Varsha Adhangale, Dr.R.D.Daruwala / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 3, Issue 4, Jul-Aug 2013, pp. 42-50

FFT PROCESSOR IMPLEMENTATION & THROUGHPUT OPTIMIZATION USING DMA & C2H COMPILER

Varsha Adhangale[1], Dr.R.D.Daruwala[2]
[1, 2] (Department of Electronics Engineering, Mumbai University)

ABSTRACT

Discrete Fourier Transform (DFT) is an important transform in signal analysis and process, but its time complexity can't be accepted under many situations. How to make DFT more fast and efficient has become an important theory. According to the algorithm characteristics of DFT, FFT was brought in and decreased the time complexity to a very large extent.

This paper presents 8-point Fast Fourier transform (FFT) processor using Altera tool & devices such as Nios II Soft processor on DE0 board, C2H Compiler & DMA. The NiosII is soft core processor which is implemented on FPGA available on Altera DE0 board. C2H Compiler is a powerful tool that generates hardware accelerators for software functions. The C2H Compiler enhances design productivity by allowing using a compiler to accelerate software algorithms in hardware. It can quickly prototype hardware functional changes in C, and explore hardware-software design tradeoffs in an efficient, iterative process. Performance was also increased by allowing accelerating only part of that software program on hardware. The C2H Compiler is well suited for improving computational bandwidth as well as memory throughput. It also provides a simpler way of computing complex multiplications, while decreasing latency time. DMA (Direct memory Access) is also one of the important concepts applied to increase the efficiency of implemented system. So in this paper performance of implemented FFT Processor is observed by three different approach & it shows how system will useful in various signals processing applications.

Keywords **-** Butterfly, C2H Compiler, DFT, DMA, FFT

I. INTRODUCTION

With increasing use of technology in every field the need for digital signal processing has increased. Nowadays, there is more demand for reduced cost, area, power and increased speed; which motivated the development of more sophisticated DSP algorithms to enhance the performance. Discrete Fourier Transform (DFT) is power tool for performing DSP Functions.

The DFT takes an N-point vector of complex data sampled in time and transforms it to an N-point vector of complex data that represents the input signal in the frequency domain. The DFT X (k), k =0,…., *N-1* of a sequence x (n), n = 0, 1, …, N-1 is defined as in equation (1),

$$X(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) W_N^{nk}, \quad k = 0 \text{ to } N-1 \qquad (1)$$

Where W_N, the twiddle factor, is defined as

$$W_N = e^{-j2\pi/N}, \quad k = 0 \text{ to } N-1$$

The DFTs are almost never computed directly, but instead are calculated using the Fast Fourier Transform (FFT), which comprises a collection of algorithms that efficiently calculate the DFT of a sequence. FFT produce results identical to the DFT in far fewer cycles. The Cooley-Turkey algorithm [1] is a widely used FFT algorithm that exploits a divide-and-conquer approach to recursively decomposes the DFT computation into smaller and smaller DFT computations until the simplest computation remains. One subset of this algorithm called Radix-2 Decimation-in-Time (DIT) breaks each DFT computation into the combination of two DFTs, one for even-indexed inputs and another for odd-indexed inputs. The decomposition continues until a DFT of just two inputs remains. The 2-point DFT is called a butterfly, and it is the simplest computational kernel of Radix-2 FFT algorithms. So In this paper FFT algorithm is implemented on Soft Core Processor which is Altera Implemented processor Nios II Soft processor on Cyclon III FPGA which is available on DE0 board. This Implemented FFT algorithm on C is hardware accelerated by using C2H compiler & DMA. By using this it will increase throughput of system in terms of no. of clock cycles.

The rest of this paper is organized as follows: Section 2, presents basics of Processor architecture .Section 3, Design of FFT Processor. Result & Comparison of Proposed method with Existing Technique in section 4. Finally, the section 5 gives conclusion of this work.

II. PROCESSOR ARCHITECTURE

FFT Processor Block diagram is as shown in Fig-1. The Nios II processor is the heart of implemented system which is Altera Implemented

Soft-Core Processor. Altera's Nios II is a soft processor, defined in a hardware description language, which can be implemented in Altera's FPGA devices by using the Quartus II CAD system. To implement a useful system it is necessary to add other functional units such as memories, input/output interfaces, timers, and communications interfaces. To facilitate the implementation of such systems, it is useful to have computer-aided-design (CAD) software for implementing a system-on-a-programmable-chip (SOPC). Altera's SOPC Builder is the software needed for this task

