending to dragonfly stage – 1. 512 data points are covered within 32 clock cycles using pipelined architecture. The twiddle factors, which are stored in ROM[2]

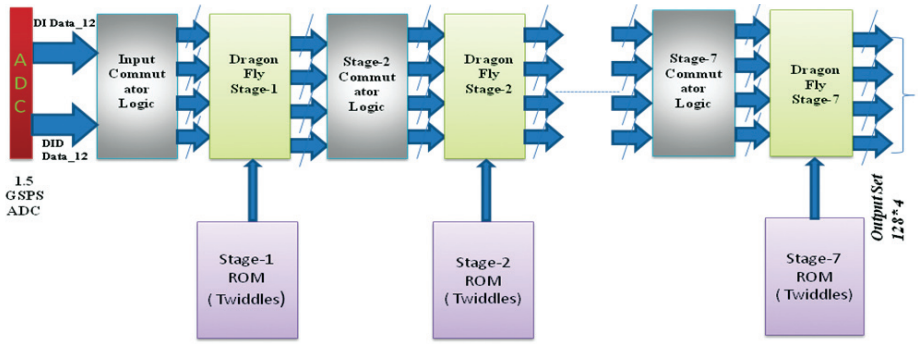

Figure 3. FPGA implementation of SRFFT – block diagram.

are read in pipeline and sent accordingly to dragonfly stage. Input data is read from four FIFOs in pipeline and provided to the dragonfly. Each dragonfly contains 4 L-shaped dragons to process 16 data points at a time. Original SRFFT algorithm is modified into 7 stages by executing the 4-pt dragonflies and 2-pt butterflies in earlier stages to maximise the resource reusability. At every stage, decomposition of the half- and quarter-length DFTs leads to full split-radix structure. As 512-pt SRFFT output is in bit reverse order, the output is stored in output RAM by giving bit reverse ordered addresses[2].

4. COMMUTATOR BLOCK

Dragonfly needs data in the format of {x(n), x(n+N/2) and x(n+N/4), x(n+3N/4)}. To take dragonfly input in required format, data is stored in 4 FIFOs in the same order and data is provided simultaneously from FIFOs and given in pipeline to the next stages[2]. To ease this data formatting, commutator is used. Figure 4 shows the block diagram of commutator logic.

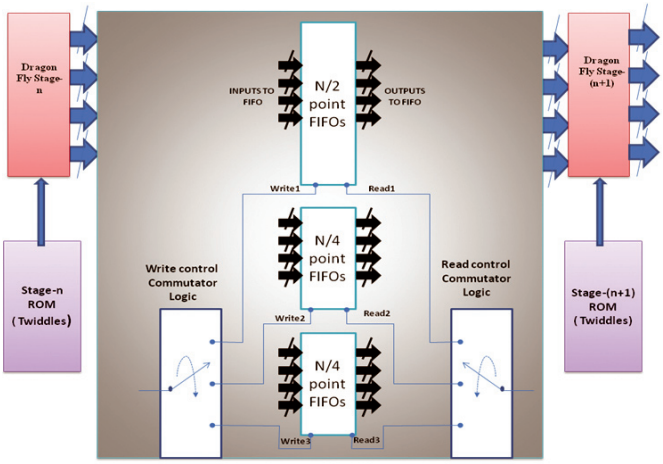

Figure 4. Commutator.

Commutator has fifos to store data from and to the dragonflies[1]. Data is stored in commutators one after the other and read at the same time for parallel operation. Commutator logic comprises of time division multiplexing (TDM) of available data at individual stages. TDM employs writing of different stage outputs from previous stage and reading of the inputs to next stage. Commutator logic also includes reading of twiddles from a particular ROM.

5. DRAGONFLY COMPUTING BLOCK

Because of the limitation on the number of DSP blocks available in FPGA, all the dragonflies cannot be implemented at a time[1]. To overcome this limitation, four dragonflies are used for computation of a stage and the same dragonflies are reused by employing pipelined architecture. Input is provided in sixteen parallel data buses, and output is taken in sixteen parallel data buses. In every stage, the first output is available after 12 clock cycles, and the final output is available at the end of 44thclock cycle. Similarly, subsequent stages get completed within 44 clock cycles each, taking the total latency to 300 clock cycles.

