International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 05 Issue: 07 | July - 2021 ISSN: 2582-3930

Design of an Efficient FFT Algorithm for Processing ECG in Cardiac Pacemakers and Evaluating the Performance Using Open Core SOC

Sudharshan K M[1], Prashant V. Joshi[2], Ashok T[3]

[1] *School of ECE, REVA UNIVERSITY, Bangalore,*
[2] *School of ECE, REVA UNIVERSITY, Bangalore,*
[3] *School of ECE, REVA UNIVERSITY, Bangalore,*

---***---

Abstract - Design of efficient Fast Fourier Transform (FFT) Techniques Was implemented Using riscv open core SoC and have written a C code for FFT Radix-2 Algorithm for processing ECG in Cardiac Pacemakers and executed the same code on open core SoC platform and found its performance cycle count and also same code has been executed on ARM cortex-M7 analysed its states. The cycle count is compared between riscv and ARM Cortex-M7 and analysed its performance

***Keywords*—** riscv ,open core SoC platform, cardiac pacemaker, electrocardiogram (ECG), fast Fourier transform (FFT), radix-2

1. INTRODUCTION

Currently a days field of digital signal processing is very vital of imparting, there is a many growing in FFT algorithms which is way in conniving a network[1].In this paper explains the implementation and simulation of 8 -point FFT using radix- 2 algorithm. Because of radix-2, FFT can get less time delay, beat down the area complexity and, also reach cost conquered execution with less grow up time[2]. DFT is used to transform a time domain signal into its frequency spectrum field. DFT is calculating tool that play a very defining role in many digital signal processing applications. The main consequence of DFT in functional applications is due to a large extent on existing of computationally proficient algorithms, known as Fast Fourier Transform. (FFT) algorithms, for numeration of DFT[10]. There are many solitude FFT algorithms consequence a large spectrum of calculated from simple difficult calculation arithmetic to set theory and computation scheme. FFTs are algorithms for speedy computation of discrete Fourier transform of a information vector. The FFT is a discrete Fourier transform (DFT) algorithm which decline the count of computation requirement for N point from O(N 2) to O(N log N) all over log is the base-2 logarithm[3].

Fourier transform is surely one of the most leading signal processing algorithms, especially so for biomedical applications [10], [11]. The developments of digital signal sampling and the appearance of discrete Fourier transform (DFT) enable biomedical investigators to examine the frequency components of physiological signals and detect secrete belongings to identify physical irregularity. Such methods have been particularly effective in examining physical signals such as the electrocardiogram (ECG) [12]–[14]. A practical application of the above-mentioned signal survey can be found inside an inculcate cardiac pacemaker, which is a device tasked with continuous monitoring of ECG signals to detect heart rhythm abnormality [15]. In the event of a slow inherent rhythm or lack of normal impulse conduction, the pacemaker generates electrical stimulations to pace the heart muscle and reserve an adequate heart rate. A pacemaker has to operate in a highly power introverted environment and may last a very long period (~7–10 years) on a single battery. Therefore, the embedded system inside a cardiac pacemaker needs to progression the ECG signal with increase energy efficiency to ensure a sufficiently long battery life.

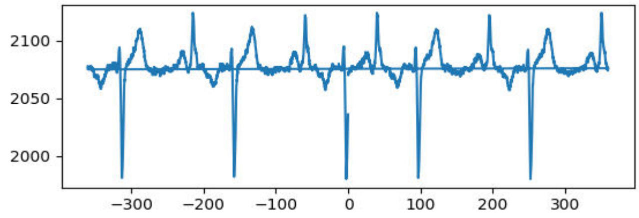


Figure 1:ecg signal

2. RADIX 2,FAST FOURIER TRANSFORM(FFT)

Radix-2 algorithms are the mostly recycled in FFT algorithms. While numeration DFT, we every time calculate N point DFT. The calculation of N can be component as,

N=r1, r2, r3,……….. rv ---(A)

Here r is a prime,

Now if r1,=r2=r3= …… rv =r

We can write, N=rv ----(B)

Here r is denomination as radix (base) FFT algorithm and v is shows number of stages in FFT algorithm.

Radix indicate base and if it's rate is '2' then it is said as radix-2 FFT algorithm. put the rate of r=2,

N=2v ------(C)

If we are calculating 2 point DFT then N=16

16=2v V=8

Therefore, designed for 2 point DFT, suitable are 8 stages of FFT Algorithm. If we are calculating 2 point DFT then N=32

32=2v

V=16

3. RISC-V

Architecture:

RISCY is an so as , one issue core with 4 pipeline stage and it's an IPC on the brink of 1, full maintenance for the bottom (integer instruction set)RV32I, (compress instructions)RV32C and (multiplication instruction set extension)RV32M. It can be arranged to have single-precision floating point instruction set extension (RV32F). It implement several instruction set architecture(ISA) extensions such as: hardware loops, post incrementing load and store instructions, bit-manipulation instructions, MAC operations, support fixed-point operations, packed SIMD instructions and the dot product. It has been intended to extend the energy efficiency of in ultra low power signal handling applications.

