Accelerated FFT computation for GNU radio using GPU of raspberry Pi

Article · January 2015

DOI: 10.1007/978-81-322-2208-8_60

C. Aneesh

4 PUBLICATIONS **7** CITATIONS

SEE PROFILE

Gandhiraj Rajendran

Amrita Vishwa Vidyapeeth…

33 PUBLICATIONS **54** CITATIONS

SEE PROFILE

Accelerated FFT Computation for GNU Radio Using GPU of Raspberry Pi

S. Sabarinath, R. Shyam, C. Aneesh, R. Gandhiraj and K.P. Soman

Abstract This paper presents the effective exploitation of Graphical Processing Unit (GPU) in Raspberry Pi for fast Fourier transform (FFT) computation. Very fast computation of FFT is found useful in computer vision based navigation system, Global Positioning System (GPS), HAM radio and on Raspberry Pi. A comparison is performed over the speed of FFT computation on BCM2835 GPU with that of 700 MHz ARM processor available in Raspberry Pi and also with intel-COREi5 processors. The FFT is computed for any one dimensional input signal and its analysis is done on different processors with varying signal lengths. The GNU radio is installed on Raspberry Pi, and the FFT computation done on GNU radio is accelerated using GPU of Raspberry Pi. Even though the Raspberry Pi GPU is primarily built for video enhancement, the parallel computational ability of GPU is utilized in this paper for accelerated FFT computation.

Keywords GPU · FFT · GNU radio · Raspberry Pi

S. Sabarinath (✉) · R. Shyam · C. Aneesh · K.P. Soman
Centre for Excellence in Computational Engineering and Networking,
Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore
641112, Tamil Nadu, India
e-mail: sabarin8@gmail.com

R. Shyam
e-mail: shyam.neezhoor@gmail.com

C. Aneesh
e-mail: a4aneeshc@gmail.com

R. Gandhiraj
Department of Electronics and Communication Engineering, Amrita School
of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore
641112, Tamil Nadu, India
e-mail: r_gandhiraj@cb.amrita.edu

L.C. Jain et al. (eds.), *Computational Intelligence in Data Mining - Volume 2*,
Smart Innovation, Systems and Technologies 32, DOI 10.1007/978-81-322-2208-8_60

1 Introduction

Fast Fourier Transform enables fast computation of Discrete Fourier Transform [1]. The pronunciation of data can be very well executed using Fourier transform. But still FFT computation is time consuming. So, faster method of computation is introduced using GPU of Raspberry Pi [2] using GNU radio software.

1.1 Raspberry Pi

Raspberry Pi is a cost effective hardware developed in UK, with the aim of teaching computer science in schools. Mainly two versions of Raspberry Pi are available, model A and model B. Model A has 256 MB RAM with single USB port and a power rating of 300 mA (1.5 W). Model B has 512 MB RAM with two USB ports and a power rating of 700 mA (3.5 W). System on chip RAM (Broadcom BCM2835) of Raspberry Pi contains CPU, GPU, DSP and SDRAM. The 256 MB RAM in model A is equally split in the ratio 50:50 in the default case. If graphical intensive work load is less, then the split ratio can be changed by the user to give the CPU some more RAM. The CPU and GPU performance are independent to each other. Communication between them is done through mail boxes. The value of flag is set when communication stops. The parameter-data is left in mail boxes. GPU even have the ability to generate its own interrupt events that stops the CPU's current execution. Code for the split is of the form,

$$sudo\ cp/boot/arm\ n\_start.elf/boot/start.elf$$

where n specifies the memory allocation for CPU and the remaining 256-n for GPU.

1.2 GNU Radio

GNU Radio is software that is purely open-source [3]. It is extremely useful in signal processing applications because it contain most of the signal processing tools as blocks, which can be used for simulation purpose. The FFT is one such block using which any signal can be given as the input of the block, that will return Fourier transform of the given input. The GNU radio is installed on Raspberry Pi. The software has an FFT block that takes any signal as its input and gives FFT of that signal as its output and also a plot for both input and output signals [4]. The FFT computation is slower process as this is done using the CPU of Raspberry Pi initially. The paper presents a method of using the GPU of Raspberry Pi for FFT computation instead of CPU and a comparison between the times of computation.

