BAB 3
ANALISIS DAN PERANCANGAN

3.1 Analisis Perangkat Lunak

Analisis perangkat lunak dapat didefinisikan sebagai penguraian dari suatu perangkat lunak yang utuh ke dalam bagian-bagian komponennya dengan maksud untuk mengidentifikasikan dan mengevaluasi permasalahan-permasalahan, kesempatan-kesempatan, hambatan-hambatan yang terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikannya.

3.1.1 Analisis Masalah

Setelah menemukan beberapa penelitian yang telah ada sebelumnya mengenai kompresi video HEVC dengan berbagai macam teknik maupun algoritma yang digunakan dalam proses kompresi maka ditemukan masalah yang akan diperbaiki oleh penggunaan algoritma lain dan juga kombinasi antara satu algoritma dengan algoritma lain.

Mengacu pada penelitian sebelumnya tentang *encode* H.265/HEVC sebelumnya yang memanfaatkan fase korelasi antara current blok dengan blok setelahnya untuk mengekstrak 3 fitur gerak berbeda yang berfokus pada tiga aspek yang berbeda pula dari gerak masing-masing CUs (Control Unit). Dari metode yang diusulkan didapat bahwa metode tersebut menghemat 1% *bit rate* atau meningkatkan 0.15dB PSNR di atas rata-rata dan juga mengurangi sekitar 30% waktu *encoding* [2].

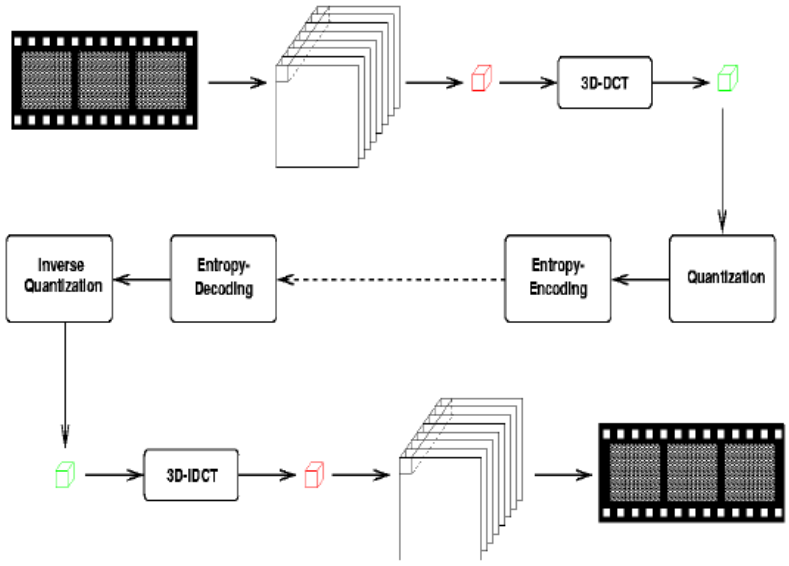
Dengan alasan demikian maka dikembangkanlah suatu kompresi video H.265/HEVC dengan menggunakan algoritma 3D-DCT pada proses *transform*-nya dan dikombinasikan dengan algoritma *fast coding unit decision* yang diterapkan pada proses partisi CU untuk menghasilkan peningkatan optimasi waktu yang lebih baik dari penelitian sebelumnya.

3.1.2 Analisis Metode

Analisis metode digunakan untuk mengetahui alur proses dari sebuah metode yang digunakan dapat diterapkan ke dalam aplikasi yang dibangun. Algoritma yang akan digunakan dalam pembangunan kompresi video ini yaitu algoritma 3D-DCT yang diterapkan pada proses *transform* dan algoritma *Fast Coding Unit Decision* yang diterapkan pada proses partisi CU (*Coding Unit*).

