IMPLEMENTASI IFFT 64 TITIK MENGGUNAKAN RADIKS-4 PADA FPGA

IMPLEMENTATION OF 64 POINT IFFT USING RADIKS-4 ON FPGA

Khairul Anas[1], Rita Purnamasari, S.T., M.T.[2], Estananto, S.T., M.Sc., M.B.A.[3]
[1,2]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
[3]Prodi S1 Teknik Elektro, Fakultas Teknik Elektro, Universitas Telkom
[1]khairulanas@student.telkomuniversity.ac.id, [2]ritapurnamasari@telkomuniversity.ac.id,
[3]estananto@telkomuniversity.ac.id

Abstrak

IFFT (*Inverse Fast Fourier Transform*) adalah algoritma komputasional yang merupakan invers dari FFT (*Fast Fourier Transform*). Metode IFFT tersebut berfungsi sebagai algoritma untuk mempercepat perhitungan dari IDFT (*Inverse Discrete Fourier Transform*). Dengan adanya IFFT, perhitungan dari IDFT dapat berkurang secara signifikan sehingga perhitungannya menjadi sederhana dan efisien dalam implementasinya. Beberapa implementasinya adalah pada bidang kesehatan, navigasi, telekomunikasi, dan pengolahan suara.

Pada tugas akhir ini, penulis telah merancang sistem IFFT 64 titik dengan menggunakan algoritma radiks-4 dan berhasil diimplementasikan pada FPGA Altera Cyclone II – EP2C20F484C7. Implementasi ini bertujuan untuk membuktikan apakah desain sistem ini dapat dirancang pada perangkat keras FPGA. Sistem ini dirancang menggunakan VHDL, yaitu sebagai bahasa yang bertujuan untuk mengkodekan blok- blok dari sistem yang telah dibuat. Rancangan VHDL nya disimulasikan terlebih dahulu menggunakan modelsim sebelum diimplementasikan ke FPGA. Hasil simulasi tersebut akan dibandingkan dengan simulasi pada MATLAB dan perhitungan manual pada Microsoft Excel.

Setelah simulasi dilakukan dan didapatkan hasilnya, maka dilanjutkan dengan mensintesis desain VHDL ke dalam perangkat keras FPGA dengan menggunakan fitur *Programmer* pada *software* Quartus. Jumlah resource yang dibutuhkan berdasarkan hasil sintesis ke hardware FPGA adalah *total logic elements* berjumlah 9648 (51%), *total combinational functions* berjumlah 8535 (46%), *dedicated logic registers* berjumlah 4406 (23%), *total pins* berjumlah 65 (21%), *total memory bits* sebesar 71680 (30%), dan *embedded multiplier 9-bit elements* berjumlah 52(100%). Hasil ini menunjukkan bahwa sistem IFFT 64 titik menggunakan radiks-4 dapat diimplementasikan ke perangkat keras FPGA Altera Cyclone II – EP2C20F484C7.

Kata kunci : IDFT, IFFT, VHDL, FPGA

Abstract

IFFT (Inverse Fast Fourier Transform) is a computational algorithm which is an inverse of FFT (Fast Fourier Transform). The IFFT method serves as an algorithm to speed up calculations from IDFT (Inverse Discrete Fourier Transform). With the IFFT, the calculation of IDFT can be significantly reduced so the calculation becomes simple and efficient in its implementation. Some of the implementation is in the field of health, navigation, telecommunications, sound processing.

In this final project, the author has designed the 64 point IFFT system using the radix-4 algorithm and successfully implemented on FPGA Altera Cyclone II - EP2C20F484C7. This implementation aims to prove whether the design of this system can be designed on FPGA hardware. This system is designed using VHDL, which is a language that aims to encode the blocks of a system that has been created. This design simulation uses MODELSIM as a pre-implementation treatment to FPGA hardware. The simulation results will be compared with the simulation on MATLAB and manual calculations in Microsoft Excel.

After the simulation is done and the results are obtained, it is continued by synthesizing the VHDL design into FPGA Altera Cyclone II - EP2C20F484C7 using Programmer features in Quartus software. The total number of resources required from the synthesis result is total logic elements, total combinational functions, dedicated logic registers totaling, total pins totaling, total memory bits, and embedded multiplier 9-

bit elements. These results indicate that the IFFT 64 point system using radix-4 algorithm can be implemented to FPGA Altera Cyclone II - EP2C20F484C7.

Keywords **: IDFT, IFFT, VHDL, FPGA**

1. Pendahuluan

Perkembangan *Digital Signal Processing* (DSP) kini semakin pesat yaitu dengan diterapkannya DSP yang semakin luas, seperti pada bidang kesehatan, navigasi, telekomunikasi, pengolahan suara dan gambar serta pengolahan video. Peralatan pada bidang-bidang tersebut kini semakin canggih dengan adanya rangkaian digital dengan komputasi yang relatif cepat, kompleksitas rangkaian yang lebih sederhana, dan lebih murah. Hal ini berbeda pada 30 tahun terakhir dimana rangkaian digital masih relatif besar dan mahal.