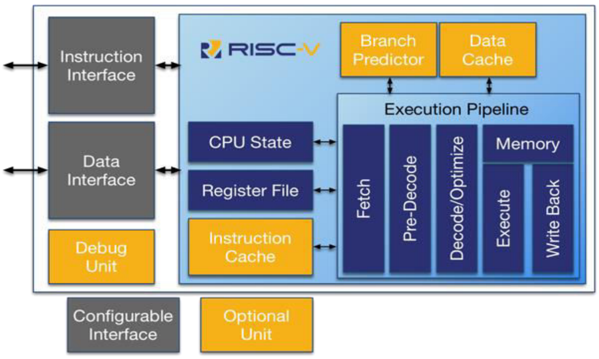


Figure 2: risc v architecture

Zero riscy is an in order, single issue core with 2 pipeline stages and it has full assistance for the base integer instruction set (RV32I) and compress instructions (RV32C). It are time and again have multiplication instruction set extension (RV32M) and cheap number of registers extension (RV32E).

4. METHODOLOGY

RISCV is a open core typical (ISA) based on originated reduced instruction set computer (RISC) rules. Unlike most other instruction set architecture(ISA) schemes, the RISC-V instruction set architecture(ISA) is delivered under open source licenses that do not need fees to use. A number of organizations are grant or have declare RISC-V HW(hardware), open source operating systems with RISC-V support have and the instruction set is supported in several standard software toolchains.

Detecting and categorizing irregularity in the ECG are key to recognize various types of irregular heartbeats known as Arrhythmia. The process of cataloging requires certain spectral features to be taken from the ECG [16], [17]. The process of FFT-based feature extraction and cataloging continual throughout the operating life cycle of the pacemaker, consuming a importance portion of the available energy. The power is less in the FFT algorithm is highly preferred.

In Fast Fourier transform algorithms, a butterfly configuration is a part of a calculation that jointly the Outcome of compressed discrete Fourier transforms (DFTs) during a massive DFT, or contrarily (demolition a massive DFT above during sub transforms). The title "butterfly" move nearer from a assembly of the data-flow resemblance in radix-2 case, as detailed beneath[5]. The before time event in punch of the period is clue to be in a 1969 MIT technical report. The similar type formation may as well be create in the Viterbi algorithm, utilized for locating the very likewise series of mystic states [6][7]. Twiddle factor mostly mentioned to the root of unity difficult multiplier repeated in butterfly stimulus of cooley-Tukey FFT Algorithm. Twiddle factor can as well be practical for every details free multiplier repeated in FFT[8].

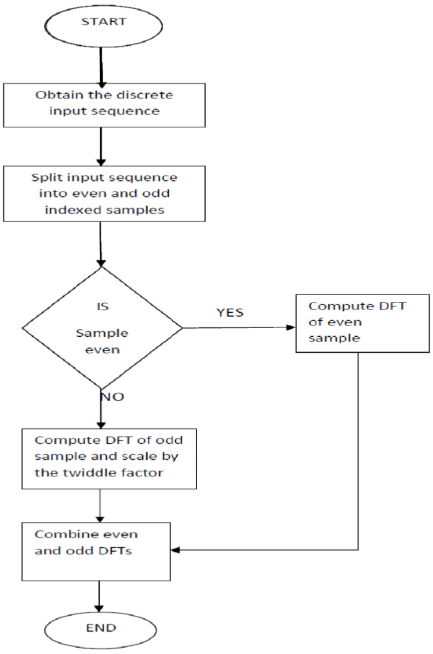

START
Obtain the discrete input sequence
Split input sequence into even and odd indexed samples
IS Sample even
YES
Compute DFT of even sample
NO
Compute DFT of odd sample and scale by the twiddle factor
Combine even and odd DFTs
END