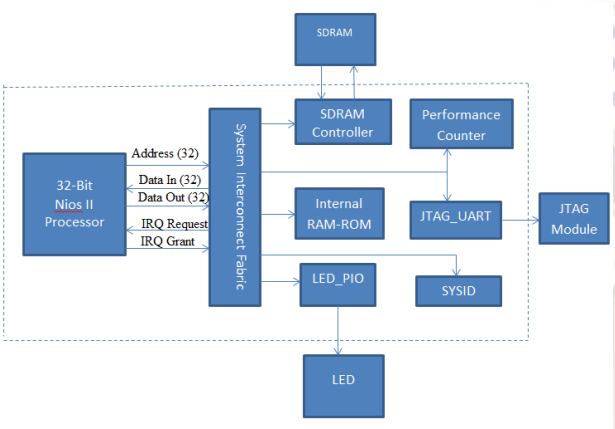

Figure-1 Processor architecture

2.1 Nios-II Soft core processor

The Nios II soft- processor [2] is a general-purpose RISC processor core. Soft means the processor core is not fixed in silicon and can be targeted to any Altera FPGA family. It provides the following features:

- Full 32-bit instruction set, data path, and address space
- 32 general-purpose registers
- 32 interrupt sources
- Access to a variety of on-chip peripherals, and interfaces to off-chip memories and peripherals.
- Hardware-assisted debug module enabling processor start, stop, step, and trace under control of the Nios II software development tools.

Nios II processor is very flexible and has some configurations that can be implemented at the design Stage. These configurations allow the user to optimize the processor for various applications. The most relevant are the clock frequency, the debug level, the performance level and the user defined instructions. The performance level configuration enables the user to choose one of the three performances available: The NIOSII/f (fast), which results in a larger processor that uses more logic elements, but is faster and has more features, such as multiplication and others; The NIOSII/s (standard), which creates a processor with balanced relationship between speed and area, and some special features;

The NIOSII/e (economic), which generates a very economic processor in terms of area, but very simple in terms of data processing capability. The user-defined instructions allow the user to import hardware designs and attach them to the processor hardware, making them accessible by means of customizable instructions. The NIOS II Processor arithmetic and logic operations are performed on operands in the general purpose registers. The operands are moved from memory to these registers by means of Load and Store instructions.

2.2 system Interconnect fabric-Avalon Bus

Nios II processor system consists of a Nios II processor core, a set of on-chip peripherals, on-chip memory, and interfaces to off-chip memory, all implemented on a single Altera device. These components are interconnected by means of the interconnection network called the Avalon Switch Fabric .The Avalon bus is used to connect processors and peripherals in a system. It is asynchronous bus interface that specifies the interface and protocol used between master and slave components. A master component (e.g. a processor) can initiate bus transfers, while a slave component (e.g. memory) only accepts transfers initiated by the master. Multiple masters and slaves are allowed on the bus. In case two masters try to access the same slave at the same time, the bus arbitration logic determines which master gets access to the slave based on fixed priorities. The bus arbitration logic is generated automatically based on the user defined master-slave connections and arbitration priorities. The Avalon bus contains decoding logic that generates a chip-select signal for each peripheral. The Avalon bus supports dynamic bus sizing, so the peripherals with different data widths can be used on a single bus. If a master attempts to read a slave that is narrower than the master, the bus logic automatically issues multiple read transfers to get the requested data.

2.3 SDRAM Controller

The SDRAM controller core with Avalon interface provides an Avalon Memory-Mapped (Avalon-MM) interface to off-chip SDRAM. The SDRAM controller [3] allows designers to create custom systems in an Altera device that connect easily to SDRAM chips. The SDRAM is relatively inexpensive, control logic is required to perform refresh operations, open-row management, and other delays and command sequences. The SDRAM controller connects to one or more SDRAM chips, and handles all SDRAM protocol requirements. Internal SDRAM Core shown in Fig-2.

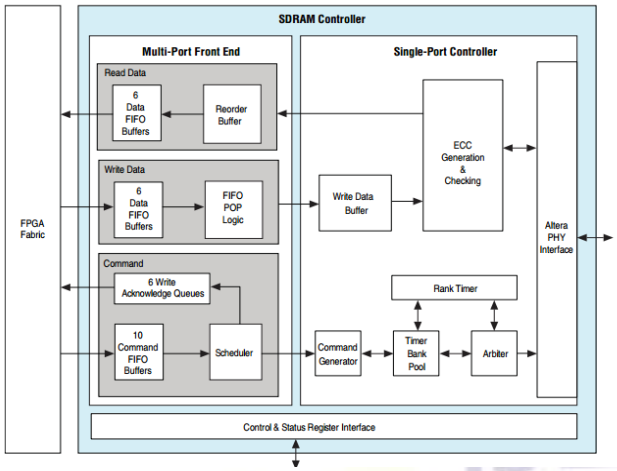

Figure-2 SDRAM Controller Core

It is divided into two parts Multi-port front end (MPFE) & Single-port controller (SPC)

2.3.1 MPEF (Multi-Port front end)

The MPFE [4] consists of the following three primary sub-blocks.

