PERANCANGAN DAN SINTESIS ARSITEKTUR HARDWARE IFFT (INVERSE FAST FOURIER TRANSFORM) 32 TITIK BERBASIS BAHASA PEMROGRAMAN VHDL

Amalia Rizka Darmayanti[1], Achmad Hidayatno, S.T., M.T.[2], Darjat, S.T., M.T[2]

Jurusan Teknik Elektro, Fakultas Teknik, Universitas Diponegoro
Jln. Prof Sudharto, Tembalang, Semarang, Indonesia

Abstrak

Metode IFFT (Inverse Fast Fourier Transform) adalah inverse atau kebalikan dari FFT (Fast Fourier Transform), yang mana FFT merupakan metode untuk pemecahan sinyal diskret. IFFT merupakan algoritma komputasional yang cepat untuk menghitung IDFT (Inverse Discrete Fourier Transform). Subcarrier pada IFFT memiliki frekuensi harmonisasi kelipatan bulat dari frekuensi dasarnya seperti halnya komponen deret Fourier pada sinyal komposit. Salah satu penggunaan IFFT adalah pada sistem OFDM (Orthogonal Frekuensi Division Multiplexing), disini IFFT berperan sebagai modulator pada pemancar.

Pada tugas akhir ini akan dibuat perancangan struktur hardaware dari IFFT dengan mengkodekan setiap blok-blok dalam IFFT menggunakan bahasa VHDL . Rancangan sistem dengan VHDL ini akan memodelkan sistem sesuai dengan kebutuhan dari sistem IFFT dan mensimulasikan sebelum perangkat sintesis menterjemahkan rancangan dalam hardware secara nyata dengan ModelSim sebagai software pendukung . Dari hasil permodelan dan simulasi maka akan dilakukan sintesis pada tingkat hardware FPGA dengan Xilinx.

Sebagai input dari IFFT berupa sinyal diskret hasil keluaran kuantizer dan keluarannya akan didapatkan sinyal diskret dalam domain waktu. Diharapkan akan dapat dilihat hasil output dari IFFT. Hasil implementasi di FPGA akan dianalisis performansinya yaitu meliputi : parameter kinerja hasil rancangan, jumlah slice yang diperlukan, delay proses dan keberhasilan algoritma untuk perhitungan IFFT.

kata kunci : IFFT**,** Xilinx**,** FPGA**,** VHDL

[1] Mahasiswa Jurusan Teknik Elektro Undip
[2] Staf Pengajar Jurusan Teknik Elektro Undip

I. PENDAHULUAN

1.1 Latar Belakang

Pengolahan sinyal digital adalah suatu bagian dari sains dan teknik yang berkembang pesat selama 30 tahun terakhir. Perkembangan yang pesat ini merupakan hasil kemajuan teknologi komputer digital dan industri rangkaian terintegrasi. Pada perkembangan DSP (Digital Signal Processing) yang dapat mensintesis penjumlah sinyal termodulasi dengan akurat, maka Modulasi Multicarrier selalu sukses dilaksanakan pada dasawarsa terakhir ini. Salah satu metode yang digunakan untuk mengatasi kendala dalam modulasi multicarrier adalah metode IFFT.

IFFT merupakan invers atau kebalikan dari metode FFT. FFT adalah algoritma komputasional yang efisien untuk menghitung DFT (Discrete Fourier Transform) bila ukuran N adalah pangkat 2 dan bila pangkat 4. Pada saat pengolahan sinyal DFT mempunyai peran yang penting. Dapat dikatakan bahwa IFFT merupakan algoritma cepat untuk menghitung IDFT (Invers DFT). Algoritma cepat dalam IFFT adalah kemampuan menghitung lebih cepat jumlah perkalian kompleks pada perhitungan langsung yaitu N^2 menjadi (N/2) $\log_2 N$ perkalian. Metode IFFT ini akan memberikan perhitungan yang lebih efisien sehingga mempercepat proses sinyal.

Penelitian ini akan diaplikasikan algoritma IFFT 32 titik pada FPGA. Dengan melakukan permodelan sistem atau desain hardware pada IFFT, melakukan simulasi dan sintesis hardware pada FPGA dengan bahasa VHDL. Dari implementasi ini akan dapat dilihat hasil keluaran dari IFFT 32 titik.

1.2 Tujuan

Tujuan dari tugas akhir ini adalah merancang arsitektur hardware dan mensimulasikan algoritma IFFT, mensintesis hasil rancangan algoritma IFFT serta mendapatkan hasil sintesis pada IFFT.

1.3 Pembatasan Masalah

Batasan-batasan masalah yang digunakan dalam tugas akhir ini adalah :

1. Menggunakan algoritma IFFT Radiks 2.
2. Jumlah titik inputan dari IFFT (N) yang digunakan adalah 32 titik.
3. Menggunakan algoritma peruraian dalam frekuensi (*Decimation in Frequency*)
4. Bahasa pemrograman yang digunakan adalah VHDL (*VHSIC Hardware Description Language*), menggunakan software *ModelSim SE 6.0.*
5. Sintesa hardware yang menggunakan software *Xilinx ISE 7.1i.*

II. Dasar Teori

2.1 DFT (Discret Fourier Transform)

DFT bermanfaat sebagai metode numerik untuk menghitung transformasi Fourier dari suatu fungsi kontinu. Tiap-tiap barisan N titik ini dapat digambarkan secara tersusun mengelilingi sebuah lingkaran. Analisis frekuensi sinyal diskret merupakan yang paling cocok dilakukan dalam pengolahan sinyal digital. Untuk melakukan analisis frekuensi pada suatu sinyal waktu maka diperlukan konversi dari deret domain waktu ke deret domain frekuensi. Untuk mentransformasikan suatu deret dalam domain waktu x(n) dengan panjang L≤N menjadi suatu barisan dengan cuplikan-cuplikan frekuensi x(n) dengan panjang N maka diperlukan evaluasi transformasi Fourier $X(\omega)$ pada himpunan N sinyal diskret. Hal ini desebut dengan Transformasi Fourier Diskret (DFT) dari x(n). Rumus DFT adalah sebagai berikut :

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N} \qquad k = 0,1,\ldots,N-1 \qquad (2.1)$$

Sebaliknya untuk konversi dari deret doamain frekuensi X(n) ke deret domain waktu x(n) dinamakn DFT Invers (IDFT), rumusnya sebagai berikut :

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j2\pi nk/N} \qquad n = 0,1,\ldots,N-1 \qquad (2.2)$$

2.2 FFT (Fast Fourier Transform)

Secara mendasar, masalah komputasional untuk DFT adalah menghitung deret X(k) dari jumlah bernilai kompleks N yang diberikan deret data x(n) lain dengan panjang N, sesuai dengan rumus

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \qquad 0 \le k \le N-1 \qquad (2.3)$$

Dengan

Pada algoritma FFT radiks-2 ada 2 struktur algoritma penting, yaitu algoritma penguraian dalam waktu dan algoritma penguraian dalam frekuensi.