1.3 Continuous, Discrete, Fast Fourier Transforms

Any signal can be represented in time domain, the values of the physical process as some quantity *h*, e.g., *h*(*t*), or in Fourier domain, where *H* is the amplitude of the process which is specified, that is *H*(*f*), where *f* is the frequency represented in cycles per second. A transformation from *h*(*t*) to *H*(*f*) or vice versa is given by the Fourier transform equation as,

$$H(f) = \int_{-\infty}^{\infty} h(t)\mathrm{e}^{-j\omega t}dt \tag{1}$$

$$h(t) = \int_{-\infty}^{\infty} H(f)\mathrm{e}^{j\omega t}df \tag{2}$$

ω and f are related as, $\omega = 2\pi f$.

The spectrum will be continuous in nature. Fourier transform uses complex bases and it's conjugate to represent the signal. Complex values of information has both phase and magnitude values. This kind of representation is found helpful for the analysis of multi-tone, non-periodic signals which are time invariant in nature.

Fourier transform of a signal from finite number of sample points can be estimated, called discrete Fourier transform [5]. The function has non zero values at these discrete levels of time. Levels are chosen based on Nyquist criteria. The number of independent outputs obtained will evidently be the same as that of number of input given that is *N*. Fourier transform need to be only estimated at these discrete values. The discrete Fourier transform H_n,

$$H_n = \sum_{k=0}^{N-1} h_k \mathrm{e}^{2\pi ikn/N} \tag{3}$$

The discrete Fourier transform [6] also exhibits symmetry property exactly like continuous Fourier transform. Recovering back h_k s from H_n s called discrete inverse Fourier transform. Transformation relation is given by,

$$h_n = \frac{1}{N}\sum_{n=0}^{N-1} H_n \mathrm{e}^{-2\pi ikn/N} \tag{4}$$

Computational speed of fast Fourier transform is much more than that of discrete Fourier transform. It defines a complex number *W*,

$$W = \mathrm{e}^{2\pi i/N}$$

Each value of h_k is multiplied by matrix which has $W^{(n\times k)}$ as its components, *n* and *k* are the number of rows and columns of the matrix containing complex values.

Discrete Fourier transform is $O(N^2)$ process, that is, the matrix multiplication need N^2 complex multiplications, and additional small operation to calculate the required power values of W. Using fast Fourier transform the discrete Fourier transform can be computed in $O(N\log_2 N)$ operations. Fourier transform having length N can be represented as sum of two $N = 2$ length discrete Fourier transform. One is even numbered point and other is odd numbered point of the original N. That is,

$$F_k = \sum_{j=0}^{N-1} \mathrm{e}^{2\pi ik/N} f_j \tag{5}$$

$$F_k = F_k^e + W^K F_k^o \tag{6}$$

where, W is complex constant, kth component of the Fourier transform is denoted by F_k^e and F_k^o having length $N = 2$ generated from even and odd components. Both the components are periodic in k having lengths $N = 2$. So repetitions of two cycles are necessary to retrieve F_k.

2 FFT in Raspberry Pi and Proposed Methodology

FFT calculation need to be executed in an accelerated way because most of the real time application of signal processing need FFT computation, which consumes much of the execution time. FFT calculation is done initially over the normal Raspberry Pi processor or the CPU of Raspberry Pi using GNU radio software. Then done over conventional MATLAB software and both results are compared with the speed of FFT computed from GNU radio using GPU of the same Raspberry Pi. Videocore IV the graphical processing unit (GPU) in BCM2835, which is the microchip available in Raspberry Pi, is effectively made use for writing general purpose code for accelerated fast Fourier transform library. Main application of GPU in Raspberry Pi was for video streaming. The FFT block used in GNU radio is forced to work using FFT library that access GPU rather than Fastest Fourier Transform in the West (FFTW) library, which is commonly used in most of the operating systems. The time of computation is accurately calculated and studied.

The architecture of GPU (BCM2835 Videocore IV) has 2 processors that are distinct in nature. One is Programmable Graphic Core (PGC) and other is its co-processor. The programmable portion has many slices that are organized. Each slice has four single input-multiple data (SIMD) units, each is capable of executing different instructions per cycle, thus making the processing parallel.

2.1 FFT in GPU of Raspberry Pi

The use of processing power of GPU for calculating FFT can reduce the computational limitation of normal CPU. Andrew Holme has designed such library which uses the GPU for calculating the FFT in Raspberry Pi [1]. This library is purely open source and can be installed on to Raspberry Pi platform by running its command in command prompt of Raspberry Pi. We implemented code for FFT computation of an input signal of given length on this library. The code has three main functions *GPU_FFT_prepare* that creates structure for storing the input. The memory is effectively utilized by using a *maloc* function, *GPU_FFT_execute* is the core function that is used to execute the FFT and also it computes the time of computation, and finally *GPU_FFT_release* that will release all the previously stored memory locations for further reuse. The program is written in such a way that it take input of size as power of 2, and also takes the number of batches containing the number of arrays of input. Each row of the batch corresponds to each input and result is calculated row wise. This will return the FFT of each row along with the time it took to calculate the same. Any signal that has been generated randomly is given as its input to compute its FFT. This is compared with the previously generated time of execution obtained from Raspberry Pi CPU and MATLAB software.