Salah satu teknik dari DSP yang sekarang masih digunakan adalah *Inverse Fast Fourier Transform* (IFFT). Pada dasarnya, IFFT merupakan *inverse* atau kebalikan dari teknik FFT, yaitu sebuah algoritma yang sangat efisien dalam formulasi sehingga dapat mempercepat perhitungan pada teknik *Inverse Discrete Fourier Transform* (DFT). DFT adalah teknik transformasi matematis dengan mengubah sinyal diskrit dari domain waktu ke dalam domain frekuensi. IFFT disebut juga algoritma cepat dalam perhitungan IDFT karena perhitungan IFFT mencapai Nlog2N proses, sedangkan IDFT hanya mencapai N2 proses. Pengolahan sinyal dapat berlangsung dengan cepat menggunakan teknik IFFT.

Algoritma IFFT radiks-4 merupakan salah satu algoritma generalisasi dari IFFT radiks-2, sehingga seluruh sifat yang dimiliki FFT radiks-2 juga dimiliki oleh IFFT radiks-4. IFFT radiks-2 adalah salah satu proses efisien dalam perhitungan IDFT dengan membagi formulasi dasar dari DFT menjadi 2 proses penjumlahan dan IFFT radiks-4 dengan pembagian menjadi 4 proses penjumlahan. FFT radiks-4 pada dasarnya merupakan dua tahap FFT radiks-2 untuk DFT 4 titik. Pada DFT dengan titik yang banyak dan merupakan kelipatan 4, komputasi akan menjadi lebih efisien jika menggunakan FFT radiks-4 daripada menggunakan FFT radiks-2.

Penulis juga meninjau penelitian- penelitian sebelumnya. Desia Ilmina Suprapto (2008) dan Siska Cucu Maesa (2011), kedua orang tersebut melakukan penelitian yang menggunakan FFT dan IFFT 64 titik sebagai metodenya[2][9]. Dan yang berbeda adalah algoritma radiks yang digunakan. Desia Ilmina Suprapto (2008) menggunakan radiks-2 IFFT dan Siska Cucu Maesa (2011) menggunakan radiks-8 FFT/IFFT[2][9]. Kedua penelitian tersebut dilakukan dengan tujuan untuk mendapatkan hasil sintesa dengan kebutuhan jumlah slice, IOB, LUT. Flip- Flop, GCLK, FIFO/RAM, dan DSP untuk N titik. Serta mendapatkan hasil delay yang terjadi pada saat simulasi.

Pada tugas akhir ini akan di implementasikan IFFT menggunakan radiks-4 ke *board* FPGA (*Field Programmable Gate Array*). IFFT dengan input 64 titik di desain menggunakan bahasa pemrograman VHDL (VHSIC *Hardware Description Language*) dan disintesa ke *hardware*.

2. Dasar Teori

2.1. FFT/ IFFT[1]

FFT (*Fast Fourier Transform*) merupakan algoritma komputasional untuk mempercepat dan mempersingkat perhitungan pada teknik DFT (*Discrete Fourier Transform*). DFT adalah prosedur matematis yang digunakan untuk menentukan harmonik, atau frekuensi serta isi dari urutan sinyal diskrit. Berikut rumus dasar dari DFT :

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{j2\pi kn}{N}} = \sum_{n=0}^{N-1} x(n) {W_N}^{kn} \qquad (1)$$

Transformasi sinyal dari domain waktu ke domain frekuensi dapat digunakan prosedur DFT. Proses ini dapat dibalik untuk mendapatkan sinyal pada domain waktu dengan menggunakan IDFT (*Inverse Discrete Fourier Transform*). Berikut rumus perhitungan dari IDFT :

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j2\pi\left(\frac{k}{N}\right)n} = \frac{1}{N} \sum_{k=0}^{N-1} X(k) {W_N}^{-kn} \qquad (2)$$

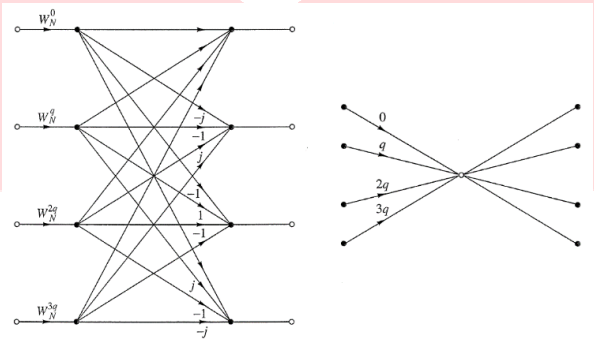
dimana $W_N = e^{-j\frac{2\pi}{N}}$, dan W_N adalah *twiddle factor*.

