# Perancangan dan Simulasi Sistem Kompresi Suara dengan Transformasi DCT

**Syaiful Alam**
Jurusan Teknik Elektro Fakultas Teknik Univeristas Lampung
*alams@unila.ac.id*

***Abstrak**--Tulisan ini membahas perancangan sistem kompresi suara menggunakan bahasa perancangan perangkat keras VHDL dan tools XACT Step 6.0 dari Xilinx Inc. dan Proseries dari Viewlogic Systems Inc. Proses ini meliputi design entry, simulasi, baik secara fungsional maupun pewaktuan statis. Frekuensi klok maksimum yang dapat digunakan untuk sistem kompresi ini adalah 2,8 MHz. Hasil simulasi pewaktuan sistem memiliki delay maksimum 44,5 ns. Hasil kompresi mendekati nilai optimal.*

***Kata kunci** : kompresi suara, transformasi DCT, VHDL*

***Abstract**—This paper discuss design of sound compression system using VHDL syntax, XACT tools step 6.0, Xilinx Inc. and Proseries View Logic System. This process covered entry design, simulation by functionally or static clock. The compression system using the maximum clock frequency 2,8 MHz. The simulation result of clock system has maximum delay 44,5 ns. The compression result close to optimum value.*

***Keywords**: sound compression, DCT transformation, VHDL.*

## A. Pendahuluan

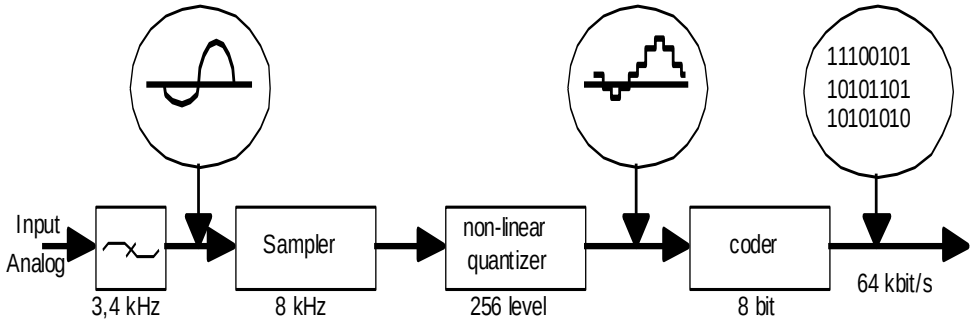
Ada beberapa metode yang telah dikembangkan untuk melakukan kompresi terhadap data audio khususnya suara. Metode-metode itu dapat dikelompokkan secara umum menjadi dua, yaitu *waveform coding* dan *analysis-synthesis*. Selain dua kelompok itu, beberapa literatur melakukan pengelompokan yang ketiga, yaitu kombinasi dari kedua metode di atas. Masing-masing metode tersebut memiliki kelebihan dan kekurangan.

Untuk aplikasi praktis komersial, metode-metode tersebut dapat diterapkan pada berbagai bidang, mulai dari keperluan pengembangan teknologi lebih lanjut, hiburan, hingga keperluan rumahtangga. Penerapan atas berbagai metode tersebut tentu juga sangat bergantung pada teknologi yang mendukungnya, sehingga diperlukan beberapa pertimbangan sebelum suatu metode diaplikasikan. Pertimbangan yang paling penting ditinjau dari aspek teknologinya adalah kemudahan implementasi dengan tetap memperhatikan tujuan aplikasinya. Pertimbangan ini tentu tidak dapat dilepaskan dari teknis pertimbangan-pertimbangan lainnya, disamping yang bersifat konseptual.

Kompresi sinyal suara dengan menggunakan transformasi DCT adalah salah satu metode yang memenuhi pertimbangan di atas. Jenis kompresi dengan transformasi ini dapat dikelompokkan ke dalam *waveform coding*. Salah satu kelebihan metode *waveform coding* adalah kualitas suara yang lebih tinggi.

Secara umum proses kompresi sinyal audio adalah terdiri atas pengolahan awal, kuantisasi dan pengkodean. Sedang untuk dekompresi adalah proses kebalikan proses kompresi sebelumnya. Dalam pengolahan awal inilah digunakan transformasi DCT yang telah dikenal mendekati sifat alihragam Hoblling/Principle Components yang optimal untuk pemampatan data, untuk kebanyakan isyarat yang dapat

Naskah ini diterima pada tangal 28 September 2007, direvisi pada tanggal 3 Nopember 2007 dan disetujui untuk diterbitkan pada tanggal 1 Desember 2007



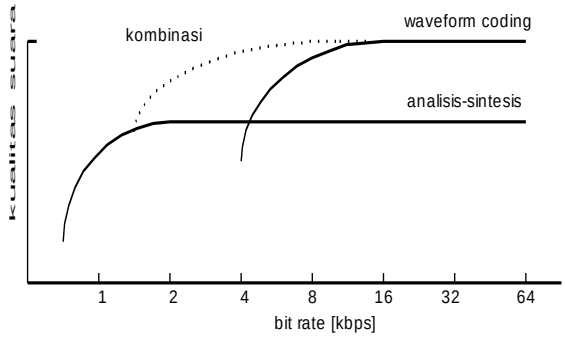

Gambar 1 Pengkodean *Pulse Code Modulation*

dimodelkan dengan runtun Markov. Sebelum disimpan ke dalam memori penyimpan, dilakukan pengkodean dengan *runlength encoding*.

**B. Tinjauan Pustaka**

Sebelum dicuplik, sinyal audio analog itu dilewatkan pada sebuah filter. Salah satu proses digitalisasi atau konversi sinyal analog ke digital ditunjukkan oleh Gambar 1 berikut.

Kompresi Sinyal Suara (Speech)




Gambar 2 Hubungan antara *bitrate* dengan kualitas suara (*speech*).[2]

Hubungan antara *bit rate* pengkodean dengan kualitas *speech* ditunjukkan oleh Gambar 2. Pada gambar terebut, terlihat bahwa metode analisis-sintesis dapat mengkodekan hingga sekitar 2,4 kbps, walaupun kualitas *speech* yang dihasilkannya terbatas. Pengkodean dengan *bitrate* yang lebih kecil dari ini, akan sangat mengurangi kualitas *speech*. Sementara itu, metode *waveform coding* memiliki kualitas *speech* yang lebih baik, meskipun kualitas ini hanya dapat dipertahankan hingga sekitar 16 kbps.

