# Real-Time Implementation of Fast Fourier Transform (FFT) and Finding the Power Spectrum Using LabVIEW and CompactRIO

Ansuman DiptiSankar Das and K. K. Mahapatra
Dept. of Electronics and Communication Engineering
National Institute of Technology, Rourkela, India – 769008

*Abstract* **-The growing popularity of adopting virtual instrumentation (modular, customizable, software-defined instrumentation) has only became possible due to the use of LabVIEW with a highly interactive process known as graphical system design. The CompactRIO programmable automation controller is an advanced embedded control and data acquisition system designed for applications that require high performance and reliability. This paper explains the real-time implementation of 256-point FFT, one of the most important transforms used in signal and image processing and finding the power spectrum using LabVIEW and CompactRIO. The proposed implementation uses only 3077 slices(21%), 2489 slice registers(8.7%), 4651 slice LUTs(16.2%) on a 400 MHz real time embedded processor.**

*Keywords* **- LabVIEW, CompactRIO, FFT, ADC Module, FPGA, Real-Time Processor, DMA- FIFO.**

## I. INTRODUCTION

Virtual instruments are the combination of user-defined software and modular hardware that implements custom systems with components for data acquisition, processing or analysis and presentation. Virtual instrumentation software is focused on the needs of the application and user defined. Applied mathematics is combined with real-time measurements, which reduce the time for innovation and, most importantly the time to market and/or time to commercialization of final products and services that result from research and development (R&D) using virtual instrumentation approach. The key element behind the success of the virtual instrumentation approach is LabVIEW which is a software development tool initially developed to support the requirements of virtual instrumentation. These activities can be implemented with a highly interactive process known as graphical system design, an approach that influences virtual instrumentation. This type of work can be complemented and enhanced by taking the graphical system design approach and using tightly integrated software and hardware tools that help to combine data analysis, visualization with measurements (data acquisition) and mining.

Designed for applications that require high performance and reliability, the National Instruments CompactRIO programmable automation controller is an advanced embedded control and data acquisition system. With the system's open, embedded architecture with small size, extreme ruggedness, and high flexibility, one can use COTS hardware to quickly build custom embedded systems. NI CompactRIO is powered by LabVIEW FPGA and LabVIEW Real-Time software tools, giving engineers the ability to design, program, test, and customize the CompactRIO embedded system with easy-to-use graphical programming tools [1].

The Fast Fourier Transform (FFT) has become almost ubiquitous and most important in high speed signal and image processing. Using this transform, signals can be moved to the frequency domain where filtering and correlation can be performed with fewer operations. When considering the alternate FPGA implementations, the FFT algorithm [2] - [4] should be chosen to consider hardware complexity, the execution speed, flexibility and precision. This paper presents the implementation of FFT of a real-time signal and finding out its power spectrum using the LabVIEW FPGA programming and the real time embedded processor of CompactRIO.

The paper is structured as follows. In section II FFT and power spectrum has been described. The RIO architecture is illustrated in Section III. In Section IV, the implementation of FFT by the proposed method will be debated. The synthesis results are shown in Section V. At last, the topic concludes with section VI.

## II. FFT AND POWER SPECTRUM

Fast Fourier Transforms (FFT) is an algorithm for speedy calculation of Discrete Fourier Transform (DFT) of a input data vector. The FFT is nothing but a DFT algorithm which reduces the number of computations needed for $N$ points from $O(N^2)$ to $O(N \log N)$ where log is the base-2 logarithm using periodicity and property. Several algorithms are there which can calculate FFT efficiently. Radix-2 algorithms are very useful when $N$ is a power of 2. There are two different Radix 2 algorithms named as Decimation in Time (DIT) and Decimation in Frequency (DIF) algorithms. Both of these are based on the successive decomposition of an $N$ point transform into 2 ($N/2$) point transforms. Split-radix, radix-$2^n$ are the other ways to calculate FFT. Many algorithms have been proposed as described in [1], [2], and [3] for efficient implementation of FFT processors. The power spectrum [1] of a signal is the power of that signal at each frequency that it contains. If we take an $N$-point sample of the function $c(t)$ at equal intervals and use the FFT to compute its Discrete Fourier Transform (DFT)

$$C_k = \sum_{j=0}^{N-1} C_j \, e^{2\pi ijk/N} \quad k=0, 1, \dots, N-1 \tag{1}$$