2.2. Algoritma Cooley-Tukey[1][3]

Algoritma ini ditemukan oleh J.W. Cooley and John Tukey, dan kemudian diberi nama berdasarkan nama mereka Cooley-Tukey. Cooley-Tukey FFT yang terkenal adalah transformasi dengan panjang N sebagai bilangan berbasis r, sebagai contoh, N= r^v. Algoritma ini sering disebut dengan algoritma radix-r.

Satu hal yang membedakan algoritma ini dengan algoritma FFT lainnya adalah bahwa nilai faktor r dapat ditentukan untuk jumlah N tertentu. Hal ini memungkinkan penggunaan sebuah algoritma radiks-r dimana N = rv . Untuk N=64 dengan menggunakan radix 4 maka dapat diperoleh nilai v=3 (64 = 43). Algoritma yang paling popular digunakan adalah algoritma dengan basis r=2 atau r=4. Dasar perhitungan dari kedua radix ini dapat diimplementasikan tanpa harus melakukan perkalian. Penggunaan algoritma ini akan sangat membantu pengurangan proses perhitungan DFT.

Perbedaan antara orde metode DFT dan FFT untuk nilai N yang sama, yaitu N2 dan Nlog2N, sehingga sangat mempengaruhi waktu yang dibutuhkan untuk menghitung.



Gambar 1. Diagram kupu-kupu FFt radix-4 DIF[3]

3. Perancangan Sistem

Perancangan sistem ini menggunakan algoritma radix-4 dengan beberapa tindakan seperti pada gambar 3.1. Dimulai dari identifikasi dan spesifikasi sistem IFFT untuk 64 titik. Selanjutnya, melakukan perhitungan manual IFFT 64 titik menggunakan algoritma radix-4 pada rumus dasar IFFT. Setelah perhitungan manual dilakukan maka dibuatlah desain blok sistem dari IFFT 64 titik. Lalu, melakukan simulasi sistem IFFT 64 titik pada perangkat lunak MATLAB dan perhitungan manual menggunakan perangkat lunak Ms. Excel serta verifikasi dari hasil kedua simulasi tersebut. Langkah selanjutnya yaitu simulasi blok- blok sistem IFFT pada quartus dengan menggunakan bahasa VHDL dan dilakukan penggabungan blok-blok sistem tersebut. Lalu simulasi kembali sistem hasil dari penggabungan blok- blok sistem dan verifikasi dari hasil pada Quartus dan Matlab. Hasil desain tersebut di sintesa ke *board* FPGA menggunakan Quartus.

Untuk perancangan IFFT 64 titik dengan menggunakan algoritma radiks-4 didapatkan 3 tahap komputasi menggunakan radiks-4.

3.1. Perhitungan Sistem IFFT 64 Titik

Berikut ini penurunan rumus dari IFFT 64 titik radiks-4.

$$X(n) = \frac{1}{N}\sum_{k=0}^{N-1} x(k)\, W_N^{-kn}$$

$$= \frac{1}{N}\left\{\sum_{k=0}^{N/4-1} x(k)\, W_N^{-kn} + \sum_{k=0}^{N/2-1} x(k)\, W_N^{-kn} + \sum_{k=0}^{3N/4-1} x(k)\, W_N^{-kn} + \sum_{k=0}^{N-1} x(k)\, W_N^{-kn}\right\}$$

$$= \frac{1}{N}\left\{\sum_{k=0}^{N/4-1} x(k)\, W_N^{-kn} + W_N^{-Nk/4} \sum_{k=0}^{N/2-1} x\left(k+\frac{N}{4}\right) W_N^{-kn} + W_N^{-Nk/2} \sum_{k=0}^{3N/4-1} x\left(k+\frac{N}{2}\right) W_N^{-kn} + W_N^{-3Nk/4} \sum_{k=0}^{N-1} x\left(k+\frac{3N}{4}\right) W_N^{-kn}\right\} \quad (3)$$

Berdasarkan definisi *twiddle factors*, dari penurunan rumus di atas didapatkan rumus perhitungan dari IFFT 64 titik radiks-4

$W_N^{-kN/4} = (j)^k$, $W_N^{-kN/2} = (-1)^k$, $W_N^{-kN/4} = (-j)^k$

Jadi,

$$X(n) = \frac{1}{N}\left\{x(k) + (j)^k\, x\left(k+\frac{N}{4}\right) + (-1)^k\, x\left(k+\frac{N}{2}\right) + (-j)^k x\left(k+\frac{3N}{4}\right)\right\} W_N^{-kn} \quad (4)$$

Untuk mengkonversi rumus di atas menjadi N/4 –*point* IFFT, maka dibagi urutan IFFT menjadi empat N/4 –point, X(4n), X(4n+1), X(4n+2), X(4n+3), k=0, 1, ...,N/4.

