PERANCANGAN DAN IMPLEMENTASI ALGORTIMA FFT 64 TITIK MENGGUNAKAN *MULTIPATH DELAY COMMUTATOR* PADA FPGA

DESIGN AND IMPLEMENTATION OF 64 POINT FFT ALGORITHM USING MULTIPATH DELAY COMMUTATOR ON FPGA

Nanda Aldira Fakhri[1], Dr.Ir. Rina Pudji Astuti, M.T.[2], Denny Darlis, S.Si. M.T.[3]

[1,2,]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
[3]Prodi D3 Teknik Telekomunikasi, Fakultas Ilmu Terapan, Universitas Telkom
[1]nandaldira@students.telkomuniversity.ac.id [2]rinapudjiastuti@telkomuniversity.ac.id
[3]dennydarlis@telkomuniversity.ac.id

Abstrak

Dalam pengaplikasian Algoritma FFT, kecepatan komputasi, sederhana dalam implementasi dan hemat memori adalah hal yang harus diperhatikan. Untuk kebutuhan tersebut teknik yang paling cocok untuk implementasi adalah *pipeline architecture*. Keunggulan dari *pipeline architecture* adalah data bisa diparalelkan saat pemrosesan, bekerja secara *real time*, pengolahan secara kontinu dan mempunyai *latency* yang kecil.

Pada penelitian ini dilakukan perancangan FFT dengan menggunakan salah satu varian dari teknik *pipeline architecture* yaitu MDC dikarenakan *control* yang lebih simpel, jumlah data yang diproses adalah 64 *subcarrier* dengan menggunakan *radix* 2 peruraian dalam frekuensi untuk pendekatan algoritma FFT. Perancangan dilakukan berdasarkan standar yang sudah ditentukan. Selanjutnya hasil rancangan disimulasikan pada *software* ModelSim. Dari hasil pemodelan dan simulasi, kemudian di implementasikan ke perangkat FPGA. Hasil Implementasi menunjukkan bahwa perancangan *prototype* Algoritma FFT menggunakan teknik MDC dapat diimplementasikan pada *board* ATLYS Spartan-6.

Hasil implementasi menunjukkan penggunaan *resource slice registers* sebesar 3,6%, penggunaan *bonded IOBs* sebesar 10,55%, dan *delay process* menghasilkan 22.900 ns, sedangkan pada penelitian sebelumnya dalam penggunaan *resource slice registers* sebesar 4,4%, penggunaan *bonded IOBs* sebesar 34,4% dan menghasilkan *delay process* 35.400 ns. Sehingga MDC merupakan solusi implementasi algoritma FFT dalam effisiensi memori dan kecepatan komputasi. *Prototype* ini menghasilkan sistem dengan periode minimum 15,895 ns dan frekuensi kerja 62,914 MHz.

Kata kunci : FFT, *Multipath Delay Commutator*, FPGA

Abstract

In the application of FFT algorithm, computing speed, simple in implementation and memory savers are things that must be considered. For those needs of the most suitable technique for implementation is the pipeline architecture. The advantages of the pipeline architecture is processing data can be parallelized, works in real time, continuous processing and have a little latency.

In this research, the design of FFT by using one of the variants of the technique because the MDC pipeline architecture that is more simple control, the amount of data processed is 64 subcarriers using a radix 2 decimation in frequency for FFT algorithm approach. The design is based on the determined standards. Furthermore, the design is simulated in the ModelSim software. From the results of modeling and simulation then implemented into the FPGA device. The results showed that the design of prototype implementation of FFT algorithm using MDC techniques can be implemented on Spartan-6 ATLYS board.

The implementation results show resource usage slice registers by 3.6%, use of bonded IOBs 10.55%, and a delay process to produce 22900 ns, whereas in previous research in resource usage slice registers at 4.4%, the use of bonded IOBs of 34.4 % and produce 35400 ns delay process. MDC is a solution so that the FFT algorithm implementation in memory efficiency and computing speed. Prototype This results in a system with a minimum period 15.895ns and 62.914 MHz operating frequency.

