Implementation of 1024-point FFT Soft-Core to Characterize Power and Resource Parameters in Artix-7, Kintex-7, Virtex-7, and Zynq-7000 FPGAs

K. S. Shashidhara. and H. C. Srinivasaiah

Abstract—This Paper presents implementation of 1024-point Fast Fourier Transform (FFT). The MatLab simulink environment approach is used to implement the complex 1024-point FFT. The FFT is implemented on different FPGAs such as the following four: Artix-7, Kintex-7, Virtex-7, and Zynq-7000. The comparative study on power and resource consumption has been carried out as design parameters of prime concern. The results show that Artix-7 FFT consumes less power of 3.402W when compared with its contemporary devices, mentioned above. The resource consumption remains same across all the devices. The resource estimation on each FPGA is carried on and its results are presented for 1024-point FFT function implementation. This Comprehensive analysis provides a deep insight with respect to power and resources. The synthesis and implementation results such as RTL Schematic, I/O Planning, and Floor Planning are generated and analyzed for all the above devices.

Index Terms—1024-point FFT, Subsystems for MIMO-OFDM, Multicarrier Modulation, Reconfigurable Hardware Simulation, Applications of FPGA, Doppler Shift, Time Variant Communication Channels, FFT Soft-Core.

I. INTRODUCTION

In the last few decades an extensive research work has been carried out on FFT algorithm implementation particularly in the field of communication evolving into generation of technologies ahead. The FFT and its inverse algorithm are derived from DFT in early 1965 by Cooley and Tukey [1] gained much interest in the Field of signal processing which proves to be superior than DFT or Short time Fourier Transform (STFT) alone to handle non-stationary signal. On par with STFT where window size is fixed, the FFT techniques allows the variable sizes or points for analyzing the signal frequency of interest with highest accuracy without any loss of timing information [2]. In Long Term Evolution (LTE) wireless communication system service providers strive hard to accommodate required number of users at a given point of time within allocated and available BW [3]. To meet the demand of ever-growing users, BW management became a challenging task such as in modern wired/wireless communication systems which works in accordance with IEEE 802.11 a/b/g/n OFDM [3] techniques are adopted. The FFT and IFFT play a vital role in OFDM and MIMO OFDM systems [4,5]. The inherent characteristic features of FFT/IFFT i.e., mapping from time to frequency and vice versa is more suitable for OFDM implementation where FFT plays major role pertaining to the requisite spectral efficiency [6]. The soft FFT core is implemented in Xilinx Vivado environment for the above said FPGAs for comparison, investigation with respect to power and resource parameters. In the state-of-art, bit error rate (ber) improvement, improved gain performance in terms of BW, and system speed are the advantages of FFT-OFDM systems [4]. The FFT features such as orthogonality leads to the realization of Multicarrier Modulation (MCM) in broader perspective. Hence, dedicated hardware accelerators are the need of the today's Modern WC system. The FFT implementation basically follows the splitting up of complex algorithm into sub-blocks interpreted as stages. The FFT variants are being chosen specific to domain requirement and specifications, for instance a Split radix or Mixed radix FFT architecture are more suitable for modern wireless communication system [2]. The high data rate computation meeting the constraint of throughputs and quality of service (QoS) [6] is the requirement of the modern WC systems. The typical approach to derive architecture of a system, firstly, to represent conceptual mathematical model by incorporating all the variables considering the hypothetical scenario such as transform length. The mathematical model is then suitably converted to algorithm which enables the designer to investigate the functionality by making use of higher level abstraction module. The functionality so verified at the First level is so generic and not concerned to any domain. For an exemplary comparison, a sine signal of 1 KHz is considered as shown in Fig. 1.