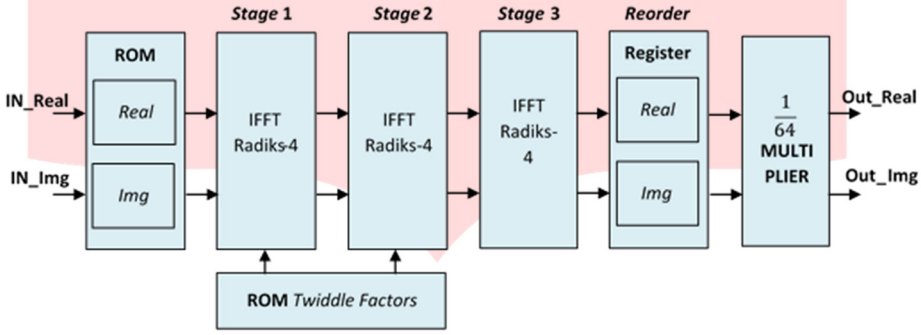
$$X(4n) = \frac{1}{N}\left\{x(k) + x\left(k+\frac{N}{4}\right) + x\left(k+\frac{N}{2}\right) + x\left(k+\frac{3N}{4}\right)\right\} W_N^0 W_{N/4}^{-kn} \quad (5)$$

$$X(4n+1) = \frac{1}{N}\left\{x(k) + jx\left(k+\frac{N}{4}\right) - x\left(k+\frac{N}{2}\right) - jx\left(k+\frac{3N}{4}\right)\right\} W_N^{-n} W_{N/4}^{-kn} \quad (6)$$

$$X(4n+2) = \frac{1}{N}\left\{x(k) - x\left(k+\frac{N}{4}\right) + x\left(k+\frac{N}{2}\right) - x\left(k+\frac{3N}{4}\right)\right\} W_N^{-2n} W_{N/4}^{-kn} \quad (7)$$

$$X(4n+3) = \frac{1}{N}\left\{x(k) - jx\left(k+\frac{N}{4}\right) - x\left(k+\frac{N}{2}\right) + jx\left(k+\frac{3N}{4}\right)\right\} W_N^{-3n} W_{N/4}^{-kn} \quad (8)$$

3.2. Blok Sistem IFFT 64 Titik

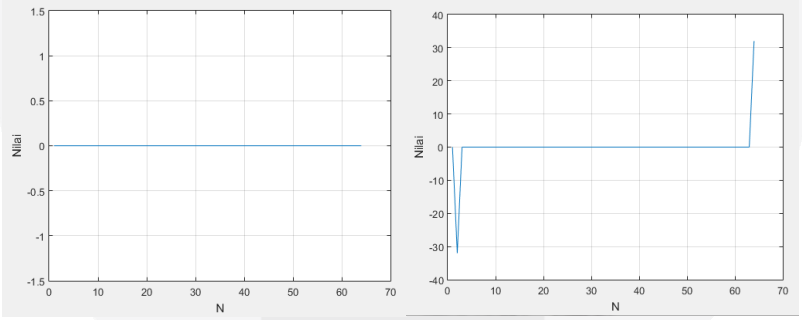


Gambar 2. Blok Sistem IFFT 64 Titik Radiks-4

4. Simulasi IFFT 64 Titik Radiks-4

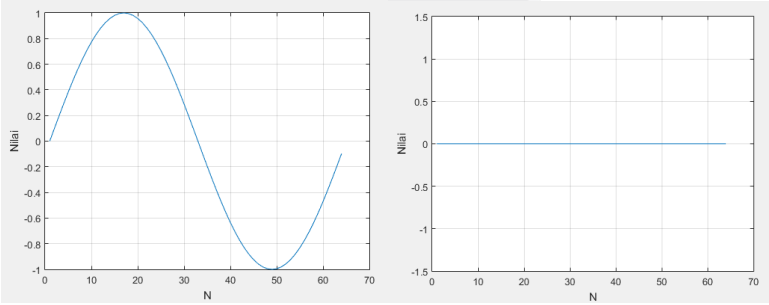
4.1. Input Sistem IFFT 64 Titik Radiks-4

Di dalam sistem ini, input yang digunakan penulis yaitu dari hasil keluaran pada Tugas Akhir sebelumnya. Rahadian Susandi (2017) membuat penelitian yaitu FFT 64 titik menggunakan radiks 4 yang menggunakan *input* berupa sinus dan menghasilkan keluaran berupa bilangan kompleks[10]. Dan hasil keluaran bilangan kompleks tersebut penulis ambil sebagai *input* dari sistem dengan tujuan untuk membuktikan bahwa data yang diproses dengan FFT dapat dikembalikan ke bentuk semula dengan menggunakan IFFT.



