PERANCANGAN DAN IMPLEMENTASI TURBO DECODER PADA TEKNOLOGI *LONG TERM EVOLUTION (LTE)* BERBASIS *FPGA*

DESIGN AND IMPLEMENTATION TURBO DECODER FOR LONG TERM EVOLUTION (LTE) TECHNOLOGY BASED ON FPGA

Ahmad Eko Ardianto[1] **Dr. Rina Pudjiastuti, Ir., M.T.**[2] **Denny Darlis S.Si.,M.T.**[3]

(ardianto.psht@gmail.com) (rpa@ittelkom.ac.id) (ddarlis@yahoo.com)

Fakultas Teknik Elektro–Universitas Telkom
Jl. Telekomunikasi, Dayeuh Kolot Bandung 40257 Indonesia

ABSTRAK

Long Term Evolution (LTE) adalah sebuah teknologi yang telah di rilis oleh 3GPP dengan kemampuan pengiriman data mencapai kecepatan 100 Mbit/s untuk *downlink* dan 50 Mbit/s untuk *uplink*. Teknologi LTE dirancang untuk menyediakan efisiensi spektrum yang lebih baik, peningkatan kapasitas radio, *latency* dan biaya operasional yang rendah bagi operator serta layanan pita lebar nirkabel bergerak kualitas tinggi untuk pengguna. LTE dimaksudkan sebagai solusi jaringan komunikasi yang komprehensif dan aman dengan kecepatan data yang jauh lebih tinggi. Untuk kebutuhan data rate dan *througput* yang tinggi pada LTE, teknik pengkodean yang paling cocok adalah *turbo coding*. Keunggulan *Turbo Code* adalah penggunaan power yang minimum pada tiap modulasi sehingga memungkinkan pengiriman sinyal dengan level daya yang sangat rendah.

Berdasarkan penjelasan diatas, maka dalam tugas akhir ini dilakukan perancangan *prototype* rangkaian decoder *Turbo Code* yang digunakan pada teknologi LTE menggunakan *software* Xilinx ISE Design Suite 14.5 dengan bahasa pengkodean *VHSIC Hardware Description Language (VHDL)* yang kemudian di implementasikan pada *Field Programmable Gate Array (FPGA) board* ATLYS Spartan-6 XC6SLX45 CSG324C.

Dari hasil implementasi ditunjukkan bahwa perancangan *prototype Turbo Decoder* dapat dilakukan pada board ATLYS Spartan-6 XC6SLX45 CSG324C. Hasil implementasi menunjukkan penggunaan resource sebesar 23% pada board FPGA. Prototype ini menghasilkan sistem dengan periode minimum 19.662 ns dan frekuensi kerja dibawah frekuensi kerja FPGA Spartan-6, yaitu 50.963 MHz.

Kata Kunci : ***LTE, 3GPP, Turbo Code, FPGA***

ABSTRACT

Long Term Evolution (LTE) is a technology that has been released by 3GPP with the ability to achieve data transmission speeds of 100 Mbit / s for downlink and 50 Mbit / s for uplink. LTE technology is designed to provide better spectrum efficiency, increased radio capacity, latency and low operating costs for operators as well as mobile wireless broadband services of high quality to users. LTE is intended as a comprehensive network solutions and secure communication with data rates much higher. For the needs of data rate and high throughput on LTE, the most suitable coding technique is turbo coding. The advantages of Turbo Code is the minimum power usage at each modulation that allows the transmission of signals with very low power levels.

Based on the above explanation, it is in this final project to design a prototype Turbo Code decoder circuit that used in LTE technology using software Xilinx ISE Design Suite 14.5 with the coding language VHSIC Hardware Description Language (VHDL) and then be implemented on a Field Programmable Gate Array (FPGA) board ATLYS Spartan-6 XC6SLX45 CSG324C.

From the results indicated that the design of a prototype implementation of Turbo Decoder can be done on board ATLYS Spartan-6 XC6SLX45 CSG324C. The results indicate the implementation of resource usage by 23% on the FPGA board. This results in a system prototype with a minimum period of 19.662 ns and frequency of work under the working frequency of the Spartan-6 FPGA, namely 50.963 MHz.

Key Words: **LTE**, ***3GPP, Turbo Code, FPGA***

I. PENDAHULUAN

1.1 Latar Belakang

Long Term Evolution (LTE) adalah sebuah teknologi komunikasi nirkabel yang telah di rilis oleh 3GPP dengan kemampuan pengiriman data mencapai kecepatan 100 Mbit/s untuk *downlink* dan 50 Mbit/s untuk *uplink*[1]. LTE menerapkan teknik pengkodean kanal (*channel coding*) pada proses transmisinya. Teknik *channel coding* yang diterapkan pada LTE adalah *convolutional coding* dan *turbo coding*. Perbandingan antara c*onvolutional coding* dan *turbo coding* terletak pada *reliability* dan *efficiency* pada proses transmisi di teknologi LTE [2]. Untuk kebutuhan data rate dan *througput* yang tinggi pada LTE, teknik pengkodean yang paling cocok adalah *turbo coding*. *Turbo Code* dapat diterapkan pada perangkat di sisi *transmitter* maupun *receiver*. Pada sisi *receiver*, perangkat yang di lengkapi dengan teknik pengkodean *Turbo Code* adalah *Turbo Decoder*. Penggunaan *Turbo Decoder* pada sisi *receiver* dapat dirancang dengan beberapa pendekatan algoritma, diantaranya adalah Log-MAP *algoritm*, MAP *algoritm* dan *Soft Output Viterbi Algoritm* (SOVA).