1. Algoritma penguraian dalam waktu.

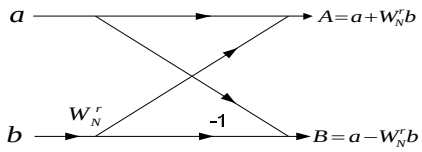
Untuk algoritma penguraian dalam waktu ini biasanya masukannya disimpan dalam orde kebalikan bit dan keluarannya berupa orde natural. Dan jika dilihat dari butterfly masukannya memiliki pola teretentu. Kita tinjau FFT 8-titik, dengan deretan data awal adalah *x(0) x(1) x(2) x(3) x(4) x(5) x(6) x(7).* Setelah dilakukan penguraian pertama maka akan menghasilkan suatu deret sebagai berikut :

x(0) x(2) x(4) x(6)} {x(1) x(3) x(5) x(7)}

Kemudian dilakukan penguraian kembali seperti dibawah ini :

[{x(0) x(4)} {x(2) x(6)}] [{x(1) x(5)} {x(3) x(7)}]

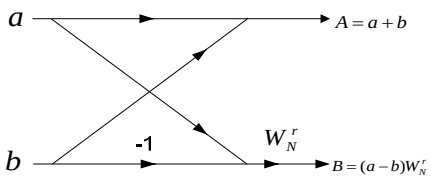
Maka untuk masukan N=8-titik didapatkan pola seperti diatas. Selain dengan cara diatas kita dapat menyatakan indeks n, dalam deret *x(n)*, dalam bentuk biner. Dari orde deret dalam bentuk biner tersebut kita baca kebalikannya untuk mendapatkan posisinya. Misal: $x(3) \equiv x(011)$, dibaca kebalikannya *m = 110 = 6* maka *x(3)* ditempatkan pada posisi m=6 dalam array yang diuraikan. Dengan deret data masukan disimpan dalam orde kebalikan bit dan komputasi kupu-kupu yang dilakukan ditempat, deret DFT *X(k)* yang dihasilkan dalam orde natural (sesuai urutan *k = 0,1,…, N-1*)



Gambar 2.1 Butterfly FFT peruaraian dalam waktu

2. Algoritma Penguraian dalam Frekuensi

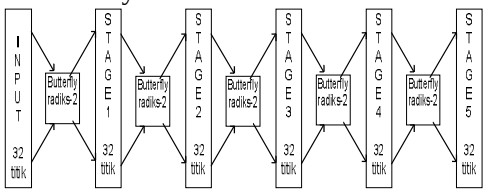
Untuk algoritma penguraian dalam frekuensi data masukan x(n) terjadi dalam orde natural tetapi DFT keluaran terjadi dalam orde kebalikan bit.



Gambar 2.2 Butterfly peruraian dalam frekuensi

2.3 IFFT 32 titik

Pada perhitungan IFFT 32 titik digunakan FFT radiks 2 sebagai dasar komputasinya. Proses keseluruhan untuk IFFT 32 titik meliputi $v = \log_2 N$ tahap penguraian maka v = 5 tahap penguraian dan meliputi N/2 = 16 butterfly FFT radiks 2 atau IFFT 2 titik.



Gambar 2.3 Butterfly IFFT 32 titik

2.4 Format Bilangan Fixed-Point

Dalam perhitungan IFFT terdapat proses komputasi dengan *twiddle factor* dan pembagian dengan N, yang mana hasil dari komputasi tersebut berupa pecahan seperti 1/32=0,03125 dalam perancangan digital. Dalam perancangan digital format pecahan tersebut harus dinyatakan kedalam bentuk biner, sedangkan dalam bentuk biner hasilnya dapat berbeda sesuai dengan format fraksinya. Sebagai contoh :

100.100 = +4,50 dalam desimal.

100100 = +36 dalam desimal.

Selain itu format biner akan mempunyai nilai yang berbeda lagi jika formatnya dalam *signed* (*two's complement*) atau *unsigned*.

Dalam format signed : 10.100 = -1,50.

Dalam format unsigned : 10.100 = +2,50.

2.5 VHDL

Very High Speed Integrated Circuits Hardware Description Language atau sering disingkat VHDL merupakan bahasa pemrograman yang digunakan untuk menggambarkan hardware elektronika. VHDL dapat digunakan untuk menggambarkan desain elektronika digital pada beberapa tingkat abstraksi, dari skala tingkat algoritma hingga tingkat gate.

VHDL pertama kali dikembangkan untuk Departemen Pertahanan US dan di standarisasi pertama kali oleh IEEE (Institute of Electrical and Electronica Engineering) pada tahun 1987, dengan nama IEEE Std 1076-1987. Kemudian di standarisasi ulang pada tahun 1993 dengan nama IEEE Std 1076-1993. Kemudian sebuah standar tambahan, IEEE 1164 untuk

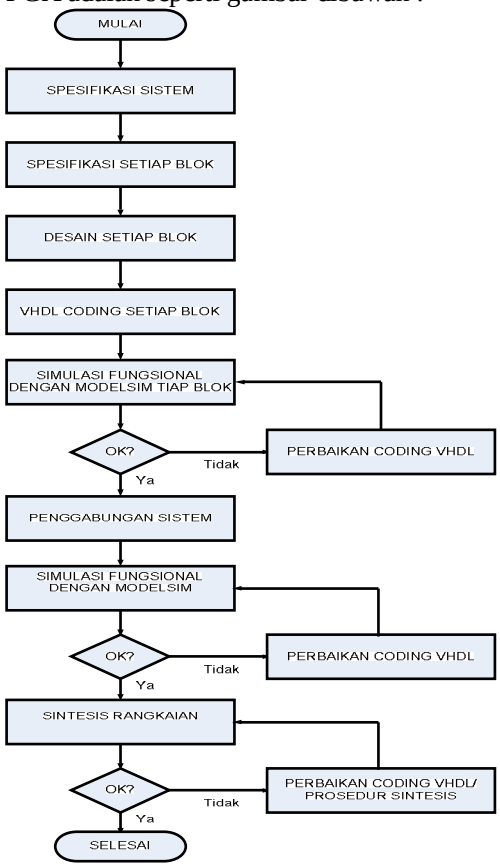
mengenalkan nilai sistem logika. Struktur dasar kode VHDL ada tiga, yaitu:

1. Library: berisi semua library yang digunakan pada rancangan.
2. Entity: spesifikasi pin input dan output pada rancangan rangkaian
3. Architecture: berisi kode utama VHDL yang menggambarkan bagaimana rangkaian bekerja.