3.1.2.1 Analisis Algoritma 3D-DCT *(Three Dimension-Discrete Cosine Transform)*

Algoritma 3D-DCT (*Three Dimension-Discrete Cosine Transform*) pada proses kompresi video ini berjalan pada proses terakhir di sisi *encoder* yaitu proses transform. Berikut ini adalah tahap-tahap algoritma 3D-DCT yang berjalan pada sisi proses encoder dan decoder kompresi video seperti **Error! Reference source not found.** yang telah dipaparkan di bab 2 :



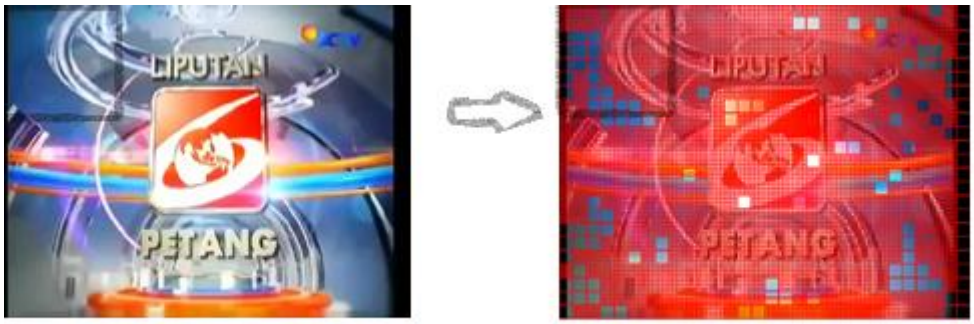
Dalam proses transformasi block images ke dalam 3D-DCT berlaku rumus seperti Persamaan (2.7) :

$$C\ (i,j) = \begin{cases} \frac{1}{\sqrt{N}} & if\ i{=}0 \\ \sqrt{\frac{2}{N}}\ x\ cos\frac{(2j+1)i\pi}{2N} & if\ i{\neq}0 \end{cases}$$

Sebagai ilustrasi transformasi coding MB [8x8] dimana akan mentransmisikan bit-bit biner saja ‘0’ atau ‘1’, maka akan dibahas proses yang terjadi dari setiap block pada *transform coding*.

Berikut adalah proses yang terjadi dari setiap blok pada *transform coding* menggunakan algoritma 3D-DCT :

1. Sebelum dilakukan proses *transform* pada proses *encode* kompresi video telah dilalui proses *partition* dimana sebuah masukan video dibagi menjadi beberapa frame atau gambar terlebih dahulu.
2. Gambar dibagi menjadi beberapa blok, dan masing-masing blok memiliki 8 piksel x 8 piksel.



(a) (b)

Gambar 3. 1. (a) Data Citra Original ; (b) Data Citra yang Telah Dikelompokkan Menjadi Beberapa Blok

3. Data matriks *original* dikurangi dengan 128 karena algoritma 3D-DCT bekerja pada rentang -127 sampai 127 sesuai dengan ketentuan pengolahan citra digital pada citra berwarna. Matriks original dapat dilihat pada Gambar 3. 2.

104	101	99	97	95	95	96	99
101	96	95	95	96	98	99	99
99	92	91	92	97	102	103	99
98	90	89	91	98	104	104	98
98	90	89	92	97	103	104	98
99	91	90	92	97	102	103	98
99	92	91	93	97	102	102	98
100	93	92	94	98	102	102	98

Gambar 3. 2. Original Image [8x8]

Matriks *original* yang sudah dikurangi dengan 128 dapat dilihat pada Gambar 3. 3.

$$X = \begin{vmatrix} -24 & -27 & -29 & -31 & -33 & -33 & -32 & -29 \\ -27 & -32 & -33 & -33 & -32 & -30 & -29 & -29 \\ -29 & -36 & -37 & -36 & -31 & -36 & -25 & -29 \\ -30 & -38 & -39 & -37 & -30 & -24 & -24 & -30 \\ -30 & -38 & -39 & -36 & -31 & -25 & -24 & -30 \\ -29 & -37 & -38 & -36 & -31 & -26 & -25 & -30 \\ -29 & -36 & -37 & -35 & -31 & 26 & -26 & -30 \\ -28 & -35 & -36 & -34 & -30 & -26 & -26 & -30 \end{vmatrix}$$

Gambar 3. 3 Matriks X

4. Buat dan cari nilai untuk matriks 3D-DCT untuk matriks C dan buat matriks *transpose* nya untuk matriks C^t.

$$C\ (i,j) = \begin{cases} \frac{1}{\sqrt{N}} & if\ i=0 \\ \sqrt{\frac{2}{N}}\ .\ cos\frac{(2j+1)i\pi}{2N} & if\ i\neq 0 \end{cases}$$

Maka dengan menggunakan rumus matriks diatas dapat dihitung nilai matriks C mulai dari C(0,0) sampai C(7,7).