- The Command Block accepts read and write transactions from the FPGA fabric. For each pending transaction, the command block calculates the next SDRAM burst needed to progress on that transaction.
- The write data block transmits data to the single-port controller. The write data block informs the command block of the amount of pending write data for each transaction.
- The read data block receives data from the single-port controller. In order to prevent the read FIFO buffer from overflowing, the read data block informs the command block of the available buffer area so the command block can pace read transaction dispatch.

2.3.2 SPC (Single Port Controller)

The single-port logic is responsible for following actions: Queuing the pending SDRAM bursts, choosing the most efficient burst to send next, Keeping the SDRAM pipeline full, and ensuring all SDRAM timing parameters are met .It consist of following block & their function:

- The command generator accepts commands from the MPFE and from the internal ECC logic, and provides those commands to the timer bank pool.
- The timer bank pool is a parallel queue that operates with the arbiter to enable data reordering.
- The arbiter determines the order in which requests are passed to the memory device.
- The rank timer performs the following functions: Maintains rank-specific timing information, Ensures that only four activates occur within a specified timing window
- The write data buffer receives write data from the MPFE and passes the data to the PHY, on approval of the write request.
- The ECC block consists of an encoder and a decoder-corrector, which can detect and correct single-bit errors, and detect double-bit errors.

2.4 PLL (Phase Lock Loop)

In synchronous transaction revelent signal may be valid for small window of time. However accessing SDRAM involves additional issues. First since SDRAM controller & SDRAM [3] reside on two separate devices, timing parameter, such as clock to q delay, set up time ,hold up time are not identical. Off chip access introduces additional delay and has significant impact on controller clock to output time. Finally clock skew also exist because rising edge may not be able to arrive at SDRAM controller & SDRAM devices at the same time. One way to overcome this problem is to adjust phase between controller clock and SDRAM devices. The required clock adjustment is done by FPGA internal PLL circuit as show on following Fig-3.

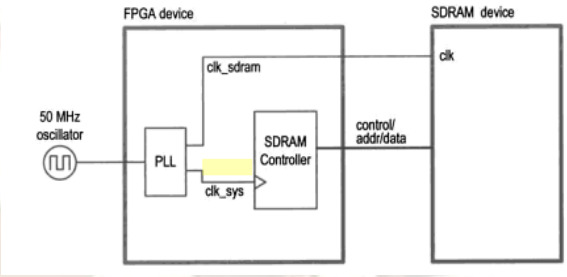

Figure-3 PLL Circuit

In above method external 50MHz of clock is fed to PLL which generates clk_sysclk to drive SDRAM controller and clk_sdram clk which leds by 3ns to drive external SDRAM devices.

2.5 JTAG UART

The JTAG universal asynchronous receiver/transmitter (UART) core with Avalon interface implements a method to communicate serial character streams between a host PC and an SOPC Builder system on an Altera FPGA. Fig-4 Shows JTAG UART Core.

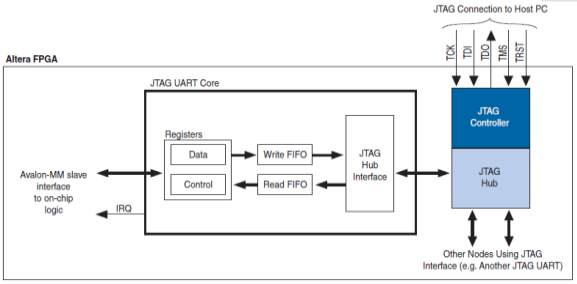

Figure-4 JTAG UART Core

The JTAG UART core consists of two 32-bit registers, data and control that are accessed through an Avalon slave port. An Avalon master, such as a Nios II processor, accesses the registers to control the core and transfer data over the JTAG connection. The core operates on 8-bit units of data at a time; The JTAG UART core provides bidirectional FIFOs to improve bandwidth over the JTAG connection. The JTAG controller can connect to user-defined circuits called "nodes" implemented in the FPGA. Because there may be several nodes that need to communicate via the JTAG interface, a JTAG hub (i.e., a multiplexer) becomes necessary.

2.6 JTAG Module

The Nios II architecture supports a JTAG debug module that provides on-chip emulation features to control the processor remotely from a host PC. PC-based software debugging tools communicate with the JTAG debug module and provide facilities, such as the following features:

- Downloading programs to memory
- Starting and stopping execution
- Setting breakpoints and watch points
- Analyzing registers and memory
- Collecting real-time execution trace data

2.7 Performance Counter

A performance counter is a block of counters in the hardware that measures the execution time of the code sections that user choose. A performance counter component can track up to seven code sections. By default, the component tracks three code sections. A pair of counters tracks each code section:

- Time—A 64-bit time (clock tick) counter that counts the number of clock ticks during code section runs.
- Occurrences—A 32-bit event counter that counts the number of times the code section runs.

