International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Comparative Performance Analysis of Fast Fourier Transform on ARM and DSP Core Using Standard Benchmarks

Vinay B K[1], Manjula N Harihar[2]

[1]M.Tech Student (SP & VLSI), Department of Electronics and Communication Engineering, Jain University, Karnataka, India

[2]Assistant Professor, Department of Electronics and Communication Engineering, Jain University, Karnataka, India

Abstract: ***This paper focuses on a sort of benchmarking and is intended to compare the performance of ARM core and DSP core using some Standard Benchmark or program. First a Standard Benchmark "Dhrystone" was used, which is an open source benchmark. It was tailored to be run on these cores. Later it was optimized using hardware and compiler options. As this benchmark was targeted for non-floating point systems it does compare only one aspect of a processor. Another Benchmark DSPlib which was available on Texas Instruments Website for public access. This library consists of various commonly used Digital Signal Processing Functions such as FFT, IIR Filter, FIR Filter, LMS and some Vector Functions such as Vector Addition, Vector Multiplication, etc. DSPlib has separate directory for each functions and algorithms and folder consists of header files, C program, and precompiled executable object file. There were two version of programs; one were written using only C language contain no intrinsics and pragma directives called Natural C Program others version was optimized for DSP core containing intrinsics function related to core and pragma directives for optimizing the code.***

Keywords: FFT, DIT, *Benchmark, Dhrystone, ILP, Intrinsics, Pragma, VLIW*

1. Introduction

The Benchmarking can help in critical decisions making in some crucial businesses. Benchmarking is the process of comparing one's business processes and performance metrics to industry bests or best practices from other industries, but in electronics industry Benchmarking is an act of running a computer program (Benchmarks), a set of similar programs in order to assess the relative performance of a system, normally by running a number of standard tests and trials against it. The Benchmarks or Programs that were used are Dhrystone, Whetstone and FFT Program, vector Functions. The Cores which are to be compared are ARM core and DSP core. The ARM core processor is a high-performance, low-power processor which consists of Vector Floating-Point architecture extension is for floating-point Computation that is fully compliant with the IEEE 754 standard. The DSP core has advanced Very long instruction word (VLIW) architecture with 8 functional units (two multiplier units and six arithmetic logic units) that operate in parallel on the whole it consists of 64 general-purpose 32-bit registers.

VLIW refers to a processor architecture designed to take advantage of instruction level parallelism (ILP). Whereas conventional processors mostly only allow programs that specify instructions to be executed one after another, a VLIW processor allows programs that can explicitly specify instructions to be executed at the same time. This architecture is intended to allow higher performance without the inherent complexity of some other approaches.

2. Review of FFT Algorithm

The basic principle behind most Radix based FFT algorithms is to exploit the symmetry properties of a complex exponential that is the cornerstone of the Discrete Fourier Transform (DFT), These algorithms divide the problem into similar sub-problems (butterfly computations), and achieve a reduction in computational complexity. All Radix algorithms are similar in structure differing only in the core computation of the butterflies. The FFT differs from the other algorithms in that it uses a real kernel, as opposed to the complex exponential kernel used by the Radix algorithms. The DITF algorithm uses both the Decimation- In-Time (DIT), and Decimation-In-Frequency (DIF), frameworks for separate parts of the computation to achieve a reduction in the computational complexity.

A. Radix-2 Decimation in Time Algorithm

Radix-2 DIT- FFT is the simplest and most common form of the Cooley–Tukey algorithm, although highly optimized Cooley–Tukey implementations typically use other forms of the algorithm as described below. Radix-2 DIT divides a DFT of size N into two interleaved DFTs (hence the name "radix-2") of size $N/2$ with each recursive stage. The discrete Fourier transform is defined by the formula

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\Pi i}{N} nk} \quad (1)$$

Where k is an integer ranging from 0 to N-1.

Radix-2 DIT first computes the DFTs of the even-indexed inputs and odd-indexed inputs and then combines those two results to produce the DFT of the whole sequence. This idea can then be performed recursively to reduce the overall runtime to (Nlog N). This simplified form assumes that N is a power of two.