**Sistem Kompresi dengan Transformasi DCT**

Blok pengolahan sinyal digital untuk tujuan kompresi, secara umum terdiri atas *Signal Preprocessing, Quantization, Lossless Data Compression.*

Berdasarkan teori yang telah berkembang, data sinyal yang memiliki nilai dengan distribusi *Gauss* dan *uncorrelated*, dapat menghasilkan faktor kompresi yang terbaik. Karena itu usaha-usaha ditujukan pada *signal preprocessing* yang menghasilkan sinyal yang *uncorrelated* dan memiliki distribusi *Gauss*[2].

Data sinyal x[n] yang memiliki N cuplikan adalah *uncorrelated* jika fungsi autokorelasi :

$$R_{xx}(k) = \frac{1}{N} \sum_{i=0}^{N-|k|-1} x[i]x[i+|k|] \qquad (1)$$

adalah 0 atau sangat kecil untuk semua k kecuali pada k=0. Transformasi Fourier dari fungsi autokorelasi tersebut dikenal sebagai *power spectral density* (PDF) $S_{xx,}$ yang tersebar merata seragam.

Untuk menentukan apakah suatu sinyal dapat diproses menjadi *uncorrelated*, dapat dilakukan analisis terhadap *spectral flatness measure* (sfm) dari sinyal tersebut. Sfm suatu sinyal didefinisikan:

$$\gamma_x^2 = \frac{\eta_x^2}{\sigma_x^2} \qquad (2)$$

dengan:

$$\ln(\eta_x^2) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \ln\left(S_{xx}(e^{j\omega})\right) d\omega \qquad (3)$$

$$\sigma_x^2 = \frac{1}{N}\sum_n \left|x[n]\right|^2 \qquad (4)$$

Jika sfm sama dengan 1, maka sinyal mencapai kondisi *uncorrelated* dan tak ada *preprocessing* yang dapat memperbaikinya lagi. Pada ekstrem yang lain, jika sfm sama dengan 0, maka sinyal benar-benar *predictable*. Dengan demikian sinyal tidak perlu ditransmisikan atau disimpan, karena melalui proses tersebut sinyal dapat dibangkitkan kapan saja.

Ada dua metode *preprocessing* yang paling penting untuk menghasilkan sinyal *uncorrelated*, yaitu *predictions* dan *transforms* [3]. *Prediction* dapat dilakukan secara linear atau nonlinear. *Prediction* yang linear melihat suatu cuplikan sinyal x[n] sebagai kombinasi linear dari cuplik-cuplik sinyal lainnya. Misalnya, untuk *prediction* orde-P:

$$x[n] = \sum_{i=1}^{P} a_k x[n-i] \qquad (5)$$

Dengan menggunakan pendekatan ini, skalar $a_k$ menjadi representasi watak sinyal tersebut. Semua sinyal dapat direkonstruksi dari $a_k$ dan kondisi awal.

**Transformasi DCT**

*Discrete Cosine Transform* (DCT) merupakan jenis transformasi yang *signal independent* dan hampir mendekati kinerja KLT untuk sinyal audio [3].

Persamaan transformasi DCT satu dimensi yang sesuai dengan sinyal audio secara umum adalah :

$$X(k) = c(k)\sum_{n=0}^{M-1} x(n)\cos\frac{(2n+1)k\pi}{2M} \qquad (6)$$

dengan:

M : jumlah cuplikan
x(n) : cuplik *speech*
c(k) = 1 untuk k=0
c(k) = $\sqrt{2}$ untuk selain k=0

Persamaan ini mentransformasikan cuplik *speech* **x**(n) ke dalam matriks kolom *X*(*k*). Dari persamaan di atas dapat ditulis kembali dalam bentuk matriks dengan mengganti indeks k dengan i yang mewakili baris dan indeks n dengan j yang mewakili kolom, elemen-elemen dari matriks DCT tersebut.

$$K_{i,j} = 1, \text{ untuk i = 0} \qquad (7)$$

$$K_{i,j} = \sqrt{2}\cos\left(\frac{(2j+1)i}{2M}\right), \text{ untuk i} \neq 0 \qquad (8)$$

Untuk melakukan transformasi balik, digunakan persamaan:

$$x(k) = \frac{c(k)}{M}\sum_{n=0}^{M-1} X(n)\cos\frac{(2n+1)k\pi}{2M} \qquad (9)$$

Karakteristik hasil transformasi DCT adalah bahwa data yang dihasilkan akan tersusun sedemikian rupa sehingga nilai-nilai koefisien frekuensi rendah akan cenderung besar. Sementara nilai-nilai koefisien frekuensi tingginya akan mendekati nol.

Dengan proses kuantisasi, nilai-nilai koefisien tersebut akan dikelompokkan ke dalam suatu jangkauan nilai untuk kemudahan proses selanjutnya.

**Kuantisasi**

Untuk keperluan kompresi dengan transformasi DCT ini, formula yang digunakan adalah:

$$NilaiKuantisasi = Pembulatan\left[\frac{NilaiKoefisienDCT}{FaktorKuantisasi}\right] \quad (10)$$

Sementara itu, untuk proses dekuantisasi, nilai koefisien DCT diperoleh dengan mengalikan nilai kuantisasi dengan faktor kuantisasi. Terlihat bahwa dalam proses dekuantisasi, nilai koefisien yang diperoleh tidak benar-benar sama dengan nilai asalnya.

Karena ukuran matriks transformasi yang digunakan adalah 16x16, maka setiap saat, jumlah cuplikan yang dapat dikalikan dengan matriks ini adalah 16. Hasil dari setiap kali perkalian keenambelas cuplikan ini dengan matriks koefisien merupakan hasil transformasi dari cuplikan-cuplikan tersebut. Dengan demikian, untuk setiap blok data *speech* yang berjumlah 256 cuplikan, terdapat 16 kali transformasi.

**Pengkodean**

Untuk kompresi *speech* menggunakan transformasi DCT, algoritma yang paling cocok digunakan adalah RLE [4]. Disamping karena algoritmanya relatif sederhana, juga pengolahan awal yang terdiri atas transformasi DCT, pengubahan urutan data dan kuantisasi, memberikan data yang memungkinkan pengkodean yang efisien untuk RLE.