2.2 FFT in CPU of Raspberry Pi

The CPU in Raspberry Pi belongs to 700 MHz ARM1176JZF-S core. It is more similar in configuration to that of old Pentium II processor, which is comparatively slower as compared with the newer versions available and hence it cannot provide much of execution speed. The execution power in the CPU cannot be up to the expectation while computing a high mathematical computation incorporated FFTs. FFT is ideally used to make DFT having computation of $O(N^2)$ faster. FFT has number of computation of $O(N\log_2 N)$. This is incorporated into CPU of Raspberry Pi using executables like *FFT_processor*, *GNU plot_driver*, *Signal_Source* and *Sound_Source*. The time taken for execution of a real time signal is computed for various lengths of the signal and is tabulated for comparison. The real time input signal is shown in Fig. 1. The obtained FFT plot is depicted in Fig. 2.

2.3 FFT in Intel-COREi5

FFT computed on intel-COREi5 is done using MATLAB software version R2013a (64 bit). The processor is having a clock speed of 3.3 GHz and maximum turbo frequency of 3.7 GHz. MATLAB has got capability of reading any one dimensional signal which can be passed as the input for computing FFT. We generated a signal

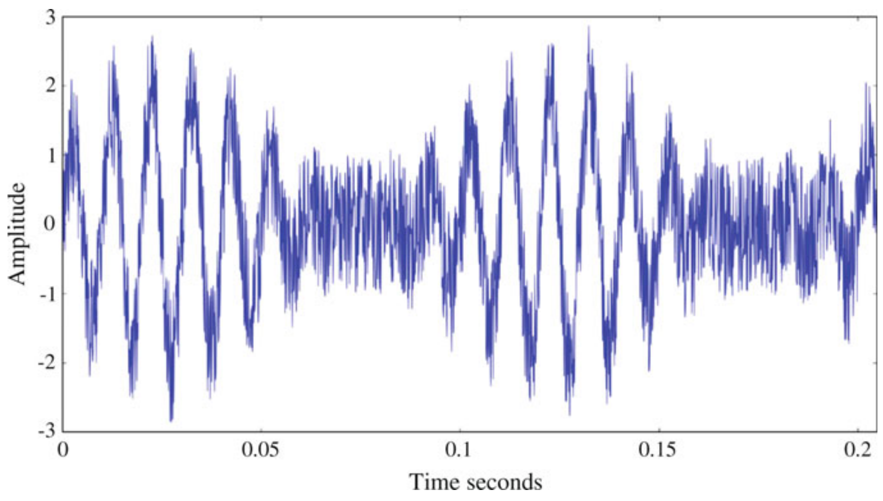

Fig. 1 Noisy input signal in time domain

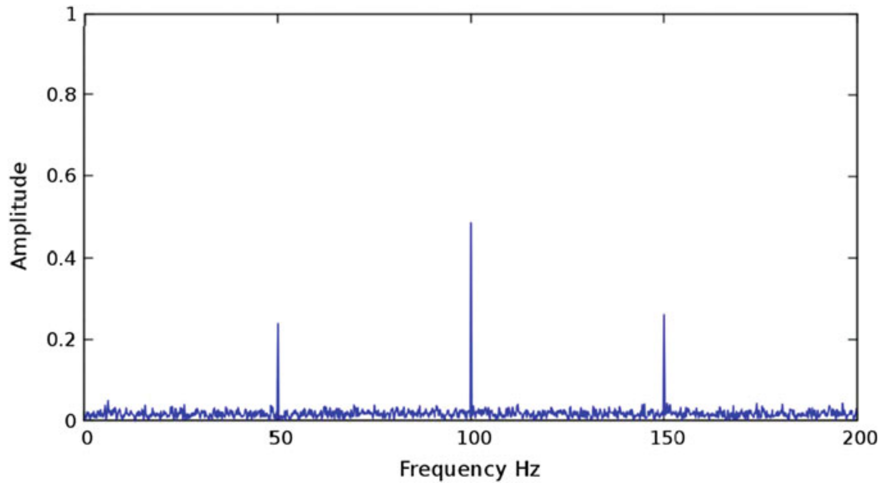

Fig. 2 Frequency spectrum

using this software and passed as the input of FFT function. MATLAB software already has inbuilt FFT command which is used to calculate the FFT. Time of computation for each input is calculated within MATLAB itself. Batches of inputs are given each batch having a particular number of inputs. So a batch 2 will contain two inputs. The computation time of both these inputs are calculated together in a batch.