L-shaped dragonfly requires 12 DSP blocks to compute

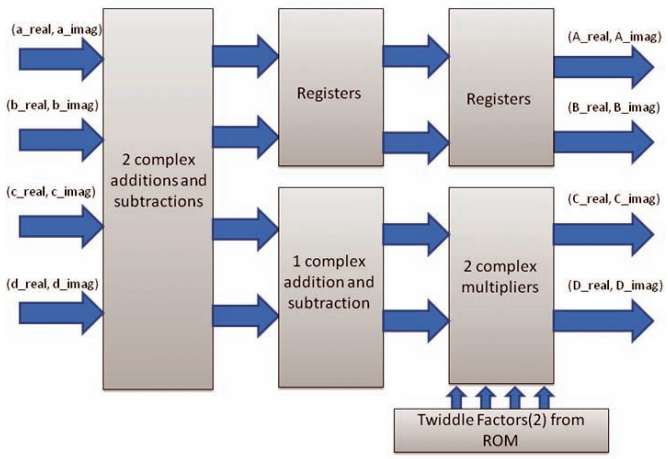

Figure 5. L-shaped dragonfly.

three complex add-sub operations and two complex multiplications as shown in the Fig. 5. Since each stage contains four L-shaped dragonflies, 48 DSP blocks are utilized. Thus, a total of 336 DSP blocks are used to complete 7 stages of 512-pt SRFFT.

6. SIMULATION RESULTS

The SRFFT algorithm is developed in MATLAB with fixed point tool box for 12-bit input and the results are verified with built-in MATLAB FFT function. VHDL simulation has been carried out using Modelsim SE. Xilinx ChipScope is used for verification in the hardware.

For N-pt FFT, the output data width is [input data width +log2(N) + 1]. For an input data width of 12, FFT output data width will be 22-bits. Growth of fractional bits created from the multiplication is truncated after the every multiplication to 22-bits. For an input sinusoid of frequency 500 MHz, the VHDL outputs shown in Fig. 7 are compared with the results of MATLAB in-built FFT function outputs shown in Fig. 6.

ChipScope capture of the FFT output shows maximum peak at xk index 323, which is same as the MATLAB output's peak index. Error in magnitude between MATLAB's built-in FFT and the FPGA implemented 512-pt SRFFT is shown in Fig. 8. Maximum error in magnitude is ±23, which is due to the fact that MATLAB functions in 64-bit operating environment

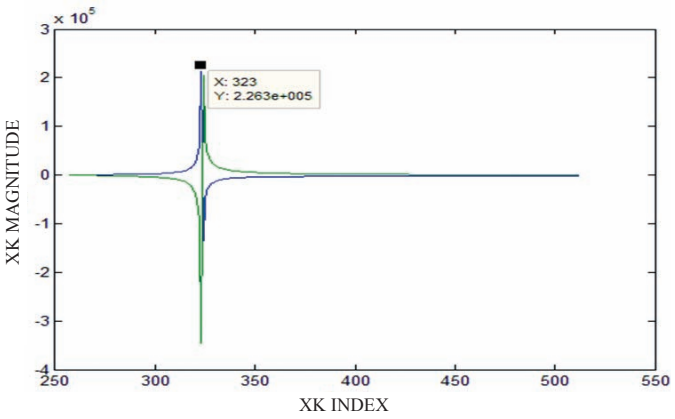

Figure 6. MATLAB inbuilt FFT output.

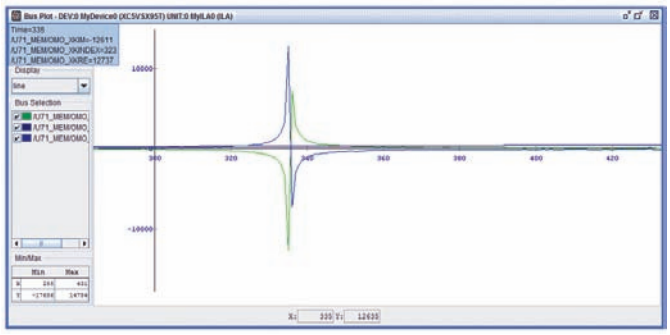

Figure 7. SRFFT output in XC5VSX95T chipscope.