Algoritma RLE tersebut adalah sebagai berikut:

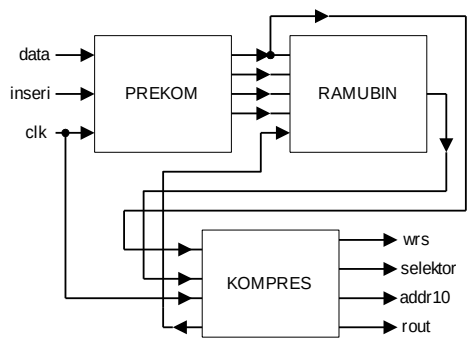
- Pembacaan data awal, *runlength* diset sama dengan nol dan pemberian alamat pada data tersebut.
- Pembacaan data berikutnya. Jika data baru sama dengan data sebelumnya, maka *runlength* dinaikkan satu (*increment*) dan alamat tidak berubah. Jika data baru tidak sama dengan data sebelumnya, maka alamat dinaikkan satu dan *runlength* direset ke nol. Demikian seterusnya hingga data terakhir.

## C. Perancangan

Rancangan sistem, didasarkan pada blok kompresi secara umum yang menggunakan metode *Transform Coding*, seperti yang telah di jelaskan pada bagian Tinjauan Pustaka.

Pada modul kompresi terdapat modul Prekom, modul Ramubin dan modul Kompres sebagai modul inti dalam proses kompresi. Modul Prekom berfungsi mengkonversi input suara serial (**inseri**) menjadi paralel disamping menghasilkan alamat dan sinyal-sinyal penulisan ke ram bufer input (modul Ramubin) yang berkapasitas 256x1 byte. Disamping data serial suara, input modul Prekom adalah sinyal header data suara (**data**) yang segera mendahului data tersebut setiap 256 byte data.

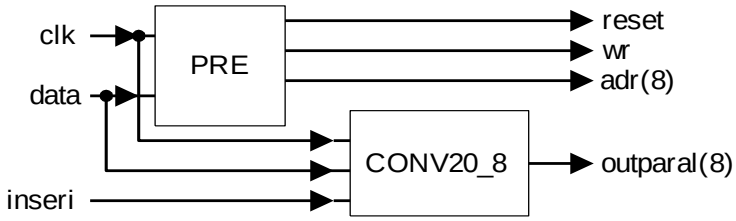
Modul Kompres adalah modul pengolah data untuk kompresi. Inputnya berupa data paralel keluaran bufer input. Sementara outputnya adalah data (8 bit) yang terkompresi dengan pengkodean RLE (**rout**) dan siap disimpan pada memori hasil kompresi.



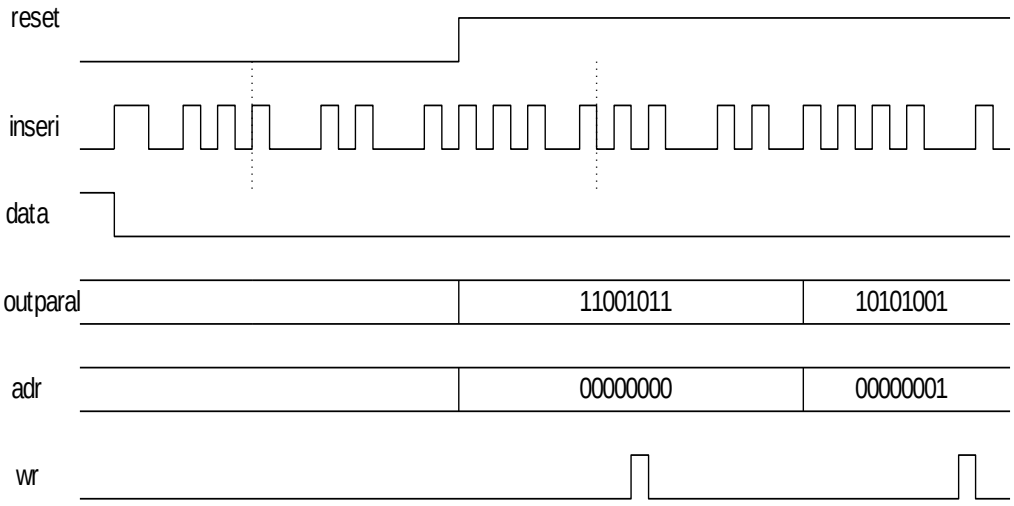

Gambar 3. Modul Penyusun Sistem Kompresi

**Modul Prekom**

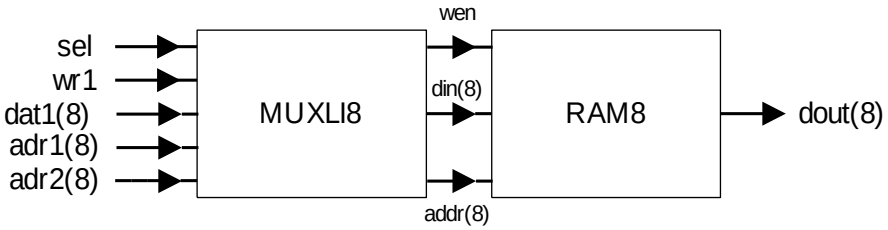
Modul Prekom terdiri dari beberapa submodul. Submodul utamanya adalah