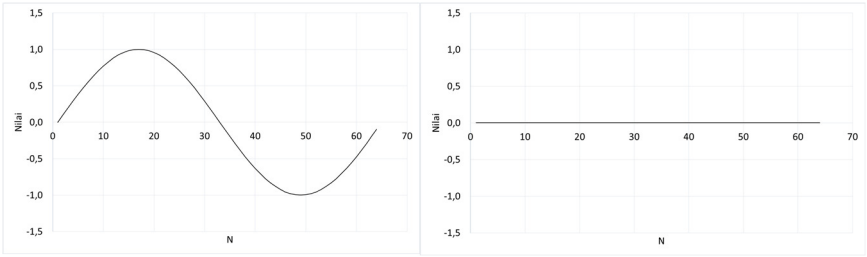
Gambar 3. Plot Grafik *Input Real* dan *Imaginer* pada Matlab

4.2. Simulasi Sistem IFFT 64 Titik Radiks-4 di MATLAB



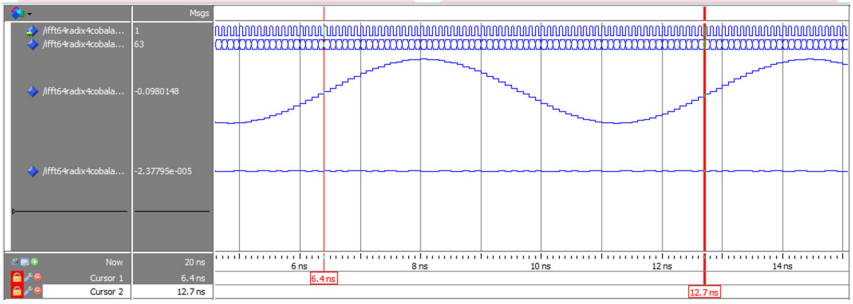
Gambar 4. Plot Grafik *Output Real* dan *Imaginer* pada Matlab

4.3. Simulasi Sistem IFFT 64 Titik Radiks-4 di Microsoft Excel



Gambar 5. Plot Grafik *Output Real* dan *Imaginer* pada Microsoft Excel

4.4. Simulasi Sistem IFFT 64 Titik Radiks-4 di Modelsim



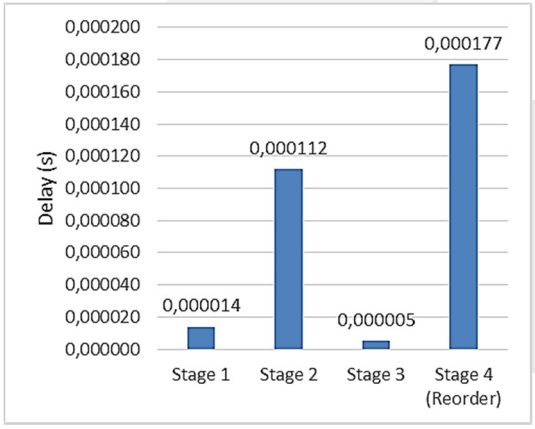
Gambar 6. Plot Grafik *Output Real* dan *Imaginer* pada Modelsim

4.5. Hasil Simulasi Sistem IFFT 64 Titik Radiks-4

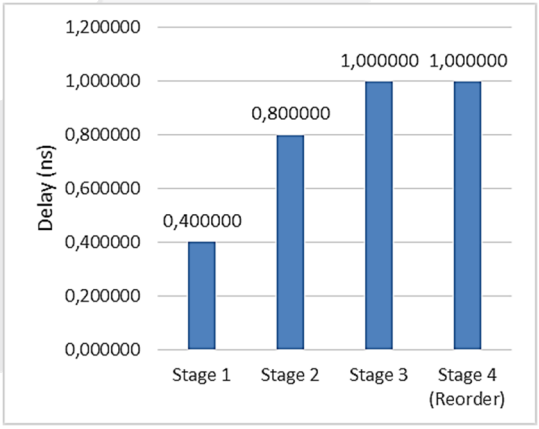
Dapat dilihat *delay* proses tiap *stage* di tabel (1) di bawah ini. *Delay* proses pada *stage* 1, *stage* 2, dan *reorder* lebih besar daripada *delay* proses pada *stage* 3. Hal tersebut dapat terjadi karena pada *stage* 1, *stage* 2, dan *stage reorder* terdapat komponen perkalian dengan *twiddle factor* atau pembagian dengan suatu angka. Sedangkan pada *stage* 3 tidak terdapat perkalian dengan *twiddle factor*.