Keywords : FFT, *Multipath Delay Commutator*, FPGA

1. Pendahuluan

2.1. Latar Belakang

Dalam implementasi perangkat keras banyak teknik arsitektur untuk mengaplikasikan Algoritma FFT dengan syarat mempunyai kecepatan komputasi, kompleksitas *hardware*, fleksibilitas dan akurasi yang tinggi. Dikarenakan dalam implementasi, kecepatan komputasi yang tinggi adalah hal utama maka teknik arsitektur yang paling cocok untuk implementasi adalah arsitektur *pipeline* [15]. Selain bekerja secara *real time*, pengolahan data secara kontinu dan mempunyai *latency* yang kecil ketika diimplementasikan, arsitektur ini juga bisa memproses secara paralel sehingga juga bisa menghemat memori [15]. Struktur *Pipeline architecture* sendiri diklasifikasikan

dalam tiga varian yaitu *Single Delay Feedback* (SDF), *Single Delay Commutator* (SDC) dan *Multipath Delay Commutator* (MDC). Untuk *Single Delay Feedback input*nya dari *feedback output* sebelumnya. Sedangkan *Single Delay Commutator* dan *Multipath Delay Commutator*, *output* di *forward* ke blok berikutnya untuk dijadikan *input* [3]. Walaupun SDF lebih hemat memori tetapi SDC dan MDC lebih cepat dalam komputasi dibanding dengan SDF. Meskipun SDC dan MDC sama-sama cepat dalam dalam komputasi tetapi MDC mempunyai kecepatan komputasi yang lebih tinggi daripada SDC, karena data diproses secara paralel sedangkan SDC data diproses secara serial. Oleh karena itu untuk tujuan implementasi perangkat keras, maka MDC sering digunakan sebagai arsitektur perangkat keras [15].

Dalam Implementasi Algoritma FFT selain kecepatan komputasi tinggi yang diperhitungkan, kapasitas memori juga harus diperhitungkan karena jika mempunyai memori yang besar maka *hardware* untuk implementasi harus mempunyai memori yang besar juga. Pada penelitian sebelumnya [14], telah dilakukan perancangan dan implementasi Algoritma FFT dengan jumlah *subcarrier* sebesar 64 dan digunakan *radix* 2 tanpa menggunakan teknik arsitektur. Pada penelitian tersebut menghasilkan pemakaian penggunaan *resource slice registers* sebesar 4,4%, penggunaan *bonded IOBs* sebesar 34,4% dan menghasilkan *delay process* 35400 ns [14].

Pada penelitian yang diajukan ini jumlah *subcarrier* dan penggunaan radix sama dengan penelitian sebelumnya[14] yaitu 64 *subcarrier* dan *radix* 2. Tetapi pada penelitian ini digunakan salah satu teknik *Pipeline Architecture* yaitu MDC (*Multipath Delay Commutator*) sebagai arsitekturnya untuk efisiensi *resource* dan peningkatan kecepatan komputasi pada FPGA. Teknik MDC sendiri sesuai dengan namanya yaitu menggunakan *Delay* yang *multipath* dan juga menggunakan *Commutator* (*switch*) untuk perancangannya. Setelah *prototype* sistem berhasil dirancang, selanjutnya di-*load* pada *board* FPGA Spartan-6.

2. Dasar Teori

2.1. DFT (*Discrete Fourier Transform*)

DFT merupakan prosedur matematika yang digunakan untuk menentukan frekuensi di suatu sinyal diskrit[4].

Persamaan untuk DFT yaitu [4]:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \qquad 0 \leq k \leq N-1 \qquad (2.1)$$

Dimana

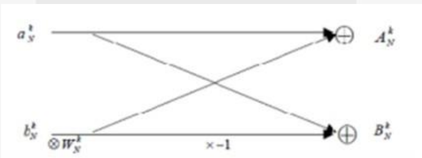
N = jumlah subcarrier

W_N = twiddle factor = $e^{-j2\pi/N}$

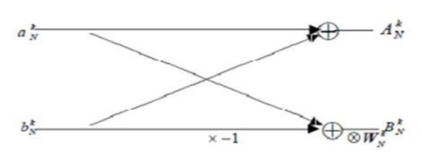
Dengan nilai $e^{j\theta} = \cos\theta + j\sin\theta$