III. Perancangan dan Implementasi

3.1 Perancangan

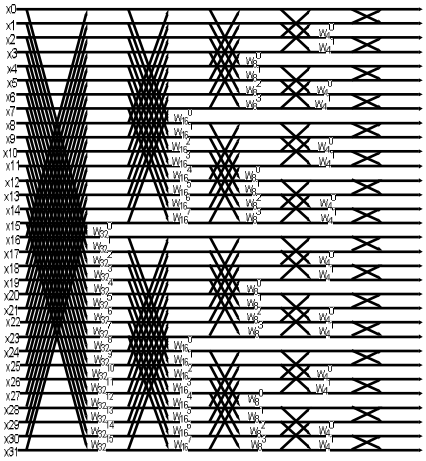
Perancangan dan implementasi pada IFFT ini dimulai pada blok dasar dari IFFT ini yaitu IFFT radiks 2, kemudian dikembangkan menjadi blok yang lebih besar. Dari blok dasar IFFT akan dibuat blok–blok pendukungnya sesuai dengan kebutuhan perancanagan sehingga menjadi blok besar. Tahap–tahap dalam perancangan dan implementasi pada FPGA adalah seperti gambar dibawah :



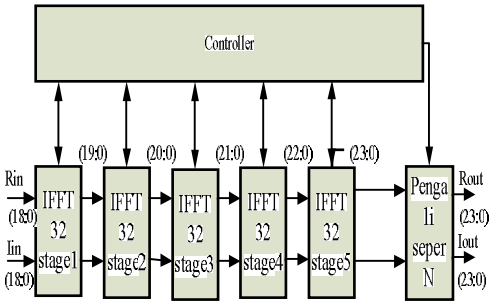
Gambar 3.2 Diagram tahapan perancangan program utama

3.2 IFFT 32 titik

Blok IFFT 32 titik ini merupakan blok top level, yang mencakup semua blok pendukungnya untuk mendapatkan hasil perhitungan IFFT 32 titik. Pada blok top level ini terdapat komponen–komponen atau blok–blok kontroller, stage 1 sampai dengan 5 dan pengali seper N. Dalam setiap stage terdapat blok datasel, RAM, IFFT Radiks 2, ROM, dan *twiddle multipier*, kecuali pada stage 5 tidak ada blok ROM dan *twiddle multiplier*. Perhitungan IFFT ini menggunakan algoritma IFFT radiks 2. Maka setiap stage akan terjadi perhitungan IFFT radiks 2 sebanyak 16 *butterfly* sehingga total ada 80 *butterfly*. Masukan dari IFFT menggunakan algoritma peruraian dalam frekuensi dimana masukan terjadi dalam orde natural tetapi keluaran terjadi dalam orde kebalikan bit yang akan diatur oleh kontroller untuk setiap stagenya.



Gambar 3.2 Butterfly IFFT 32 titik



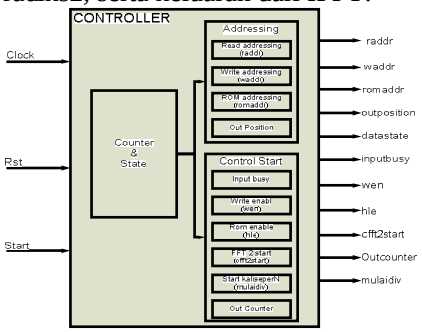
Gambar 3.3 Blok IFFT 32 titik

Masukan pada blok IFFT 32 titik mempunyai panjang 19 bit dan keluaran 24 bit. Setiap stage mempunyai masukan dengan jumlah bit yang bertambah dalam setiap stage. Pada stage 1 masukan berasal dari masukan awal top level IFFT 32 titik yaitu 19 bit dan outputnya 20 bit, adanya penambahan 1 bit pada keluaran dari IFFT radiks 2. Kemudian masuk pada stage 2 data dari 20 bit menjadi 21 bit hanya ada penambahan 1 bit dari komputasi IFFT radiks 2. Pada stage 3 dari 21

bit menjadi 22 bit, stage 4 dari 22 bit menjadi 23 bit, dan pada stage 5 dari 23 bit menjadi 24 bit. Dari 24 bit masuk ke dalam blok pengaliseperN dan keluarannya 24 bit.

a. Controller (Addressing)

Controller merupakan salah satu komponen pendukung pada IFFT. Dalam hal ini kontroller mempunyai peranan yang sangat penting, karena berfungsi mengatur aliran data proses yang sedang berlangsung. Dalam kontroller ini terdapat pengaturan counter, pengaturan orde data yang masuk dan data yang akan keluar, mengontrol RAM, ROM, IFFT radiks2, serta keluaran dari IFFT.

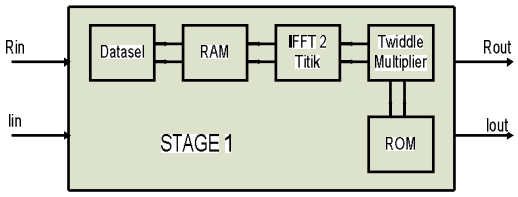


Gambar 3.4 Blok Controller

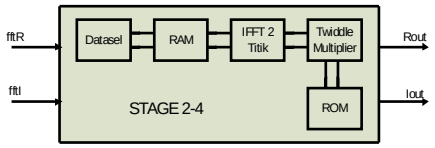
Pada blok kontroller ini diatur oleh clock, reset (rst) dan start. Reset (rst) berfungsi atau aktif saat low, saat bernilai ‘0’. Begitu juga dengan start, start aktif saat low ‘0’. Program yang dirancang akan berjalan ketika clock dalam kondisi aktif event dan aktif high ‘1’. Keluaran dari kontroller ini ada 11 output yang digunakan untuk mengatur blok–blok selanjutnya.

b. Stage

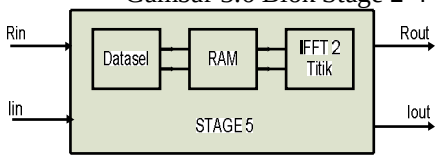
Pada IFFT N titik maka peruraian dapat dilakukan $N = 2^v$ maka $v = \log_2 N$, sehingga didapatkan untuk IFFT 32 titik v = 5 yang berarti terdapat 5 stage dalam proses komputasi IFFT 32 titik ini. Dalam setiap stage terdapat beberapa komponen, diantaranya adalah komponen datasel, RAM, IFFT Radiks 2, ROM, dan *twiddle multiplier*. Untuk stage 1 sampai 4 berisi komponen-komponen tersebut, namun untuk stage 5 untuk komponen *twiddle multiplier* tidak ada. Masukan dari tiap stage mempunyai panjang bit yang berbeda–beda sesuai dengan pertambahan panjang bit pada setiap stage.



Gambar 3.5 Blok Stage 1



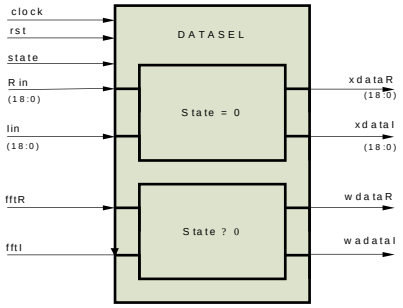
Gambar 3.6 Blok Stage 2-4



Gambar 3.7. Blok Stage 5

c. Data Sel

Blok data sel merupakan blok untuk penyimpanan data sementara pada register untuk pengaksesan RAM. Sehingga data sebelum masuk pada RAM data disimpan dulu pada register xdata atau wdata.