The Radix-2 DIT algorithm rearranges the DFT of the function x_n into two parts: a sum over the even-numbered indices n=2m and a sum over the odd-numbered indices n=2m+1.

$$X_k = \sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\Pi i}{N}(2m)k} + \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\Pi i}{N}(2m+1)k} \quad (2)$$

The full radix-2 decimation-in-time decomposition is illustrated in Figure1 using the simplified butterflies involves $M=\log_2 N$ stages, each with $N/2$ butterflies per stage. The total computational cost of radix-2 algorithm is $N^2 \log_2 N$ complex multipliers and $N \log_2 N$ complex adders.

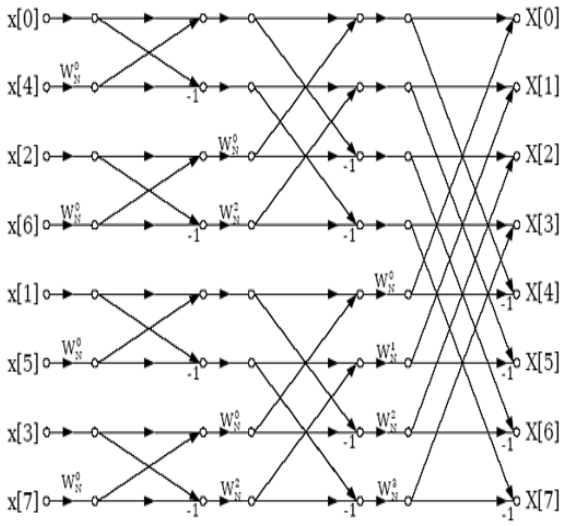

Figure 1: Radix-2 Decimation-in-Time FFT algorithm for a length-8 signal

B. Radix-4 Decimation in Time Algorithm

The Radix-4 algorithm is very similar to the Radix-2 algorithm in concept. Instead of dividing the DFT computation into halves as in Radix-2, a four-way split is used. The N-point input sequence is split into four subsequences. The decimation process is similar to the RAD2 algorithm, and uses $v=\log_4 N$ stages, where each stage has N/4 butterflies. TheRadix-4 butterfly involves 8 complex additions and 3 complex multiplications, or a total of 34 floating point operations. Thus, the total number of floating point operations involved in the Radix-4 computation of an N-point DFT is 4.25log2N, which is 15% less than the corresponding value for the Radix-2 algorithm.

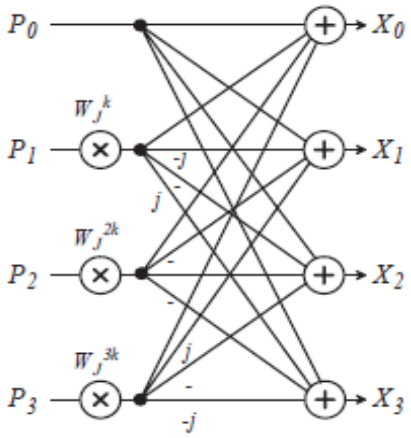

Figure 2: Signal flow graph of radix-4 DIT butterfly.

Radix-4 algorithms have a computational advantage over radix-2 algorithms because one radix-4 butterfly does the work of four radix-2 butterflies, and the radix-4 butterfly requires only three complex multipliers compared to four complex multipliers of four radix-2 butterflies. Radix-2 and radix-4 FFTs are the most commonly used algorithms, it is also possible to design FFTs with even higher radix butterflies. The reason they are not often used is because the control and dataflow of their butterflies are so complicated that the additional efficiency gained is lost.

The DITF algorithm is based on the observation that in a DIF implementation of a Radix-2 algorithm, most of the computations (especially complex multiplications), are performed during the initial stages of the algorithm. In the DIT implementation of the Radix-2 algorithm, the computations are concentrated towards the final stages of the algorithm. Thus, starting with the DIT implementation and then shifting to the DIF implementation at some transition stage intuitively seems to be a computation saving process.

3. Benchmarking Criteria

Most preceding FFT complexity studies have been conducted on special purpose hardware such as digital signal processing. Typically, the primary benchmarking criteria have been the number of mathematical operations (multiplications and additions), and/or the overall computation speed. The efficiency of an algorithm is most influenced by the arithmetic complexity, usually expressed in terms of a count of real multiplications and additions. However, on general purpose computers this is not a very good benchmark and other factors need to be considered as well. For instance, the issue of memory usage is very important for memory constrained applications.