Figure 3: FFT flow chart

Flow diagram

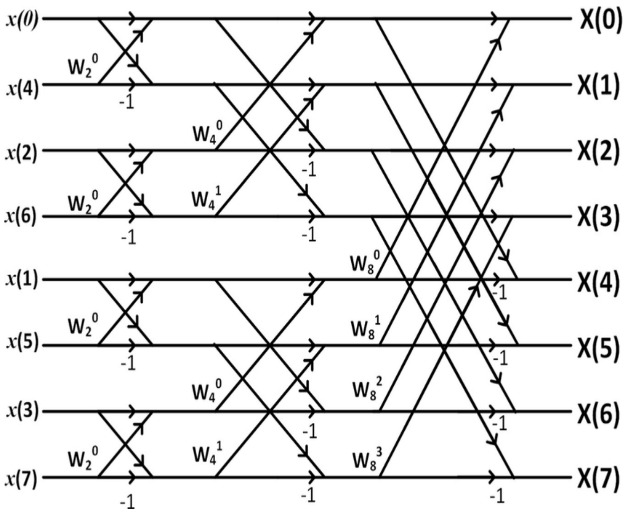


Figure 4: FFT flow diagram

4.1. Twiddle factor

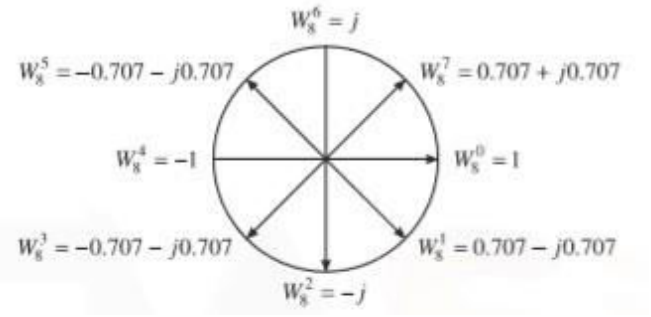


Figure 5:Twidle factor values

Twiddle factors are a set of values that is used to speed up DFT and IDFT computations which is represented with the letter W.

For a discrete sequence x(n), we can compute its Discrete Fourier Transform and Inverse Discrete Fourier Transform using the following formulas.

DFT x(k) = $\sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi nk}{N}}$ -----------(1)

IDFT x(n) = $\frac{1}{N} \sum_{n=0}^{N-1} x(k) e^{\frac{j2\pi nk}{N}}$ -----------(2)

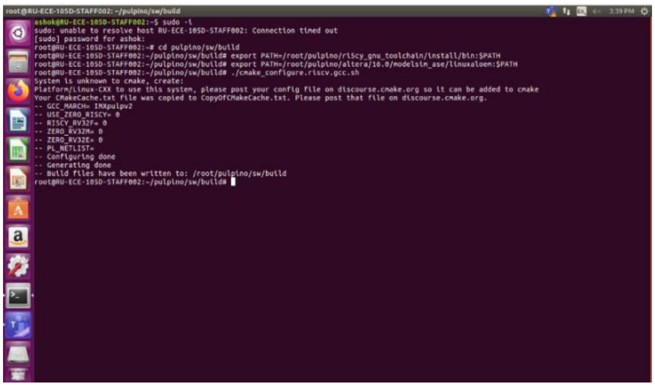
4.2 GNU Toolchain

One of the starting steps in the reorder virtual platform is setting up the GNU Tool chain for building the real-time kernel and its application. The Tool chain used in the embedded systems is known as cross-platform Tool chain This process of accumulating on a host to get code for the target system is called cross assembling and the compiler used for the purpose is called a cross-compiler [4]. The compiler requires lot of assistance libraries and binaries which are collective called cross-platform toolchain. The GNU Toolchain consists of elements like binutils, GNU Cross Compiler Collection (GCC), GDB, newlib. Toolchain binaries for certain stage already available in the form of binaries. Since it is not possible to add the new functionality in the binaries, so the Tool chain is made for ARM processor and a virtual timer patch is added to the GDB.

These are the following steps for building Tool chain as follows:

1. Target should be decided first
2. Version of binutils, GCC, GDB, Newlib should be decided
3. Suitable patches are to be downloaded and installed (if necessary)
4. Compile and build binutils
5. Compile and build minimal GCC
6. Compile and build Newlib
7. Compile and build GDB