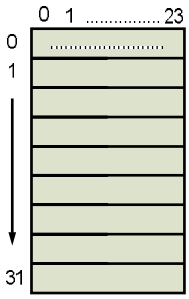
Gambar 3.8. Blok Datasel

Pada blok datasel ini bergantung pada adanya nilai clock, reset dan state. Nilai state diambil dari blok kontroller. Untuk xdataR dan xdataI selalu diisi dengan Rin dan Iin. Sedangkan untuk wdataR dan wdata I didisi dengan data fftR dan fftI yang merupakan data keluaran dari setiap stage. Panjang data untuk fftR dan fftI tergantung dari panjangnya data hasil keluaran dari setiap stage.

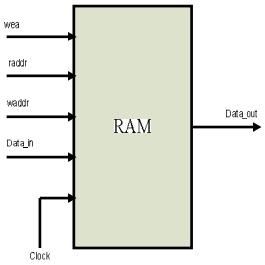
d. RAM

Pada blok RAM ini digunakan untuk menyimpan data masukan dimana data ini akan digunakan sebagai masukan untuk IFFT sebagai proses komputasi. Hasil komputasi dari IFFT ini akan disimpan kembali pada RAM dengan alamat yang sama. Wea, raddr,

dan waddr dikontrol oleh blok kontroller. Untuk IFFT 32 titik RAM ini mempunyai alamat sampai dengan 32 alamat yaitu dari alamat 0 sampai dengan 31. Dalam satu alamat dapat menyimpan data maksimal adalah 24 bit. Dalam setiap stage RAM dapat menyimpan data sesuai dengan panjang data masukan pada stage tersebut. Misalkan pada stage 1 IFFT 32 titik, panjang data masukanya adalah 19 bit maka RAM dapat menyimpan data sebanyak 32 alamat dengan setiap alamat tersimpan data 19 bit. Data_out akan mengeluarkan data sesuai dengan nilai raddr yang diminta. Sedangkan data_in akan disimpan pada alamat sesuai dengan nilai waddr.



Gambar 3.9 Kapasitas RAM



Gambar 3.10 Blok RAM

e. Butterfly IFFT Radiks 2

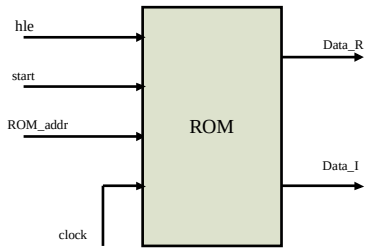
Pada blok FFT radiks 2 ini merupakan proses komputasi dasar atau utama. Data masukan berupa bilangan kompleks yang setiap masukan terdapat bilangan real dan imajiner. Masukan dari IFFT ini adalah 2 titik yang berasal dari data yang telah disimpan pada RAM yang alamat data yang masuk disesuaikan dengan urutan masukan pada IFFT dan hasil keluaran dari IFFT radiks 2 ini akan disimpan pada RAM sesuai dengan alamat pada masukannya. Pada blok butterfly IFFT radiks 2 titik ini terdapat counter yang akan mengatur aliran data dan proses komputasinya.

f. ROM

Blok ROM ini berfungsi untuk menyimpan nilai sinus dan cosinus yang digunakan pada proses komputasi IFFT. Pada ROM ini terdapat 32 nilai sinus dan cosinus. ROM ini digunakan dengan mengakses alamat ROM yang diatur pada kontroller.

Dalam perancangan ini terdapat 4 ROM untuk 5 stage IFFT dimana setiap ROM menyimpan nilai yang berbeda–beda untuk setiap alamat ROM. Nilai yang tersimpan dalam ROM disini disebut dengan nilai *twiddle factor* dimana nilai twiddle ini berupa bilangan kompleks, terdapat nilai real dan imajiner. Nilai twiddle ini merupakan representasi dari rumus :

$$W_1(x) = e^{(2j\pi(k/N))}$$

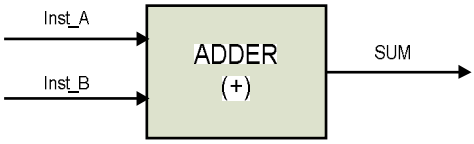


Gambar 3.11 Blok ROM

Pada ROM ini tersimpan data *twiddle factor* dalam bentuk biner dengan format fixed point dan mempunyai panjang 14 bit untuk data real dan imajiner. Untuk IFFT 32 titik maka dalam satu ROM mempunyai alamat dari 0 sampai 31 dan dalam satu alamat terdapat data dengan panjang 14 bit.

g. Adder

Adder merupakan blok terkecil dalam perancangan IFFT ini, dimana blok ini berfungsi untuk menjumlahkan dua buah masukan dengan panjang bit yang sama. Panajng bit nilai masukan dengan keluaran pada adder ini berbeda, karena adanya kemungkinan bertambahnya jumlah bit setelah proses penjumlahan.

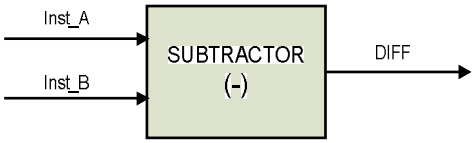


Gambar 3.12 Blok Adder

h. Substractor

Substractor merupakan blok yang berfungsi untuk pengurangan pada proses komputasi IFFT. Panjang bit pada masukan dan keluaran dari blok substractor ini sama dengan blok adder. Penggunaannya juga

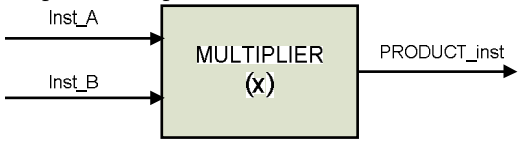
bersamaan dengan blok adder pada blok *compleks multiplier*.



Gambar 3.13 Blok Substractor

i. Multiplier

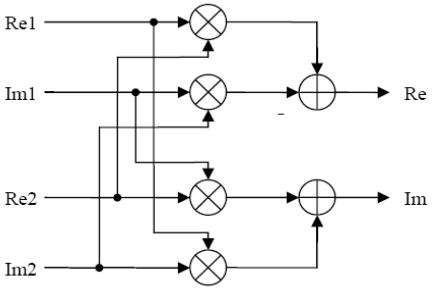
Blok multiplier berfungsi sebagai blok pengali dalam proses komputasi IFFT. Dua buah inputan akan dikaliakan dengan panjang bit awal adalah *inst_width-1 downto 0* dan ada keluarannya panjang bitnya bertambah menjadi *ins_width1+inst_width2-1 downto 0*. Blok multiplier ini juga terdapat pada blok *compleks multiplier*.



Gambar 3.14 Blok Multiplier

3.1.3 Complex Multiplier

Hasil perhitungan IFFT pada setiap stage kecuali pada stage terakhir selalu akan dikalikan dengan *twiddle factor* yang mana *twiddle factor* berupa bilangan kompleks seperti halnya pada keluaran hasil perhitungan IFFT pada setiap stage. Maka pada blok *complex multiplier* ini berfungsi sebagai proses perkalian kompleks dua buah masukan. Pada blok ini terdapat tiga blok yang mendukung, yaitu blok adder, blok substractor, dan blok multiplier.