A. Number of Computations

Since many general purposes CPUs have significantly different speeds on floating point and integer operations, we decided to individually account for floating point and integer arithmetic. It is a well-known fact that most new architectures compute floating point operations more efficiently than integer operations. Also, most indexing and loop control is done using integer arithmetic. Therefore the

integer operations count directly measures the cost of indexing and loop control. Many FFT algorithms require a large number of divisions by- two operations which is efficiently accomplished by using a binary shift operator.

B. Computation Speed

In most present-day applications for general purpose computers, with easy availability of faster CPUs and memory not being a primary constraint, the fastest algorithm is by far treated as the best algorithm. Thus, a common choice to rank order algorithms is by their computation speed. One of the classic trade-offs seen in algorithm development is that of memory usage versus speed. In most portable signal processing applications, the FFT is a core computational component. However, few applications can afford a large memory space for evaluating FFTs. While memory usage is important for specification of hardware, memory accesses also account for a significant portion of computation time.

4. Benchmarking Results and Analysis

Each of the algorithms was implemented under a common framework using common functions for operations such as bit-reversal and lookup table generation so that differences in performance could be attributed solely to the efficiency of the algorithms on different cores. Following this, we comprehensively benchmarked each algorithm according to the criteria discussed in the previous section. Computation speed is typically the most prominent aspect of an FFT algorithm in current DSP applications. The computation speed of an algorithm for large data sizes can often be heavily dependent on the clock speed, RAM size, cache size and the operating system. Hence, these factors must be taken into account. In this paper we try to bring out the comparison between ARM and DSP cores for number of CPU Cycles required to execute the FFT algorithm using various memory units.

FFT algorithm both radix-2 and radix-4 were implemented on ARM and DSP cores for comparison in three different variations of coding i.e. in Natural C, Optimized C and in Assembly Language. All these variations were experimentally carried on both On Chip Memory and on Cache memory. In Natural C program none of the intrinsics, core specific header files and optimizing directives options were used whereas in optimized c program all of these were used.