Tabel 1. Perbandingan *Delay* Proses Tiap *Stage* Sistem IFFT 64 Titik Radiks-4

	Stage 1	*Stage* 2	*Stage* 3	*Stage* 4 (*Reorder*)
Matlab (s)	0,000014	0,000112	0,000005	0,000177
Modelsim (ns)	0,400000	0,800000	1,000000	1,000000



Gambar 7. Grafik Data *Delay Proses* di Matlab



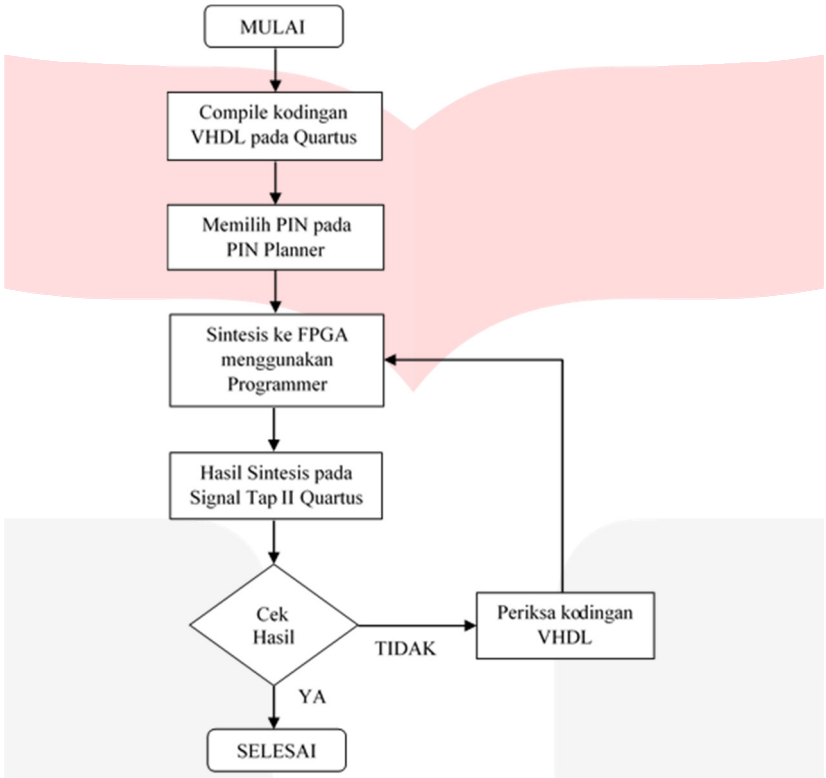
Gambar 8. Grafik Data *Delay Proses* di Modelsim

4.6. Sintesa dan Implementasi Sistem IFFT 64 Titik Radiks-4 pada FPGA

4.6.1. Perancangan Sistem IFFT 64 Titik Radiks-4 di Quartus

Untuk perancangan sistem di Quartus, terdapat 2 bagian file kodingan VHDL nya yaitu ifft64_radiks4.vhd dan ifft_radiks4_formulation.vhd.

1.) *Top Level Entity* (ifft64_radiks4.vhd), berisi kodingan untuk membuat *port port* yang akan dikeluarkan di FPGA yaitu seperti fpga_ifft_real, fpga_ifft_imajiner, dan fpga_clk. Dan juga untuk menghubungkan *file* perhitungan yang berisi *formula* perumusan dari IFFT 64 titik radiks-4.
2.) *File* perhitungan dari IFFT 64 titik radiks-4 (ifft_radiks4_formulation.vhd), untuk menghitung sistem IFFT 64 titik radiks-4 dengan kodingan yang hampir sama dengan simulasi sebelumnya pada ModelSim. Perbedaannya adalah pada Modelsim menggunakan jenis bilangan *Integer* sedangkan pada *file* ini menggunakan jenis bilangan *signed* dan *unsigned* pada *library* IEEE.std_logic_arith.all..



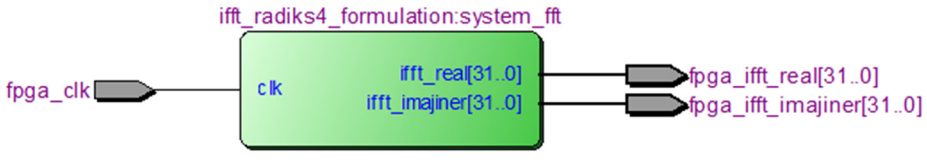
Gambar 9. *Flowchart* Implementasi ke hardware FPGA

4.6.2. *Design Entry*

Langkah pertama pada implementasi yaitu memasukkan kodingan VHDL ke project yang dibuat dan compile kodingan terlebih dahulu. Setelah proses compile selesai, maka kita dapat melihat jumlah penggunaan resource di tabel (3) di bawah ini.