Nilai i=0 maka rumus menggunakan $C_{ij} = \frac{1}{\sqrt{N}}$, dimana nilai N adalah banyaknya blok. Sehingga didapat hasil sebagai berikut :

C(0,0)= $\frac{1}{\sqrt{N}} = \frac{1}{\sqrt{8}}$ = 0,3536,

…

C(0,7)= $\frac{1}{\sqrt{N}} = \frac{1}{\sqrt{8}}$ = 0,3536,

Namun apabila nilai $i \neq 0$ maka rumus yang digunakan untuk menghitung nilai matriks yaitu $Cij = \sqrt{\frac{2}{N}}\ .\ cos\frac{(2j+1)i\pi}{2N}$, dimana nilai π = 180°. Kemudian hitung nilai C(1,0) hingga C(7,7):

C(1,0)= $\sqrt{\frac{2}{N}}\ .\ cos\frac{(2j+1)i\pi}{2N} = \sqrt{\frac{2}{N}}\ .\ cos\frac{(2.0+1)1.180°}{2.8}$ = 0,4904

…

C(7,7)= $\sqrt{\frac{2}{N}}\ .\ cos\frac{(2j+1)i\pi}{2N} = \sqrt{\frac{2}{N}}\ .\ cos\frac{(2.7+1)7.180°}{2.8}$ = -0,0975

Maka dari perhitungan diatas didapatkan nilai untuk matriks C seperti ditunjukkan pada Gambar 3. 4. Dan matriks transpose C ditunjukkan pada Gambar 3. 5.

$$C = \begin{vmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1919 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ -0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & -0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{vmatrix}$$

Gambar 3. 4 Nilai Matriks C

$$C^T = \begin{vmatrix} 0.3536 & 0.4904 & 0.4619 & 0.4157 & 0.3536 & -0.2778 & 0.1913 & 0.0975 \\ 0.3536 & 0.4157 & 0.1919 & -0.0975 & -0.3536 & -0.4904 & -0.4619 & -0.2778 \\ 0.3536 & 0.2778 & -0.1913 & -0.4904 & -0.3536 & 0.0975 & 0.4619 & 0.4157 \\ 0.3536 & 0.0975 & -0.4619 & -0.2778 & 0.3536 & 0.4157 & -0.1913 & -0.4904 \\ 0.3536 & -0.0975 & -0.4619 & 0.2778 & 0.3536 & -0.4157 & -0.1913 & 0.4904 \\ 0.3536 & -0.2778 & -0.1913 & 0.4904 & -0.3536 & -0.0975 & 0.4619 & -0.4157 \\ 0.3536 & -0.4157 & 0.1913 & 0.0975 & -0.3536 & -0.4904 & -0.4619 & 0.2778 \\ 0.3536 & -0.4904 & 0.4619 & -0.4157 & 0.3536 & -0.2778 & 0.1913 & -0.0975 \end{vmatrix}$$

Gambar 3. 5. Matriks C^T

5. Dengan menggunakan persamaan 3D-DCT, cari matriks Y dimana matriks Y akan digunakan untuk kuantisasi lanjutan.

   $Y = C. X. C^T$ (Persamaan 2.6)

$$Y_{ij} = \begin{vmatrix} -248.00 & -16.1592 & 11.3996 & 19.2935 & 0.50 & 6.1245 & 2.0431 & 1.5029 \\ 2.2777 & 8.2394 & 1.8936 & -6.5075 & 0.8284 & -1.8108 & 0.0814 & -0.2827 \\ 3.9989 & 11.1516 & 0.6036 & -7.3684 & 1.5772 & -1.8480 & 0.1036 & -0.8600 \\ 0.2381 & 4.3461 & 0.8372 & -2.9047 & 0.6091 & -0.2061 & -0.2008 & -0.4762 \\ 1.0000 & 1.3390 & -0.3266 & -0.1420 & 0 & -0.1829 & -0.1353 & -0.1688 \\ 0.0710 & 0.1907 & 0.1665 & 0.3974 & -0.4070 & -0.5022 & 0.2587 & 0.0355 \\ 0.5084 & 0.2214 & 0.6036 & 0.1661 & -0.1121 & -0.0122 & -0.1036 & 0.3436 \\ 0.0179 & -0.1792 & 0.2387 & -0.3785 & -0.1648 & 0.2781 & -0.1218 & -0.3325 \end{vmatrix}$$