Dari penjelasan di atas, dilakukan sebuah perancangan rangkaian sistem elektronika *Turbo Decoder* pada kanal ideal menggunakan pendekatan *Soft Ouput Viterbi Algoritm*. Pada penelitian sebelumnya yang dilakukan oleh Manjunatha K N, Kiran B, Prasanna Kumar C[3] telah berhasil mendesain dan mengimplementasikan Turbo Decoder dengan pendekatan MAP *algoritm* pada bahasa pengkodean *verilog*. Pada penelitian lain yang telah dilakukan oleh K. Kalyani, A. Skahti Amutha Vardhini , S. Rajaram[4] juga telah berhasil mendesain

dan mengimplementasikan *Turbo Decoder* dengan pendekatan Log-MAP *algoritm*.

Dari pemaparan di atas, pada tugas akhir ini dirancang sebuah *protoype Turbo Decoder* dengan pendekatan algoritma yang berbeda. Pendekatan algoritma yang digunakan pada tugas akhir ini menggunakan *Soft Output Viterbi Algoritm* (SOVA). Penggunaan algoritma tersebut dikarenakan salah satu yang paling cocok dan suitable dengan *Very Large Scale Integration (VLSI) chip design* dan bahasa pengkodean *VHSIC Hardware Description Language* (VHDL) pada software Xilinx ISE 14.5. Setelah *prototype* sistem berhasil dirancang, selanjutnya di tanamkan pada *board* FPGA ATLYS Spartan-6 XC6SLX45 CSG324C.

II. LANDASAN TEORI

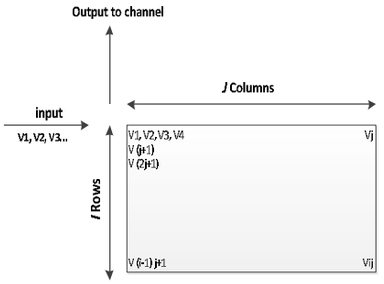
2.1 *Turbo decoder*

Turbo decoder merupakan proses pendekodean kode konvolusi pada kanal *memory-less* menggunakan prinsip iterasi. *Turbo decoder* diimplementasikan dalam 2 unit fungsi utama, yaitu *interleaver* dan *Soft Output Viterbi Algoritm* (SOVA). *Interleave*r yang digunakan berdasarkan standar dari 3GPP. *Interleaver* melakukan proses pembacaan *bits* secara baris dan mengeluarkannya berdasarkan kolom sesuai dengan ukuran matrik *interleaver*. SOVA unit melakukan proses *decoding* secara *soft decision*. SOVA unit terdiri dari 3 sub-blok, yaitu *Branch Distance Calculation, Add Compare Select,* dan *Traceback Unit*

2.1.1 *Interleaver*

Interleaving adalah salah satu cara yang efektif untuk mengatasi burst error. Ide dibalik interleaving adalah untuk memisahkan simbol-simbol data terkode dalam domain waktu. Simbol-simbol terkode dari encoder akan diterima dalam blok-blok oleh sebuah blok interleaver.

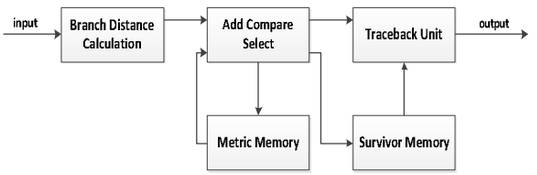
Pengurutan yang bisa dilakukan adalah dengan mengisi deretan terkode pada baris-baris pada susunan *i* baris dan *j* kolom. Data terkode sejumlah *j* pertama akan menempati baris pertama, dan untuk data ke-(j+1) sampai dengan data ke-(2j) menempati baris kedua, dan begitu seterusnya sampai semua baris terisi semua. Setelah itu, data akan dibaca per kolom pada satu waktu dan akan dikeluarkan ke kanal.



Gambar 2.2 *Interleaver Block*[3]

2.1.2 *Soft Output Viterbi Algoritm (SOVA)*

SOVA unit terdiri dari 3 sub-blok, yaitu *Branch Distance Calculation, Add Compare Select,* dan *Traceback Unit. Branch Distance Calculation* melakukan perhitungan jarak antara bit yang diterima dengan bit yang ideal pada state transisi. *Add Compare Select* menghitung dan memilih survivor *path metric* untuk setiap *state* pada *trellis. Traceback Unit* melakukan proses *traceback* untuk beberapa *survivor path* dari akhir menuju awal jalur.



Gambar 2.3 Arsitektur Umum Blok SOVA[7]

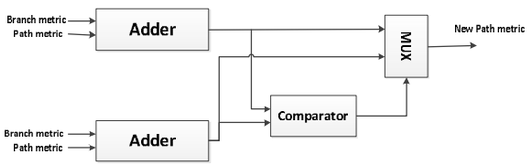
2.1.2.1 *Branch Distance Calculation*

Pada blok *Branch Distance Calculation* terjadi proses penghitungan *branch metric* (BM) untuk setiap waktu pada kode yang diterima. Pada teknik *soft decision decoding*, penghitungan BM menggunakan jarak Euclidean, dengan rumus sebagai berikut:

$$d = \sqrt{(r_x - S_x)^2 + (r_y - S_y)^2}$$

dimana r_x dan r_y merupakan koordinat informasi yang diterima setelah melalui proses transmisi pada kanal berderau. Sedangkan S_x dan S_y adalah koordinat pada diagram konstelasi dari informasi asli hasil pengkodean. Dalam proses penghitungan jarak *Euclide* ini, r dan S berbentuk multibit hasil kuantisasi. Penghitungan jarak *Euclide* dilakukan pada setiap *branch*/cabang untuk setiap waktu kode yang diterima.