The power spectral components can be calculated from parseval's theorem for power calculation as follows:

$$\begin{aligned} P(0) &= P(f_0) = \frac{1}{N^2}\,|C_0|^2 \\ P(f_k) &= \frac{1}{N^2}[|C_k|^2 + |C_{N-k}|^2] \\ P(f_c) &= P(f_{N/2}) = \frac{1}{N^2}\,|C_{N/2}|^2 \end{aligned} \tag{2}$$

Where $f_c$ is the nyquist frequency and $f_k$ is defined only for the zero and positive frequencies and is given by

$$f_k \equiv \frac{k}{N\Delta} = 2f_c\frac{k}{N} \quad k=0, 1, \dots, \frac{N}{2} \tag{3}$$

## III. THE RIO ARCHITECTURE

CompactRIO combines a real-time embedded processor, a high-performance FPGA, and hot-swappable I/O modules mounted on the chassis. Each I/O module present on the chassis is connected directly to the FPGA. The above provides low-level customization of timing and I/O signal processing. The FPGA is connected to the real-time embedded processor via a high-speed duplex PCI bus. This provides open access to low-level hardware resources. This also represents a low-cost architecture. LabVIEW contains built-in data transfer mechanisms to pass data to the FPGA from the I/O modules and also to the embedded processor from the FPGA for real-time analysis, post data processing, data logging, or communication to a linked host computer. CompactRIO combines with LabVIEW, Real-time module, C-series I/O modules, chassis, CompactRIO device driver and FPGA module to function as a complete development suite for implementing any customized design.

The CompactRIO embedded system [5] - [7] features an industrial 400 MHz Freescale MPC5200 processor. It deterministically executes LabVIEW Real-Time applications on the reliable Wind River VxWorks real-time operating system. Built-in functions of LabVIEW transfer data between the FPGA and the real-time embedded processor within the CompactRIO embedded system. Existing C/C++ code can also be integrated with LabVIEW Real-Time code to save on development time.

A variety of swappable I/O types are available including voltage, RTD, current, thermocouple, accelerometer, and strain gauge inputs. There are analog I/O modules, digital I/O modules, counter/timers, high voltage/current relays, and pulse generation units with a wide range current and voltage rating. Sensors and actuators can be wired directly with C series modules as the modules contain built-in signal conditioning for extended voltage ranges for industrial signal types.

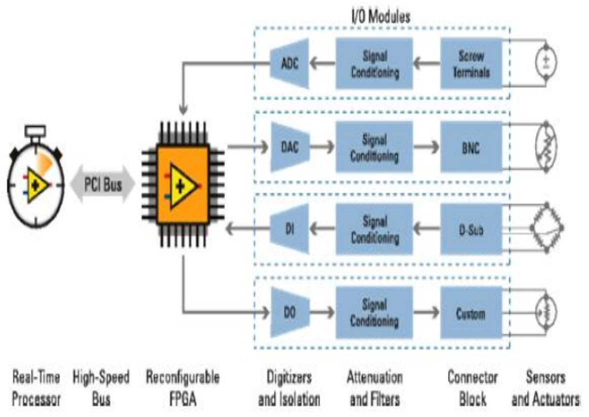


Fig. 1. The Rio architecture

The embedded FPGA is a high-performance, ultra rugged reconfigurable chip that can be programmed with LabVIEW FPGA tools. Historically, FPGA designers were forced to learn and use complex design languages such as Verilog or VHDL to program the FPGAs. Now, the problem can be solved by using LabVIEW tools to program for self-customized FPGAs. One can implement custom timing, synchronization, triggering, control, and signal processing for your analog and digital I/O using the FPGA hardware embedded in CompactRIO.

