IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 5, Issue 2, Ver. II (Mar. - Apr. 2015), PP 26-30
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

VLSI Implementation of Split-Radix Fast Fourier Transform: ASurvey

Siddhartha Bhavaraju[1], MousamiVaibhav Munot[2], Lalit P.Patil[3], S. PrabhuKumar[4]
[1]VelTech Dr RR & Dr.SR Technical University, Chennai, CDAC-ACTS, Pune, India, [2]Pune Institute of Computer Technology, Pune, India,
[3]Pune Institute of Computer Technology, Pune, India,
[4]VelTech Dr RR & Dr.SR Technical University, Chennai, India,

Abstract:*The purpose of a transform is to consider an algorithm or a fixed procedure or a set of rules or anyequation that changes one set of data to another set of data. The Fast Fourier Transform which is an efficient way to calculate the Discrete Fourier Transform produces results by first dividing the equation into even and odd terms. A type of Fast Fourier Transform algorithm is Split-Radix where, it uses both radix-2 and radix-4 for its simplicity and efficiency respectively The purpose of this paper is to give review of work done on split-radix algorithm and its advantages when compared to other radix implementations.*

Index Terms: *Cooley-Tukey algorithm, DFT, FFT, FPGA, split-radix.*

I. Introduction

The main question to answer first is why we need the transforms and then why FFT and also why split-radix. So, first we see about transform. Transform is used when we use real time signal is present i.e. if it is time domain then we have to convert to frequency domain. So the time domain is converted to samples. This information is got out from samples using the transforms it just enables to see the information at the output. So the transformation aides in visibility of information in one way or the other. There are different kinds of transforms used out of which DFT is preferred and among DFT algorithms FFT algorithm is preferred as it is faster than other algorithms. We consider speed as our age considers or prefers the speed of a system as it helps in the performance. The discrete Fourier transform is the most important discrete transformation used to perform Fourier analysis in many applications. In digital signal processing i.e. DFT, a function is any quantity or signal that varies over time, just like the pressure of a sound wave, a radio signal, or daily temperature reading, sampled over a finite time interval. The DFT is also used to efficiently solve partial differential equations, and not only that it is used to perform other operations such as convolutions or multiplying large integers. The equation of DFT is as follows

$$X_k = \sum_{n=0}^{N-1} x_n \omega_N^{nk}$$

Advantage of DFT is it uses finite set of data so it can easily be implemented on dedicated hardware. Also the DFT is discrete both in frequency and time which is used easily for implementing a hardware. A FFT is an algorithm to compute the DFT and its inverse. Fourier analysis converts a time signal to a frequency signal and vice-versa and FFT rapidly computes such transformations by factorizing the DFT matrix into a product of spare (mostly zero) factors. As a result, FFT is widely used for many applications in engineering, science and mathematics, form simple complex-number arithmetic to group theory and number theory. The DFT is obtained by the decomposition of a sequence of values into components of different frequencies. This operation is used in many fields but computing it directly from the definition is often too slow to be practical. An FFT is one of the ways to compute the same result more quickly i.e. computing the DFT of N points in a naive way, using the definition, takes order of N^2 arithmetical operations, while a FFT can compute the same DFT in only the order of N log2 N operations. The difference in speed can be effectively huge, especially for long data sets where N maybe in thousands or millions.

II. Split-Radix Algorithm

In general several many new algorithms can be categorized into two main classes, in first class N equals to 2^m where m is natural number. The basic idea of split-radix algorithm is that it, uses radix-2 for even sequence number and radix-4 for odd sequence number. Split-radix algorithm is the best algorithm with least number of arithmetic operations [1].

$$X_k = \sum_{n_2=0}^{N/2-1} x_{2n_2}\omega_{N/2}^{n_2k} + \omega_N^k \sum_{n_4=0}^{N/4-1} x_{4n_4+1}\omega_{N/4}^{n_4k} + \omega_N^{3k} \sum_{n_4=0}^{N/4-1} x_{4n_4+3}\omega_{N/4}^{n_4k}$$

Where x2n2 is for even indices and x4n4 is for odd indices. And the radix-2 and radix-4 butterfly equations are as follows

$$X_{2k} = \sum_{n=0}^{(N/2)-1} (x_n + x_{n+(N/2)})W^{2nk}$$

$$X_{4k+1} = \sum_{n=0}^{(N/4)-1} [(x_n - x_{n+(N/2)}) - j(x_{n+(N/4)} - x_{n+3(N/4)})]W^{n}W^{4nk}$$

$$X_{4k+3} = \sum_{n=0}^{(N/4)-1} [(x_n - x_{n+(N/2)}) + j(x_{n+(N/4)} - x_{n+3(N/4)})]W^{3n}W^{4nk}$$

