VHDL Implementation of A Flexible and Synthesizable FFT Processor

¹ Gatla Srinivas, ²P.Masthanaiah, ³P.Veeranath, ⁴R.Durga Gopal,

^{1,2}[M.Tech], ³Associate Professor, J.B.R E.C, ⁴Associate Professor, CVR College of Engineering, Hyderabad, AP, India.

Abstract – FFT is a very expensive operation in terms of MAC units if you require FFT calculation to achieve very good spectral resolution (e.g., many points) with a high sample rate data stream the MACs requirement could not be matched by even the most powerful DSP so a possible solution is to use dedicated hardware (or FPGA) as a co-processor to perform efficient FFT computation at high data rate, while the DSP or GPP could perform the less intensive parts of the processing. A parallel and pipelined Fast Fourier Transform (FFT) Processor for use in the Direction of Arrival (DOA) estimation of a wideband waveform is presented. The selected DOA algorithm follows the Coherent signal subspace Method (CSSM). Although the parallel and pipelined architecture uses a large portion of the available FPGA resources, the architecture does not yield a high throughput. Low speed because of the serial operation of the DSP Processors. The number of input and output pins also very low so higher number of bits are not possible to implement possibly. High speed due to parallel architecture and a hardware description oriented coding. High numbers of Mac units are available for implementation in FPGA in future through which we can achieve a parallel and pipelined architecture.

I. INTRODUCTION

This paper proposes the design of 32-points FFT & 64-points FFT processing block. The work of the project is focused on the design and implementation of FFT [13] for a FPGA kit. This design computes 32-points FFT and all the numbers follow fixed point format of the type Q8.23, signed type input format is used. The direct mathematical derivation method is used for this design in this project the coding is done in VHDL [8] & the FPGA synthesis and logic simulation is done using Xilinx ISE Design suite 10.1.

The Discrete Fourier Transform (DFT) [5] plays an important role in the analyses, design and implementation of the discrete-time signal- processing

algorithms and systems it is used to convert the samples in time domain to frequency domain. The Fast Fourier Transform (FFT) is simply a fast (computationally efficient) way to calculate the Discrete Fourier Transform (DFT). The wide usage of DFT's in Digital Signal Processing applications is the motivation to implement FFT's. Almost every branch of engineering and science uses Fourier Methods. The words "frequency," "period," "phase," and "spectrum" is important parts of an engineer's vocabulary.

The Discrete Fourier transform is used to produce frequency analysis of discrete non periodic signals. The FFT is another method of achieving the same result, but with less overhead involved in the calculations. The representation of finite Fourier Transform is given in $\{X(n)\}$ of the sequence of $\{x(n)\}$ is continuous function in frequency domain Fourier Transform, it is not computationally convenient to be represented by the sequence of $\{x(n)\}$. However, the sequence of $\{x(n)\}$ can be represented by sampling the spectrum $\{X(n)\}$. This frequency domain representation leads to the discrete Fourier Transform (DFT), which is an important algorithm for performing frequency analysis of discrete-time signal.

Application using frequency analysis of discrete-time signals in digital signal processor is the most convenient method especially designed digital hardware (Praxis and Melonakos, 1996 and Schoukens et. Al, 2004). Frequency analysis is performed on a discrete-time signal $\{x(n)\}$ by converting the time-domain sequence to an equivalent frequency-domain presentation. As straightforward representation of the Fourier transform is illustrated in Figure 1.1 (Brigham, 1974). From the figure, it shows the essence of the Fourier transform of a waveform is to decompose or separate the waveform into a sum of sinusoids of

different frequency. Thus, the pictorial representation of the Fourier transforms is a diagram which displays the amplitude and frequency of each of the determined sinusoids.

Figure 1.1 (Brigham, 1974).

The Fourier Transform identifies or distinguishes the different frequency sinusoids and their respective amplitudes which combine to form an arbitrary waveform. Mathematically, this relationship is stated as (Brace well, 1978).

II. FOUNDAMENTALS

Radix-2 FFT algorithm is called the decimation-in-frequency algorithm, is obtained by using the divide-and-conquer approach. To derive the algorithm, we begin by splitting the DFT formula into two summations, one of the which involves the sum over the first $N/2$ data points and the second sum involves the last $N/2$ data points. Thus we obtain

$$X(k) = \sum_{n=0}^{(N/2)-1} x(n) W_N^{kn} + \sum_{n=0}^{(N/2)-1} x(n) W_N^{k(N/2+n)}$$

$$= \sum_{n=0}^{(N/2)-1} x(n) W_N^{kn} + W_N^{Nk/2} \sum_{n=0}^{(N/2)-1} x(n + \frac{N}{2}) W_N^{kn}$$