2.1.2.2 *Add Compare Select*



Gambar 2.4 Blok *Add Compare Select*[8]

Pada blok ACS (*Add Compare Select*), atau yang juga dikenal dengan blok PMU (*Path Metric Unit*), terjadi proses penghitungan *path metric.* Penghitungan *path metric* (PM) dengan prosedur ACS ini dilakukan berulang untuk setiap *state*

1) Untuk menghitung *path metric* baru, jumlahkan *path metric* sebelumnya dengan *branch metric* yang bersangkutan
2) Dari penghitungan *path metric* diatas akan didapat dua buah *path metric* baru pada t+1 untuk setiap *state* (untuk menuju satu *state*, selalu ada dua transisi yang mungkin). Bandingkan dan pilih antara dua PM tersebut yang memiliki nilai lebih kecil pada masing-masing *state* dan membuang PM yang lainnya. Transisi yang dipertahankan ini disebut jalur *survivor*. Kemudian jalur ini disimpan.

2.1.2.3 *Traceback Unit*

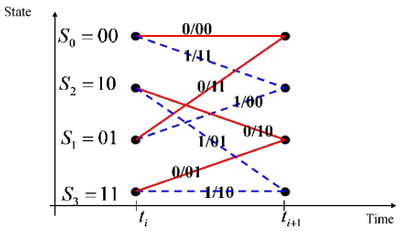
Setelah nilai *path metric* diperoleh untuk setiap *state*. Selanjutnya akan dilanjutkan dengan proses *traceback* untuk mengekstraksi urutan keluaran pendekode berdasarkan diagram *trellis*. Proses ini diawali dari *state* pada t pengamatan terakhir menuju *state* pada t-1, seterusnya hingga mencapai *state* pada t=0. Proses *traceback* ini berawal dr konsep bahwa setiap percabangan (*branch*) terkait dengan bit masukan pada pengkode.

Berikut adalah algoritma pendekode *Viterbi*. Dimana BM adalah *branch metric* dan PM adalah *path metric*

1. (a) inisialisasi waktu t=0
   (b) inisialisasi PM awal = 0 dan lainnya adalah +∞
2. (a) set waktu t=t+1
   (b) hitung *path metric parsial* untuk semua *path* akan membentuk *state* S_k pada waktu t. Pertama, hitung *branch metric* (BM). Kedua, hitung *path metric parsial* (PM) yang diperolah dari penjumlahan antara PM asal dengan BM yang berasosiasi dengan *state* tersebut. Dari sini akan didapat 2 PM untuk setiap *state*. Kemudian algoritma harus memutuskan transisi mana yang akan dipertahankan untuk setiap *state*, dengan memilih PM yang lebih kecil pada masing-masing *state*. Kemudian simpan jalur tersebut sebagai *survivor path*.

3. Jika t < L+m-1, kembali menuju langkah 2, banyaknya iterasi ini dikenal sebagai *traceback depth*. L adalah panjang informasi masukan, dan m adalah panjang maksimum *shift register*. Proses *traceback* dilakukan dengan menelusuri jalur *survivor*-nya dalam arah mundur dan mengekstrak *state-state* mana saja yang telah dikunjungi untuk mencapai *state* akhir yang memiliki akumulasi *path metric* paling kecil. Dengan mengetahui semua *state* yang dilalui oleh jalur *survivor*-nya, algoritma ini bisa membangun kembali deretan bit yang berkaitan dengan setiap transisi pada jalur *survivor*, sebagai keluaran dari pendekode itu sendiri.

Untuk menerjemahkan kembali bit-bit yang dikirim dari enkoder, maka pada dekoder diterapkan suatu teknik *trellis diagram*. Teknik ini merupakan salah satu teknik pendekodean yang terbukti handal. Diagram trellis mengadopsi dari teori perpanjangan *states diagram* yang menunjukkan telah melalui *t* tertentu. Dibawah ini akan ditunjukkan salah satu contoh bagian diagram trellis dengan *rate* ½ .



Gambar 2.5 *trellis diagram* dengan *rate* ½[9]

2.1.3 Perangkat Keras FPGA ATLYS Spartan-6 XC6SLX45 CSG324C

FPGA (*Field Programmable Logic* Array) merupakan suatu IC (*Integrated Circuit*) tipe HDL (*High speed IC Description Language*) yang dapat diprogram untuk melakukan fungsi-fungsi logika tertentu sesuai dengan kebutuhan. FPGA merupakan *Programable Logic Device* (PLD) yang dibangun dari sekumpulan sel fungsi logika dasar yang dapat diprogram. Sel-sel logika ini terhubung satu sama lain melalui suatu jaringan interkoneksi yang juga dapat diprogram.