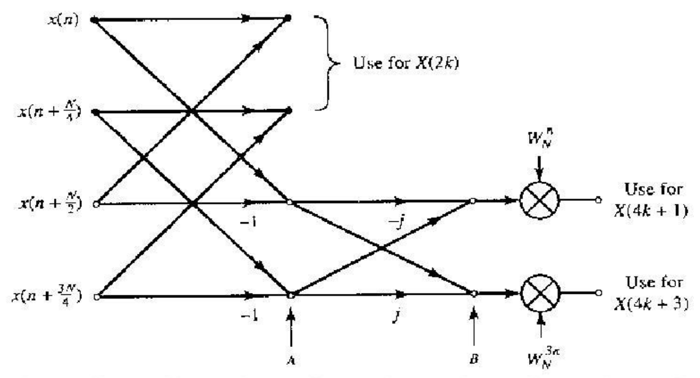

Fig.1 Split-Radix Butterfly

III. Related Work

This paper presents a novel split-radix FFT pipeline architecture design. A mapping method has been developed to obtain regular and modular pipeline for split-radix algorithm. The pipeline is re-partitioned into balance the latency between complex multiplication and butterfly operation by using carry and save addition. The number of complex multiplier is minimized via a bit-inverse and bit-reverse data scheduling scheme. One can also apply the design methodology described here to obtain regular and modular for the other Cooley-Tukey-based algorithms [2].

This paper presents a simple recursive modification of the split-radix algorithm that computes the DFT with asymptotically about 6% fewer operations when compared to Yavne, matching the count achieved by Van Buskirk's program-generation framework. It also discusses the application of our algorithm to real-data and real-symmetric transforms, where it is to achieve lower arithmetic counts than previously published algorithms [3].

This paper presents an efficient Fortran program that computes the split-radix FFT. The indexing scheme of things is used that gives a three-loop structure for the split-radix FFT that is very similar to the conventional Cooley-Tukey FFT. Both decimation-in-time (DIT) and a decimation-in-frequency (DIF) are presented. An arithmetic analysis is made for the purpose to compare the operation count of the Cooley-Tukey FFT for several different radixes to that of the split-radix algorithm. The split-radix algorithm seems to require the least total arithmetic of any power-of-two DFT algorithm [4].

In this paper, a general class of split-radix FFT algorithms for computing the length-2^m DFT is proposed by introducing a new recursive approach coupled with an efficient method for combining the twiddle factors. This encourages the development of higher split-radix FFT algorithms from lower split-radix FFT algorithms without any increase in the arithmetic complexity. Specifically, an arbitrary radix-$2/2^s$ FFT algorithm for any value of s, $4 \leq s \leq m$, is proposed and its arithmetic operations required by the proposed radix-$2/2^s$ FFT algorithm is independent of s and is $(2m-3)2^{m+1}+8$ regardless of whether a complex multiplication is carried out using four multiplication and two additions or three multiplications and three additions. This paper thus provides a variety of choices and ways for computing the length-2^m DFT with the same arithmetic complexity [5].Thispaper shows any Mathematical applications such as DFT and convolution are two main and common operations used in signal processing applications. Many of the other Signal processing algorithms such as filter, spectrum estimation and OFDM can be transformed into DFT to implement it on hardware. FFT is the collection of algorithms that performs the DFT at a higher speed. FFT is used in most signal processing applications, so the designing of an appropriate algorithm for the implementation of FFT can be most important in digital signal processing. The techniques such as pipelining and parallel calculations have some potential impacts on VLSI implementation of FFT algorithm. So by theoretical observations Split-radix algorithm is an appropriate algorithm for the implementation of FFT among all the effective algorithms of FFT, because it reduces number of arithmetic operations to great extent. This algorithm performs well with the implementation on FPGA kit and ASIC, satisfies the requirement of high speed [6].

The paper deals with the efficient computation of one-butterfly in-place complex split-radix fast Fourier transform algorithm. The proposed approach is based on the conventional three-loop index structure, in which redundancies are associated with the indexing scheme which have been removed at the expense of memory. As a result an increase in speed of 10% is achieved depending on the FFT sequence length [7].

This paper shows high performance hardware FFTs have numerous applications in instrumentation and communication systems. This paper describes a new parallel FFT architecture which combines the split-radix algorithm with a constant geometry interconnect structure. The split-radix algorithm is known to have lower multiplicative complexity than both radix-2 and radix-4 algorithms. However, it conventionally involves an"L-shaped" butterfly data path whose irregular shape has uneven latencies and makes scheduling difficult. This work proposes a split-radix data path that avoids the L-shape. With this, the split-radix algorithm can be mapped onto a constant geometry interconnect structure in which the wiring in each FFT stage is identical, resulting in low multiplexing overhead. Further, we exploit the lower arithmetic complexity of split-radix to lower dynamic power, by gating the multipliers during trivial multiplications. The proposed FFT achieves 46% lower power than a parallel radix-4 design at 4.5GS/s when computing a 128-point real-valued transform [8].