Tabel 3. Jumlah Penggunaan *Resource*

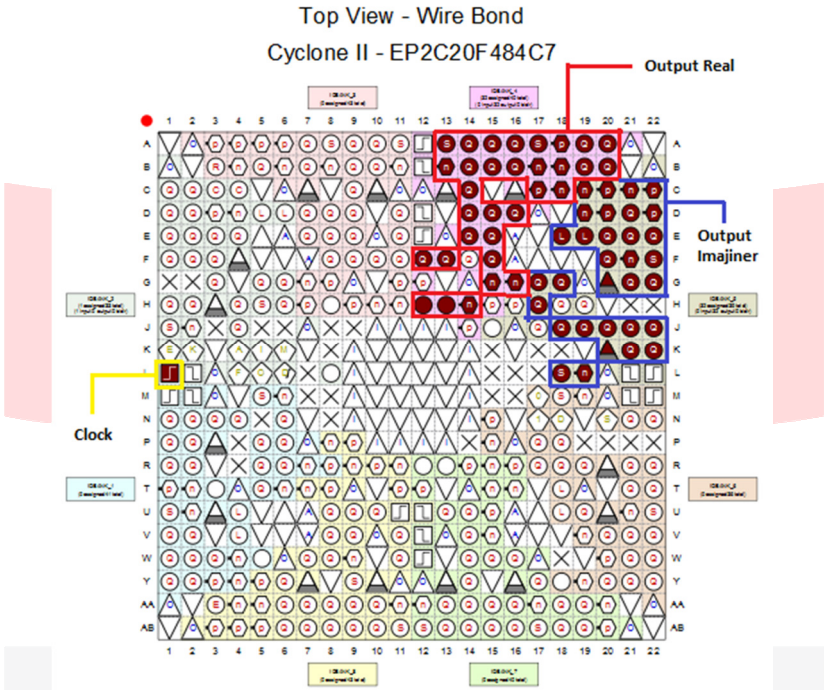
Resource	Penggunaan *Resource*
Total Logic Elements	9648 / 18752 (51%)
Total Memory Bits	71680 / 239616 (30%)
Total Combinational Functions	8535 / 18752 (46%)
Dedicated Logic Registers	4406 / 18752 (23%)
Total PINS	65 / 315 (21%)
Embedded Multiplier 9- bit Elements	52 / 52 (100%)



Gambar 10. Bentuk RTL dari sistem IFFT 64 titik Radiks-4

4.6.3. *Assigned Package* PIN (PIN *Planner*)

Pada tahap ini dilakukan pemilihan pin keluaran (*output*) real, imajiner dan *clock* sesuai dengan nama pin pada FPGA tersebut.



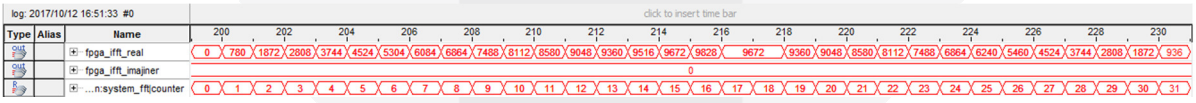
Gambar 11. Pemetaan PIN *Planner* pada Altera Cyclone II EP2C20F484C7

4.6.4. Programmer

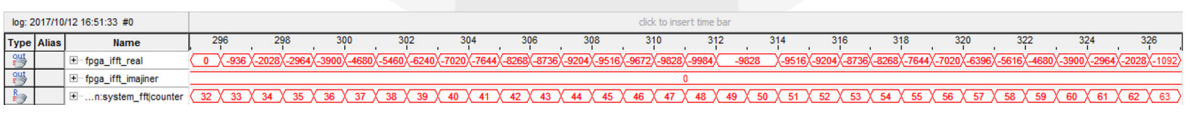
Setelah selesai melakukan penamaan dan penempatan *output* dan *clock* maka dilanjutkan dengan *compile* VHDL *coding* di Quartus agar penempatan pin di PIN *Planner* dapat dibaca oleh VHDL *coding*. Setelah *compile* selesai dilanjutkan dengan *Programmer*, yaitu *sintesis*/ memasukkan kodingan VHDL tersebut ke *board* FPGA. Klik *Start* untuk memulai pengiriman kodingan VHDL ke FPGA.

4.6.5. Signal Tap II Logic Analyzer

Setelah proses *transfer* ke FPGA selesai, saatnya menampilkan hasil keluaran dari sistem IFFT 64 Titik radiks-4. Untuk melihat hasil keluaran data nilai maupun grafik dapat ditampilkan di aplikasi Signal Tap II Logic Analyzer pada Quartus. Terdapat 2 hasil keluaran yaitu fpga_ifft_real untuk hasil keluaran real dan fpga_ifft_imajiner untuk hasil keluaran yang imaginer.