Pada penelitian ini digunakan FPGA seri ATLYS Spartan-6 XC6SLX45 CSG324C dengan spesifikasi 1Gbit (64MB x 16) DDR2, 128Mbit x 4 Quad SPI Flash, *Multiple* HMDI *ports, Trimode Ethernet Interface, Audio Codec* dan USB-UART and USB-HID *host*.

Gambar 4.18 Spartan-6 XC6SLX45 CSG324C[10]

III. PERANCANGAN SISTEM TURBO DECODER

1.1 Diagram Alir Perancangan Sistem Turbo Decoder

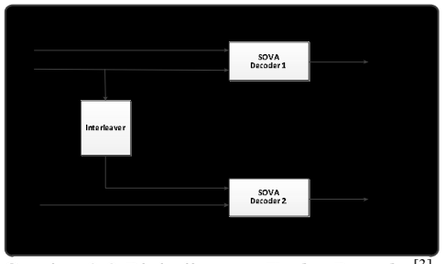
Pada sub bab ini dibahas tahapan-tahapan perancangan sistem *Turbo Decoder*. Perancangan sistem diawali dengan penentuan spesifikasi *Turbo Decoder*. Selanjutnya dibuat blok-blok sub sistem *Turbo Decoder* dari spesifikasi tersebut. Kemudian blok-blok sub sistem yang telah ada dibuat pada VHDL. Blok-blok sub sistem tersebut disatukan menjadi rangkaian sebuah *prototype Turbo Decoder*. Maka dari hasil perancangan ini didapatkan *source-source* VHDL dari dekoder yang dirancang.

Gambar 3.1 Diagram Alir Sistem *Turbo Decoder*

3.2 Penentuan Spesifikasi Sistem *Turbo Decoder*

Parameter sistem yang digunakan pada penelitian ini adalah :

- Kanal yang digunakan dianggap ideal
- Pendekatan algoritma yang digunakan adalah *Soft Output Viterbi Algoritm* (SOVA)
- Mode yang digunakan pada SOVA adalah QPSK
- Interleaver yang digunakan adalah matriks 4 x 4
- Input decoder adalah deretan bit yang di bangkitkan dari *generator* konvolusi dengan K = 7

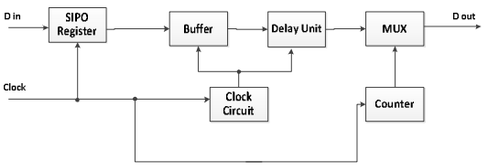


Gambar 3.2 Blok diagram *Turbo Decoder*[3]

3.2.1 Interleaver

Spesifikasi sistem interleaver yang digunakan pada penelitian ini adalah :

- Input interleaver adalah 16 bits data, merupakan keluaran dari *mapper* pada sisi *encoder*
- Matriks yang digunakan pada interleaver adalah 4 x 4
- Pola bits *interlevaer* berdasarkan standar 3GPP tentang LTE

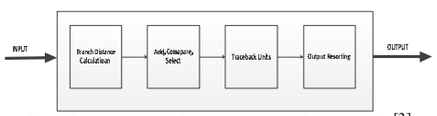


Gambar 3.4 *Scematic* bagian *Interleaver*[3]

3.2.2 SOVA Decoder

Spesifikasi SOVA yang digunakan pada sistem *Turbo Decoder* adalah :

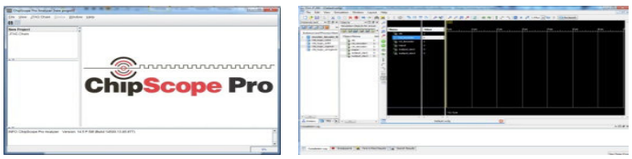
- SOVA dengan kode rate ½
- Register *memmory* pada SOVA adalah 28, berdasarkan *K* pada *encoder* dan memenuhi aturan *4K*
- Mode yang digunakan pada SOVA adalah QPSK



Gambar 3.6 Arsitektur dekoder *SOVA*[3]

3.2.3 Validasi Hasil simulasi *Software ISim* dan *Chipscope*

Sebelum dilakukan implementasi pada *board* FPGA Spartan-6 XC6SLX45 CSG324C, terlebih dahulu dilakukan proses validasi pada *software ISim* dan *chipscope*. Tujuan validasi ini adalah untuk mencocokkan hasil simulasi pada software ISim dan chipscope, bila hasilnya sama maka prototype *Turbo Decoder* dapat diimplementasikan pada *board* FPGA. Berikut ini adalah tampilan software *ISim* dan *chipscope*.



(a) (b)

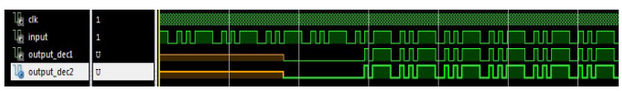
Gambar 3.10 (a) *software chipscope* (b) *software ISim*

IV. PENGUJIAN DAN ANALISIS SISTEM *TURBO DECODER*

Dalam bab ini, hasil perancangan yang telah dilakukan akan diuji. Pengujian yang dilakukan berkaitan dengan hasil simulasi dengan bahasa VHDL pada perangkat lunak *xilinx*.