2.2. FFT (*Fast Fourier Transform*)

Fast Fourier Transformation atau Transformasi Fourier Cepat digunakan untuk penghitungan komputasi yang lebih cepat dan mampu mereduksi jumlah perkalian kompleks dari N^2 menjadi $(N/2)\log_2 N$ perkalian[6]. Dalam pemrosesan sinyal digital dikenal *butterfly* yang digunakan untuk mendeskripsikan *decimation* dalam FFT. Ada dua varian *decimation* dalam FFT yaitu *Decimation In Time* (DIT) dan *Decimation In Frequency* (DIF) [4]. Berikut gambar yang merepresentasikan *decimation*.



Gambar 2.1 ***Decimation In Time*** (Sumber : Syafiq Irsyadi,2007 [5])



Gambar 2.2 ***Decimation In Frequency*** (Sumber : Syafiq Irsyadi,2007 [5])

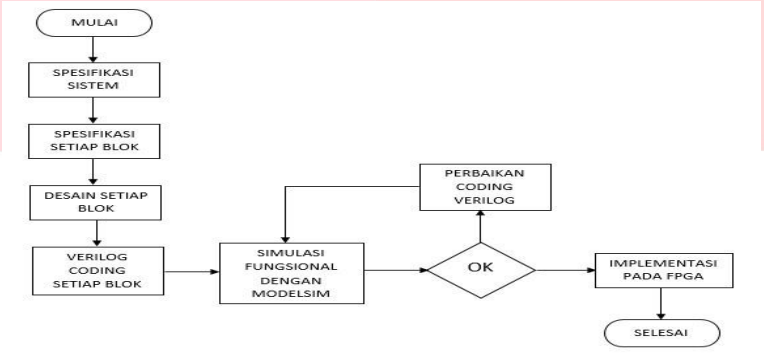
2.3 *Pipeline Architecture*

Pipeline architecture merupakan salah satu arsitektur Algoritma FFT yang dapat melakukan komputasi secara terus secara *real-time* [13]. Arsitektur ini cocok untuk implementasi Algoritma FFT karena selain cepat dalam kecepatan komputasi dan mampu bekerja secara *real-time*, arsitektur ini juga mampu menghasilkan *Throughput* yang tinggi [11][13], menghemat memori dalam Implementasi dan juga saat pemroresan data bisa diparalel [13]. *Pipeline architecture* sendiri terbagi menjadi 3 varian yaitu SDF *(Single Path Delay Feedback*), SDC (*Single Path Delay Commutator*) dan MDC (*Multiple Path Delay Commutator*) [17].

3. Perancangan dan Implementasi

3.1. Alur Perancangan Sistem

Dalam perancangan sistem terbagi beberapa tahap. Tahapan alur desain perancangan FFT 64 titik dilakukan dengan menggunakan teknik MDC ditunjukkan dalam gambar berikut ini.



Gambar 3.2 Diagram Blok Perancangan Sistem

3.2. Perancangan Sistem

Pada perancangan Algoritma FFT ini digunakan algoritma Cooley-Tukey dengan teknik *Decimation in Frequency* dan dikarenakan menggunakan *radix* 2 maka $64 = 2^6$. Sehingga dalam perancangan maka akan dilakukan dalam 6 tahap perkalian kalkulus *Butterfly*. Berikut penuruan rumus secara matematis.

Dengan N = 64, maka persamaan FFT menjadi.