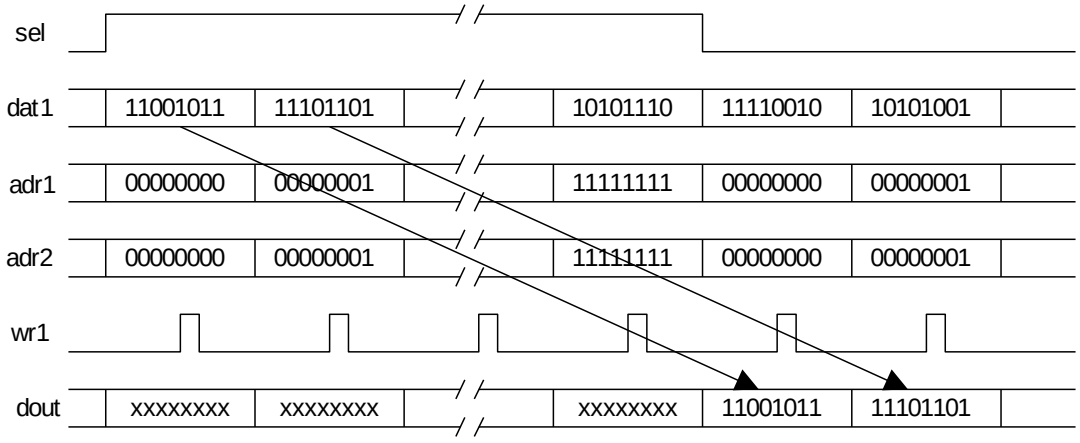

Gambar 4a. Modul Penyusun Prekom



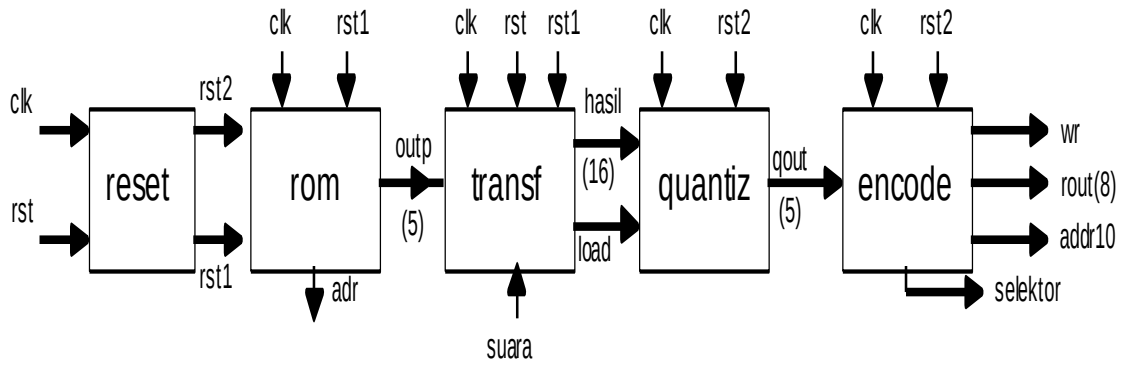

Gambar 4b. Diagram Waktu Prekom




Gambar 5a. Modul Penyusun Ramubin




Gambar 5b. Diagram Waktu Ramubin




Gambar 6. Modul Penyusun Kompres

modul pembangkit sinyal kontrol dan modul pengkoversi data serial ke paralel.

Modul penyusun modul Prekom dan diagram waktu yang diharapkan dari modul ini diperlihatkan pada gambar berikut.

**Modul Ramubin**

Modul Ramubin terdiri atas dua submodul. Modul pertama berupa multiplekser yang memilih mode operasi ram. Modul ram dibangun dari beberapa ram yang telah tersedia pada library ieee.std_logic_1164.

Setelah proses penulisan ke dalam ram input bufer ini, maka modul ini akan mengandung 256 byte data suara yang dianggap sebagai satu blok data yang siap diolah.

**Modul Kompres**

Sistem ini terdiri atas lima modul utama seperti yang diperlihatkan pada Gambar 6. Input modul kompres adalah 8 bit paralel yang merepresentasikan nilai **suara**.

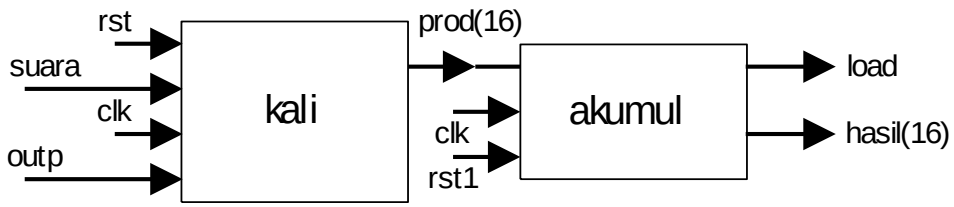
Modul Rom di atas membangkitkan alamat (**adr**) untuk membaca isi modul Ramubin yang berupa data suara. Pada modul Ramubin, **adr2** dihubungkan dengan **adr**. Melalui alamat inilah isi ram bufer input dibaca dan diinputkan ke modul Transf.

Perancangan blok transformasi Transf didasarkan pada Persamaan (2.6). Persamaan ini dapat ditulis ulang dalam bentuk matriks : $\boldsymbol{X}(k) = \boldsymbol{K}_{i,j} \cdot \mathbf{x}(n)$, dimana $\boldsymbol{K}_{i,j}$ merupakan matriks bujursangkar dari koefisien-koefisien DCT dan $\mathbf{x}(n)$ adalah matriks kolom sampel suara.

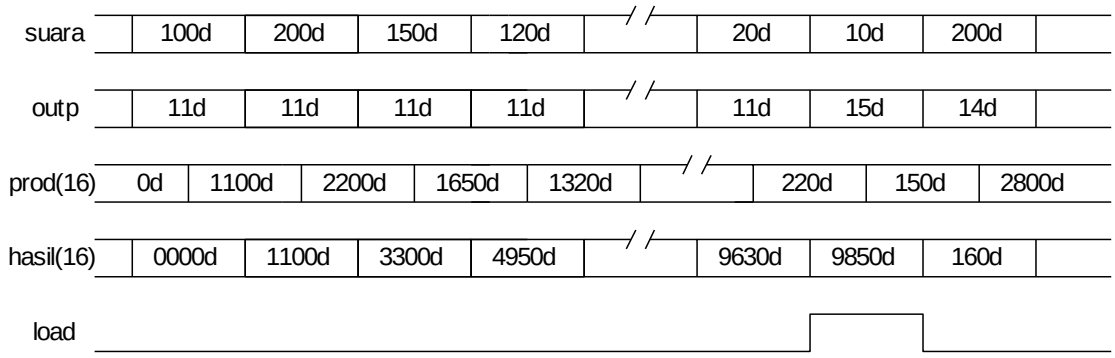
Pengolahan pertama yang dilakukan terhadap data suara adalah perkalian dengan matriks koefisien DCT yang disimpan dalam modul Rom yang merupakan bagian modul kompresi. Agar dalam pengolahan selanjutnya data tersebut langsung siap dikodekan sebelum disimpan dalam memori penyimpanan, maka perkalian dilakukan dengan cara mengakses bufer input sebanyak 16 kali secara berturut-turut. Dengan demikian satu blok data suara yang disimpan dalam input bufer tersebut, perlu dipertahankan selama proses perkalian berlangsung. Berikut ini adalah gambaran perkalian matriks tersebut, dengan k adalah koefisien dan s adalah data suara.

$$\begin{bmatrix} k_{1,1} & \cdot & k_{1,16} \\ \cdot & \cdot & \cdot \\ k_{16,1} & \cdot & k_{16,16} \end{bmatrix} \begin{bmatrix} s_{1} & \cdot & s_{241} \\ \cdot & \cdot & \cdot \\ s_{16} & \cdot & s_{256} \end{bmatrix}$$