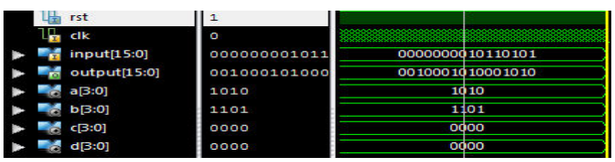
1.1 Simulasi Sistem pada Xilinx ISE 14.5

Untuk menguji sistem yang telah dirancang, digunakan sebuah *generator* konvolusi yang membangkitkan deretan bit tertentu. *Generator* konvolusi bekerja pada *clock rising edge*, artinya sistem akan aktif ketika clk berubah dari '0' ke '1'. Dari hasil simulasi diketahui bahwa untuk kanal ideal keluaran hasil dari output_dec1 dan output_dec2 adalah sama. Keseluruhan proses dari awal sampai didapatkan hasil keluaran dari *decoder* menghasilkan *delay* proses sebesar 30960 ns atau 3096 *clock*.



Gambar 4.3 Hasil Simulasi *Turbo decoder*

1.1.1 Simulasi Keluaran *Interleaver*

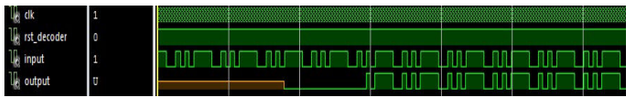


Gambar 4.4 Hasil Simulasi *Interleaver*

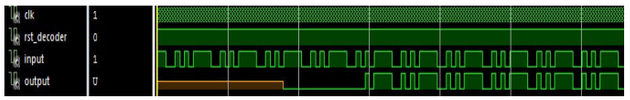
Gambar diatas menunjukkan proses interleaving dimana deretan 16 *bits* masuk akan di acak dengan pola tertentu dan kemudian akan dikeluarkan. *Bits* yang masuk akan disimpan sementara pada *signal* a,b,c, dan d dimana masing-masing dibagi menjadi 4 *bits*. Data masuk akan disimpan secara perbaris dan akan dibaca secara per kolom. Blok *interleaver* akan mulai membaca input ketika nilai rst '1' dan clk *rising edge*. *Delay proccess* yang dihaslkan adalah 1000 ns atau 100 *clock*.

4.1.2 Simulasi Keluaran *Soft Output Viterbi Algoritm* (SOVA)

Secara umum hasil simulasi blok SOVA adalah sebagai berikut :



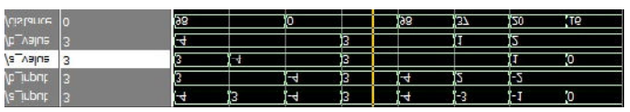
Gambar 4.5a Hasil Simulasi *Soft Output Viterbi Algoritm* (SOVA 1)



Gambar 4.5b Hasil Simulasi *Soft Output Viterbi Algoritm* (SOVA 2)

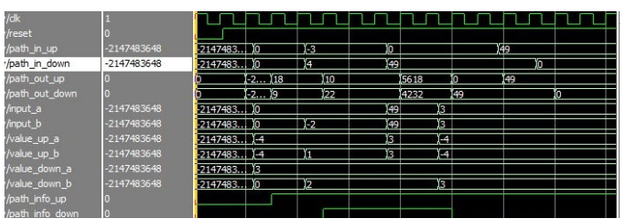
Blok ini memiliki 2 masukan, yaitu sinyal informasi dan *parity bits* hasil keluaran dari *encoder*. Bila dilihat dari gambar 4.5a dan gambar 4.5b keluaran dari SOVA adalah output, karena kanal yang digunakan adalah ideal keluran dari SOVA 1 dan SOVA 2 sama. Sistem bekerja rst_decoder bernilai '1' dan kondisi clk adalah *rising edge*. Keseluruhan proses pada blok SOVA menghasilkan *delay sistem* sebesar 29400 ns atau sama dengan 2940 *clock*.

Di bawah ini merupakan hasi simulasi dari masing-masing sub-blok penyusun *Soft Output Viterbi Algoritm* (SOVA).



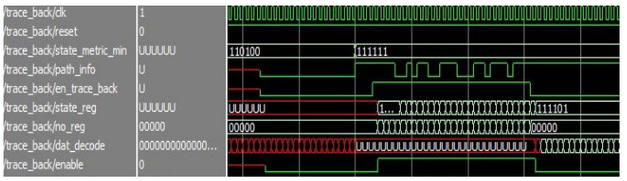
Gambar 4.6 Hasil Simulasi blok *Branch Distance Calculation* (BDC)

Pada bagian BDC terjadi proses perhitungan jarak *euclidean.* Blok BDC merupakan bagian pertama pada SOVA, oleh karena itu menerima masukan langsung dari kanal. a_input, b_input, a_value, dan b_value merupakan input dari BDC dan distance merupakan keluaran yang kemudian akan diproses pada blok *Add Compare Select.* Nilai distance yang ditunjukan pada gambar 4.6 dapat disesuaikan dengan perhitungan manual seperti yang telah dijelaskan pada Bab II.



Gambar 4.7 Hasil Simulasi blok *Add Compare Select*

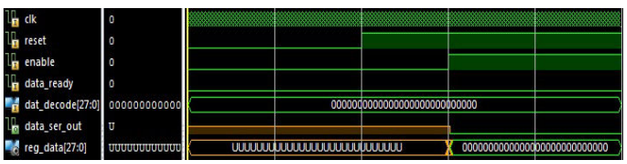
Input pada blok *Add Compare Select* adalah path_in dan distance. Distance merupakan keluaran dari blok *Branch Distance Calculation.* Keluaran dari blok *Add Compare Select* adalah path_out. Path_in dan distance merepresentasikan *state* awal sebagai *state* masukan, sedangkan path_out merepresentasikan hasil perpindahan *state* masukan. Pada gambar 4.7 juga ditampilkan *path info* yang akan digunakan pada proses *traceback* untuk menghasilkan bit terdekode sesuai dengan bit asli.