The performance counter component occupies a substantial number of logic elements (LEs) on device, and requires software implementation to obtain performance measurements. These counters enable to measure the execution time of the designated sections of C/C++ code. Macros enable us to mark the start and the end of the code sections in program.

2.8 System ID (SYSID)

The system ID peripheral safeguards against accidentally downloading software compiled for a different Nios II system. If the system includes the system ID peripheral, can prevent us from downloading programs compiled for a different system.

III. IMPLEMENTATION METHODOLOGY

The proposed method in this paper implements FFT processor on FPGA using ALTERA tool & DE0 Development board. With the help of such tool & SOPC Technology makes it easier for design the system as per the requirement with less complexity. ALTERA provides various range of Development board with different feature such as Cyclone, Stratix edition, designer can choose according to the application requirements. In this performance of implemented system is compare by three different approaches:

- Software only approach
- Hardware Accelerator approach- Using C2H compiler
- DMA Approach

Performance comparison in each approach is done by no. of clock cycles required to get final output.

3.1 Requirement of tool for complete System Development

Software:

- Altera Quartus II software version 7.1 or later.
- Nios II Embedded Design Suite version 7.1 or later

Hardware

- Altera DE0 Development board

The Complete Design Flow is as shown in Fig-5.

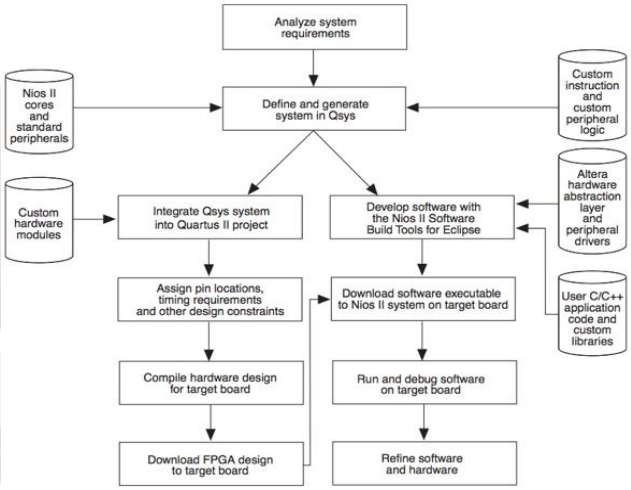

Figure-5 Design Flow of System

After the initial system specification, the design flow is divided into a hardware and software development. Processor, memory and peripheral properties are defined during the hardware development process. User programs and other system software are developed and built during the software development process. The software development is dependent on the hardware development results. After the system hardware and software have been built, the system prototype is tested on a development board featuring an FPGA

device and other components useful for prototyping. If the system meets the specification, the system design is complete. Otherwise, either hardware or software needs to be redesigned.

3.2 Hardware Development

The Development flow is depend on the application requirement such as, Computational performance, bandwidth, throughput, types of interface & software which is implemented on such system, Based on this requirement user can determine the system at design stage. Hardware Development is as shown in following Fig-6

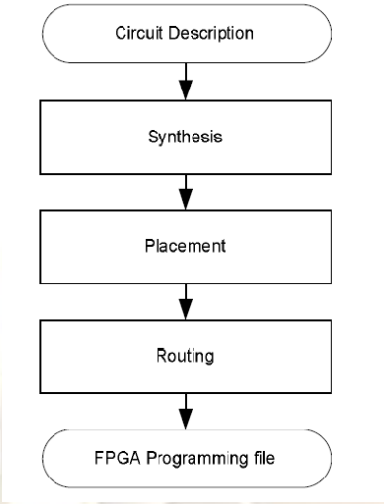

Figure-6 Hardware Development Flow

Input to a system is a high-level circuit description, which is typically provided using a hardware description language (HDL). An HDL circuit description is converted into a netlist of basic gates in the synthesis step of the design flow. The placement algorithm maps logic blocks from the netlist to physical locations on an FPGA. Once the placement has been done, the routing algorithm determines how to interconnect the logic blocks using the available routing. The routing algorithm produces a set of programming bits determining the state of all the interconnection switches inside an FPGA. The final output produce is the FPGA programming file, which is a bit stream determining the state of every programmable element inside an FPGA. Design flow, including synthesis, placement and routing is referred to as the design compilation.

This FPGA Programming is done by using Quartus-II Tool. Quartus II provides a set of tools for circuit designs targeting Altera programmable devices. These tools include design editors, compilation and simulation tools, and device programming software. After analysing the system requirement use SOPC builder tool which is included in Altera Quartus II software.

3.2.1SOPC Builder

SOPC Builder is a tool for the integration and configuration of a bus-based system consisting of library and user components. Library components include processors, memories, bus arbiters and bridges, standard peripherals, and other IP cores.