3 Results and Discussion

A comparison of time taken for FFT computation by CPU, GPU and MATLAB software is done here. Signal having various lengths are considered. Here we consider length 256, 512, 1,024 and 2,048. Each signal contains four batches, each batch having array of input signal whose FFT need to be computed. Results obtained are tabulated for easy comparison. Table 1 shows the execution time for FFT in CPU of Raspberry Pi, Table 3 shows the execution time for FFT in intel-COREi5. The computation of FFT using GNU radio is found to be time consuming, as it is done using GPU in Raspberry Pi. This paper presents an effective solution to the above mentioned problem. From detailed analysis of sequences having various lengths (for different batches), it is found that CPU of Raspberry Pi consumes much more time as compared with MATLAB software which runs in conventional CPU as shown in Tables 1 and 3. But at the same time GPU of Raspberry Pi performs faster FFT computation than both of the above methods. Execution time of GPU is found to come in the order of micro seconds, which is shown in Table 2.

Table 1 Execution time of FFT in CPU of Raspberry Pi given in seconds

Batches	Length of FFT			
	256	512	1,024	2,048
1	0.2235	0.3134	0.534	1.0335
2	0.4229	0.7378	1.817	2.1096
3	0.6602	1.08	1.20	3.1788
4	0.9298	2	2.448	4.185

Table 2 Execution time of FFT in GPU of Raspberry Pi given in seconds

Batches	Length of FFT			
	256	512	1,024	2,048
1	0.23×10^{-6}	36×10^{-6}	57×10^{-6}	117×10^{-6}
2	46×10^{-6}	76×10^{-6}	112×10^{-6}	228×10^{-6}
3	72×10^{-6}	113×10^{-6}	174×10^{-6}	346×10^{-6}
4	114×10^{-6}	154×10^{-6}	238×10^{-6}	455×10^{-6}

Table 3 Execution time of FFT in intel-COREi5 given in seconds

Batches	Length of FFT			
	256	512	1,024	2,048
1	0.014089	0.052547	0.038113	0.086689
2	0.039796	0.043302	0.095895	0.188929
3	0.036306	0.075456	0.167499	0.278827
4	0.098299	0.081684	0.149205	0.320916

4 Conclusion and Future Work

In this paper, a comparison is done on the speed of FFT computation in GPU with that of CPU of Raspberry Pi and computation performed in intel-COREi5 processor (using MATLAB software). It was found that GPU allows us to compute the FFT much faster than both CPU of Raspberry Pi and intel-COREi5 processor. In future this GPU can be synchronized with normal processor in such a way that whenever faster computational power is needed, processor automatically shifts to GPU. The motivation behind doing this work was the slower computation speed of FFT in GNU Radio Companion. The slow computational speed is compensated by the use of GPU in Raspberry Pi. This computation can even be performed on environments where there is absence of computer, since Raspberry Pi itself act as a mini computer. As a future work this FFT computation could be utilized in computer vision based navigation system using Raspberry Pi for path tracking applications.

References

1. Duhamel, P., Vetterli, M.: Fast Fourier transforms: a tutorial review and a state of the art. Sig. Process. **19**, 259–299 (1990)
2. Accelerating Fourier transforms using the GPU | Raspberry. http://www.raspberrypi.org/accelerating-fourier-transforms-using-the-gpu/
3. WhatIsGR—GNU Radio—gnuradio.org. http://gnuradio.org/redmine/projects/gnuradio/wiki/WhatIsGR/
4. Gandhiraj, R., Soman, K.P.: Modern analog and digital communication systems development using GNU radio with USRP. Telecommun. Syst. 1-15 (2013)
5. Brigham, E.O., Yuen, C.: The fast Fourier transform. Syst. Man Cybern. IEEE Trans. **8**, 146–146 (1978)
6. Sorensen, H.V., Jones, D.L., Heideman, M., Burrus, C.S.: Real-valued fast Fourier transform algorithms. Acoust. Speech Sig. Process. IEEE Trans. **35**, 849–863 (1987)
7. Gandhiraj, R., Ram, R., Soman, K.P.: Analog and digital modulation toolkit for software defined radio. Proc. Eng **30**, 1155–1162 (2012)