Gambar 12. Nilai Keluaran Sistem pada *Counter* 0 s.d. 31



Gambar 13. Nilai Keluaran Sistem pada *Counter* 32 s.d. 63

Gambar 14. Grafik Nilai Keluaran Sistem

5. Kesimpulan dan Saran

5.1. Kesimpulan

Berikut ini beberapa kesimpulan yang didapatkan penulis dari hasil sintesis dan implementasi sistem IFFT 64 titik radiks-4.

1.) Hasil keluaran sistem IFFT 64 titik radiks-4 dari perhitungan manual menggunakan Microsoft Excel sama dengan simulasi menggunakan MATLAB dan ModelSim.
2.) Menurut data hasil simulasi, hasil keluaran IFFT 64 titik radiks-4 sesuai dengan masukan FFT 64 titik radiks 4.
3.) Jumlah resource yang dibutuhkan berdasarkan hasil sintesis ke hardware FPGA adalah *total logic elements* berjumlah 9648 (51%), *total combinational functions* berjumlah 8535 (46%), *dedicated logic registers* berjumlah 4406 (23%), *total pins* berjumlah 65 (21%), *total memory bits* sebesar 71680 (30%), dan *embedded multiplier 9-bit elements* berjumlah 52 (100%).
4.) Berdasarkan simulasi pada MATLAB dan ModelSim, didapatkan *delay* rata- rata sebesar 77 µs pada MATLAB dan 0,0008 µs pada ModelSim.
5.) Sistem IFFT 64 titik radiks-4 dapat diimplementasikan pada *board* FPGA Altera Cyclone II EP2C20F484C7.

5.2. Saran

Adapun saran sebagai pengembangan dan kemajuan dari sistem ini yaitu :

1.) Untuk pengujian selanjutnya dapat memakai jenis jenis bilangan yang berbeda pada kodingan VHDL dengan harapan didapatkan jumlah *resource* dan *delay* proses yang lebih kecil.
2.) Untuk pengembangannya dapat menggunakan algoritma radix-4 atau kombinasi antara radix-4 dengan radix-8 atau radix-2 dengan jumlah masukan yang lebih besar dari sebelumnya.

Daftar Pustaka

[1] Manalu, Manoto J F. 2011. Perancangan dan Implementasi Prosesor I/FFT 512 titik Radix-8 pada FPGA. Tugas Akhir pada Institut Teknologi Telkom : Tidak Diterbitkan.
[2] Suprapto, Desia Ilmina. 2008. Desain dan Sintesis Arsitektur Hardware IFFT (Inverse Fast Fourier Transform) 64 titik berbasis Bahasa Pemrograman VHDL. Tugas Akhir pada Institut Teknologi Telkom : Tidak Diterbitkan.
[3] Chu, Eleanor, dan Alan George. 2000. Inside FFT Black Box: Serial & Parallel FFT Algorithms. Ontario, Canada.
[4] Irsyadi, Muh Syafiq. 2007. Desain dan Implementasi 2k Pipeline FFT-IFFT Core untuk DVB-T. Tugas Akhir pada Institut Teknologi Bandung : Tidak Diterbitkan.
[5] John G. Proakis and Dimitris G. Manolakis. Digital Signal Processing: Principles, Algorithms, and Applications, 4th edition, 2007.
[6] Pedroni, Volney A. Circuit Design With VHDL. MIT Press Cambridge, Massachusetts London, England, 2004.
[7] Wada, Tomohisa. The 10th International LSI Design Contest. The University of The Ryukyus. [Online] October 2006. Available: http://www.ie.uryukyu.ac.jp/~wada/design07/spec_e.html.
[8] The 15th International LSI Design Contest. The University of The Ryukyus. [Online] 2012. Available : http://www.lsi-contest.com/2012/shiyou_3-3.html
[9] Maesa, Siska Cucu. 2011. Perancangan dan Implementasi Sistem Multicarrier Code Division Multiple Access (MC-CDMA) menggunakan VHDL pada FPGA. Tugas Akhir pada Institut Teknologi Telkom : Tidak Diterbitkan.
[10] Susandi, Rahadian. 2017. *Implementasi FFT 64 Titik menggunakan Radiks-4 pada FPGA*. Tugas Akhir pada Universitas Telkom : Tidak Diterbitkan.
[11] J.J. Zhang; Z.H. Tang; R.P. Giddings; Q. Wu; W.L. Wang; B.Y. Cao; Q.W. Zhang; J.M. Tang. 2016. *Stage dependent DSP Operation Range Clipping-induced Bit Resolution Reductions of Full Parallel 64-point FFTs Incorporated in FPGA-based Optical OFDM Receivers*. Paper. IEEE
[12] *Cyclone II FPGA Starter Development Board Reference Manual*. Altera. [Online] 2006. Available : http://www.altera.com/