## IV. IMPLEMENTATION OF FFT AND FINDING POWER SPECTRUM USING LabVIEW AND CompactRIO

The proposed implementation is divided into two parts: writing the code in FPGA mode and controlling it through HOST [7]. Here NI-9104 has been taken which is an intelligent real-time embedded controller for CompactRIO. It is a 400 MHz processor with 128 MB of DRAM and 2 GB of memory in which data transfer can take place at a rate of 10 Mbps. An 8-slot chassis has been taken on which NI-9201 and NI-9263 are mounted. The NI-9201 is an 8-channel, 12-bit analog input module which uses successive approximation register (SAR) of analog to digital converter (ADC). The NI-9263 is a 4-channel, ±10V, 16-bit analog output module that works as a digital to analog converter (DAC).

Figure 2 shows the FPGA VI for finding the FFT of a real-time signal and storing its value in memory and using DMA-FIFO [5] and [7] for the stored data to be transferred to the HOST. Signal generation VIs are placed outside the case structure in order to avoid duplicating logic by having the same functions in multiple cases. The signal is given to NI-9201 (MOD-3 AI). The digital output of NI-9201(ADC) is given to the FFT express VI which calculates the 256-point FFT and gives real and imaginary values. Those values are stored in memory separately and delivered coherently to the host in 256-element frames by checking the DMA FIFOs status before writing to them, so that the FPGA DMA buffer never overflows. The advantage of DMA is that the host computer processor can perform calculations while the FPGA target transfers data to the host computer memory through bus mastering. A DMA-FIFO allocates memory on both the host computer and the FPGA target, but yet acts as a single FIFO. The FPGA VI writes to the DMA- FIFO one element at a particular instant of time with the Write method of the FIFO Method Node or reads from the FIFO one element at a particular instant of time with the Read method. While invoking, the host VI reads from or writes to the FIFO one or more elements at a time. A DMA Engine is used by LabVIEW to transfer DMA-FIFO data between the FPGA and the host computer.