$$X(k) = \sum_{n=0}^{63} x(n) \cdot W_N^{nk} \quad (3.1)$$

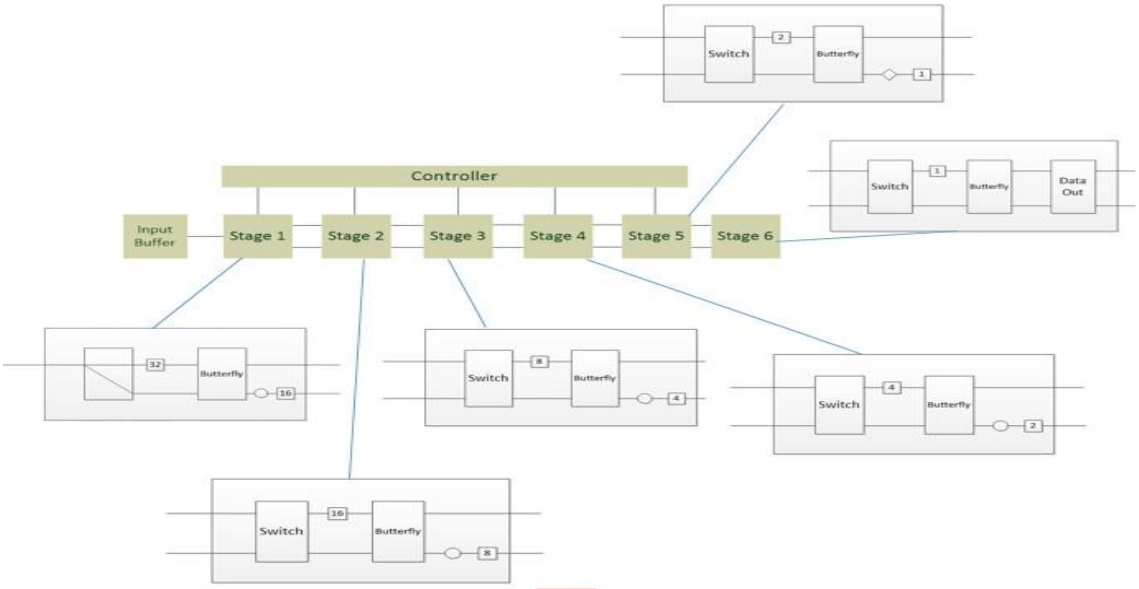
$$n = n_0 + 2n_1 + 4n_2 + 8n_3 + 16n_4 + 32n_5 \qquad (n_0, n_1, n_2, n_3, n_4, n_5 = 0,1)$$

$$k = k_0 + 2k_1 + 4k_2 + 8k_3 + 16k_4 + 32k_5 \qquad (k_0, k_1, k_2, k_3, k_4, k_5 = 0,1)$$

Sehingga dapat diperoleh.

$$X(k_0 + 2k_1 + 4k_2 + 8k_3 + 16k_4 + 32k_5) = \sum_{n_0=0}^{1} \sum_{n_1=0}^{1} \sum_{n_2=0}^{1} \sum_{n_3=0}^{1} \sum_{n_4=0}^{1} \sum_{n_5=0}^{1} x(n_0 + 2n_1 + 4n_2 + 8n_3 + 16n_4 + 32n_5) W_{64}^{(n_0+2n_1+4n_2+8n_3+16n_4+32n_5)(k_0+2k_1+4k_2+8k_3+16k_4+32k_5)} \quad (3.2)$$

Dari persamaan (3.2) dapat disimpulkan bahwa Perancangan FFT 64 titik dengan *radix* 2 terdiri dari 6 blok *stage* dan dikarenakan digunakan MDC (*Multipath Delay Commutator*) sebagai arsitekturnya, maka rancangan tidak hanya terdiri dari *Butterfly* tapi terdapat blok *Delay* dan *Switch* (*Commutator*). Kemudian untuk jumlah jalur (*path*) untuk melakukan komputasi untuk lebih detailnya tergambar pada gambar (3.2).



Gambar 3.2 Blok Sistem FFT 64 titik menggunakan teknik MDC

3.3. Blok-Blok Penyusun Desain Sistem FFT 64 titik

Pada sistem yang akan dirancang, masing-masing desain terdiri dari dua blok utama, yaitu *controller* dan *stage*. Keduanya terprogram secara independen yang kemudian diintegrasikan dalam satu wadah.

3.3.1. Controller

Blok ini bukan hanya berfungsi sebagai pengontrol, tetapi juga berfungsi sebagai pengalamatan (*addressing*). Karena dalam perancangan ini fungsi pengontrolan dan pengalamatan digabung menjadi dalam satu blok dan dinamakan blok *controller*. Sehingga dalam blok ini yang mengatur semua aliran data.