This signal is sampled at Nyquist rate of 2 Kilosamples/sec with sample interval of Ts=0.5 ms. The sampled values are ported to input of FFT function in MatLab environment whose FFT and its power spectral density (PSD) in Watts/Hz are calculated and plotted as shown in Fig. 2. The FFT spectrum is indicated with plot line in blue color, whereas PSD in black color as shown in Fig. 2. The Purpose of calculating the PSD in watts/Hz is in view of SNR calculation requirement in communication theory. For characterizing the signal and channel characteristics, the requirement is to compute FFT of them on-the-fly, this necessitates a dedicated FFT processor to capture the signal samples for computing FFT. This paper, implements a 1024 point FFT processor [7] using Field Programmable Gate Array (FPGA) [8] for such a high speed applications as discussed in rest of this paper.
II. THE FAST FOURIER TRANSFORM ALGORITHM REALIZATION: AN OVERVIEW

The FFT algorithm is assumed to be a block box at higher level abstraction. Precisely, this block box comprises of many processing elements (PEs) or generally butterfly units [9] which process the data as soon as the data is made available at its input. This process takes place in combinatorial manner asynchronous with clock and the twiddle factors are supplied on-the-fly by way of storing, precomputed and stored values. Except in some scenario, such as in case of interoperability of FFT modules, the control signals are rarely used [2].

Fig. 3 shows the comprehensive algorithmic level design flow for a typical FFT realization [7]. The pre-processed data are supplied at the input of FFT processor. DIT-FFT needs the data to be in bit reversed order whereas the DIF-FFT needs the data in natural order. However, a suitable data format has to be maintained at the input ports. Depending on the type of architecture: parallel, burst I/O etc., the storage space will be configured at the input point. The various data format such as canonical signed digit(CSD), minimal signed digit(MSD), or Logarithmic data representation adopted in case of precise data path unit such as FIR filter reported in [10], however a straightforward data format representation is considered in this work in view of observing the artefact such as s creating numerical noise as a result of round-up and round-down approximations (i.e. finite word length effects) during hardware FFT implementation, computation, and analysis.

The complex real, and imaginary signals are ported into input of FFT processor. These signals traverse through the network of PE’s with computations at predefined stages. The cumulative results from the intermediate stages lead to final multiply accumulate (MAC) processor output with latency defined by the individual PEs. Each stage PEs performs identical computations but with different twiddle factor arguments. Hence, these homogeneous operations enable the designer to adopt the very sophisticated mechanism such as Pipelining, commonly seen in DSP algorithm implementation. The Pipelining mechanism leverages the higher throughput operations [9].

As depicted in the algorithm flow in Fig. 3, the input sequence indexed with ‘n’ a time domain samples and the output sequence indexed with ‘k’ a frequency domain samples. The time domain complex signals are transformed to frequency domain complex signals by multiplying with complex sine and cosine signals called as twiddle factors (W_N). The linear transformation takes place through stages with the help of leaf PE unit i.e., butterfly unit [9]. The product of ‘n’ and ‘k’ values when used in conjunction with W_N gives rise to distant values in vector form. These values are used appropriately at each stage to compute the intermediate transformed values, which leads to final transformed output in frequency domain. The BF unit alternatively referred as PEs at each stage denoted in line with stage number are shown in the flow as BF0, BF1……BF_N (N=0……9).

The number of butterfly required at each stage is equal to N/2, where N represents the N-point FFT. Each stage FFT requires same number of butterfly [2](where N=1024, the transform length). Though, the number of butterfly stages remains same but spanning at each stage will vary as

 DOI: http://dx.doi.org/10.24018/ejers.2019.4.9.1515
depicted in the flow above.

A flowchart depicting the implementation methodology to realize a complex 1024 point FFT algorithm is as shown in Fig. 4. In accordance with this flowchart methodology, a HDL code generation from simulink environment \([11]\) is thoroughly interpreted and modified as needed by this work. The modules at intermediate stages of this flowchart herein are called as PEs are investigated to verify the stipulated FFT functionality. High level abstraction modelling (design) approach enables the designer to build the system with designated time and verify the functionality on-the-fly. The pre-designed and optimized blocks which are readily available are plugged into simulink model window in the form of Xilinx blockset to build the model of interest to simulate and implement desired functionality such as FFT \([12]\) considered herein interactively.

All the sub modules are picked from DSP system tool boxes. The required complex signals are generated by combining multiple sinusoidal signals using complex adder block at the input source points. Similarly, the computed results are ported to MatLab using “To workspace blocks” from sink blocks from DSP System Tool boxes. Each of these blocks, such as source block, FFT processor block, and sink blocks are configured through system block properties. Configurations of these parameters ensure a proper data format hence faithful data transfer through the sub modules.