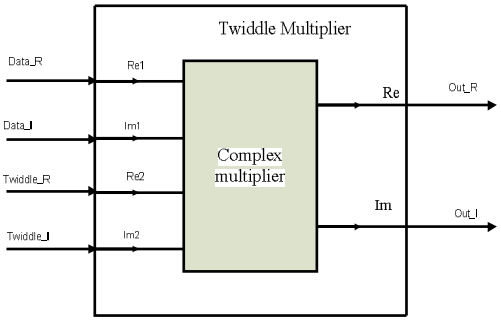


Gambar 3.15 Complex Multiplier Structure

Masukan pertama dari blok *complex multiplier* berasal dari blok IFFT Radiks 2 dan masukan kedua adalah *twiddle factor* yang tersimpan dalam ROM. Kedua masukan tersebut mempunyai panjang bit yang berbeda.

j. Twiddle Multiplier

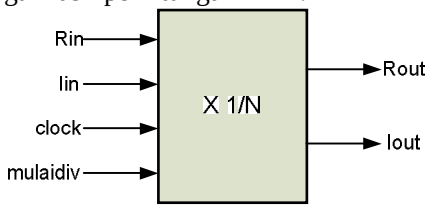
Blok *twiddle multiplier* ini terdiri dari komponen *complex multiplier*. Masukan dari blok *twiddle multiplier* ini akan menjadi masukan untuk blok *complex multiplier*. Keluaran dari complex multiplier yang mana panjang bitnya telah bertambah akan disamakan dengan panjang bit masukan pada setiap stage, kecuali pada stage pertama. Pada stage pertama panjang bit saat keluaran bertambah sama seperti panjang bit keluaran blok *complex multiplier*.



Gambar 3.16 Blok Twiddle Multiplier

k. Pengali Seper N

Pada blok ini berfungsi sebagai pengali dengan 1/N (N=32). Dalam hal ini hasil dari stage terakhir atau stage 5 yang akan dikali dengan 1/N. Masukan dari blok ini parameter panjang bitnya adalah *width* dengan panjang bit sebesar *width-1 downto 0* dan keluarannya juga mempunyai panjang bit yang sama. Hasil dari blok ini yang akan keluar sebagai hasil perhitungan IFFT.



Gambar 3.17 Blok Pengali SeperN

3.1.4 Test Bench

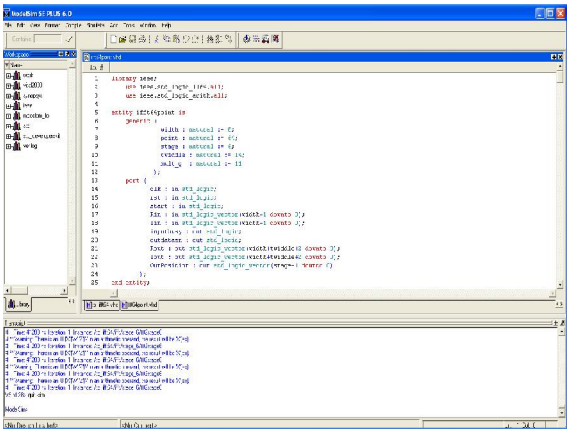
Test bench untuk perhitungan IFFT ditulis dalam bahasa VHDL dan akan disimulasikan pada ModelSim. Pada *testbench* akan dibangkitkan nilai pulsa dan data–data sebagai sinyal uji. Nilai pulsa yang dibangkitkan seperti *clock, reset, start* dan memasukan data-data masukan *R_in*. Dengan *testbench* ini akan dapat dengan mudah

melihat dan mengoreksi hasil keluaran dari perhitungan IFFT.

3.2 Simulasi

Dari program yang telah dibuat perlu dilakukan pengujian, pengujian dilakukan pada *ModelSim SE 6.0*. Pengujian yang dilakukan untuk mengetahui timing dari perhitungan yang dilakukan serta untuk mengetahui kesalahan sintaks dan error yang terjadi. Selain itu juga untuk mengetahui jika terdapat kesalahan perhitungan pada setiap stage maupun hasil keluaran dari IFFT.

Untuk simulasi dilakukan dengan menggunakan *testbench*, yaitu dengan memberikan tes vector pada program top level yang telah dibuat. Dari *testbench* tersebut kemudian disimulasikan dan data yang dihasilkan sesuai dengan sinyal tes vector yang diberikan.

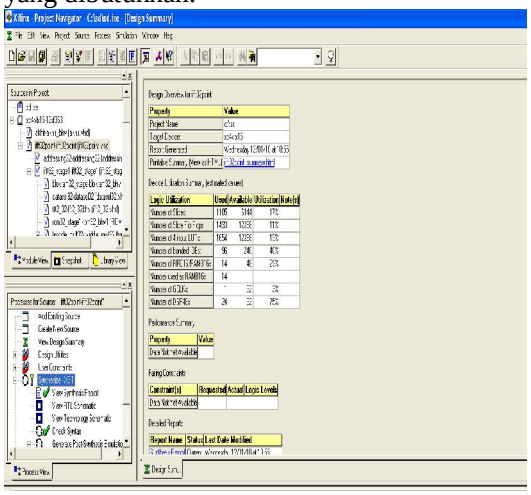


Gambar 3.18 ModelSim SE 6.0

3.3 Sintesis

Proses sintesis dilakukan dengan menggunakan software *Xilink ISE 7.1*. Sintesis yang dilakukan adalah penterjemahan bahasa VHDL menjadi bentuk rangkaian dalam RTL (*Register Transfer Level*) Netlis. Hasil dari sintesis akan didapatkan beberapa parameter yng dibutuhkan dalam implementasi ke FPGA.

Setelah perancangan dituankan kedalam bahasa pemrograman VHDL maka file program tersebut siap untuk di sintesis untuk diketahui parameter-parameter yang dibutuhkan serta rangkaian RTL hasil perancangan. Beberapa parameter yang diamati adalah *number of slice, number of slice flip–flop, number of 4 input LUTs, number of bonded IOB, number of FIFO 16/RAMB 16s, number of of GCLKs.* Rangkaian hasil sintesis yang didapatkan dari penterjemahan bahasa VHDL akan didapatkan dari rangkaian terbesar sampai dengan komponen terkecil atau gerbang logika dasar yang dibutuhkan.



Gambar 3.19 Sintesis Xilinx ISE 7.1

Berikut ini merupakan tabel kapasitas masing-masing parameter di atas yang terdapat pada *Xilinx ISE 7.1* menggunakan hardware FPGA seri *Virtex xc4vlx15*.

Tabel 3.1 Kapasitas parameter sintesis.