Gambar 7. Proses Perkalian dalam Modul Transf




Gambar 8a. Modul Penyusun Transf




Gambar 8b. Diagram Waktu Transf

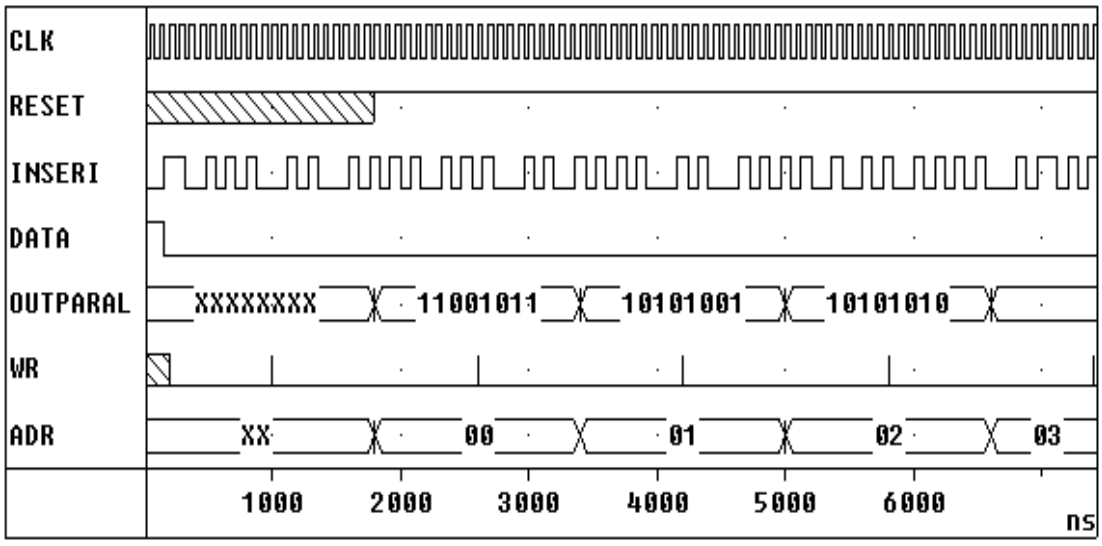
Perkalian antara matriks koefisien DCT dengan input sampel suara dilakukan dengan modul **transf**. Pada prinsipnya, cara kerja modul ini diadaptasi dari proses perkalian matriks.

Mula-mula dilakukan perkalian titik antara baris pertama dengan matriks kolom sampel suara untuk mendapatkan elemen pertama hasil transformasi. Demikian juga untuk baris kedua, dan seterusnya. Proses ini dalam rancangan dibuat dengan modul perkalian dan modul akumulasi. Hasil perkalian antara elemen matriks dengan elemen sampel suara diakumulasikan untuk setiap baris, sehingga akan dihasilkan elemen pertama hasil transformasi setelah akumulasi enam belas penjumlahan dari hasil perkalian tersebut atau setelah enam belas perkalian dan lima belas kali penjumlahan untuk setiap baris. Demikian seterusnya untuk setiap elemen hasil transformasi. Berikut ini adalah modul-modul penyusun modul Transf dan diagram waktu yang dirancang untuk modul tersebut.

Modul Akumul melakukan akumulasi terhadap hasil perkalian yang dilakukan oleh modul sebelumnya. Setelah melakukan perkalian matriks baris pertama pada matriks koefisien dengan enam belas sampel suara, akan dibangkitkan sinyal **load**.

Pada rancangan ini, kuantisasi dilakukan dengan faktor kuantisasi fq=2048. Pemilihan nilai faktor kuantisasi tersebut didasarkan pada pertimbangan spesifikasi rancangan dan kemudahan pelaksanaannya. Operasinya adalah dengan melakukan pembagian dan pembulatan terhadap hasil akumulasi dari modul sebelumnya. Hasil kuantisasi tersebut akan berlangsung selama 16 klok, yaitu selama proses perkalian dan akumulasi dari 16 hasil perkalian.

Selain melakukan pengkodean, modul **encoding** ini juga menyiapkan penulisan ke memori penyimpanan. Karena itu selain




Gambar 9. Hasil Simulasi Modul Prekom

menghasilkan data hasil kompresi (**rout8)**, juga menghasilkan alamat penyimpanan dan sinyal penulisan.

**D. Simulasi**

**Simulasi Fungsional**

Langkah-langkah yang ditempuh dalam simulasi fungsional adalah sebagai berikut:

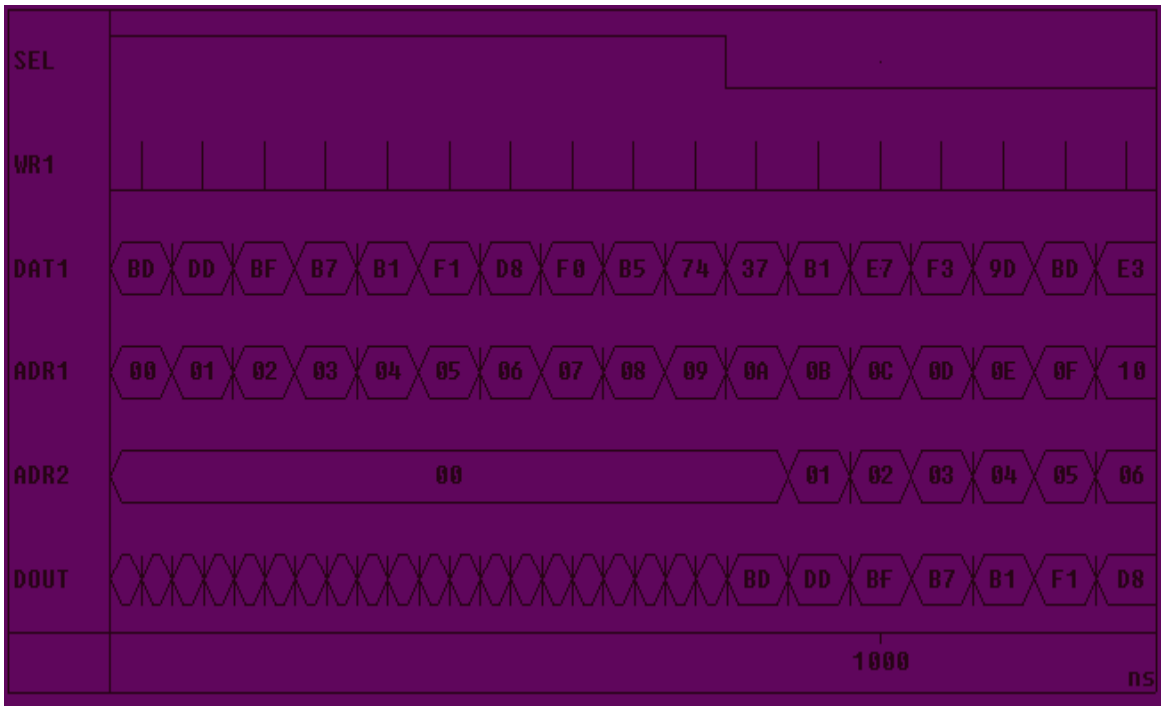
- Skematik rangkaian yang dihasilkan dari proses sintesis dibuka dalam *Procapture*.
- Setalah itu *Prosim* dipanggil dengan salah satu dari dua cara, yaitu dengan memilih *tools* "*link to Prosim*" dari *Procapture*, atau dengan langsung memilih *Prosim* dari *Xilinx Proflow*.
- Bila *Prosim* dipanggil dari *Xilinx Proflow*, maka dipilih *option* untuk menjalankan *netlister* yang akan membuat jaringan dari rangkaian yang akan disimulasi. Informasi jaringan dari rangkaian ini disimpan dalam *file* *.*vsm*
- Setelah *netlister* selesai dijalankan dan *.*vsm* telah tersedia, maka program akan membuka *Notepad* untuk memperlihatkan *error* atau *warning* yang terjadi selama *netlister* dijalankan. Jika *notepad* ini ditutup, maka *Prosim* akan langsungdijalankan dan *file* *.*vsm* langsung dibuka. Bila *Prosim* dipanggil dari *Procapture, Prosim* akan langsung dijalankan tanpa membuka *notepad* setelah *netlister* selesai.
- Setelah langkah-langkah persiapan simulasi di atas dilaksanakan, maka rancangan siap disimulasi dengan vektor tes yang telah terlebih dahulu disiapkan dalam bentuk *file* *.*cmd.* Sebaiknya *notepad* hasil simulasi tidak ditutup, untuk mempermudah pengecekan fungsi logika pada tiap komponen rangkaian rancangan.
- Diagram waktu hasil simulasi fungsional dapat dilihat melalui *Prowave.*

**Analisis Hasil Simulasi Fungsional**

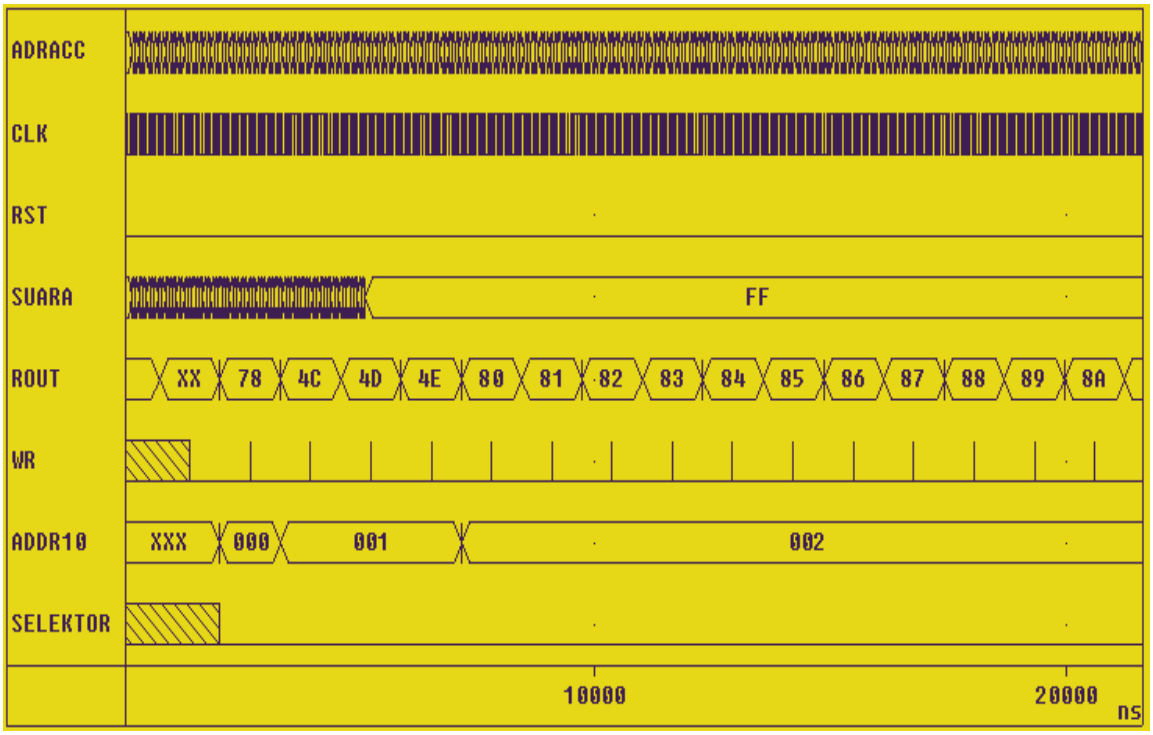
Dalam bagian ini akan dijelaskan analisis terhadap hasil simulasi fungsional pada modul-modul penyusun sistem kompresi dan dekompresi suara yang telah dirancang pada Bab 3.

**Modul Prekom**

Dalam simulasi fungsional ini, karena digunakan hanya untuk mengetahui kebenaran logika rangkaian, frekuensi klok yang digunakan tidaklah penting.