Since $W_N^{Nk/2} = (-1)^k$

$$X(k) = \sum_{n=0}^{(N/2)-1} \left[x(n) + (-1)^k x\left(n + \frac{N}{2}\right) \right] W_N^{kn}$$

Now, let us split (decimate) $X(k)$ into the even- and odd-numbered samples. Thus we obtain

$$X(2k) = \sum_{n=0}^{(N/2)-1} \left[x(n) + x\left(n + \frac{N}{2}\right) \right], \quad k = 0, 1, \dots, \frac{N}{2} - 1$$

$$X(2k+1) = \sum_{n=0}^{(N/2)-1} \left[x(n) - x\left(n + \frac{N}{2}\right) \right], \quad k = 0, 1, \dots, \frac{N}{2} - 1$$

Where we have used the fact that $W_N^2 = W_{N/2}$

The computational procedure above can be repeated through decimation of the $N/2$ -point DFTs $X(2k)$ and $X(2k+1)$. The entire process involves $v = \log_2 N$ stages of decimation, where each stage involves $N/2$ butterflies of the type. Consequently, the computation of the N -point DFT via the decimation-in-frequency FFT requires $(N/2)\log_2 N$ complex multiplications and $N\log_2 N$ complex additions, just as in the decimation-in-time algorithm.

Figure 2.5: Basic Butterfly computation in the decimation-in-frequency.

III. RADIX-4 DIF FFT ALGORITHM

In this paper, two improved radix-4 FFT algorithms have been proposed. It has been shown that by re-indexing a subset of the output samples resulting from the conventional decompositions in the radix-4 FFT algorithms, a substantial reduction in the number

of twiddle factor evaluations or accesses to the lookup table can easily be obtained. These savings have been achieved without imposing any additional computational or structural complexity in the algorithms. The basic idea for improved FFT algorithms introduced in this paper can also be applied to other higher radices DIT and DIF FFT algorithms as well as to multidimensional FFT algorithms.

When the number of data points N in the DFT is a power of 4 (i.e., $N = 4^v$), we can, of course, always use a radix-2 algorithm for the computation. However, for this case, it is more efficient computationally to employ a radix-4 FFT algorithm. Let us begin by describing a radix-4 decimation-in-time FFT algorithm briefly. We split or decimate the N -point input sequence into four subsequence's,

$$x(4n), x(4n+1), x(4n+2), x(4n+3), n = 0, 1, \dots, N/4 - 1$$

$$X(p, q) = \sum_{l=0}^3 [W_N^{lq} F(l, q)] W_N^{lp}$$

$$F(l, q) = \sum_{m=0}^{N/4-1} x(l, m) W_{N/4}^{mq}$$

$$p=0, 1, 2, 3; l=0, 1, 2, 3; q=0, 1, 2, \dots, \frac{N}{4}-1$$

and

$$x(l, m) = x(4m + l)$$

$$X(p, q) = X\left(\frac{N}{4}p + q\right)$$

Thus the four $N/4$ -point DFTs $F(l, q)$ obtained from the above equation are combined to yield the N -point DFT. The expression for combining the $N/4$ -point DFTs defines a radix-4 decimation-in-time butterfly, which can be expressed in matrix form as

$$\begin{bmatrix} X(0, q) \\ X(1, q) \\ X(2, q) \\ X(3, q) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} W_N^0 F(0, q) \\ W_N^{N/4} F(1, q) \\ W_N^{N/2} F(2, q) \\ W_N^{3N/4} F(3, q) \end{bmatrix}$$

The radix-4 butterfly is depicted in figure TC.3.9a and in a more compact form in figure TC.3.9b. Note that each butterfly involves three complex multiplications, since $W_N^0 = 1$, and 12 complex additions.

Figure: Basic Butterfly computation in a Radix-4 FFT Algorithm

By performing the additions in two steps, it is possible to reduce the number of additions per butterfly from 12 to 8. This can be accomplished by expressing the matrix of the linear transformation mentioned previously as a product of two matrices as follows:

$$\begin{bmatrix} X(0, q) \\ X(1, q) \\ X(2, q) \\ X(3, q) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -j \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & j \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} W_N^0 F(0, q) \\ W_N^{N/4} F(1, q) \\ W_N^{N/2} F(2, q) \\ W_N^{3N/4} F(3, q) \end{bmatrix}$$

Fig. 3: Sixteen-point, radix-4 decimation-in-frequency algorithm with input in normal order and output in digit-reversed order.