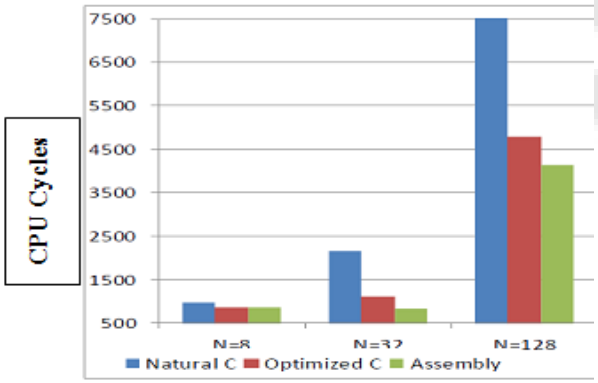

Figure 3: Radix-2 FFT on DSP core using On chip Memory at 600 MHz

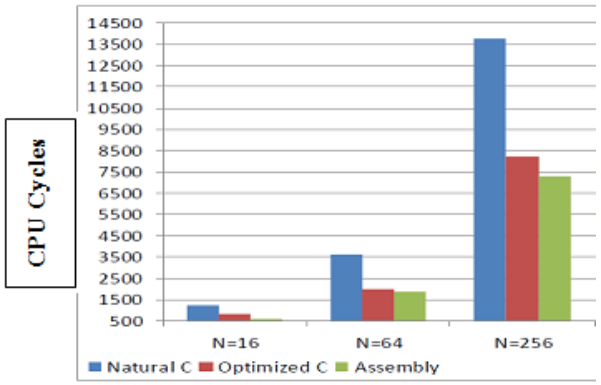

Figure 4: Radix-4 FFT on DSP core using On chip Memory at 600 MHz

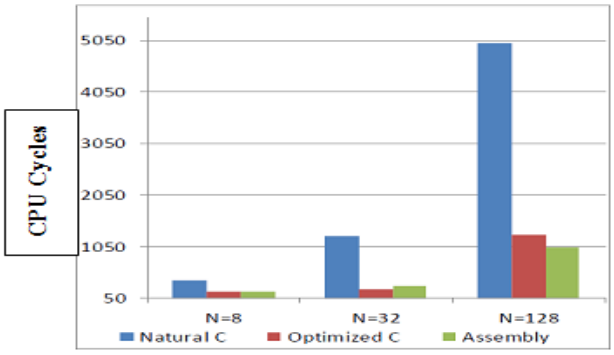

Figure 5: Radix-2 FFT on DSP core using Cache Memory at 600 MHz

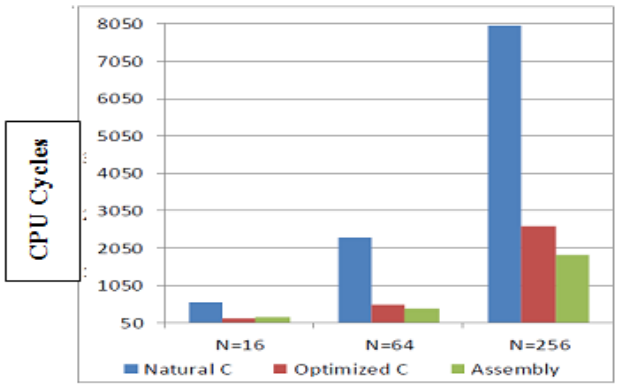

Figure 6: Radix-4 FFT on DSP core using Cache Memory at 600 MHz

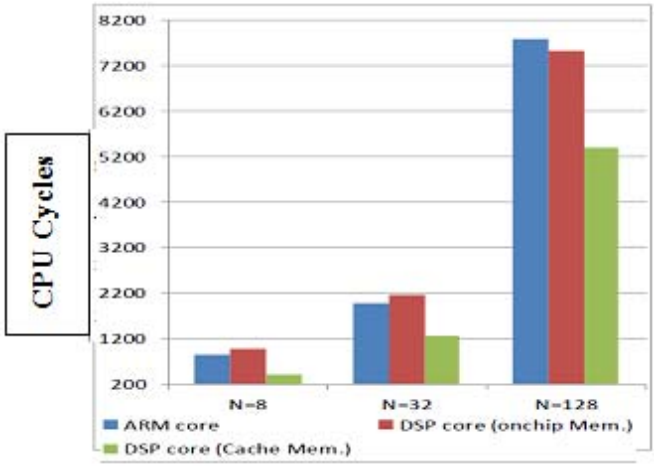

Figure 7: Radix-2 FFT on DSP core and ARM core 600 MHz

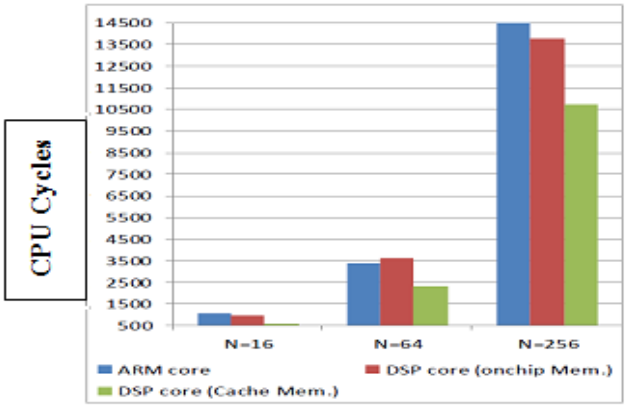

Figure 8: Radix-4 FFT on DSP core and ARM core at 600 MHz

5. Conclusion

Some SOC's include both ARM and DSP cores and subsystems. In order to reduce redundancy at performing some common required tasks at comparable efficiency and speed, their performance metrics will be compared and analyzed and then decision could be taken which could save cost, area, and power consumption. The scores for Dhrystone conclude that ARM core is better than DSP for running non-numeric programs like Dhrystone. The scores for FFT conclude that DSP core is better than ARM core when it comes to execution of non vectorizable floating point calculations. The scores for vector functions conclude that ARM core gives a good combat to DSP in a program which involves vector operations.