This paper presents a new algorithm is presented for the fast computation of the Discrete Fourier Transform. This algorithm belongs to that class of recently proposed 2 -FFTs which present the same arithmetic complexity (the lowest among any previously published one). Moreover, this algorithm has the advantage of being performed in-place, by repetitive use of a "butterfly"- type structure, without any data reordering inside the algorithm. Furthermore, it can easily be applied to real and real symmetric data with reduced arithmetic complexity by removing all redundancy in the algorithm [9].

This paper presents Fast Fourier Transform (FFT) is a very common operation used for various signal processing units. Many efficient algorithms are being designed to improve the architecture of FFT. Among the different algorithms, split-radix FFT has shown considerable improvement in terms of reducing hardware complexity of the architecture compared to radix-2 and radix-4 FFT algorithm. The performance in terms of throughput of the processor is limited by the multiplication. Therefore multiplier is optimized to make the input to output delay as short as possible. Distributed arithmetic (DA) is one of the most used techniques in implementing multiplier-less architectures of many digital systems. In this paper a Split Radix FFT without the use of multiplier is designed. All the complex multiplications are done by using Distributed Arithmetic (DA) technique. For faster calculation parallel prefix adder is used. These algorithms reduces overall arithmetic operations in FFT, but increases the number of operations and complexity of each butterfly. In Split Radix FFT, mixed-radix approach helps to achieve low number of multiplications and additions. The advantage of DA is its efficiency of mechanization. A method is incorporated to overcome the overflow problem introduced by DA method [10].

In this paper, the split-radix approach for computing the one-dimensional (1-D) discrete Fourier transform (DFT) is to compute the two-dimensional (2-D) DFT of size $2^{r1}x2^{r2}$ using a radix-2x2 index map and a radix-8x8 map instead of a radix-2x2 index map and a radix-4x4 map as is done in the classical split vector radix FFT algorithms. Since a radix-8x8 index map and a method for combining the twiddle factors are used, the new algorithm provides significant compared to the 2-D FFT algorithms previously reported in terms of the arithmetic complexity, data transfer, index generation and twiddle factor evaluation or access to the lookup table. In addition, since the algorithm is expressed in a simple matrix form using the Kronecker product, it facilitates an easy implementation of the algorithm, and allows for an extension to the multidimensional case [11].

In this paper the split-radix algorithm for the discrete Fourier transform of length-2^{m}" is by now fairly popular. First, we give the reason why the split-radix algorithm is better than any single-radix algorithm on length-2^{m} DFT's. Then, the split-radix approach is generalized to Iength-p ^{m} DFT's. It is shown that whenever a radix-p^2 outperforms a radix-p algorithm, then a radix-p/p^2 algorithm will outperform both of them. As an example, a radix-3/9 algorithm is developed for length-3^{m} DFT's [12].

In this patent an apparatus for performing a FFT is provided. The apparatus comprises a reorder matrix, symmetrical butterflies, and a memory. The reorder matrix is configured to have a constant geometry, and the butterflies are coupled in parallel to the reorder matrix. The memory is also coupled to the reorder matrix and each butterfly. The reorder matrix, the butterflies, and the memory can then execute a split radix algorithm [13].

In this patent an SR-2/8 FFT apparatus includes a memory, an SRFFT processor and a control unit. The control unit includes an input control block, an SRFFT control block and an output control block. The input control block loads memory banks with the input data in a first order, such that the SRFFT processor is able to retrieve data from the memory banks simultaneously in a single clock cycle. The SRFFT control block determines a decomposition structure of a 2^{M}-point FFT and controls the SRFFT processor to repeatedly perform a butterfly computation along the decomposition structure. The order of the input data of each butterfly computation fits in with the first order. The SRFFT control block controls output results of each butterfly computation to be written back into the memory, banks corresponding to the input data. The output control block controls the output to be output-ted in a second order [14].

In this patent a first number is multiplied by a second number, by representing the first number as a first set of one or more W-bit wide numbers, and representing the second number as a second set of one or more W-bit wide numbers. Each of the W-bit wide numbers from the first set is paired with each of the W-bit wide numbers from the second set. For each pair of W-bit wide numbers, a set of sub-partial products is generated. Combinations of sub-partial products are formed such that each combination is representable by a W-bit wide lower partial product and a carry out term that has fewer than W-bits. The W-bit wide lower partial products and the carry out terms are combined to form the product of the first number and the second number. The carry out term is advantageously representable by (W/2)+1 bits. In this way split-radix multiplication is done in this patent [15].