For illustrative purposes, let us re-derive the radix-4 decimation-in-frequency algorithm by breaking the N -point DFT formula into four smaller DFTs. We have

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{kn} \\ &= \sum_{n=0}^{N/4-1} x(n) W_N^{kn} + \sum_{n=N/4}^{N/2-1} x(n) W_N^{kn} + \\ &\quad \sum_{n=N/2}^{3N/4-1} x(n) W_N^{kn} + \sum_{n=3N/4}^{N-1} x(n) W_N^{kn} \\ &= \sum_{n=0}^{N/4-1} x(n) W_N^{kn} + \\ &\quad W_N^{kN/4} \sum_{n=0}^{N/4-1} x\left(n + \frac{N}{4}\right) W_N^{kn} + W_N^{kN/2} \sum_{n=0}^{N/4-1} x\left(n + \frac{N}{2}\right) W_N^{kn} \\ &\quad + W_N^{3N/4} \sum_{n=0}^{N/4-1} x\left(n + \frac{3N}{4}\right) W_N^{kn} \end{aligned}$$

From the definition of the twiddle factors, we have

$$W_N^{kN/4} = (-j)^k, W_N^{kN/2} = (-1)^k, W_N^{3N/4} = j^k$$

Thus

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

The relation is not an $N/4$ -point DFT because the twiddle factor depends on N and not on $N/4$. To convert it into an $N/4$ -point DFT we subdivide the DFT sequence into four $N/4$ -point subsequences $X(4k), X(4k+1), X(4k+2), X(4k+3), k=0, 1, \dots, N/4$. thus we obtain the radix-4 decimation-in frequency DFT as

$$X(4k) = \sum_{n=0}^{N/4-1} \left[x(n) + x\left(n + \frac{N}{4}\right) + x\left(n + \frac{N}{2}\right) + x\left(n + \frac{3N}{4}\right) \right] W_N^{kn} W_{N/4}^{kn}$$

$$X(4k+1) = \sum_{n=0}^{N/4-1} \left[x(n) + x\left(n + \frac{N}{4}\right) + x\left(n + \frac{N}{2}\right) + x\left(n + \frac{3N}{4}\right) \right] W_N^{kn} W_{N/4}^{kn}$$

$$X(4k+2) = \sum_{n=0}^{N/4-1} \left[x(n) + x\left(n + \frac{N}{4}\right) + x\left(n + \frac{N}{2}\right) + x\left(n + \frac{3N}{4}\right) \right] W_N^{kn} W_{N/4}^{kn}$$

$$X(4k+3) = \sum_{n=0}^{N/4-1} \left[x(n) + x\left(n + \frac{N}{4}\right) + x\left(n + \frac{N}{2}\right) + x\left(n + \frac{3N}{4}\right) \right] W_N^{kn} W_{N/4}^{kn}$$

Where we have used the property $W_N^{4kn} = W_N^{kn} N/4$. Note that the input to each $N/4$ -point DFT is a linear combination of four signal samples scaled by a twiddle factor. This procedure is repeated v times, where $v = \log_4 N$.

IV. SPLIT-RADIX FFT ALGORITHMS

An inspection of the radix-2 decimation-in-frequency flow graph shown in figure TC.3.8 indicates that the even-numbered points of the DFT can be computed independently of the odd-numbered points. This suggests the possibility of using different computational methods for independent parts of the algorithm, with the objective of the number of computations. The split-radix FFT (SRFFT) algorithms exploit this idea by using both radix-2 and radix-4 decomposition in the same FFT algorithm. First, we recall that in the radix-2 decimation-in-frequency FFT algorithm, the even-numbered samples of the N -point DFT are given as

$$X(2k) = \sum_{n=0}^{N/2-1} \left[x(n) + x\left(n + \frac{N}{2}\right) \right] W_N^{kn} \quad k = 0, 1, \dots, \frac{N}{2} - 1$$

A radix-2 suffices for this computation. The odd-numbered samples $\{X(2k+1)\}$ of the DFT require the pre-multiplication of the input sequence with the twiddle factors W_N^{kn} . For these samples radix-4 decomposition produces some computational efficiency because the four-point DFT has the largest multiplication-free butterfly. Indeed, it can be shown that using a radix greater than 4 does not result in a significant reduction in computational complexity.

If we use a radix-4 decimation-in-frequency FFT algorithm for the odd-numbered samples of the N -point DFT, we obtain the following $N/4$ -point DFTs:

$$X(4k+1) = \sum_{n=0}^{N/4-1} \left[x(n) + x\left(n + \frac{N}{4}\right) + x\left(n + \frac{N}{2}\right) + x\left(n + \frac{3N}{4}\right) \right] W_N^{kn} W_{N/4}^{kn}$$

$$X(4k+3) = \sum_{n=0}^{N/4-1} \left[x(n) + x\left(n + \frac{N}{4}\right) + x\left(n + \frac{N}{2}\right) + x\left(n + \frac{3N}{4}\right) \right] W_N^{kn} W_{N/4}^{kn}$$

Fig-4 shows the flow graph for an in-place 32-point decimation-in-frequency SFFT algorithm.