Further, the solver options are carefully configured specific to continuous or discrete data processing to avoid likely simulation errors.

III. IMPLEMENTATION OF 1024 POINT FFT ON FPGA.

The Simulink MATLAB environment tool suite provides a way of mapping higher level abstraction module to hardware realizable components. This intern facilitates the designer to build the model in less turnaround time and verify the functionality. These designed models are implemented on target FPGA hardware device by generating the suitable hardware compatible language (HDL) from Simulink environment. The flowchart of Fig. 4 shows one such methodology adopted to realize 1024-point FFT on FPGA device starting from high level module to implement on FPGA \([13]\). The FFT blocks are plugged into Simulink editor window and a complex valued (discrete time domain signals) signal source is connected to FFT processor input. Each block is then configured for given specifications appropriately, and simulation environment are configured through solver options (such as ode4 or none) in the main tab menu. These initial setting ensures the proper simulation process which meets the desired results. The blocks are interfaced appropriately considering the data format between each block to avoid any erroneous results. Simulate the design using “Run Simulation” option from the main menu bar and observe the results through the sink blocks, and verify the results in MATLAB environment, if it is exported through sink blocks. Generate the HDL code with the options available in the main menu bar. The various options such as generating the test bench can be chosen, if needed. The generated code is saved in the target directory used to create project in the Vivado environment. Thus creating the project and further analysis will be dealt in subsequent sections.

The intended design project can be created in many possible ways could be HDL or "C" language. In this work an HDL Verilog code is used to build the project. A set of source files have been added along with a top module, top module will have instantiated module placed in the hierarchically structured manner. As depicted in the Fig. 5 Vivado project window, gm_t_1024-point file constitutes the main source file and its expanded view of the same is shown in Fig. 6 below. The initial settings made are visible in the project summary window \([14]\). The FPGA design flow goes through several design phases such as simulation, synthesis, implementation (place and route), and finally programming the device. Optimizations at each level happens inherently adopted in the tool flow, however a provision is made where designer can invoke and optimize with trade-off, if needed. Xilinx Vivado tool has interactive graphical user interface to enable the designer to manage the
design through all phases of design flow. The Vivado has features of usage of Tcl commands. All the operations which are performed in GUI mode generate an equivalent Tcl command. These Tcl commands are reused to perform the same set of operations as in GUI mode. Fig. 5, above shows the Vivado project window, mainly consists of flow navigator, source window, properties window, and project summary window. The flow navigator window assists the designer and takes through the standard design flow adopted by the tool [15,5,16,17].

The main source file and instantiated files are shown in Fig. 6. This facilitates the user to scale up the design by adding the required number of file over the available Instantiated file. The modular approach is followed in most of the complex design, this window helps to invoke and modify the design in the process of scaling up of the design.

The first level transformation and representation of given design which is in algorithmic form is RTL schematic as shown in Fig. 7. The schematic shows the mapping of design from conceptual block box into a set of logic components with appropriate interconnection. These set of logic components called as library elements (leaf cells) are standardized and independent of target device technology. Hence, the functionality of the design can be verified using these schematic without any constraint attached to it at this stage.

The RTL schematic shown in Fig. 8 gives the finer details about the design under implementation in terms of logic primitive specific to particular technology.

The main notion behind the synthesis is firstly, to modify the design so as to suit to specific devices or migration from one technology to another technology, secondly, to fine tune some portions of the complex design so as to improve overall performance, and finally, to keep design constraint within the tolerable limit of course with trade-off between the power, area and performance. The concept of folding, re-timing pertaining to optimum DSP design can be incorporated at this stage [9].

The RTL schematic after simulation

Fig. 7. The RTL schematic after simulation

The detailed device I/O Layout view after implementation with viewer pane undocked

Fig. 9. The detailed device I/O Layout view after implementation with viewer pane undocked

The device package view of 1024-point FFT implementations. The I/O banks utilized for this implementations are I/O Bank- 0,14,15,16, 34,35 , and the remaining are Bankless Pins (bank-67). These banks are grouped based on the operating voltages. Bank-0, specifies the pins configured mainly for clocks, similarly the pins from bank 14-16 allocated to various nets/nodes of the design.