Gambar 4.8 Hasil Simulasi blok *Traceback*

Blok *traceback* berfungsi untuk menerjemahkan keluaran dari blok *Add Compare Select* yang sebelumnya telah disimpan dalam suatu blok *register memory*. Masukan yang dibutuhkan pada blok *traceback* ini adalah *state* akhir minimum yang dihasilkan oleh blok *Add Compare Select* yang disimpan secara *temporary* pada *register memory*. *State* akhir minimum tersebut, nantinya akan menjadi *state* awal dalam proses *traceback*. Blok *traceback* mulai bekerja saat sinyal *enable* aktif *high* atau bernilai '1'.

Waktu berlangsung *clock* tergantung dari *constraint length* pada *encoder*. Pada simulasi, *constraint length encoder* adalah 7, sehingga memenuhi aturan *traceback* yaitu 4*K,* sehingga *clock* berlangsung selama 28 *clock* atau sama dengan 280 ns, ditambah delay 1 clock sehingga *delay* total menjadi 29 *clock* atau sama dengan 290 ns.



Gambar 4.9 Hasil Simulasi blok *Output Resorting*

Pada blok *Output Resorting* terjadi proses pengubahan format keluaran dari blok *traceback* yang masih dalam format paralel untuk kemudian di ubah ke dalam format serial. Dari gambar 4.9 terlihat bahwa data_ser_out aktif ketika enable bernilai '1', reset bernilai '1' dan kondisi *clock* adalah *rising edge*.

1.2 Sintesis Sistem

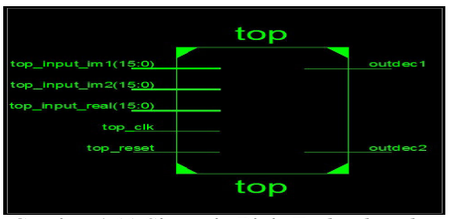
Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	12611	54576	23%
Number of Slice LUTs	13377	27288	49%
Number of fully used LUT-FF pairs	7105	18883	37%
Number of bonded IOBs	52	218	23%
Number of BUFG/BUFGCTRL/BUFHCEs	2	16	12%
Number of DSP48A1s	36	58	62%

Gambar 4.10 Hasil Sintesis Sistem *Turbo decoder*

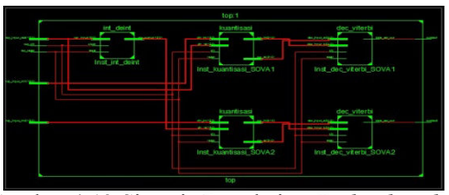
Dari hasil sintesis dapat diperoleh beberapa parameter yang dibutuhkan dalam implementasi pada FPGA. Dari gambar 4.10 diperoleh bahwa penggunaan *Number of Slices Registers* mencapai 23% dari jumlah resources yang tersedia. Setiap slices terdiri dari 2 LUT dan 2 flip flop. Dari gambar 4.8 terlihat bahwa *Number of Slice LUTs* adalah 49% dan *Number of fully used LUT-FF* adalah 37%, hal itu berarti desain yang telah dirancang lebih sebagai *combinational logic* daripada *sequential logic*.

Proses Sintesis juga memberikan hasil periode minimum dan frekuensi maksimum sistem. Periode minimum yang dihasilkan adalah 19.622 ns dan frekuensi maksimum

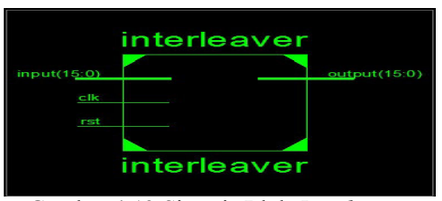
sebesar 50.963 MHz, sehingga dengan frekuensi tersebut *prototype Turbo decoder* dapat diimplementasikan dibawah frekuensi FPGA Spartan-6 yaitu 100 MHz.



Gambar 4.11 Sisntesis Blok *Turbo decoder*



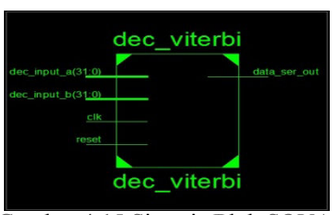
Gambar 4.12 Sintesis Rangkaian *Turbo decoder*



Gambar 4.13 Sintesis Blok *Interleaver*

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	32	54576	0%
Number of Slice LUTs	1	27288	0%
Number of fully used LUT-FF pairs	0	33	0%
Number of bonded IOBs	34	218	15%
Number of BUFG/BUFGCTRL/BUFHCEs	1	16	6%

Gambar 4.14 Hasil Sintesis Sistem Blok *Interleaver*



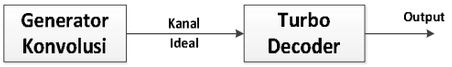
Gambar 4.15 Sintesis Blok SOVA

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	6308	54576	11%
Number of Slice LUTs	5518	27288	20%
Number of fully used LUT-FF pairs	3437	8389	40%
Number of bonded IOBs	67	218	30%
Number of BUFG/BUFGCTRL/BUFHCEs	2	16	12%
Number of DSP48A1s	24	58	41%

Gambar 4.16 Hasil Sintesis Sistem Blok SOVA

4.3 Pengujian pada Kanal Ideal

Pada skenario pengujian *Turbo decoder* dengan kanal ideal, digunakan sebuah *generator* konvolusi yang dihubungkan langsung dengan *decoder*. Artinya keluaran dari *generator* konvolusi akan langsung dijadikan masukan pada sisi *decoder*.