Gambar 3. 6 Matriks Y

Matriks Y yang ditunjukkan Gambar 3. 6 berisi koefisien DCT, yang kemudian akan dikuantisasi dengan level kuantisasi yang dipilih.

6. Setelah didapatkan data Matriks Y atau F(u,v,w)dari hasil 3D-DCT transform maka proses selanjutnya adalah kuantisasi. Dengan menggunakan rumus pada Persamaan 2.8 :

$$F_q(u,v,w) = \left\lfloor \frac{F(u,v,w)}{Q(u,v,w)} \right\rfloor$$

Dimana tanda […] hasil akan dibulatkan ke nilai integer yang terkecil.

Oleh karena Q(u,v,w) merupakan matriks [8 8] dengan nilai konstan = 16, karena menerjemahkan kedalam blok Simulink agak sulit maka diganti dengan proses perkalian dengan nilai konstan = 1/16. Sehingga rumus proses kuantisasi akan menjadi :

$$F_q(u,v,w) = \left\lfloor \frac{F(u,v,w)}{16} \right\rfloor$$

Dari hasil perhitungan kuantisasi aka didapatkan nilai-nilai koefisien yang kemudian akan digunakan CABAC untuk mengkonversikan kedalam bentuk biner, matriks kuantisasi dan hasil kuantisasi seperti terlihat pada Gambar 3. 7 dan Gambar 3. 8.

$$Q = \begin{vmatrix} 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \end{vmatrix}$$

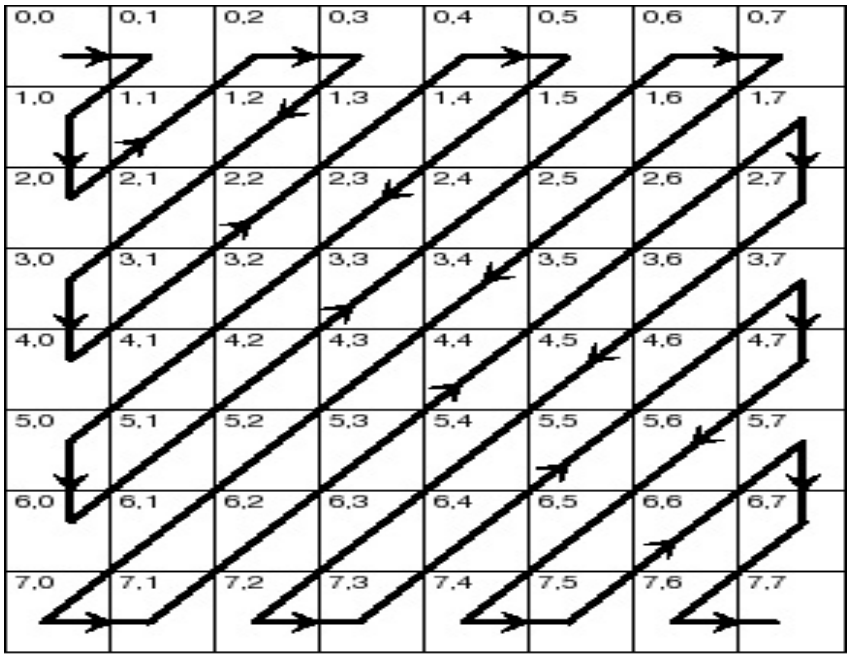
Gambar 3. 7 Matriks Kuantisasi

$$F_q(u,v,w) = \begin{vmatrix} -15 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

Gambar 3. 8 Matriks Hasil Kuantisasi

Dari hasil kuantisasi seperti terlihat pada Gambar 3. 8 terlihat bahwa terdapat beberapa angka integer '*zero'* dan *'non zero'*. Macroblock [8x8] yang telah terkuantisasi tersebut kemudian akan dilakukan pemrosesan zigzag scan blok.