The SOPC Builder consists of two parts: graphical user interface (GUI), and a system generator program. The GUI is used to select system components, and to set various component and system parameters. Depending on the peripheral type, memory mapping and the interrupt priority may be defined. System components are automatically interconnected using one of the available buses. The SOPC Builder supports two bus types: AMBA–AHB and Avalon, in this we use Avalon interconnects Bus. The system configuration defined in the SOPC Builder GUI is stored in a system PTF file, which is a plain text file that completely describes the system. The system generator program uses the information in the system PTF file to generate HDL code and software for the system. By using SOPC builder components are added which are require to implement system. SOPC builder automatically generates interconnect logic to integrate component into hardware system. User can also add custom instruction logic to system. Final implemented system overview in SOPC Builder is as shown in Fig-7.

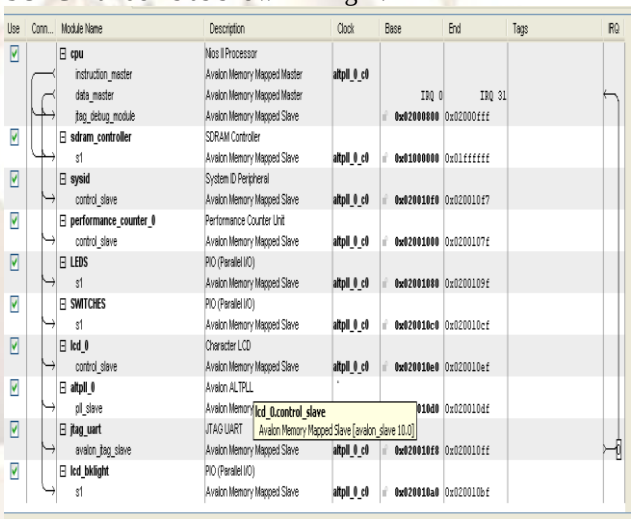

Figure-7 SOPC System Overview

3.2.2 Schematic Design

The Quartus II Graphic Editor can be used to specify a circuit in the form of a block diagram or schematic. In this connection of Nios II processor pin according to their function has been done like Input, Output & Bidirectional as shown in Fig-8. Once system is compile successfully by using SOPC Builder Nios II soft Processor is ready with pin & their functionality.

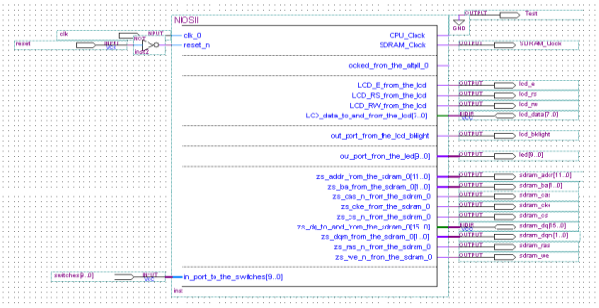

Figure-8 Schmetic of System

3.2.3 Pin Planner

Pin assignments are made by using the Assignment Editor. The DE0 board has fixed pin assignments. Recompile the circuit, so that it will be compiled with the correct pin assignments. A simple file format that can be used for this purpose is the comma separated value (CSV) format, which is a common text file format that contains comma-delimited values. According to pin assignment changes has been done as shown in Fig-9.

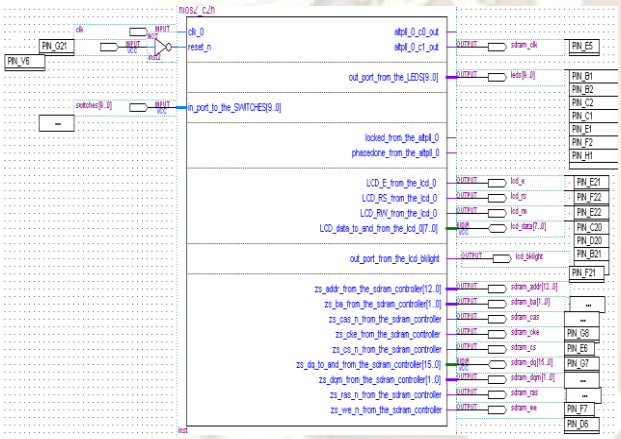

Figure-9 Pin Planner

3.2.4 JTAG Programming

The FPGA device must be programmed and configured to implement the designed circuit. The required configuration file is generated by the Quartus II Compiler's. Altera's DE0 board allows the configuration to be done by JTAG as shown in Fig-10. The configuration data is transferred from the host computer (which runs the Quartus II software) to the board by means of a cable that connects a USB port on the host computer. Quartus II software places the configuration data into the configuration device on the DE0 board. Then, this data is loaded into the FPGA upon power-up or reconfiguration.