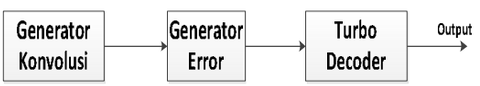
Gambar 4.1 Skenario Pengujian pada kanal ideal

Pengujian kanal ideal dilakukan dengan melakukan pengamatan pada data yang dikirim, dalam hal ini adalah 10 bit masukan yang diamati setelah pengkode benar-benar telah bekerja. Percobaan dilakukan dengan mengambil 10 data uji yang diberian secara acak pada input encoder. Pada kanal ideal 10 bit masukan yang diberikan pada encoder mampu di dekodekan dengan baik pada sisi *decoder*. Hal ini terlihat seperti pada tabel 4.1 bahwa output *decoder* 1 dan output *decoder* 2 menghasilkan keluaran yang sama seperti masukan pada encoder.

Percobaan	Input Encoder	Output *Decoder* 1	Output *Decoder* 2
1	0001010011	0001010011	0001010011
2	1100101011	1100101011	1100101011
3	1011010010	1011010010	1011010010
4	1110001110	1110001110	1110001110
5	1010100011	1010100011	1010100011
6	0100011011	0100011011	0100011011
7	0001101011	0001101011	0001101011
8	1100110100	1100110100	1100110100
9	1111000100	1111000100	1111000100
10	1111100000	1111100000	1111100000

1.3 Pengujian menggunakan *Generator Error*

Pengujian menggunakan *generator error* ditujukan untuk mengetahui kemampuan koreksi dari blok SOVA pada *turbo decoder*.



Gambar 4.2 Skenario pengujian dengan *generator error*

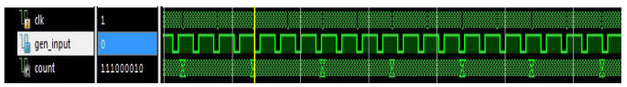
Pengujian dengan menggunakan *generator error* dilakukan untuk mengetahui kemampuan koreksi dari blok SOVA pada *Turbo decoder*. Pada pengujian ini ditambahkan sebuah *generator error* pada kanal diantara *generator* konvolusi dan *Turbo decoder*. *Generator error* dirancang dengan pola tertentu sehingga akan mengacak keluaran dari encoder yang kemudian akan dijadikan sebagai masukan pada *Turbo decoder*.

Gambar 4.17 Hasil *Generator Error*

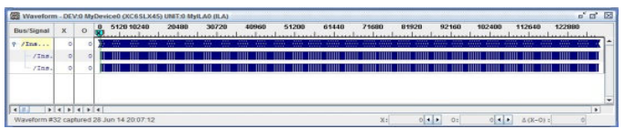
Data Encoder	Simbol Keluaran Encoder	Jumlah *Error*	Data yang diterima (kuantisasi generator error)	Data yang diterima (kuantisasi seharusnya)	Data Decoder
10101001	00 01 01 11 **11** 00 00 01	1	3,3 3,-4 3,-4 -4,-4 **0,-4** 3,3 3,3 3,-4	3,3 3,-4 3,-4 -4,-4 - **4,-4** 3,3 3,3 3,-4	10101001
10101001	00 01 01 11 **11 00** 00 01	2	3,3 3,-4 3,-4 -4,-4 **0,-4** **3,-1** 3,3 3,-4	3,3 3,-4 3,-4 -4,-4 - **4,-4 3,3** 3,3 3,-4	10101001
10101001	00 01 **01** 11 **11** 00 00 01	2	3,3 3,-4 **-1,-4** -4,-4 **0,-4** 3,3 3,3 3,-4	3,3 3,-4 **3,-4** -4,-4 - **4,-4** 3,3 3,3 3,-4	10101001
10101001	00 01 01 11 **11 00 00** 01	3	3,3 3,-4 3,-4 -4,-4 **0,-4** **3,-1 -1,3** 3,-4	3,3 3,-4 3,-4 -4,-4 - **4,-4 3,3 3,3** 3,-4	10101001
10101001	**00** 01 **01** 11 11 **00** 00 01	3	**-1,-1** 3,-4 **-1,0** -4,-4 - 4,-4 **-1,-1** 3,3 3,-4	**3,3** 3,-4 **3,-4** -4,-4 - 4,-4 **3,3** 3,3 3,-4	10101001

1.4 Validasi Data ISim dan Chipscope

Untuk mengetahui apakah program aplikasi dapat dijalankan dan berhasil mengolah data, maka digunakan sebuah software chipscope yang terdapat pada xilinx. Pada pengujian menggunakan *chipscope*, digunakan sebuah *prototype* blok tambahan yang berfungsi sebagai signal *generator*. Signal *generator* berfungsi untuk membangkitkan deretan bit secara acak pada periode waktu tertentu. Pada prototype signal *generator* dibangkitkan sinyal dengan bit '1' selama 448 clock atau 4480 ns, sedangkan bit '0' dibangkitkan selama 152 clock atau 1520 ns. Proses pembangkitan sinyal ini akan terus dilakukan selama clock dari sistem juga terus berjalan, dan kondisi *clock*nya adalah *rising edge*.