The detailed device I/O Layout view after implementation with viewer pane docked

Fig. 10. The detailed device I/O Layout view after implementation with viewer pane docked

From Fig. 10, it is observed that a differential pair of pins are highlighted by connecting wire with adjacent pins. Clock regions for each I/O bank are designated as X0Y1 for Bank14, X0Y2 uses Bank 0, and 15, X0Y3 uses Bank 16, X1Y1 uses Bank 34, and X1Y2 uses Bank 35. These coordinates are highlighted when moused over to desired area upon implementation in Vivado Tool environment.

The Floor plan for 1024 point FFT implementation shows the placement and routing details as in the figure.11. The input and output signals through Input I/O Buffer are shown in green color, and are mapped to slices (SLICE- L and SLICE-M) of each CLBs. The usage of various logic primitives apart from CLBs viz DSP slices, distributed RAM etc., are obtained from utilization report generated after implementation. The floor plan optimization are done
by the tool by default (unconstrained), however, tweaking of placement and routing can be done by providing the constraint files to the design.

![The Floor plan view after implementation](image1.png)

**Fig. 11:** The Floor plan view after implementation

![Artix-7 Project window](image2.png)

**Fig. 12.** Artix-7 (a) Project window (b) RTL Schematic (c) Device IO Planning, and (d) Floor Planning.

Fig. 12 shows the results obtained at each design flow steps with target FPGA board from Artix-7 device family. Project summary window as in Fig. 12(a) shows the configured information such as chosen target device, source/node properties, hierarchical nature of files instantiated into the design, object properties, flow navigator panes to guide the designer through standard design flow steps. The output window prompts the user about any error messages encountered after compilation and progress status during synthesis and implementation under design run tabs. The first level transformation from algorithm is RTL schematic as in Fig. 12(b), the default pins are reset, clk, and ceout, Out2. The RTLMUX is used to select the input sources, but as it depicted in Fig. 12(c), Mux selects the input based on the select line generated by address counterreg (cntreg). The explanation follows the same as in Fig. 9 and 10 for Fig. 12(c) and (d).

Fig. 13. shows the results obtained at each design flow steps with target FPGA board from Kintex-7 device family. Project summary window as in Fig. 13(a) shows the configured information such as chosen target device, source/node properties, hierarchical nature of files instantiated into the design, object properties, flow navigator panes to guide the designer through standard design flow steps. The output window prompts the user about any error messages encountered after compilation and progress status during synthesis and implementation under design run tabs. The first level transformation from algorithm is RTL schematic as in Fig. 13(b), the default pins are reset, clk, and ceout, Out2. The RTLMUX is used to select the input sources, but as it depicted in Fig. 13(c), Mux selects the input based on the select line generated by address counterreg (cntreg). Fig. 13(c) and 13(d) are same as described section above.

![Kintex-7 Project window](image3.png)

**Fig. 13.** Kintex-7 (a) Project window (b) RTL Schematic (c) Device IO Planning, and (d) Floor Planning

Fig. 14 shows the results obtained at each design flow steps with target FPGA board from Virtex-7 device family. Project summary window as in Fig. 14(a) shows the configured information such as chosen target device, source/node properties, and hierarchical nature of files instantiated into the design, object properties, flow navigator panes to guide the designer through standard design flow steps. The output window prompts the user about any error messages encountered after compilation and progress status during synthesis and implementation under design run tabs. The first level transformation from algorithm is RTL schematic as in Fig. 14(b), the default pins are reset, clk, and ceout, Out2. The RTLMUX is used to select the input sources, but as it depicted in Fig. 14(c), Mux selects the input based on the select line generated by address counterreg (cntreg). Fig. 14(c) and 14(d) are same as described above.

![Virtex-7 Project window](image4.png)

**Fig. 14.** Virtex-7 (a) Project window (b) RTL Schematic (c) Device IO Planning, and (d) Floor Planning
messages encountered after compilation and progress status during synthesis and implementation under design run tabs. The first level transformation from algorithm is RTL schematic as in Fig. 14(b), the default pins are reset, clk, and ceout, Out2. The RTL MUX used to select the input sources, but as it depicted in Fig. 14(c), Mux selects the input based on the select line generated by address cntreg. Fig. 14(c) and (d) is same as described in previous section.