3.3.2. Stage

Pada sistem yang akan dirancang terdapat 6 buah *stage* yang sudah dijelaskan pada persamaan (3.1) dan (3.2). Dalam setiap *stage* terdapat beberapa komponen (*sub* sistem). Diantaranya adalah *Butterfly*, *Delay*, *Switch* (*Commutator*), *Inverter*, dan blok pembualatan pada *stage* terakhir.

- **Butterfly**

Blok *butterfly* ini merupakan proses komputasi dasar dalam algoritma FFT. Dalam blok ini terdapat sebuah yaitu blok *Complex Multiplier* yang berfungsi untuk melakukan penjumlahan, pengurangan, perkalian dalam blok *butterfly*.

- **Twiddle Factor**

Blok ini berfungsi untuk menyimpan nilai *sinus* dan *cosinus* yang digunakan pada proses komputasi FFT.

- **Delay**

Blok ini berfungsi menunda data dengan waktu tertentu untuk mengontrol sinyal pada siklus *clock* yang sesuai.

- **Commutator (Switch)**

Blok ini dilakukan pertukaran data, supaya dapat mengurangi ukuran silikon *chip* dan mengurangi konsumsi daya saat implementasi.

- **Inverter**

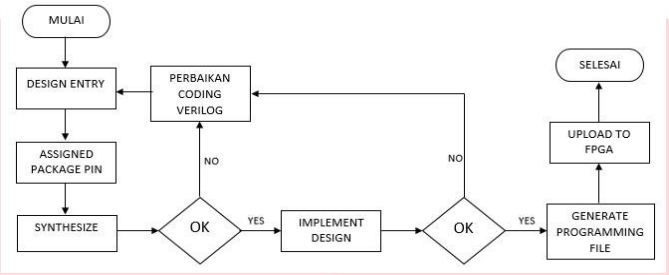
Pada perancangan MDC *radix* 2 ditambahkan blok *inverter*. Blok ini digunakan untuk mengembalikan nilai FFT dari domain negatif karena pengaruh dari nilai imajiner.

- **Data Out**

Dalam blok ini dilakukakan pemotongan bit akhir, pembulatan dan pengurutan data. Hal ini dilakukan untuk mendapatkan hasil FFT yang sesuai dengan *input* FFT.

3.4. Implementasi FPGA

Dalam Implementasi sistem terbagi beberapa tahap. Tahapan alur implementasi sistem ditunjukkan dalam gambar berikut ini.



Gambar 3.3 Diagram alir Implementasi Sistem

3.4.1. Design Entry

Design Entry yaitu menambahkan sumber kode Verilog yang telah dirancang kedalam Xilinx 14.5

3.4.2. Assigned Package Pin

Pada tahap ini dilakukan pemetaan *port input/output*, yaitu atau dari modul yang dirancang pada *port input/output* yang disediakan pada *evalution board* FPGA Spartan 6.

3.4.3. Synthesize

Pada tahap ini dilakukan *synthesize* pada rancangan, yaitu menerjemahkan kode Verilog ke dalam daftar jaringan (*netlist*) dan gerbang dasar pada FPGA.

3.4.4. Implement Design

Setelah melakukan proses sintesis, selanjutnya dilakukan proses *implement Design*. *Implement design* dilakukan dengan cara :

a. *Translate*, yaitu menerjemahkan *netlist* dan blok dasar berdasarkan *constraint*nya.
b. *Map*, yaitu pemetaan rangkaian ke blok Xilinx FPGA, gerbang logika dan distribusinya.
c. *Place* and *Route*, yaitu penempatan hasil pemetaan pada blok dan gerbang yang dimaksud pada *floorplan*.

3.4.5. Generate Programming File

Pada tahap ini dilakukan *Generate Programming File*, yaitu membuat *file* **.bit** dari rancangan FFT kemudian akan dikirimkan pada *board* FPGA melalui *interface* JTAG FPGA.

3.4.6. Upload to FPGA

Hasil *file* dari proses sebelumnya yang berupa **.bit** kemudian akan di *load* ke perangkat FPGA melalui rantai JTAG.