Gambar 10 Hasil Simulasi Modul Ramubin




Gambar 11 Hasil Simulasi Modul Kompres

Sesuai dengan rancangan yang telah dijelaskan pada bab yang lalu, sinyal **data** yang merupakan salah satu input pada rancangan modul kompresi, dianggap sebagai sinyal yang mendahului data input serial suara. Jika tepi naik master klok

mendapatkan sinyal ini, maka akan dibangkitkan dua sinyal lainnya yaitu; sinyal pencacah akan mulai menghitung input serial yang dibaca melalui input **inseri**, sinyal klok dengan periode 20 periode klok master untuk membangkitkan sinyal **wr**. Klok yang terakhir inilah yang digunakan untuk membangkitkan sinyal **reset** dan **adr** dengan menggunakan pencacah pada tepi klok turun. Sementar itu, konversi bit input serial ke paralel dilakukan dengan membaca input serial tersebut bit per bit dan memasukkannya ke dalam suatu register 20 bit melalui LSBnya dengan menganggap bit pertama sebagai MSB data suara. Dan isi register ini akan digeser ke kiri setiap kali pembacaan bit input pada tepi klok naik. Pengisian register ini dikontrol oleh sebuah pencacah. Jika register telah terisi penuh yaitu pada saat (pencacah="10011"), maka isi register tersebut ditransfer ke output 20 bit.

Komponen selanjutnya pada modul ini adalah pengkode 20 bit menjadi 8 bit. Komponen ini bekerja asinkron. Ia akan langsung mengkodekan setiap kali ada input 20 bit menjadi 8 bit.

**Modul Ramubin**

Modul ini adalah ram input bufer modul kompresi. Seperti telah dijelaskan pada bagian perancangan, ram ini bekerja asinkron. Karena itu setiap kali ada sinyal **wr** untuk penulisan, selama ram berada pada mode penulisan, ram akan ditulisi dengan data yang telah siap pada inputnya di alamat yang telah ditentukan. Begitu juga pada mode pembacaan. Setiap kali ada alamat yang menjadi masukan modul ini, ram segera mengoutputkan data dari alamat itu.

Modul ini berada dalam mode penulisan jika **sel**='1' dan sebaliknya, jika **sel**='0', ram dalam mode pembacaan. Pada hasil simulasi di atas juga terlihat bahwa sinyal wr1 tidak berpengaruh pada proses pembacaan ram.

**Modul Kompres**

Modul ini merupakan salah satu modul pada hirarki menengah yang membawahi beberapa modul lainnya. Seperti yang telah disebutkan pada bagian perancangan, modul ini melakukan kompresi terhadap data paralel 8 bit (**suara**) yang diterima dari modul pengkonversi data serial ke data paralel 8 bit (**rout**). Hasil kompresi tersebut kemudian disimpan pada sebuah ram penyimpan dengan menggunakan sinyal **wr** untuk penulisan pada alamat **addr10** dan sinyal **selektor** untuk menentukan mode penulisan ke ram. Di sini digunakan sinyal penulisan dengan **selektor=**'0'. **Adracc** adalah alamat pembacaan data yang disimpan dalam modul Ramubin.

**Simulasi Pewaktuan**

Setelah memastikan rancangan benar secara logika dengan simulasi fungsional, tahap selajutnya dalam rangkain verifikasi adalah simulasi pewaktuan. Simulasi ini dapat dilakukan setelah melalui tahap "implementasi" yang disebut *Partitioning Placement and Routing* (PPR) rancangan pada divais target.

Pilihan *generate timing report* ketika memulai PPR akan memanggil program *Xdelay* untuk melakukan analisis pewaktuan statis dan hasilnya akan disimpan pada *file* .xrp sebagai salah satu hasil proses PPR.

File ini mengandung informasi frekuensi pada *critical path*. Frekuensi ini menyatakan batas frekuensi maksimum yang dapat diterapkan pada rancangan.

**Proses PPR** [5]

PPR dilakukan dengan memangggil *Design Manager* dari *Xilinx Proflow*. Proses pertama sebelum PPR adalah proses

translasi. Proses ini akan mengubah format file rancangan input menjadi *file XNF* (*Xilinx Netlis Format*). *File* inilah yang digunakan sebagai masukan untuk PPR. Dalam proses ini juga dilakukan pemilihan terhadap jenis LCA yang dipakai dan tingkat kecepatannya. Pada penelitian ini dipakai LCA jenis XC4010 dengan tipe *package* PC84 dan tingkat kecepatan -4.

Setelah proses translasi berhasil dan lengkap, maka PPR dapat segera dilakukan. Dalam pelaksanaan PPR, ada beberapa *option* yang dapat dipilih untuk mengatur jalannya proses PPR ini.

- *Placement Effort*
- *Routing Effort*
- *Use XACT Performance*
- *Trim Unconnected Signal*
- *Use Global Resources for High Fan-out Signal*
- *Create RPMs for Registered Based Logic*
- *Merge Flip-flop into I/O*

Selain opsi-opsi di atas, masih terdapat opsi lain yang berhubungan dengan *Guide/Resource*. Opsi ini akan berguna pada saat melakukan PPR dengan menggunakan *guide file.* Melakukan PPR dengan *guide file* berarti menuntun proses PPR pada saat *placement* dan *routing.* Hal ini dapat dilakukan misalnya dengan memasukkan informasi *placement* dan *routing* dari hasil PPR yang telah dilakukan sebelumnya dan dianggap baik.

Jika dalam rancangan terdapat bagian yang kritis, maka bagian ini dapat di-PPR terlebih dulu. Kemudian bagian ini dapat dimasukkan ke proses PPR selanjutnya sebagai *guide file*, sehingga tidak diperlukan pengubahan terhadap hasil *placement* atau *routing* yang sebelumnya telah berhasil dengan baik.

Untuk rancangan modul kompresi ini digunakan pilihan *placement effort*=4 dan *routing effort*=4 serta *use global resources fan out signal*. Sedang plilihan-pilihan lainnya dibiarkan dalam keadaan *default.*