4.3 Selecting the core



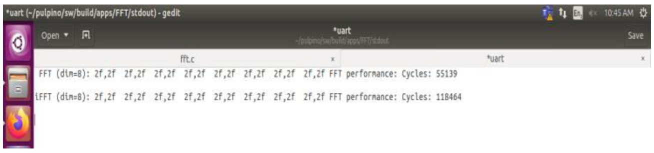
```
root@RU-ECE-105D-STAFF002: ~/pulpino/sw/build
ashok@RU-ECE-105D-STAFF002:~$ sudo -i
sudo: unable to resolve host RU-ECE-105D-STAFF002: Connection timed out
[sudo] password for ashok:
root@RU-ECE-105D-STAFF002:~# cd pulpino/sw/build
root@RU-ECE-105D-STAFF002:~/pulpino/sw/build# export PATH=/root/pulpino/riscy_gnu_toolchain/install/bin:$PATH
root@RU-ECE-105D-STAFF002:~/pulpino/sw/build# export PATH=/root/pulpino/altera/16.0/modelsim_ase/linuxaloem:$PATH
root@RU-ECE-105D-STAFF002:~/pulpino/sw/build# ./cmake_configure.riscv.gcc.sh
System is unknown to cmake, create:
Platform/Linux-CXX to use this system, please post your config file on discourse.cmake.org so it can be added to cmake
Your CMakeCache.txt file was copied to CopyOfCMakeCache.txt. Please post that file on discourse.cmake.org.
-- GCC_MARCH= IMXpulpv2
-- USE_ZERO_RISCY= 0
-- RISCY_RV32F= 0
-- ZERO_RV32M= 0
-- ZERO_RV32E= 0
-- PL_NETLIST=
-- Configuring done
-- Generating done
-- Build files have been written to: /root/pulpino/sw/build
root@RU-ECE-105D-STAFF002:~/pulpino/sw/build#
```

Figure 6:selecting the core

we have written C code for 8-point FFT using radix-2 algorithm and have compiled using ARM cortex-M7 and RISC V GNU toolchain in open core SoC platform And also compared its performance cycle count.

5. SIMULATION RESULTS

5.1 Output in UART file



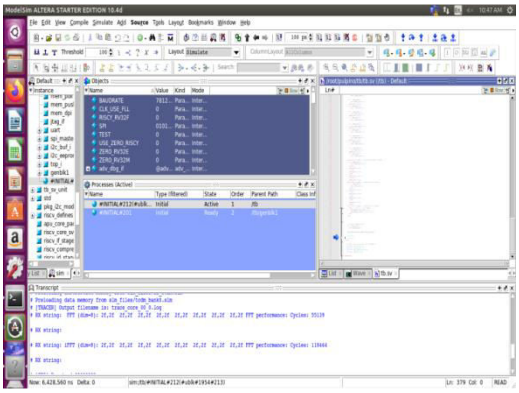
*uart (~/pulpino/sw/build/apps/FFT/stdout) - gedit

```
FFT (dim=8): 2f,2f  2f,2f  2f,2f  2f,2f  2f,2f  2f,2f  2f,2f  2f,2f FFT performance: Cycles: 55139

iFFT (dim=8): 2f,2f  2f,2f  2f,2f  2f,2f  2f,2f  2f,2f  2f,2f  2f,2f FFT performance: Cycles: 118464
```

Figure 7:UART output

5.2 Transcript window



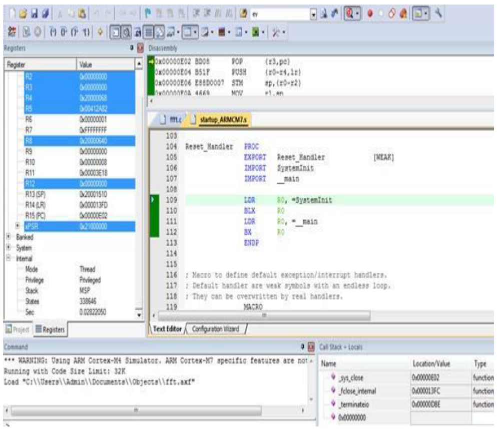
```
# Preloading data memory from slm_files/tcdm_bank0.slm
# [TRACER] Output filename is: trace_core_00_0.log
# RX string:  FFT (dim=8): 2f,2f  2f,2f  2f,2f  2f,2f  2f,2f  2f,2f  2f,2f  2f,2f FFT performance: Cycles: 55139

# RX string:

# RX string: iFFT (dim=8): 2f,2f  2f,2f  2f,2f  2f,2f  2f,2f  2f,2f  2f,2f  2f,2f FFT performance: Cycles: 118464

# RX string:
```

Figure 8: Transcript window



```
*** WARNING: Using ARM Cortex-M4 Simulator. ARM Cortex-M7 specific features are not
Running with Code Size Limit: 32K
Load "C:\\Users\\Admin\\Documents\\Objects\\fft.axf"
```