IV. Observations

Most of the conventional radix-r FFT pipeline has the restriction that the length of FFT has to be power of r. But by using split-radix algorithm such restriction can be removed [2]. When coming up with new and modified versions of split-radix two things should be answered. First, can one do better still and simplify the algorithm? Second, will the new and modified version result in practical improvements to actual computation times for the FFT? [3]

If we write a program (e.g. like Fortran program) for split-radix algorithm we have to make sure that it is an improvement on Cooley-Tukey algorithm and the arithmetic complexity should be improved when compared to the Cooley-Tukey algorithm[4]. For the purpose of implementation we have to make sure that the butterfly diagram of the slit-radix algorithm is exact and compact and no changes should be made once fixed otherwise the implementation might go haywire.

V. Conclusion

The implementation of split-radix FFT can be efficiently done on a suitable FPGA kit when compared to other implementations. Since DFT is carried out in the digital domain, there are several methods which are related to implementing the system. First, we can implement using ASIC technology because ASIC are the fastest, smallest, and lower power way to implement DFT into hardware, but it has limitations such as inflexibility of design process involved and the longer time to market period for the designed chips. Another method that can be used to implement DFT is general purpose Microprocessor or Micro Controller. The disadvantages of using this kind of hardware are, it needs memory and also a few other peripheral chips to support operation. Besides that, it uses the most power usage and memory space, and would be slowest in terms of time to produce the output compared to other hardware. By FPGA implementation of DFT flexibility is obtained to the program design besides the low cost hardware component compared to others.

Acknowledgment

The author wishes to thank Mrs.VaishaliMaheshkar for her counsel in the preparation of this work.

References

[1]. P. Duhamel and H. Hollmann, "Split-radix FFT algorithm," Electron. Lett. 20 (1), 14–16 (1984).
[2]. Wen-Chang Yeh and Chein-Wei Jen, “High Speed and Low-Power Split-Radix FFT”, IEEE TRANSACTIONS ON
[3]. SIGNAL PROCESSING, VOL. 51, NO. 3, MARCH 2003.
[4]. Steven G. Johnson and Matteo Frigo, “Modified Split-Radix FFT With Fewer Arithematic Operations”, IEEE
[5]. TRANSACTIONS ON SIGNAL PROCESSING, VOL. 55, NO. 1, JANUARY 2007.
[6]. Henrik V. Sorensen, Michael T. Heideman, and C. Sidney Burrus, “On Computing the Split-Radix FFT”, IEEE
[7]. TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING, VOL. ASSP-34, NO. 1, FEBRUARY 1986.
[8]. SaadBouguezel, M. Omair Ahmad, M. N. S. Swamy, “A General Class of Split-Radix FFT Algorithms for theComputation of the DFT of Length-2^m, IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 55, NO. 8, AUGUST 2007.
[9]. NandyalaRamanatha Reddy, Lyla B. Das, A. Rajesh, SriharshaEnjapuri, “ASIC Implementation of High speed Fast
[10]. Fourier Transform Based on Split-Radix algorithm”, International Conference on Embedded Systems- (ICES-2014).
[11]. A.N. Skodras, A.G. Constantinides, “Efficient coputation of the split-radix FFT”, IEEE PROCEEDINGS-F, VOL. 139,NO. 1, FEBRUARY 1992.
[12]. Joyce Kwng, Manish Goel, “A High Performance Split-Radix FFT with Constant Geometry Architecture”, International
[13]. Conference on Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012.
[14]. Pierre Duhamel and HenkHollmann, “Implementation of Split-Radix FFT Algorithms for Complex, Real, and Real-Symmetric Data”, IEEE Internationational Conference on Acoustics, Speech, and Signal Processing (ICASSP) 1985.
[15]. NishaLaguri, K. Anusudha, “VLSI Implementation of Efficient Split Radix FFT Based on Distributed Arithmetic”,
[16]. International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE) 2014.
[17]. SaadBouguezel, M. Omair Ahmad, M. N. S. Swamy, “ A Split-Radix Algrithm for 2-D DFT”, International Symposiumon Circuits and Systems (ISCAS) 2003.
[18]. Martin Vetterli and Pierre Duhamel, “Split-Radix for Length-p^mDFT’s”, IEEETRANSACTIONS on Acoustics, Speechand Signal Processing, VOL. 37, NO. 1, JANUARY 1989.
[19]. Joyce Y. Kwong, Manish Goel, “Constant Geometry Spit-Radix FFT”, U.S Patent 12/335,256, (KR) Nov. 22, 2011.
[20]. Heng-Tai Tang, “Apparatus and Method for Split-Radix-2/8Fast Fourier Transform”, U.S Patent 13/048,344, (TW) Dec.3, 2013.
[21]. Anders Berkeman, “Split Radix Multiplication”, U.S Patent 10/701,634, Jan. 8, 2008.