Logic Utilization	Available
Number of Slices	6144
Number of Slice Flip Flops	12288
Number of 4 input LUTs	12288
Number of bonded IOBs	240
Number of FIFO16/RAMB16s	48
Number of GCLKs	32

- *Number of slice* adalah menujukkan jumlah slice atau komponen utama yang akan digunakan pada FPGA yang terdiri dari CLB (*Configurable Logic Blok*).
- *Number of slice flip–flop* adalah jumlah bagian suatu flip–flop, dimana flip- flop itu sendiri adalah dua state logika buffer yang diaktifkan oleh *clock* dan diberi masukan tunggal yang akan bekerja berkombinasi dengan *clock*. Posisinya adalah *high* dan *low*. Ketika *clock high*, flip–flop ini akan bekerja sebagai buffer untuk nilai output dari nilai input D pada saat *clock rises* dan akan dijaga sampai dengan *clock rises*

berikutya. Output tidak dipengaruhi ketika *clock low* (*falling clock*).

- *Number of LUTs* adalah menunjukkan jumlah LUTs yang digunakan. LUTs digunakan untuk menerapkan fungsi generator pada CLBs. Terdapat empat masukan untuk dua buah fungsi generator. Fungsi generator sendiri dapat diterapkan pada beberapa fungsi boolean dengan empat masukkan.
- *Number of IOB* ini menunjukaan jumlah elemen unsur–unsur dasar yang menerapkan fungsi keluaran dan masukan suatu alat.
- *Number of FIFO 16/RAMB 16s* menunjukkan jumlah RAM yang digunakan, yaitu memori yang dapat membaca dan menulis yang diakses pada waktu yang independen pada lokasi fisik sebuah data. RAM dapat mengubah nilai alamat pada fungsi generator.
- *Number of GCLK* menunjukkan jumlah global clock (clock utama) yang digunakan pada perancangan.

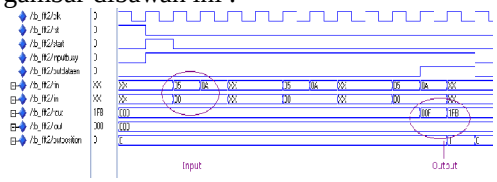
IV. Pengujian dan Analisa

Pengujian yang dilakukan adalah pengujian dan analisa terhadap hasil pemodelan algoritma IFFT pada bahasa VHDL untuk dilihat hasil simulasi dan sintesisnya. Simulasi dengan bahasa VHDL menggunakan software *ModelSim*. Pengujiannya menggunakan *testbench* yang diberi sinyal test vector pada arsitektur blok–blok pendukung IFFT 32 titik.

4.1 Hasil Simulasi

4.1.1 Pengujian IFFT 2 titik

Untuk IFFT 2 titik ini dibutuhkan 2 masukan yang terdiri dari bilangan real dan imajiner. Yang masukan hanya berupa bilangan real maka untuk imajiner bernilai nol. Masukan tiap titik panjangnya 8 bit dengan format fixed point <8,7,t>. Keluaran sistem panjang bit menjadi 10 bit dengan format fixed point <16,8,t>. Simulasi yang dilakukan dberikan dua buah data dan dapat dilihat pada gambar dibawah ini :

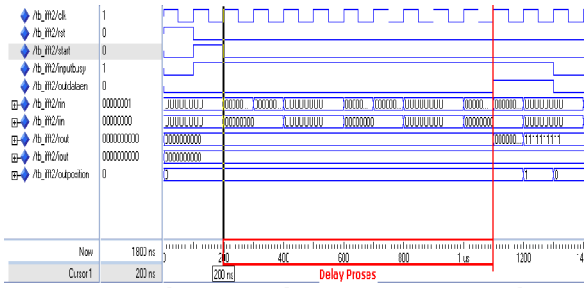


Gambar 4.1 Simulasi testbench IFFT 2 titik

Data masukan dalam simulasi IFFT 2 titik dengan *testbench* ini adalah sebagai berikut :

Tabel 4.1. Data input dan output simulasi IFFT 2 titik.

Input	Dalam integer <8,7,t>	Output	Dalam integer <10,8,t>
x(0) = 00000001	1	X(0) = 0000000011	1,5
x(1) = 00000010	2	X(1) = 1111111111	-0,5

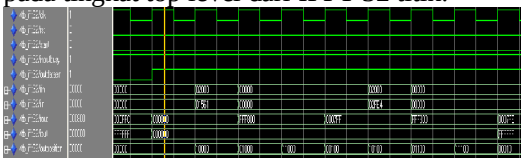


Gambar 4.2 Delay proses IFFT 2 titik

Berdasarkan hasil pengujian diatas didapatkan delay proses pada IFFT 2 titik adalah sebesar 900 ns (9 pulsa clock).

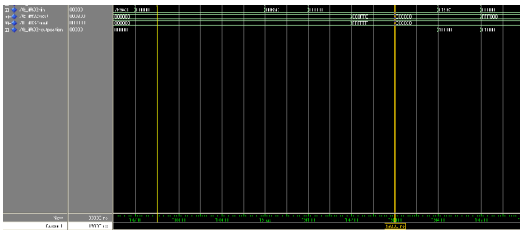
4.1.2 Pengujian IFFT 32 titik menggunakan sinyal kotak

Dalam perhitungan IFFT 32 titik dengan algoritma FFT radiks 2 terjadi penguraian sebanyak 5 stage. Input IFFT berupa bilangan dengan format fixed point 19 bit <19,11,t> dan output dari IFFT akan bertambah jumlah bitnya menjadi 24 bit fixed point <24,11,t>. Pemodelan IFFT 32 titik ini terdiri dari beberapa blok pendukung. Di bawah ini akan ditunjukkan simulasi *testbench* pada tingkat top level dari IFFT 32 titik.



Gambar 4.3. Output IFFT 32 titik

Data yang menjadi input dan hasil output dari simulasi IFFT 32 titik ini adalah sebagai berikut :

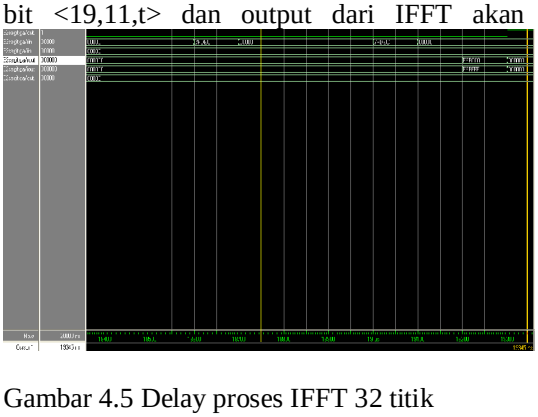


Gambar 4.4. Delay proses IFFT 32 titik

Delay proses yang terjadi untuk IFFT 32 titik dapat dilihat pada gambar diatas, dimana besaranya adalah 19300 ns (193 pulsa clock).