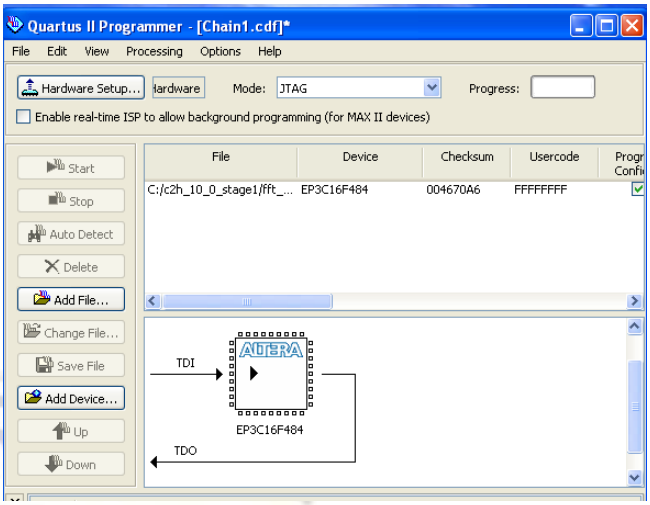

Figure-10 Programming Window

To complete the hardware design, perform the following:

- Instantiate the module generated by the SOPC Builder into the Quartus II project.
- Assign the FPGA pins.
- Compile the designed circuit.
- Program and configure the Cyclone III device.

3.3 Software Development

The Nios II EDS provides software development environment that works for all Nios II processor system. With Nios II running on a host computer, an Altera FPGA & JTAG download cable we can write a program for & communicate with any Nios II processor system The Nios II EDS provides the following development flows for creating Nios II programs:

- Nios II Software Build Tools(SBT)
- Nios II integrated development environment(IDE)

These flows differ in how they create makefile. These file in two flows are different and are not compatible. In this project we use Nios II IDE development flow, which provides integrated environment in which we can create, modify, build, run and debug Nios II programs with the Nios II IDE GUI. The makefile it creates cannot be user managed. This flow provides limited control over build process & project setting, with no support for customized scripting. The Nios II EDS perform very useful task to run the software program on hardware DE0 Board.

3.4 Different Implementation Approach

3.4.1Software only

With the help of Nios II IDE we had Implemented FFT Algorithm & that will run on DE0 board cyclone III FPGA. Initialization of performance counter is done in this algorithm which we had already specified in hardware development

so they can run in synchronism to get no. of clock cycles required for final output.

3.4.2Hardware Accelerator Approach using C2H compiler

A hardware accelerator is a block of logic that implements a C function in hardware. Hardware accelerator implemented in a field programmable gate array (FPGA), Based on SOPC Builder and Avalon system interconnect fabric. The C2H Compiler uses SOPC Builder as the infrastructure to connect hardware accelerators into Nios II systems. A C2H accelerator becomes a component within an existing Nios II system. SOPC Builder automatically generates system interconnect fabric to connect the accelerator to the system.

The Nios II IDE allows carrying out the following important tasks:

- Debug function prior to accelerating
- Generate the accelerator and incorporate it into hardware
- Test and profile software and hardware with the C2H accelerator

The C2H Compiler converts only sections of code that user specify. A typical program contains a mix of performance-critical code and other code. Performance-critical sections are often iterative and simple, but consume the majority of a program's execution time on a processor. The best use of hardware resources is to accelerate only the performance-critical functions of a program, rather than converting an entire program to hardware. After using C2H compiler FFT accelerator component is get added in system as hardware unit and its connection with other component is done automatically by SOPC builder by using Avalon switch fabric. Final system overview in SOPC builder after adding Hardware Accelerator is as shown in Fig-11.

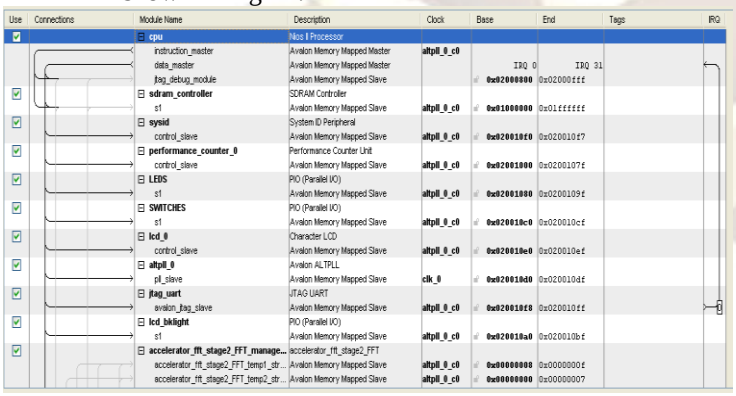

Figure-11 Hardware Accelerator System overview

3.4.3 DMA Approach

Embedded systems frequently employ DMA engines to increase data throughput and offload memory copy operations from the processor [9]. In many cases the memory locations being accessed are dispersed throughout the address space.