Fig. 15. Zynq-7000 (a) Project window (b) RTL Schematic (c) Device IO Planning, and (d) Floor Planning.

Fig. 15 shows the results obtained at each design flow steps with target FPGA board from Zynq-7000 device family. Project summary window as in Fig. 15(a) shows the configured information such as chosen target device, source/node properties, hierarchical nature of files instantiated into the design, object properties, flow navigator panes to guide the designer through standard design flow steps. The output window prompts the user about any error messages encountered after compilation and progress status during synthesis and implementation under design runs. The first level transformation from algorithm is RTL schematic as in Fig. 15(b), the default pins are reset, clk, and ceout (chip enable out), Out2. The RTLMUX is used to select the input sources, but as it depicted in Fig. 15(c), Mux selects the input based on the select line generated by address cntreg. Fig. 15(c) and (d) is same as described in the previous section that is Fig. 14 (c) and (d).

IV. POWER ANALYSIS OF 1024-POINT FFT ALGORITHM

The power consumption in FPGA typically ranges from mW to Watts and it mainly depends on factors such as design description or design entry, operating frequency, switching operations involved in the design, and target FPGA board used. Power consumption in FPGA will be dealt under four components; Device Static power, Core (or device) Dynamic power, and I/O(and Transceiver) power. Device static power accounts to power consumption even without any design being active on the device, it is due to inherent leakage due to technology node of the FPGA device under consideration. Technology node has great impact on the leakage power due to sub-micron regime of today's CMOS technology shrinkage of devices. Static power is a function of leakage in the silicon Core. Dynamic power is a measure of power consumed when the device (FPGA) is functioning excluding the power consumed by I/O bank. I/O power consumption is more prominent when the data transfer is between the on-chip components. In summary, the tool has provision to specify the constraint parameter to get optimal and best results in the context of overall power consumption of the device. The various factors which influence the consumption of power in given design are categorized as physical parameters, and functional parameters. The Physical parameters are Target Board-design, the type of packaging, and mainly on target device for implementation of the intended design, similarly functional parameters mainly depend on the RTL coding style adopted in the design. This section presents the tool features of Xilinx Vivado to estimate the power and also the methodology adopted for power optimization pertaining to 1024 point FFT algorithm. The quantitative analysis carried out in the implementation of 1024 point FFT are also accounted in this section. Estimation of Power can be carried out at various phases of the design (granularity). Further, the accuracy of the estimation depends on the configuration parameters provided during the project configuration, as a thumb rule, more information in the configuration, constraint leads to accurate estimates. These accurate results are very much nearer to the power consumption in the target FPGA when the design is implemented. Xilinx Vivado tools basically provide three methods to estimate the power [18]: Xilinx Power Estimator (XPE), Vivado Report Power, and Vivado Power Optimization (VPO). The XPE method is more suitable for power estimation (or power budgeting analysis) and applicable only in the pre-design phase.

It makes use of spreadsheet processing mechanism and analysis is solely depends on the information provided by end-user during configuration time. This method also facilitates the end user to analyse the design with varying parameters. The Vivado Report Power (VRP) generated after synthesis is for post-design stage power analysis.

The VRP is considered as accurate tool because it gathers the information from synthesis, placement and /or routing netlist nodes. In order to get accurate results, switching activity information (sai) needs to be provided in VRP method of power estimation. Hence, this considered to be power estimation at the lowest level of abstraction.

DOI: http://dx.doi.org/10.24018/ejers.2019.4.9.1515
Fig. 16. The table of power and resource utilization report for the Artix-7 FPGA embodied in this figure.

Fig. 17. The table of power and resource utilization report for the Kintex-7 FPGA embodied in this figure.

Fig. 18. The table of power and resource utilization report for the Virtex-7 FPGA embodied in this figure.

Fig. 19. The table of power and resource utilization report for the Zynq-7000 FPGA embodied in this figure.