4. Simulasi dan Hasil Implementasi

4.1. Sintesis

Setelah dilakukan perancangan dan diuji melalui simulasi, maka dilakukan síntesis, hasil síntesis dapat diperoleh beberapa parameter yang dibutuhkan dalam implementasi pada FPGA. Berikut adalah tabel penggunaan *resource* rancangan pada FPGA.

Tabel 4.1 Hasil Sintesis Perancangan Algoritma FFT

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	1975	54576	3%
Number of Slice LUTs	5150	27288	18%
Number of fully used LUT-FF pairs	1786	5339	33%
Number of bonded IOBs	23	218	10%
Number of BUFG/BUFGCTRL/BUFHCEs	2	16	12%

Selanjutnya hasil pemakaian *resource* pada penelitian ini yang ditampilkan pada tabel 4.1 akan dibandingkan dengan penelitian sebelumnya [14] untuk melihat perbedaanya. Berikut tabel perbandingan utilisasi *resource*nya.

Tabel 4.2 perbandingan utilisasi *resource*

Logic Utilization	Proposed	[14]
Number of Slice Registers	1975	2411
Number of bonded IOBs	23	75
Delay Process	22900 ns	35400 ns
Jumlah Clock	2290 clock	3540 clock

Utilization Number of Slice Registers untuk penggunaan FPGA Atlys Spartan 6

- Proposed = $\frac{19\,75}{54576} \times 100\ \% = 3{,}6\ \%$
- [14] = $\frac{2411}{54576} \times 100\ \% = 4{,}4\ \%$

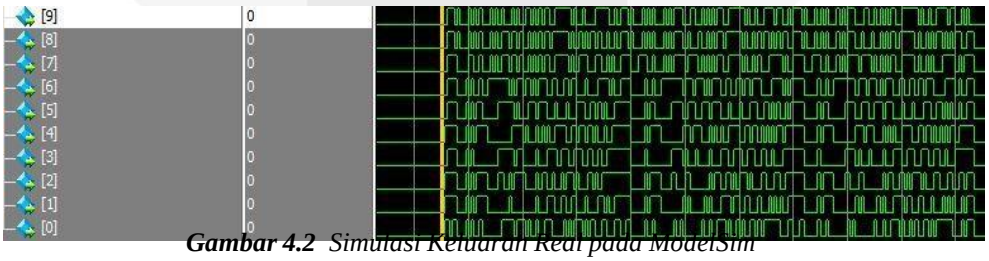
Utilization Number of bonded IOBs untuk penggunaan FPGA Atlys Spartan 6

- Proposed = $\frac{23}{218} \times 100\ \% = 10{,}55\ \%$
- [14] = $\frac{75}{218} \times 100\ \% = 34{,}4\ \%$

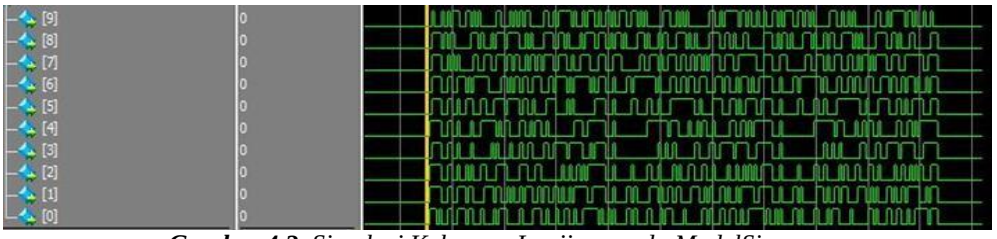
Dari tabel 4.2 diatas terlihat bahwa perancangan FFT menggunakan teknik MDC yang merupakan salah satu varian dari *pipeline architecture* selain menghemat memori, penggunaan arsitektur tersebut juga menghasilkan kecepatan komputasi yang lebih tinggi dari perancangan sebelumnya [14].

4.2. Simulasi ModelSim

Untuk menguji sistem yang telah dirancang, digunakan *Test bench* untuk membangkitkan deretan bit tertentu. *Test bench* bekerja pada clock *rising edge*, artinya sistem akan aktif ketika *clock* berubah dari '0' ke '1'.