The Butterfly operation, which requires the processor to execute a large number of memory accesses, is time consuming. Because the primary bottleneck in the Butterfly computation is memory throughput, combining the FFT algorithm and DMA operation into a hardware accelerator maximizes system performance. The C code to transfer the data in each buffer is replaced by an accelerated function that reads from the data buffers residing in main memory (SDRAM). The DMA operation is implemented in Software algorithm.

Final Implemented System block diagram after applying three different approach is as shown below Fig-12.

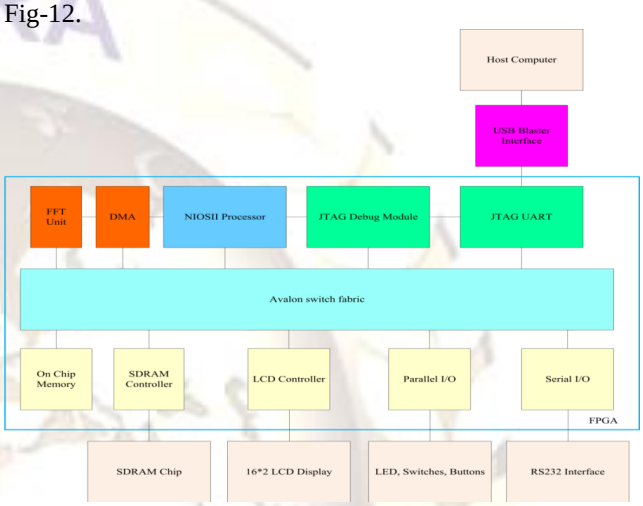

Figure-12 Final System Block Diagram

IV. RESULT

For comparing the performance in this 3 different approach we input data in power of 2 like [1 2 4 8 16 32 64 128] & observe their corresponding ouput & no. of clock cycles in each approach.

```
--Performance Counter Report--
Total Time: 0.00011294 seconds  (5647 clock-cycles)
+--------------+-----+----------+--------------+-----------+
| Section      |  %  | Time (sec)|  Time (clocks)|Occurrences|
+--------------+-----+----------+--------------+-----------+
|Software Only | 99.6|    0.00011|          5625|          1|
+--------------+-----+----------+--------------+-----------+
Real data index 0 = 255, Imaginary Data index 0 = 0
Real data index 1 = 480, Imaginary Data index 1 = 1650
Real data index 2 = -510, Imaginary Data index 2 = 1020
Real data index 3 = -780, Imaginary Data index 3 = 450
Real data index 4 = -850, Imaginary Data index 4 = 0
Real data index 5 = -780, Imaginary Data index 5 = -450
Real data index 6 = -510, Imaginary Data index 6 = -1020
Real data index 7 = 480, Imaginary Data index 7 = -1650
```

Figure-1 Software Only Approach Result

```
--Performance Counter Report--
Total Time: 9.706E-05 seconds  (4853 clock-cycles)
+---------------+-----+-----------+---------------+-----------+
| Section       |  %  | Time (sec)|  Time (clocks)|Occurrences|
+---------------+-----+-----------+---------------+-----------+
|Hardware Only  | 99.8|    0.00010|           4841|          1|
+---------------+-----+-----------+---------------+-----------+
Real data index 0 = 255, Imaginary Data index 0 = 0
Real data index 1 = 480, Imaginary Data index 1 = 1650
Real data index 2 = -510, Imaginary Data index 2 = 1020
Real data index 3 = -780, Imaginary Data index 3 = 450
Real data index 4 = -850, Imaginary Data index 4 = 0
Real data index 5 = -780, Imaginary Data index 5 = -450
Real data index 6 = -510, Imaginary Data index 6 = -1020
Real data index 7 = 480, Imaginary Data index 7 = -1650
```

Figure-2 Hardware Accelerator approach

```
--Performance Counter Report--
Total Time: 9.32E-05 seconds  (4660 clock-cycles)
+---------------+-----+-----------+---------------+-----------+
| Section       |  %  | Time (sec)|  Time (clocks)|Occurrences|
+---------------+-----+-----------+---------------+-----------+
|Hardware + DMA | 99.4|    0.00009|           4630|          1|
+---------------+-----+-----------+---------------+-----------+
Real data index 0 = 255, Imaginary Data index 0 = 0
Real data index 1 = 480, Imaginary Data index 1 = 1650
Real data index 2 = -510, Imaginary Data index 2 = 1020
Real data index 3 = -780, Imaginary Data index 3 = 450
Real data index 4 = -850, Imaginary Data index 4 = 0
Real data index 5 = -780, Imaginary Data index 5 = -450
Real data index 6 = -510, Imaginary Data index 6 = -1020
Real data index 7 = 480, Imaginary Data index 7 = -1650
```

Figure-3 DMA Approach

With different sets of data we had observed that there is almost 900 to 1000 no. of clock cycles reduction after applying DMA & hardware accelerator approach for implementation of FFT algorithm on Nios II soft core processor.