In this paper, we propose to design an 32-point FFT and find its computation time. Here our goal is to implement Radix-4 64-point FFT in hardware using hardware language (VHDL) here time constrain is measured with the help of Xilinx FPGA (Field Programmable Gate Array). First we consider single butterfly suited for FPGA implementation. Then we had seen the result of complete FFT. From above result we get the time taken by each block in hardware language. Thus, FPGA resources are effectively utilized because the designed structures fitted for FPGA structures.

This paper concentrates on the development of the fast Fourier Transform (FFT), based on Decimation-in-time (DIT) domain, Radix-2 algorithm, Radix-4 Decimation-in-Frequency algorithm this paper uses VHDL as a design entity, and their synthesis by Xilinx Synthesis Tool on Vertex kit has been done. The input of fast Fourier transform has been given by a PS2 KEYBOARD using a test bench and output has been displayed using the waveforms on the Xilinx ISE 10.1. the synthesis results show that the computation for calculating the 32-point Fast Fourier transform is efficient in terms of speed.

V. FPGA

The introduction of field programmable gate arrays (FPGAs), has made it feasible to provide hardware for application specific computation design. The changes in designs in FPGA's can be accomplished within a few hours, and thus result in significant savings in cost and design cycle. FPGAs offer speed comparable to dedicated and fixed hardware systems for parallel algorithm. The 32-point FFT proposed in this paper is been simulated and synthesized using the Xilinx ISE10.1.

Overview	
Generated on	Sat Nov 24 13:46:26 2012
EDK Version	13.1
FPGA Family	spartan3
Device	xc3s200tq144-4
# IP Instances	13
# Processors	1
# Buses	3

Device Utilization Summary:

Number of BSCANs	1 out of 1	100%
Number of BUFMUXs	2 out of 8	25%
Number of DCMs	1 out of 4	25%
Number of External IOBs	62 out of 97	63%
Number of LOCed IOBs	62 out of 62	100%
Number of MULT18X18s	3 out of 12	25%
Number of RAMB16s	4 out of 12	33%
Number of Slices	1174 out of 1920	61%
Number of SLICEMs	280 out of 960	29%

VI CONCLUSION

This paper proposes a novel radix-4 FFT processor based on FPGA for WLAN, using Verilog HDL as Hardware description language and Quartus. II as Design and Synthesis tool. To achieve high-throughput, pipelined architectures have been used in the butterfly unit and the whole system performance can be greatly improved due to adopting a novel simple address mapping scheme. For radix-2 system. This mapping scheme is better and simpler than most of others. The design is implemented on a FPGA chip. And this pipelined FFT completes a complex 64-point FFT within 2.1 μ s. The hardware testing result explains that it can meet the requirements of the WLAN.

VII REFERENCES

- [1] Alan V. Oppenheim, Ronald W. Schaffer and John R. Buck, Discrete-Time Signal Processing, Prentice Hall, second edition, 1999.
- [2] Mandeep Singh Balwinder Singh Pawan Verma, Harpreet Kaur, "VHDL implementation of FFT/IFFT blocks for OFDM", *International Conference on Advances in Recent Technologies in Communication and Computing*, 2009.
- [3] James W. Cooley and John W. Tukey, "An algorithm for the machine calculation of complex fourier series", *Math Comput.*, pp. 297 – 301, 1965.
- [4] Volnei A. Pedroni, Circuit Design with VDHL, MIT Press, ISBN 0- 262-16224-5, 2004.
- [5] T. Starr, M. Sorbara, J. M. Cioffi and P. J. Silverman, *DSL Advance*, Prentice Hall, 2003.
- [6] Rd. J.; Ordaz-Moreno A.; Vite-Frias, J. A.; Romero-Troncoso, "VHDL core for 1024-point radix-4 fft computation", *International Conference on Reconfigurable Computing and FPGAs 2005, ReConFig 2005*, pp. 4 – 24, 9 2005.
- [7] A. Álvarez-Marquina E. Martinez de Icaya C. González- Consejero, V. Rodellar and P. Gonzalez-Vilda, "A portable hardware design of a FFT algorithm", *Latin American Applied Research*, 2007.
- [8] Randy Yates, "Fixed-point arithmetic: An introduction", 2009, <http://www.digitalsignallabs.com>.
- [9] David Bishop, "Fixed point package user's guide".
- [10] Frank Vahid, Digital Design with RTL Design, VDHL and Verilog, John Wiley and Sons, second edition, 2011.
- [11] J. A. C. Bingham, "Multicarrier modulation for data transmission: an idea whose time has come," *IEEE Communication Magazine*, vol. 28, no. 5, pp. 5-14, May 1990.
- [12] J. Palicot and C. Roland, "FFT: a basic function for a reconfigurable receiver," 10th International Conference on Telecommunications, vol. 1, pp. 898-902, March 2003.