7. Susun bilangan menggunakan fungsi *zig zag scanning* dimana ini merupakan langkah terakhir dalam proses transform.

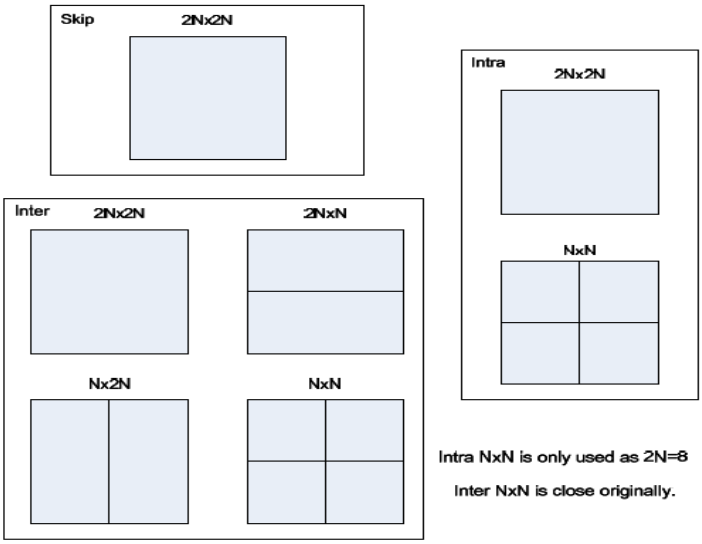


Gambar 3. 9. Metode Zig Zag Scanning

Matriks R yang terkuantisasi sekarang akan dikonversi sekarang akan dikonversi oleh encoder ke data biner (01101011 …). Koefisien 3D-DCT terkuantisasi mengatur sehingga bit yang paling kiri berisikan nilai-nilai yang tidak 0, dan yang paling kanan berisikan bit yang bernilai 0. Hasil zig-zag kemudian akan dikirimkan ke block CABAC untuk dilakukan proses yang menghasilkan *codeword* bit-bit biner ‘0’ atau ‘1’.

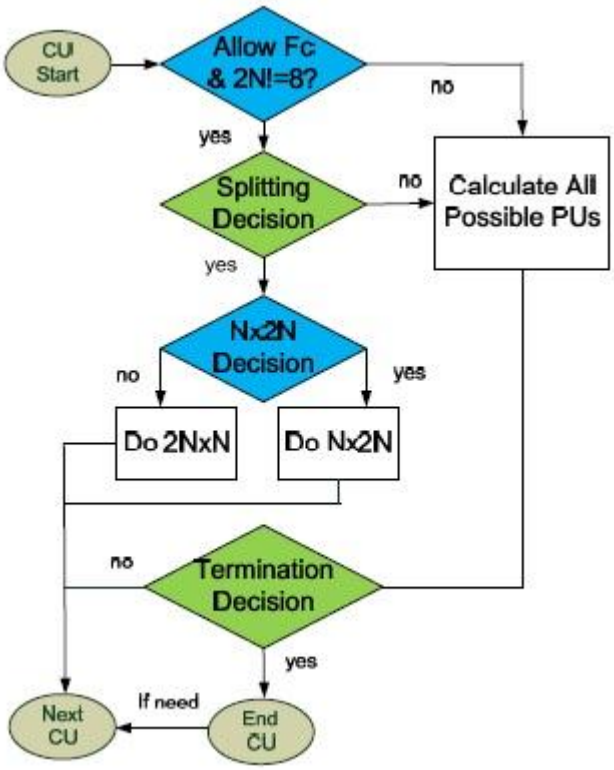
3.1.2.2 Analisis Algoritma *Fast Coding Unit Decision*