TABLE-1 Comparison with Different sets of data

Sets of Data	Software Only approach	Hardware Accelerator Approach	DMA+Hard ware accelerator approach
[1 2 3 4 5 6 7 8]	5620	4854	4625
[1 2 3 4 4 3 2 1]	5608	4854	4822
[1+1j 2+2j 3+3j 4+4j 5+5j 6+6j 7+7j 8+8j]	5617	4856	4618

4.1 Comparison with Existing Method

Various implementation techniques of FFT Algorithem & their performance characteristics is as follows:

- HDL (Streaming)

The Pipelined Streaming [10] I/O solution pipelines several Radix-2 butterfly processing engines to offer continuous data processing. Each processing engine has its own memory banks to store the input and intermediate data. We can continuously stream in data and, after the calculation latency, can continuously unload the results.

- HDL (Pipeline)

Pipelined FFT architecture called Radix-2^2 Single Path Delay Feedback ($R2^2$ SDF) and it is implemented on FPGAs using a high-level C-to-hardware programming language for rapid prototyping.

- Software

FFT Algorithm Implemented using Software requires less silicon area. It has drawback as the no. of input increased butterfly complexity also increases which decrease the speed of operation hence degrades the system performance.

- C2H Compiler

The C2H Compiler translates FFT Algorithm implemented in C constructs to their hardware equivalents. C2H accelerators consume hardware resources such as LE (Logic elements), multipliers, and on-chip memory.

Following Table gives Comparison of all above Technique in terms of advantages & disadvantages.

Table-5 Comparison with Existing Technique

Technique	Performance	Advantages	Disadvantage
HDL (Streaming)	Very high	Optimized for speed	1) Large Silicon area FPGA/ASIC 2) Difficult to design
HDL (Pipeline)	High	Optimized for silicon area	Less speed compared to streaming mode
Software	Poor	1) Very less silicon area 2) Very Easy to design	Very low speed
C2H	Moderate	1) Moderate silicon area 2) Very Easy to design 3) Much better performance than software approach	Not recommended for very high speed/ time critical application

V. CONCLUSION

The capabilities of FPGAs have increased to the level where it is possible to implement a complete computer system on a single FPGA chip. The main component in such a system is a soft-core processor. The Nios soft-core processor is intended for implementation in Altera FPGAs.

In this project, we have design FFT processor using ALTERA DE0 board & Design Tools. It includes FPGA which is dedicated only for Altera family of FPGA. The main advantage offered by this method is flexibility & parallelism. Flexibility makes design process complex. The SOPC builder system design tool helps to manage with this complexity. SOPC builder can also provide mechanism for peripheral expansion or processor offload. Complete system can be implemented on any FPGA according to application requirement which make it to be used in real time environment. Implementing Part of Software algorithm as a Hardware unit not only increases its throughput but

also increase design efficiency. With a few straightforward code optimizations, the Nios II C2H Compiler can sharply improve the computational bandwidth and memory throughput of a software algorithm. So Part of FFT Algorithm as Hardware Accelerator & DMA is very much useful which provides flexibility for user to implement design as per the requirement and gives performance in terms of less latency period.

REFERENCES

[1] Jayasumana, G. Colorado State University, Ft. Collins, CO Loeffler, C. ,"Searching for the best Cooley-Tukey FFT algorithms" *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '87* , vol.12, pp. 2408- 2411, : Apr 1987

[2] Joshi, N.N. Y.C.C.E., Nagpur, India Dakhole, P.K. ; Zode, P.P.," Embedded Web Server on Nios II Embedded FPGA Platform", ?" *proc IEEE ICETET*, vol. 55, pp. 372-377,DEC 2009

[3] Pong P. Chu *Embedded SOPC Design with Nios II Processor and VHDL Examples* Wiley Publisher, August 2011

[4] SDRAM Controller Subsystem http://www.altera.com/literature/hb/cyclone-v/cv_54008.pdf

[5] JTAG UART Core ftp://ftp.altera.com/up/pub/Altera_Material/11.1/Laboratory_Exercises/Computer_Organization/DE0/lab7.pdf

[6] Franjo Plavec(2004) *Soft-Core Processor Design* Master Thesis, Toronto University , Toronto

[7] Sahil Sharma(2010) *Implementation of Web-Server Using Altera DE2-70 FPGA Development Kit* Bachelor Thesis, National Institute of Technology(NIT), Rourkela

[8] Altera C2H Compiler http://www.altera.com/literature/tt/tt_nios2_c2h_accelerating_tutorial.pdf?GSA_pos=1&WT.oss_r=1&WT.oss=fft%20c2h

[9] DMA http://www.altera.com/literature/an/an417.pdf?GSA_pos=7&WT.oss_r=1&WT.oss=DMA

[10] Xlinx Logic-Core IP Fast Fourier Transform http://www.xilinx.com/support/documentation/ip_documentation/ds808_xfft.pdf