Fig. 16 comprises of two tables shows the power consumed and resource utilized with target FPGA device as Artix-7, which is 3.402W (On chip power) with a split up of 3.303(Dynamic) and 0.098(Static). Resource utilized shown to be 438, (0.69 % with available resources of 63400) for slice LUTs (Logic and Memory), whereas the total of 221 slice registers as Flip flops (accounted to 0.17 % against available 126800) is used. Fig. 17 comprises of two tables embodied shows the power consumed and resource utilized with target Kintex-7 FPGA device consists of 410 pins. The total on chip power is 3.521W with a split up of 3.302(Dynamic) and 0.219(Static). Resource utilized shown to be 436, (0.17 % with available resources of 254600) for slice LUTs (Logic and Memory), whereas the total of 221 slice registers as Flip flops (accounted to 0.04% against available 508400) is used.

Fig. 18 comprises of two tables shows the power consumed and resource utilized with target FPGA device as Virtex-7, which is 3.587 W (On chip power) with a split up of 3.312(Dynamic) and 0.275(Static). Resource utilized shown to be 436, (0.14 % with available resources of 303600) for slice LUTs (Logic and Memory), whereas the total of 221 slice registers as Flip flops (accounted to 0.04 % against available 607200) is used. Fig. 19 comprises of two tables shows the power consumed and resource utilized with target FPGA device as Zynq-7000, which is 3.486 W (On chip power) with a split up of 3.279(Dynamic) and 0.207(Static). Resource utilized shown to be 436, (0.81% with available resources of 53200) for slice LUTs (Logic and Memory), whereas the total of 221 slice registers as Flip flops (accounted to 0.2 % against available 106400) is used.

V. CONCLUSION

In this paper a detailed discussion on implementation of 1024-point FFT algorithm targeting FPGA devices namely, Artix-7, Kintex-7, Vertex-7 and Zynq-7000 is presented with sufficient focus. The results from various phases of the design give an insight into implementation aspects on these reconfigurable devices. RTL schematic across all the device looks similar, whereas the I/O-planning and floor planning varies across them. The power consumption across all these devices for the same FFT function implementation is found to be different. This analysis facilitates the designer a guideline for choosing a right FPGA specific to application. Artix-7 for example is found to be marginally power economical among themselves.

ACKNOWLEDGMENT

Authors gratefully acknowledge Visvesvaraya Technological University (VTU), Belagavi, Karnataka, India for making this research work successful. They also convey gratitude to management, Dayananda Sagar Group of Institutions (DSI) for all its support and encouragement for this research work in research centre (RC), TCE, DSCE. Bengaluru, India.

REFERENCES


K. S. Shashidhara, Bengaluru, Karnataka, India. 
(14th January 1975). Obtained Bachelor of Engineering from Mysore University, Karnataka, India in the year 2000. Master of Technology in Industrial Electronics, Vivesvaraya Technological University(VTU), Karnataka, India, in the year 2002. Currently he is pursuing Ph.D in Engineering in the area of Digital Signal Processing on Reconfigurable hardware at VTU, RC, department of TCE, DSCE, Bengaluru, Karnataka, India.

He is working as an Assistant Professor in the department of Electronics and Communication Engineering, Nitte Meenakshi Institute of Technology (NMIT), Bengaluru, Karnataka, India.

H. C. Srinivasah, Bengaluru, Karnataka, India. 
(16th April 1964), has completed his bachelor degree BE in electronics and communication engineering from the university of Mysore, Karnataka, India during the year 1987. He obtained his master’s degree ME in electronics engineering from Dr. Babasaheb Ambedkar Marathwada University, Aurangabad, MS, India during 1995, He obtained his PhD in engineering from Indian Institute of Science (IISc), Bengaluru India during the year 2005.

He is currently working as professor in the department of Telecommunication engineering (TCE), DSCE, Bengaluru, India since 2009-till date. Prior to this he worked for Cypress semiconductors, Pvt. Ltd. as engineer staff, where he worked towards characterizing sub-100nm CMOS technology during the year 2007 and 2008.

Dr. Srinivasah is currently member of IEEE society of SSC/EDS. He is also a member of VLSI society of India (VSI) and member ISTE, India. He has credit of having worked as member of academic boards of Indian Universities including VTU, Belagavi, Karnataka India. He has published more than 50 papers in national/international journals and proceedings of conferences. He has supervised 3 PhD theses in the research areas of his relevance.