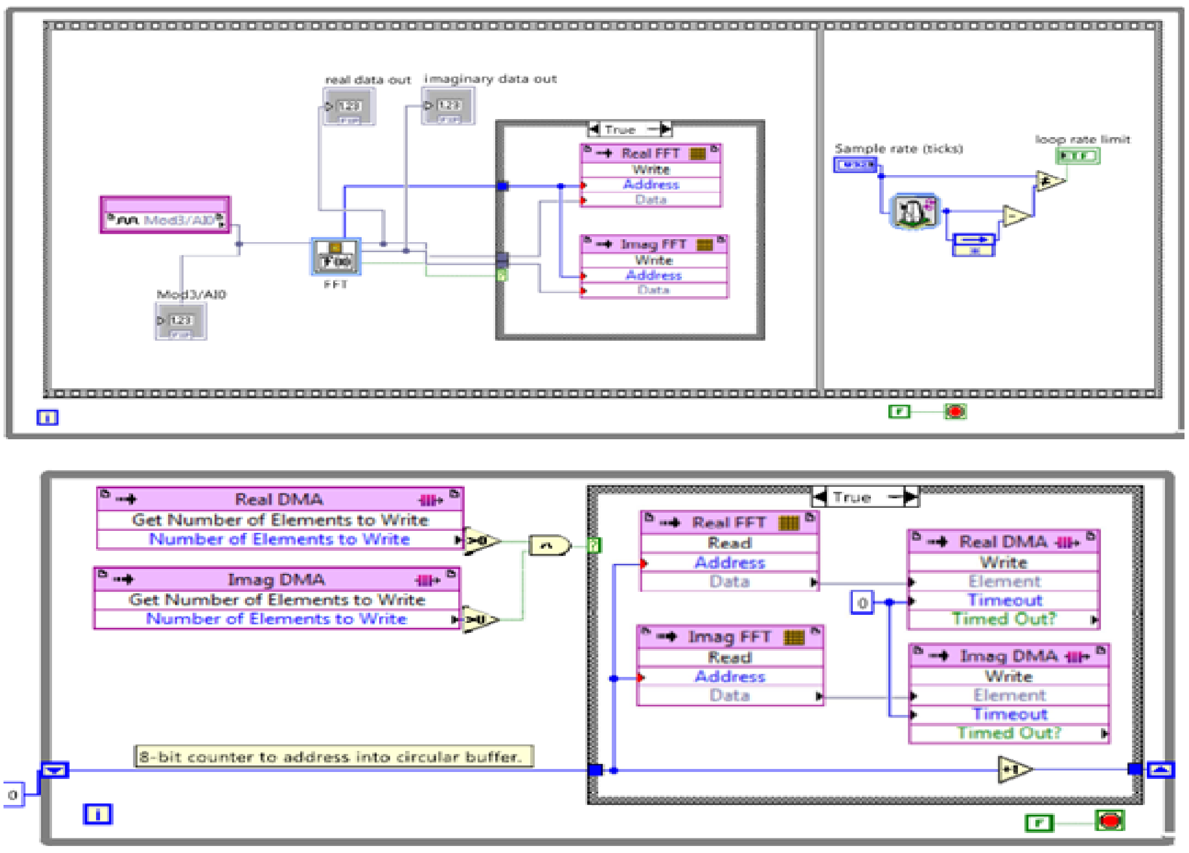


Fig. 2. FPGA VI for the proposed implementation in LabVIEW

Finally a HOST VI has been written which uses the FPGA code. The values stored in memory are now transferred to the HOST through DMA-FIFO. The open FPGA reference opens a reference to the FPGA VI without running it, in order to avoid generating data before DMA is configured. Reset the VI to guarantee that FIFOs are in a known state on the target. Here the use of small host buffers is to minimize latency for signal changes to show up in the display of HOST VI. The sample rate can be given manually or it can be determined by the loop by using different scaling VIs present in LabVIEW. The frequencies can also be seen or can be given manually. Since the FPGA VI prevents buffer overflows, the output should always be exactly one frame, starting with the DC bin. There are two memories created to store real and imaginary values of the FFT and are named as real FFT and imaginary FFT respectively. Likewise two DMA-FIFOs named as real DMA and imaginary DMA are being used to transport the values stored in the real and imaginary memory to the host respectively. Prior to the main loop in the HOST VI, real and imaginary DMA has been configured so that the main loop will run after the FIFOs get 500 values from FPGA VI, out of which

256 will be used at a time. This ensures that there will be continuous execution of the main loop in HOST VI without any data insufficiency. The values are converted to DBL (double precision float) from FXP (fixed point). The time delay module ensures clear vision of change of outputs from one frame to another. Finally the Close FPGA VI reference closes the reference to the FPGA.