Figure 9: cortex m7 output

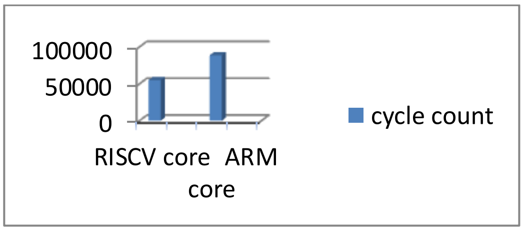


Figure 10: cycle count

Comparison between RISCV and ARM Cortex-M7

Table 1:performance analysis

Sl .n o	Performance analysis		
	CORE	*RISCV*	*ARM*
1.	**Cycle count**	**55139**	**88646**

6. conclusion

The designing of an efficient FFT radix 2 algorithm using RISCV core for processing ECG in cardiac pacemakers and executed the code on open core soc platform and the same has be executed on ARM cortex M7 and compared its performance cycle count and concluded that RISCV is more efficient than ARM.

Acknowledgment

The authors are gratefully acknowledge the facilities and support provided by the director of the school of Electronics And Communication Engineering of REVA UNIVERSITY, We also extend thanks to all teaching and non-teaching staff who had helped directly or indirectly to make this project successfully.

References

[1] Harpreet Kaur , Tarandip Singh 'Design and Simulation of 32-Point FFT Using Mixed Radix Algorithm for FPGA Implementation International Journal of Engineering Trends and Applications (IJETA) – Volume 2 Issue 3, May-June 2015.

[2] Remya Ramachandran,Vanmathi.k"simulation of radix-2 fast fourier transform using Xilinx" Remya Ramachandran et al. / International Journal of Computer Science Engineering (IJCSE) ISSN

[3] Waqar Hussain, Tapani Ahonen, Roberto Airoldi, Jari Nurmi "Energy and Power Estimation of Coarse-Grain Reconfigurable Array based Fast Fourier Transform Accelerators" 2012 - ieeexplore.ieee.org I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] P. Raghavan, et al., Embedded Linux System Design and Development. Boca Raton: Auerbach Publications, 2006.

[5] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck, DiscreteTime Signal Processing, 2nd edition (Upper Saddle River, NJ: Prentice Hall, 1989)

[6] Jump up^ C. J. Weinstein (1969-11-21). Quantization Effects in Digital Filters (Report). MIT Lincoln Laboratory. p. 42. Retrieved 2015-02-10. This computation, referred to as a 'butterfly'

[7] Jump up^ Cipra, Barry A. (2012-06-04). "FFT and Butterfly Diagram". mathoverflow.net. Retrieved 2015-02-10.

[8] W. M. Gentleman and G. Sande, "Fast Fourier transforms—for fun and profit," Proc. AFIPS 29, 563–578 (1966). doi:10.1145/1464291.1464352 R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[9] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[10] A. P. Yoganathan, R. Gupta, and W. H. Corcoran, "Fast Fourier transform in the analysis of biomedical data," Med. Biol. Eng., vol. 14, no. 2, pp. 239–245, Mar. 1976.

[11] M. Akay, Time Frequency and Wavelets in Biomedical Signal Processing (IEEE Press Series in Biomedical Engineering). 1998.

[12] P. de Carvalho, J. Henriques, R. Couceiro, M. Harris, M. Antunes, and J. Habetha, "Model-based atrial fibrillation detection," in ECG Signal Processing, Classification and Interpretation: A Comprehensive Framework of Computational Intelligence, A. Gacek and W. Pedrycz, Eds., London, U.K.: Springer, 2012, pp. 99–133.

[13] F. M. Vaneghi, M. Oladazimi, F. Shiman, A. Kordi, M. J. Safari, and F. Ibrahim, "A comparative approach to ECG feature extraction methods," in Proc. 3rd Int. Conf. Intell. Syst. Modelling Simul., 2012, pp. 252–256.

[14] N. A. Abdul-Kadir, N. M. Safri, and M. A. Othman, "Effect of ECG episodes on parameters extraction for paroxysmal atrial fibrillation classification," in Proc. IEEE Conf. Biomed. Eng. Sci., 2014, pp. 874–877.

[15] S. A. P. Haddad and W. A. Serdijn, "Ultra low-power biomedical system designs," in Proc. Ultra Low-Power Biomed. Signal Process., Dordrecht, The Netherlands: Springer, 2009, pp. 131–172.

[16] G. B. Moody and R. G. Mark, "The impact of the MIT-BIH arrhythmia database," IEEE Eng. Med. Biol. Mag., vol. 20, no. 3, pp. 45–50, May/Jun. 2001.

[17] S. Mostafa and E. John, "Reducing power and cycle requirement for FFT of ECG signals through low level arithmetic optimizations for cardiac implantable devices," J. Low Power Electron., vol. 12, no. 1, pp. 21–29, 2016.