***Gambar 4.2** Simulasi Keluaran Real pada ModelSim*

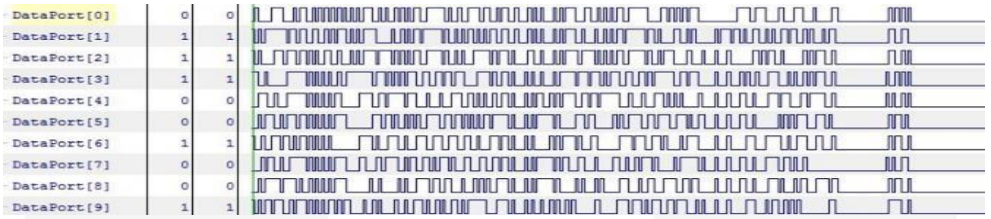


Gambar 4.3 *Simulasi Keluaran Imajiner pada ModelSim*

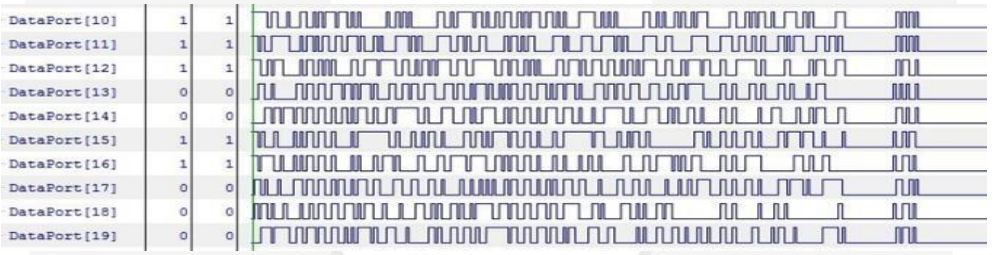
Dapat dilihat pada grafik gambar 4.2 dan 4.3 bahwa data keluaran menghasilkan panjang data 10 bit yang ditampilkan perbitnya baik hasil real maupun hasil imajiner. Keseluruhan proses dari awal sampai didapatkan hasil keluaran dari komputasi FFT menghasilkan *delay process* sebesar 22900 ns atau 2290 *clock*.

4.3. Hasil Implementasi pada Chipscope

Untuk melihat hasil hasil implementasi digunakan *software* chipscope yang terdapat Xilinx. Pada pengujian menggunakan *software* chipscope, untuk membangkitkan deretan bit secara acak pada periode waktu tertentu digunakan *signal generator*. Pada *prototype signal generator* dibangkitkan sinyal dengan kenaikan 1 bit selama 10 *clock* atau 100 ns. Proses pembangkitaan sinyal ini akan terus dilakukan selama *clock* dari sistem juga terus berjalan, dan kondisi *clock*nya adalah *rising edge*.



Gambar 4.4 *Simulasi Keluaran Real Chipscope*



Gambar 4.5 *Keluaran imajiner Chipscope*

Pada gambar 4.4 dan 4.5 adalah hasil dari rancangan yang sudah terimplementasi pada board FPGA. Dengan hasil yang tidak beda jauh dengan saat simulasi pada software ModelSim yaitu hasil keluaran untuk Real dan Imajiner menghasilkan panjang data 10 bit dengan urutan membaca nilai bit nya dimulai dari bit paling bawah ke bit paling atas.

5. Kesimpulan

Dari hasil pengujian dan analisis yang telah dilakukan pada tugas akhir ini dapat diambil beberapa kesimpulan sebagai berikut:

1. Perancangan Algoritma FFT menggunakan teknik MDC sudah dapat dilakukan sampai dengan tahap simulasi fungsional sistem.
2. Teknik *Multipath Delay Commutator* merupakan solusi implementasi Algoritma FFT dalam hal efisiensi memori dan peningkatan kecepatan sistem.
3. Perancangan prototype Algoritma FFT menggunakan teknik *Multipath Delay Commutator* (MDC) telah berhasil dilakukan dan di implementasikan pada board ATLYS Spartan-6. dari hasil simulasi, sistem dapat menjalankan fungsi dengan total waktu proses 22900 ns atau 2290 clock.