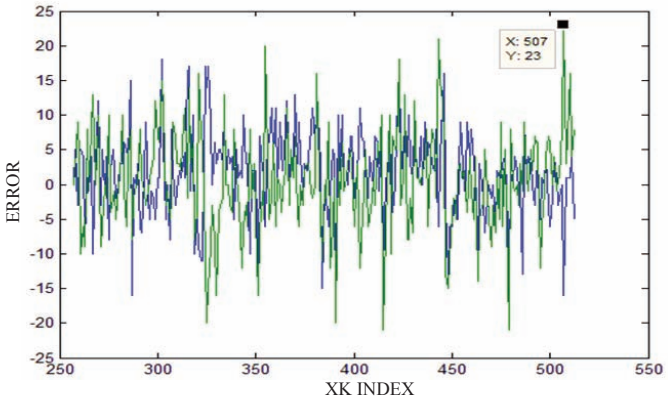

Figure 8. SRFFT error plot.

whereas in VHDL, only 22-bit output width is used. Fixed point results from MATLAB shown in Fig. 9 are matching with VHDL results shown in Fig. 7 for 12-bit input.

7. FPGA RESOURCE USAGE

Implementation of normal 512-pt SRFFT algorithms in XC5VSX95T FPGA is not feasible due to the limitations on the resources available. However, the modifications to the algorithm proposed by us have made the implementation feasible. Table 1 gives the actual FPGA utilisation for a 512-pt SRFFT for a clock rate of 187.5 MHz.

Table 1. Performance of 512-pt SRFFT

No.	Parameter	Value
1	Number of points	512
2	Output data width	22-bit
3	Number of slice registers	51,192
4	Number of slice LUTs	27,547
5	Number used as memory	7,000
6	Number of DSP48Es	336
7	Initial latency	1.65 µs
8	Frequency	187.5 MHz
9	FPGA family device	XC5VSX95T

8. CONCLUSIONS

This paper describes the design and FPGA implementation of a 512-pt pipeline SRFFT. The inputs are four parallel data buses of 12-bit width. The algorithm outputs the transform coefficients over 16 parallel data channels with a latency of 300 clock cycles. The computing elements have three complex add-subtract blocks operating in parallel with two complex multipliers, which have been implemented with the DSP48Es available within the FPGA in order to attain the highest possible operational speed. For a clock rate of 187.5 MHz, a latency of 1.65 µs is achieved in implementing the 512-pt SRFFT. split radix-based FFT algorithm for 512-pt FFT has been implemented by using 7 stages instead of 9 stages in FPGA for achieving a throughput of 1500MSPS. The output has been compared with Matlab simulation results and validated. Verification is done on Xilinx XC5VSX95T by feeding a single tone.

REFERENCES

1. García1, Jesus; Michel, Juan A.; Ruiz, Gustavo & Boron, Angel M. FPGA realization of a Split Radix FFT processor. SPIE, **6590**, 2007, P-1 - P-11.
2. Kannan M. & Srivatsa, S.K. Low power hardware implementation of high speed FFT core. *Int. Arab J. Info. Technol.*, 2009, **6**(1), 1-7.
3. Jones, Douglas L. Split-radix FFT algorithms. http://cnx.org/content/m12031/latest/ (Accessed on 12/01/2012).
4. Yeh, Wen-Chang. High-speed and low-power split-radix FFT. IEEE Trans. Signal Processing, 2003, **51**(3), 864–74.

Contributors

Mr P.S. Sai Pavan graduated from JNTU College of Engineering, Kakinada, A.P. in the year 2010. Currently working as Deputy Engineer in Bharat Electronics Ltd, Hyderabad's D&E-Core Technologies group. His area of interest include: Digital receivers and embedded systems of EW systems.

Ms B. Renuka graduated from Osmania University, Hyderabad, A.P. in 2009. Currently working as Deputy Engineer in Bharat Electronics Ltd, Hyderabad's D&E-Core Technologies group. Her area of interest include: Digital receivers for EW systems, FPGA implementation of the DSP algorithms.

Mrs B.Vinatha did her BTech from JNTU College of Engineering, Kakinada, A.P in 2000. Currently she is working as Manager in Bharat Electronics Ltd, Hyderabad's D&E-Core Technologies Group. Her area of interest include: Digital signal processing algorithms development and implementation on FPGAs for digital receivers of EW systems.