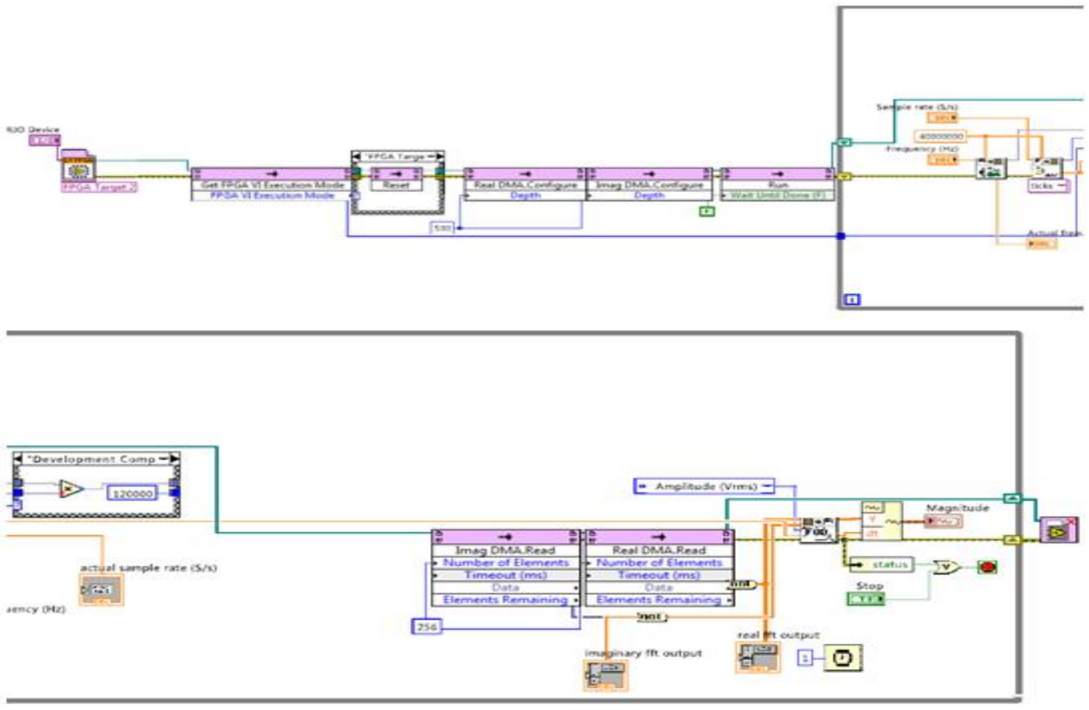


Fig. 3. HOST VI for the proposed implementation in LabVIEW

## V. Results and Consumed Resources

The FFT and power spectrum has been calculated for a 1 kHz, 10 kHz square wave and 1 kHz, 10 kHz sine wave with amplitude of 1V and the results are shown through figure 5 - 8 respectively. It is known in that the power spectrum of a square wave at a particular frequency ($f_m$) contains only odd harmonics. The amplitude of the odd harmonics is given by $2/(\pi \times n)$ where $\pi$=3.142 and n is the odd harmonic number. The first harmonic presents at that frequency with the power nearly equal to the input signal's amplitude value. The third harmonic presents at 3×$f_m$ with a power nearly equal to 0.21V, the fifth harmonic has a value around 0.12V present at a 5×$f_m$ for a 1V input signal and so on.

The results shown through figure 4, and 5 are quiet similar with the theoretical values as discussed above. The power spectrum of a sine wave only contains a harmonic at the input signal's frequency having power nearly equal to the amplitude of the input signal. This has been depicted in figure 6, and 7.

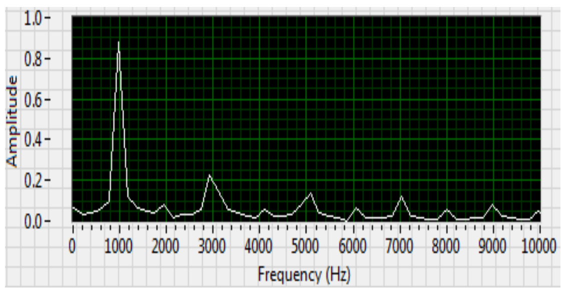


Fig. 4. Observed power spectrum for 1 kHz square wave

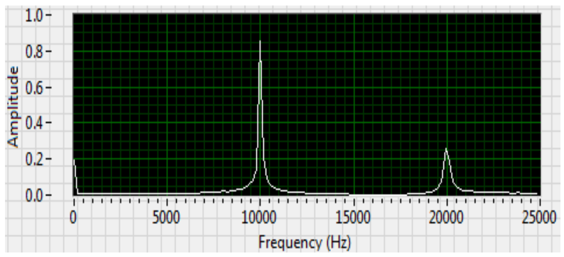


Fig. 5. Observed power spectrum for 10 kHz square wave

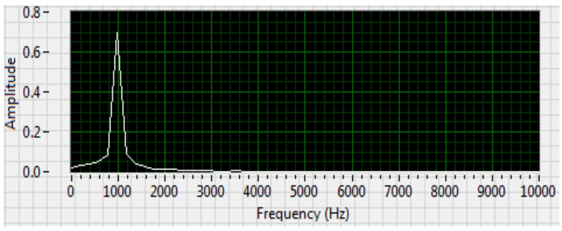


Fig. 6. Observed power spectrum for 1 kHz sine wave

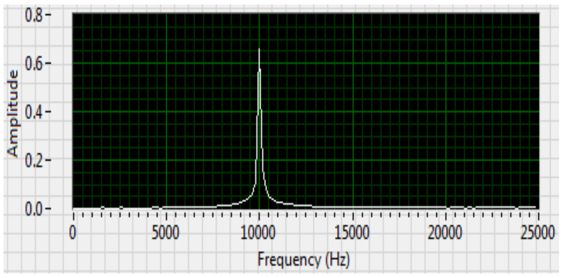
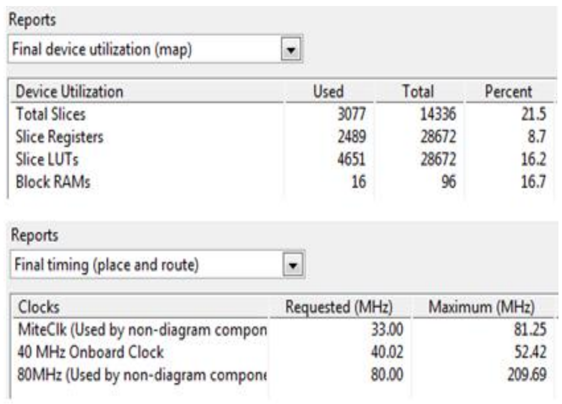