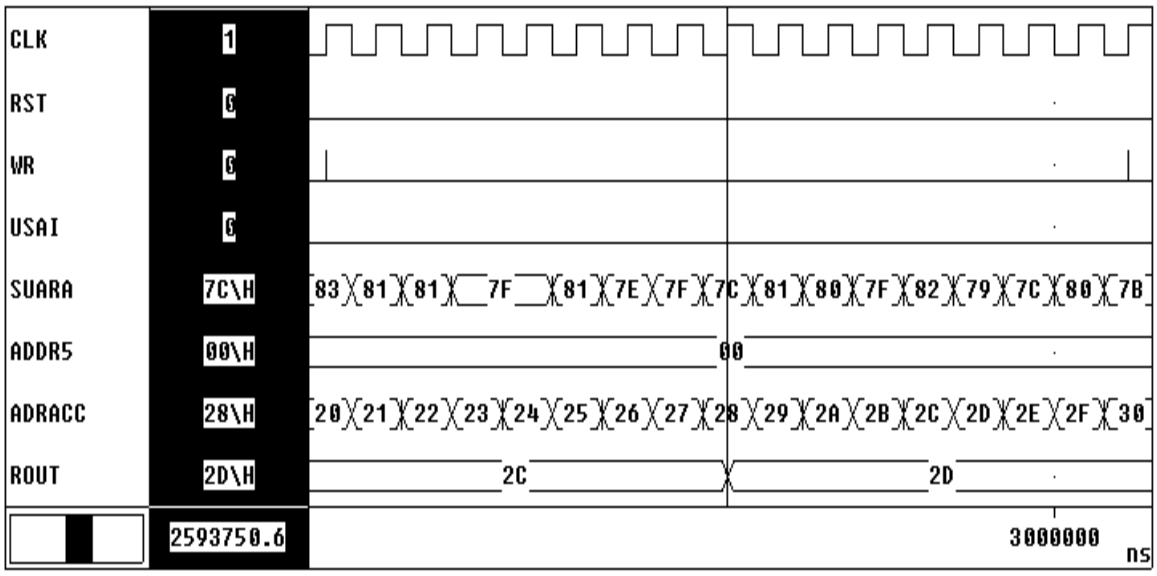
**Analisis Simulasi Pewaktuan**

Pengujian kedua terhadap rancangan sistem kompresi setelah di-PPR adalah dengan simulasi pewaktuan. Berbeda dengan simulasi fungsional, pada simulasi ini telah dipertimbangkan frekuensi maksimum yang diizinkan untuk rancangan ini. Informasi ini diperoleh dari salah satu *report* hasil PPR (file.xrp).

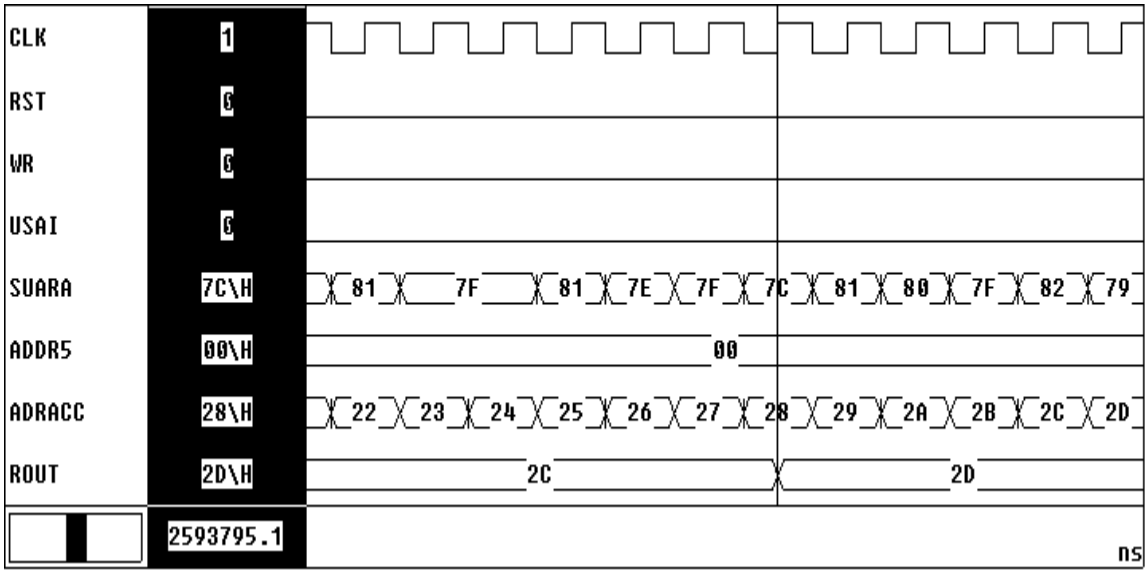
Dalam melakukan PPR, dapat dipilih opsi "*generate timing report*" yang akan memanggil program *Xdelay* untuk melakukan analisis pewaktuan statis. Hasil analisis yang dilakukan *Xdelay* disimpan dalam *file* *.xrp. *File* ini dapat dilihat dengan membuka *timing report* dari *Design Manager*. Analisis *Xdelay* yang diperoleh dari PPR yang dilakukan, dapat dilihat pada bagian lampiran.

Dari hasil analisis pewakuan statis diperoleh bahwa *critical path* untuk bagian kompresi (modul Kompres) yang telah dirancang adalah pada 2,8MHz . Dengan demikian, frekuensi klok maksimum yang dapat digunakan untuk sistem kompresi ini adalah 2,8 MHz.

Dengan data pewaktuan yang diperoleh dari proses PPR, simulasi pewaktuan dapat dilakukan dengan memperhitungkan *delay* pada rangkaian. Diagram waktu yang dihasilkan pada simulasi pewaktuan sama dengan digram waktu yang dihasilkan oleh simulasi fungsional. Hasil simulasi pewaktuan sistem kompresi memiliki *delay* maksimum 2593795,1-2593750,6 = 44,5 ns yang dapat dilihat dari waktu yang dibutuhkan oleh sinyal untuk berubah setelah sinyal penyebabnya berubah. Diagram waktu hasil simulasi pewaktuan tersebut dapat dilihat pada gambar berikut.




Gambar 12a Hasil Simulasi Fungsional Modul Kompres




Gambar 12b Hasil Simulasi Pewaktuan Modul Kompres

## E. Simpulan

1. Rancangan telah berhasil ditinjau dari simulasi fungsional dan pewaktuan menggunakan VHDL untuk tujuan kompresi data.
2. Hasil simulasi pewaktuan sistem memiliki *delay* maksimum 44,5 ns yang diperoleh dari selisih antara hasil simulasi pewaktuan dan simulasi fungsional.
3. Hasil kompresi mendekati nilai optimal.

## Daftar Pustaka

[1]. Armstrong, J.R., F.G.Gray, 1993, '*Structured Logic Design with VHDL'*, Prenctice Hall

[2]. Furui, Sadaoki, 1989, *'Digital Speech Processing, Synthesis and Recognition',* Marcel Dekker Inc.

[3]. Langi, Z.R.Armein, 1997, *'Audio Processing and Compression',* Course note no.02/DSPRestech/ITB.

[4]. Nelson, M., J.L.Gailly, 1996, *'The Data Compression Book 2nd Edition'*, M&T Books.

[5]. Xilinx, 1996, *'Hardware & Peripherals User Guide XACT Step,* Xilinx.