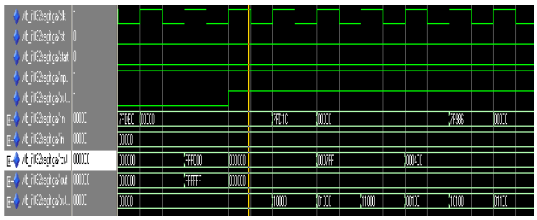
4.1.3 Pengujian IFFT 32 titik menggunakan sinyal segitiga

Dalam perhitungan IFFT 32 titik dengan algoritma FFT radiks 2 terjadi penguraian sebanyak 5 stage. Input IFFT berupa bilangan dengan format fixed pont 19 bit <19,11,t> dan output dari IFFT akan bertambah jumlah bitnya menjadi 24 bit fixed point <24,11,t>. Pemodelan IFFT 32 titik ini terdiri dari beberapa blok pendukung. Di bawah ini akan ditunjukkan simulasi *testbench* pada tingkat top level dari IFFT 32 titik.



Gambar 4.5 Delay proses IFFT 32 titik

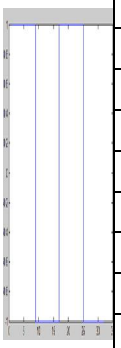
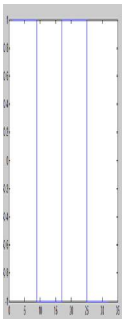
Delay proses yang terjadi untuk IFFT 32 titik dapat dilihat pada gambar diatas, dimana besaranya adalah 19300 ns (193 pulsa clock).



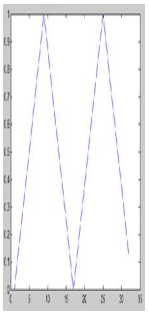
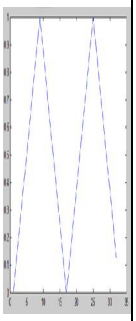
Gambar 4.6 Output IFFT 32 titik

Data yang menjadi input dan hasil output dari simulasi IFFT 32 titik ini adalah sebagai berikut :

Tabel 4.2: Tabel Perbadingan Output Ifft 32 Titik Menggunakan Sinyal Kotak dengan Output pada Matlab

No	Input <19,11,t>		Integer		Gambar sinyal input	Output <24,11,t>		Integer		Matlab		Gambar sinyal output
	Real (Rin)	Im (Iin)	Real (Rin)	Im (Iin)		Real (Rout)	Im (Iout)	Real (Rout)	Im (Iout)	Real	Im	
0	00000	00000	0	0		000800	000000	1	0	1	0	
1	00000	00000	0	0		000800	FFFFFF	1	-0,0004882	1	0	
2	02000	F5F21	4	-20,109357968503392		0007FE	FFFFFF	0,9990234	-0,0004882	1	0	
3	00000	00000	0	0		0007FE	FFFFFF	0,9990234	-0,0004882	1	0	
4	00000	00000	0	0		0007FF	000000	0,9995117	0	1	0	
5	00000	00000	0	0		0007FF	000000	0,9995117	0	1	0	
6	02000	FD0C	4	-5,986423050661956		0007FE	000000	0,9990234	0	1	0	
7	00000	00000	0	0		0007FE	000000	0,9990234	0	1	0	
8	00000	00000	0	0		FFF800	000000	-1	0	-1	0	
9	00000	00000	0	0		FFF7FF	000000	-1,000488	0	-1	0	
10	02000	FEAF	4	-2,672714551677195		FFF801	000000	-0,9990234	0	-1	0	
11	00000	00000	0	0		FFF801	000000	-0,9990234	0	-1	0	
12	00000	00000	0	0		FFF800	000000	-1	0	-1	0	
13	00000	00000	0	0		FFF800	000000	-1	0	-1	0	
14	02000	FF9A3	4	-0,795649469518633		FFF801	000000	-0,9990234	0	-1	0	
15	00000	00000	0	0		FFF801	000000	-0,9990234	0	-1	0	
16	00000	00000	0	0		000800	000000	1	0	1	0	
17	00000	00000	0	0		000800	FFFFFF	1	-0,0004882	1	0	
18	02000	0065D	4	0,795649469518633		0007FE	FFFFFF	0,9990234	-0,0004882	1	0	
19	00000	00000	0	0		0007FE	FFFFFF	0,9990234	-0,0004882	1	0	
20	00000	00000	0	0		0007FF	000000	0,9995117	0	1	0	
21	00000	00000	0	0		0007FF	000000	0,9995117	0	1	0	
22	02000	01561	4	2,672714551677195		0007FE	000000	0,9990234	0	1	0	
23	00000	00000	0	0		0007FE	000000	0,9990234	0	1	0	
24	00000	00000	0	0		FFF800	000000	-1	0	-1	0	
25	00000	00000	0	0		FFF7FF	000000	-1,000488	0	-1	0	
26	02000	02FE4	4	5,986423050661956		FFF801	000000	-0,9990234	0	-1	0	
27	00000	00000	0	0		FFF801	000000	-0,9990234	0	-1	0	
28	00000	00000	0	0		FFF800	000000	-1	0	-1	0	
29	00000	00000	0	0		FFF800	000000	-1	0	-1	0	
30	02000	0A0DF	4	20,109357968503392		FFF801	000000	-0,9990234	0	-1	0	
31	00000	00000	0	0		FFF801	000000	-0,9990234	0	-1	0	

Tabel 4.3: Tabel Perbadingan Output Ifft 32 Titik Menggunakan Sinyal Kotak dengan Output pada Matlab

No	Input <19,11,t>		Integer		Gambar sinyal input	Output <24,11,t>		Integer		Matlab		Gambar sinyal output
	Real (Rin)	Im (Iin)	Real (Rin)	Im (Iin)		Real (Rout)	Im (Iout)	Real (Rout)	Im (Iout)	Real	Im	
0	10000	00000	16	0		000000	000000	0	0	0	0	
1	00000	00000	0	0		000100	FFFFFF	0,125	-0,0004882	0,125	0	
2	FCB05	00000	-6,568535592	0		000200	FFFFFF	0,25	-0,0004882	0,25	0	
3	00000	00000	0	0		000300	FFFFFF	0,375	-0,0004882	0,375	0	
4	00000	00000	0	0		000400	000000	0,5	0	0,5	0	
5	00000	00000	0	0		000500	000000	0,625	0	0,625	0	
6	FF986	00000	-0,809957202	0		0005FF	000000	0,74951171875	0	0,75	0	
7	00000	00000	0	0		0006FF	000000	0,87451171875	0	0,875	0	
8	00000	00000	0	0		0007FF	000000	0,99951171875	0	1	0	
9	00000	00000	0	0		000700	000000	0,875	0	0,875	0	
10	FFD1C	00000	-0,361615673	0		0005FF	000000	0,74951171875	0	0,75	0	
11	00000	00000	0	0		0004FF	000000	0,62451171875	0	0,625	0	
12	00000	00000	0	0		000400	000000	0,5	0	0,5	0	
13	00000	00000	0	0		0002FF	000000	0,37451171875	0	0,375	0	
14	FFDEC	00000	-0,259891532	0		000200	000000	0,25	0	0,25	0	
15	00000	00000	0	0		000100	000000	0,125	0	0,125	0	
16	00000	00000	0	0		000000	000000	0	0	0	0	
17	00000	00000	0	0		000100	FFFFFF	0,125	-0,0004882	0,125	0	
18	FFDEC	00000	-0,259891532	0		000200	FFFFFF	0,25	-0,0004882	0,25	0	
19	00000	00000	0	0		000300	FFFFFF	0,375	-0,0004882	0,375	0	
20	00000	00000	0	0		000400	000000	0,5	0	0,5	0	
21	00000	00000	0	0		000500	000000	0,625	0	0,625	0	
22	FFD1C	00000	-0,361615673	0		0005FF	000000	0,74951171875	0	0,75	0	
23	00000	00000	0	0		0006FF	000000	0,87451171875	0	0,875	0	
24	00000	00000	0	0		0007FF	000000	0,99951171875	0	1	0	
25	00000	00000	0	0		000700	000000	0,875	0	0,875	0	
26	FF986	00000	-0,809957202	0		0005FF	000000	0,74951171875	0	0,75	0	
27	00000	00000	0	0		0004FF	000000	0,62451171875	0	0,625	0	
28	00000	00000	0	0		000400	000000	0,5	0	0,5	0	
29	00000	00000	0	0		0002FF	000000	0,37451171875	0	0,375	0	
30	FCB05	00000	-6,568535592	0		000200	000000	0,25	0	0,25	0	
31	00000	00000	0	0		000100	000000	0,125	0	0,125	0	