Gambar 4.24 Keluaran *Generator* input



Gambar 4.25 Keluaran *chipscope*

Dari gambar 4.25 terlihat bahwa dengan signal *generator* yang di bangkitkan secara acak, hasil keluaran outdec1 dan outdec2 dari chipscope selalu sama, hal itu serupa dengan keluaran outdec1 dan outdec2 ketika dilakukan simulasi dengan software *Isim* pada *xilinx*, sehingga dapat disimpulkan bahwa *prototype Turbo decoder* yang telah dirancang dapat diimplementasikan pada perangkat FPGA ATLYS Spartan-6 XC6SLX45 CSG324C dengan baik.

V. Kesimpulan dan Saran

5.1 Kesimpulan

Dari hasil penelitian yang dilakukan, dapat disimpulkan bahwa :

1. Perancangan *prototype Turbo Decoder* berdasarkan spesifikasi LTE yaitu coderate 1/3 dan *Chanel Coding* yang digunakan *Turbo Coding* telah berhasil dilakukan dan di implementasikan pada *board* ATLYS Spartan-6 XC6SLX45 CSG324C. Dari hasil simulasi, sistem dapat menjalankan fungsi dengan total waktu proses 30960 ns atau 3096 *clock*.
2. Hasil perancangan berhasil disintesiskan dan menghasilkan *resources Turbo Decoder* adalah 23% *Number of Slices Registers*, 49% *Number of Slice LUTs* dan 37% *Number of fully used LUT-FF*. Dengan presentase *Slice LUTs* yang lebih besar daripada *LUT-FF* maka *protoype* yang dirancang lebih sebagai *combinational logic* daripada *sequential logic*.
3. Sistem yang dihasilkan memiliki periode minimum 19.662 ns dan frekuensi kerja dibawah frekuensi kerja FPGA Spartan-6, yaitu 50.963 MHz. Dengan frekuensi tersebut yang masih dibawah ketersediaan resources pada FPGA, maka sistem dapat di implementasikan pada *board*.
4. Pada pengujian kanal ideal, dalam 10 kali percobaan menggunakan *generator* konvolusi, sistem mampu mendekodekan *bit* dengan baik, terbukti dari keluaran *decoder* yang sama dengan masukan pada *generator* konvolusi. Sedangkan pada pengujian dengan penambahan *generator* error, dengan masukan sebanyak 8 *bits*, sistem mampu mengoreksi kesalahan sebanyak 3 buah kesalahan.

5.2 Saran

1. Diharapkan pada penelitian berikutnya dapat ditambahkan suatu kanal acak dan blok tambahan sehingga memungkinkan proses iterasi dalam perhitungan *bit error rate*.
2. Dilakukan penelitian dengan pendekatan Algoritma lain seperti Log-MAP Algoritma atau MAP Algoritma.
3. Gunakan FPGA dengan spesifikasi yang lebih rendah, agar tidak banyak *source* yang tidak digunakan

Daftar Pustaka

[1] 3GPP *Long Term Evolution*. From http:// http://id.wikipedia.org/wiki/LTE [24 Juli 2014]

[2] Huahua, Wang dan Wenwen, Liu. (2012) "Analysis of Turbo Decoding Algorithm In LTE System". *9th International Conference on Fuzzy Systems and Knowledge Discovery*, 1741-1744.

[3] K N, Manjunatha; B, Kiran dan Kumar, Prasanna. 'Design and ASIC Implementation of a 3GPP LTE-Advance Turbo Encoder and Turbo Decoder,' *International Journal of Engineering Research and Applications (IJERA)*, Vol.2, No.4, pp. 006-010, July-August 2012.

[4] Kalyani, K; Vardhini, A. Sakthi Amutha dan Rajaram, S. 'FPGA Implementation of Turbo Decoder for LTE Standard' *Journal of Artificial Intelegence,* Vol.6, No.1, pp. 22-32, 2013

[5] ETSI TS 136 212 V10.0.0. (2011). *Multiplexing and channel coding (3GPP TS 36.212 version 10.0.0 Release 10)*. France

[6] Sklar, Bernard. *Fundamental of Turbo Codes*. Indiana: Prentice Hall, 2001. Print

[7] Meliani, H, Guellal, A. *Comparison between ViterbiAlgorithm Soft and Hard Decision Decoding*. KSA

[8] Bhanubhai, Patel Sneha; Shajan, Mary Grace dan Dalal, Upena D. 'Performance of Turbo Encoder and Turbo Decoder for LTE,' *International Journal of Engineering and Innovative Technology (IJEIT)*, Vol.2, No.6, pp. 125-128, October 2012.

[9] Lee, CheeYoon dan Wang, Zixiong. (1997). 'Viterbi Decoder for Convolutional Coded Signals' *Student CAD Doccumentation.* Departement of Electrical and Computer Engineering, University of Alberta. Not Published

[10] Atlys™ Spartan-6 FPGA Development Board. From http:// http://www.digilentinc.com/atlys [24 Juli 2014].

[11] Patmasari, Raditiana. 2012. *Desain Arsitektur dan Implementasi Pengkode Konvolusi dan Pendekode Viterbi dengan Soft Decision pada Aplikasi DVB*. Institut Teknologi Telkom : Tidak diterbitkan.