4. Sistem yang dihasilkan memiliki periode minimum 15,895 ns dan frekuensi kerja dibawah frekuensi kerja FPGA Spartan-6, yaitu 62,914 MHz. dengan frekuensi tersebut yang masih dibawah ketersediaan *resources* pada FPGA, maka sistem dapat di implementasikan pada *board*.
5. Hasil perancangan berhasil disintesiskan dan menghasilkan *resources* Algoritma FFT adalah 6% *Number of Slices Registers*, 25% *Number of Slice LUTs*, 43% *Number of fully used LUT-FF pairs*, 10% *Number of bonded IOBs*, 12% Number *of BUFG/BUFGCTRL/BUFHCEs*. Dengan presentase *Slice LUTs* yang lebih kecil daripada *LUT-FF* maka *prototype* yang dirancang merupakan rangkaian *Combinational logic.*

Daftar Pustaka

[1] Amjadha. A, E. Konguvel and J. Raja, 2014, "*Design of Multipath Delay Commutator Architecture based FFT Processor for 4th generation system*", International Journal of Computer Applications, Vol 89 no.12.

[2] Darlis, Denny, 2010, "*Perancangan dan Implementasi Prosesor OFDM Baseband untuk Prototipe modem PLC pada FPGA*", Program Pasca Sarjana teknik Telekomunikasi, Institut Teknologi Telkom.

[3] Karthick. S, Dhandapani. Dr.S., 2014, "*Parallel-Pipelined radix-6z Multipath Delay Commutator FFT Architectures*", International Journal of Emerging Technology and Advanced Engineering.

[4] Proakis, John G. dan Manolakis, Dimitris G., "*Pemrosesan Sinyal Digital*", Edisi Bahasa Indonesia Jilid 1, Prenhallindo, Jakarta. 1997.

[5] Syafiq irsyadi, Muh., 2007, "*Desain dan implementasi 2k Pipeline FFT-IFFT core untuk DVB-T*", Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

[6] Chang, K.C., 1999, "*Digital Systems Design with VHDL and Synthesis*", Matt Loeb, USA.

[7] Wada, Tom, 2006, "*64 point Fast Fourier Transform Circuit*". www.ie-u-ryuku.ac.id (diakses terakhir 28 Juli 2016).

[8] Peng, Wei. Xiang, Shen. Dan Lin, Fang., 2008, "*64 Points FFT processor*". IEEE

[9] http://de.lab.telkomuniversity.ac.id (diakses terakhir 28 Juli 2016).

[10] Khairuddin, Labib Ahmad., 2011, "*Perancangan dan Implementasi Prosesor FFT 256 titik-OFDM Baseband berbasis pengkodean VHDL pada FPGA*". IT Telkom.

[11] H. Gerez, Sabih, 2014, "*Pipeline Implementations of the Fast Fourier Transform (FFT)*", University of Twente.

[12] Stojanovic, Vladimir, 2006, "*course materials for 6.973 Communication System Design*", MIT OpenCourseWare.

[13] S. He and M. Torkelson, 1996, "*A new approach to pipeline FFT processor*" in Proceedings of the 10th International Parallel Processing Symposium (IPPS '96), pp. 766–770, Honolulu, Hawaii, USA.

[14] Nova, Wielda Safitri., 2008, "*Desain dan Sintesis Arsitektur Hardware FFT 64 titik berbasis bahasa pemrograman VHDL*", IT Telkom.

[15] Basha, T.S.Ghouse. Suneetha, L, 2014, "*Implementation of High Speed MDC FFT/IFFT Processor for MIMO-OFDM System*", International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering.

[16] Park, Hyun-cheol. Kyungki-do. Jung, Yon-ho. Kim, Jae-seok. Kyungki-do. Tak, Youn-ji. Park, Jun-hyun , 2001, "*Fast Fourier Transform Processor using High Speed Area-Efficient Algorithm*", Patent Application Publication Park et al, United States.

[17] M. Joshi, Shubhangi, 2015, "*FFT Architectures*", Sathyabhama University, International Journal of Computer Applications Volume 116-No.7.

[18] K. Malathy. Rabi, B. Justus, 2016, "A Novel VLSI Based Radix-2 Single Path Delay Commutator (R2SDC) FFT Architecture Design", Indian Journal of Science an Technology, vol 9(11).