Fig. 7. Observed power spectrum for 10 kHz sine wave

The proposed implementation of 256-point FFT and its power spectrum using CompactRIO has been allocated only 3077 slices(21%), 2489 slice registers(8.7%), 4651 slice LUTs(16.2%) operated with a maximum frequency of 52.42 MHz for on-board clock and 209.69 MHz for non-diagram components as shown in figure 8.


Reports

Final device utilization (map)

| Device Utilization | Used | Total | Percent |
|---|---|---|---|
| Total Slices | 3077 | 14336 | 21.5 |
| Slice Registers | 2489 | 28672 | 8.7 |
| Slice LUTs | 4651 | 28672 | 16.2 |
| Block RAMs | 16 | 96 | 16.7 |

Reports

Final timing (place and route)

| Clocks | Requested (MHz) | Maximum (MHz) |
|---|---|---|
| MiteClk (Used by non-diagram compon | 33.00 | 81.25 |
| 40 MHz Onboard Clock | 40.02 | 52.42 |
| 80MHz (Used by non-diagram compone | 80.00 | 209.69 |

Fig. 8. Final synthesis report of the proposed implementation

## VI. CONCLUSIONS

The implementation of 256-point FFT and finding the power spectrum has been successfully verified by using LabVIEW and CompactRIO and the results has been listed with the final synthesis report. Algorithm implemented in hardware, we can get an extremely high loop rate. Once the code is downloaded to the chassis and the power is on, the system will respond within milliseconds. There is no need to know VHDL language to program FPGA; no need to be deeply familiar with real-time operating system. The above structure also uses pipelining method to calculate 256-point FFT continuously. The use of DMA-FIFO ensures reduction in latency in display on the host and no data is lost with increased throughput.

## REFERENCES

[1] Marek Horinek, Petr Bilik, Power Analyzer for Converter Testing Based on Crio Hardware Platform, Applied Electronics International Conference, pp. 1-4, 2010.

[2] I.S. Uzun, A. Amira and A. Bouridane, FPGA implementations of fast Fourier transforms for real-time signal and image processing, IEE Proc.-Vis. Image Signal Process., Vol. 152, No. 3, June 2005.

[3] Ren-Xi Gong, Jiong-Quan Wei and Dan Sun, Ling-Ling Xie, Peng-Fei Shu and Xiao-Bi Meng, FPGA Implementation of a CORDIC-based Radix-4 FFT Processor for Real-Time Harmonic Analyzer, International Conference on Natural Computation ,Vol. 4, pp. 1832-1835, July 2011.

[4] Ahmed Saeed, M. Elbably, G. Abdelfadeel, and M. I. Eladawy: Efficient FPGA implementation of FFT/IFFT Processor, INTERNATIONAL JOURNAL OF CIRCUITS, SYSTEMS AND SIGNAL PROCESSING, Vol. 3, Issue 3, pp. 103-110, 2009.

[5] Tao Lin, Yongxing Xie, Jing Tang, Design of CompactRIO-based Acquisition System, Conference on Environmental Science and Information Application Technology, Vol. 1, pp. 678-681, 2010.

[6] Carroll Dase, Jeannie Sullivan falcon, Brain Maccleery, “Motorcycle Control Prototyping Using an FPGA-Based Embedded Control System,” IEEE Control Systems Magazine, pp.17-21, Oct. 2006.

[7] Maciej Rosol, Adam Pilat, Andrzej Turnau, Real-time controller design based on NI Compact-RIO, Proceedings of the International Multiconference on Computer Science and Information Technology, pp. 825–830, Oct. 2010.