References

[1] J. Cooley and J. Tuckey, "An algorithm for the machine calculation of the complex fourier series," Math. Comput., vol. 19, pp. 297–301, Apr. 1965.
[2] Tran-Thong and B. Liu, "Fixed-point fast fourier transform error analysis," IEEE Trans. Acoust., Speech, Signal Processing, vol. 24, no. 6, pp. 563– 573, Dec. 1976.
[3] Granata, M. Conner, and R. Tolimieri, "Recursive fast algorithms and the role of the tensor product," IEEE Trans. Signal Processing, vol. 40, no. 12, pp. 2921–2930, Dec. 1992.
[4] S. F. Gorman and J. M. Wills, "Partial column fft pipelines," IEEE Trans. IEEE Trans. Circuits Syst. II, vol. 42, no. 6, pp. 414–423, June 1995.
[5] S. He and M. Torkelson, "Design and implementation of a 1024-point pipeline fft processor," in Proc. IEEE Custom Integrated Circuits Conf., Santa Clara, CA, May 11-14 1998, vol. 2, pp. 131–134.
[6] E. H. Wold and A. M. Despain, "Pipeline and parallel-pipeline fft processors for vlsi implementations," IEEE Trans. Comput., vol. 33, no. 5, pp. 414– 426, May 1984.
[7] M. Hasan and T. Arslan, "Implementation of low power fft processor cores using a novel order-base processing scheme," in Proc. IEEE Circuits Devices Syst., June 2003, vol. 150, pp. 149–154.
[8] M. Wosnitza, M. Cavadini, M. Thaler, and G. Tr¨oster, "A high precision 1024-point fft processor for 2d convolution," in Dig. Tech Papers IEEE Solid-State Circuits Conf., San Francisco, CA, Feb. 5-7 1998, pp. 118–119.
[9] A. M. Despain, "Fourier transform computers using cordic iterations," IEEE Trans. Comput., vol. 23, no. 10, pp. 993–1001, Oct 1974.
[10] A. Berkeman, V. Owall, and M. Torkelson, "A low logic depth complex multiplier using distributed arithmetic," IEEE Solid-State Circuits, vol. 35, no. 4, pp. 656–659, Apr. 2000.
[11] L. Wanhammar, DSP Integrated Circuits, CA: Academic Press, San Diego, 1999.
[12] H.S. Hou, (1987), The Fast Hartley Transform Algorithm, IEEE Transactions on Computers, pp. 147-155, February.
[13] P. Duhamel and H. Hollomann, (1984), Split Radix FFT Algorithm, Electronic Letters, vol. 20, pp. 14-16, January.
[14] H. Guo, G.A. Sitton, and C.S. Burrus, (1994), The Quick Discrete Fourier Transform, Proceedings of International Conference on Acoustics, Speech and Signal Processing, vol. 3, pp. 445-447, Adelaide, Australia.
[15] A. Saidi, (1994), Decimation-In-Time Frequency FFT Algorithm, Proceedings of International Conference on Acoustics, Speech and Signal Processing, vol. 3, pp. 453-456, Adelaide, Australia.
[16] C.S. Burrus and T.W. Parks, (1985), DFT/FFT and Convolution Algorithms: Theory and Implementation, John Wiley and Sons, New York, NY, USA.
[17] J.G. Proakis, D.G. Manolakis, (1992), Digital Signal Processing - Principles, Algorithms and Applications, Macmillan Publishing Company, NY, USA, 1992.
[18] A.V. Oppenheim, R.W. Schafer, (1989), Digital Signal Processing, Prentice-Hall International Inc., Englewood Cliffs, NJ, USA.
[19] R.N. Bracewell, (1990), Assessing the Hartley Transform, IEEE Transactions on Acoustics Speech and Signal Processing, pp. 2174-2176.
[20] http://www.ti.com/sc/docs/dsps/literatu.htm
[21] http://www.lsidsp.com/c6x/tech/wpsy nop.htm

Author Profile

Mr. Vinay BK is a student in the Department of Electronics and Communication Engineering, School of Engineering and Technology, Jain University, Karnataka, India. He received his Bachelor degree in Electronics & Communication Engineering from VTU in 2012. He is pursuing M. Tech (SP and VLSI Design) in Electronics and Communication Engineering, Jain University, Karnataka, India. His research interest includes Low power VLSI Design; Analog and Mixed signal VLSI Design, Circuit design and simulations, DSP, and Embedded Systems Design, FPGA prototyping, Emulation, Verification and Validation.

Manjula. N. Harihar is a Assistant Professor in the Department of Electronics and Communication Engineering, School of Engineering, Jain University, Bangalore. She obtained her Bachelor degree in Electronics and Communication Engineering from S.T.J Institute of Technology, Ranebennur. Master degree in communication Systems from P.D.A College of Engineering, Gulbarga, Karnataka, India. She is pursuing Ph.D in Electronics and Communication Engineering, Jain University, Bangalore. Her research interest includes Image Processing, VLSI, Neural Networks and Image Processing.