4.2 Sintesis

Sintesis adalah menerjemahkan dari bahasa pemrograman VHDL menjadi *netlis*. Sintesis yang dilakukan adalah IFFT 2 titik dan IFFT 32 titik. Hal ini dilakukan untuk melihat jumlah parameter–parameter yang dibutuhkan untuk tiap titik. Dari xilinx *add source* file VHDL yang akan disintesis, dan dilakukan sintesis. Dari sintesis ini akan ada design summary yang menunjukkan parameter–parameter yang akan dibutuhkan serta jumlah yang diutuhkan pada saat perancangan yang disesuaikan dengan perancangan pemrograman VHDL yang telah dibuat. Setelah sintesis dilakukan view RTL schematic akan didapatkan rangkaian RTL hasil perancangan IFFT.

4.2.1. Hasil Sintesis

Dari perancangan top level IFFT 32 titik dan dilakukan sintesis pada Xilinx maka akan didapatkan parameter–parameter yang dibutuhkan pada perancangan IFFT 32 titik. Parameter–parameter yang dibutuhkan serta jumlah yang digunakan dapat dilihat pada tabel dibawah ini

Tabel 4.4. Tabel hasil sintesis IFFT 32 titik.

Device Utilization Summary (estimates values)			
Logic Utilization	Used	Available	Utilization
Number of Slice	1105	6144	17%
Number of Slice Flip Flop	1433	12288	11%
Number of 4 input LUTs	1654	12288	13%
Number of bonded IOBs	96	240	40%
Number of FIFO16/RAMB16s	14	48	29%
Number of GCLKs	1	32	3%

4.2.2 Hasil Rangkaian RTL Schematic

Setelah dilihat parameter–parameter yang dibutuhkan dalam perancangan IFFT 32 titik maka dengan view RTL schematic akan didapatkan rangkaian dari perancangan IFFT 32 titik yang telah dibuat berdasarkan pemrograman dengan bahasa VHDL.

Gambar 4.7 Model Top Hirarki RTL Schematic

V. Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan dalam tugas akhir ini dapat disimpulkan beberapa hal sebagai berikut:

1. Sudah dapat dilakukan perancangan arsitektur IFFT 32 titik sampai dengan tahap simulasi fungsional sistem.
2. Keluaran pada IFFT yang dihasilkan setelah dilakukan penelitian pada perancangan FFT hasil yang didapatkan sesuai dengan input pada IFFT.
3. Delay proses yang diperlukan dari saat mulai memasukkan inputan hingga keluar output untuk IFFT 32 titik besaranya adalah 19300 ns (193 pulsa clock).
4. Output IFFT hasil simulasi sama dengan hasil *output* dari Matlab.
5. Berdasarkan hasil sintesis IFFT 32 titik didapatkan jumlah resource yang dibutuhkan adalah jumlah slice 15%, jumlah flip–flop 10%, jumlah LUT 10%, jumlah IOB 30%, jumlah FIFO16/RAMB16 29%, jumlah GCLK 3%.
6. Panjang input adalah 19 bit sedangkan output bertambah menjadi 24 bit.

5.2 Saran

Dari serangkaian penelitian yang telah dilaksanakan, beberapa saran pengembangan yang dapat dilakukan adalah:

1. Dilakukan pengujian untuk jumlah titik input yang lebih besar untuk kemudian ditentukan optimasi hasil design yang dapat dicapai.
2. Dilakukan proses implementasi pada board FPGA.
3. Dilakukan penelitian dengan menggunakan algoritma radiks 4 yang

kemudian dibandingkan dengan sistem IFFT yang menggunakan algoritma radiks 2.

DAFTAR PUSTAKA

1) Chang, K.C., *Digital Systems Design with VHDL and Synthesis*, Matt Loeb, USA.1999

2) Ludeman, Lonnie C., *Fundamental of Digital Signal Processing*, John Wiley & Sons, Inc, Canada. 1987.

3) Miller, Adam Robert, *Development and Verificationof ParameterizedDigital Signal Processing Macros for Microelectronics Systems*, The University of Tennesee, Knoxville. 2003.

4) Pedroni, Volnei A., *Circuit Design With VHDL*, Massachusetts Institute of Technology, USA. 2004.

5) Proakis, John G. dan Manolakis, Dimitris G., *Pemrosesan Sinyal Digital*, Edisi Bahasa Indonesia Jilid 1, Prenhallindo, Jakarta. 1997.

6) Suprapto, Desia ilmina, *Desain dan Sintesis Arsitektur Hardware IFFT (Inverse Fast Fourier Transform) 64 titik Berbasis Bahasa Pemrograman VHDL*. Tugas Akhir STT Telkom. Bandung. 2008

7) Wardana, Ali, *Desain dan Implementasi IDFT (Inverse Descrete Fourier Transform) untuk OFDM dengan FPGA*. Tugas Akhir STT Telkom. Bandung. 2007.

8) Wada, Tom, *64 point Fast Fourier Transform Circuit (Version 1.0)*.

9) www.ie-u-ryuku.ac.id. 2006

10) www.opencores.org

11) www.wikipedia.com

Amalia Rizka Darmayanti (L2F 308005)

Saat ini sedang menempuh studi pendidikan strata I di Jurusan Teknik Elektro, Fakultas Teknik Universitas Diponegoro. Konsentrasi yang ditekuni adalah pada bidang Elektronika dan Telekomunikasi

Mengetahui,

Dosen Pembimbing I	Dosen Pembimbing II
Achmad H, S.T., M.T.	Darjat, S.T., M.T.
NIP.196912211995121001	NIP.197206061999031001
Tanggal : _____	Tanggal : _____