Dalam spesifikasi HEVC, CU adalah sebuah blok persegi 2Nx2N dan 2N bisa 64, 32, 16 atau 8. CU terbesar juga disebut LCU (*Large Coding Unit*). Di dalam HEVC, slice dalam frame terdiri dari banyak LCUs, dan sebuah CU besar dapat dibagi menjadi 4 CUs kecil. Masing-masing CU dipartisi secara rekursif hingga ukuran terkecil dari CU tercapai. Salah satu CU 2N x 2N yang diproses di setiap kedalaman yaitu analisis encoder *Rate-Distortion* (RD) mode. *Prediction Unit* (PU) hanya didefinisikan pada daun node dari CU dari setiap level kedalaman [12]. PU yang memungkinkan untuk CU 2Nx2N pada *low complexity setting* dapat dilihat seperti Gambar 3. 10 berikut :



Gambar 3. 10 PU yang memungkinkan untuk CU 2Nx2N pada low complexity setting

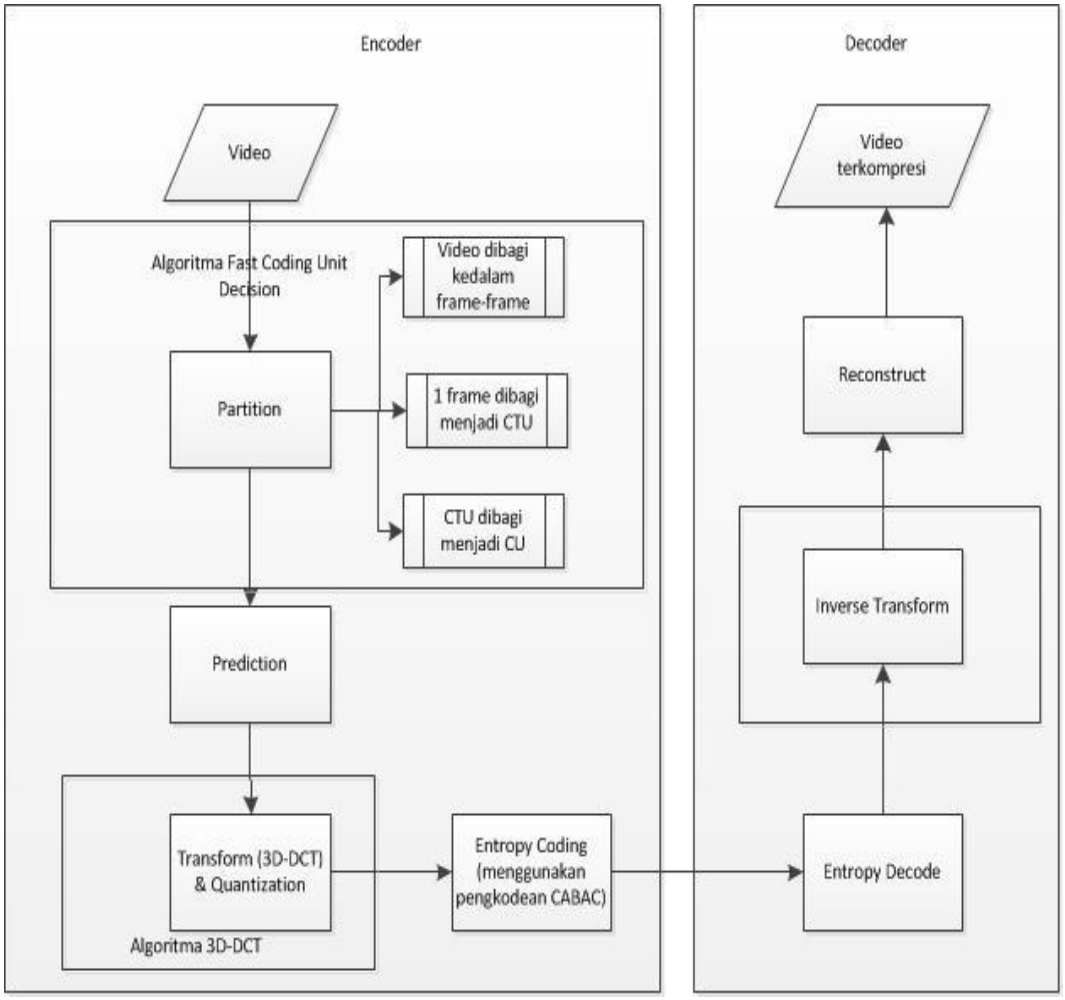
Algoritma dasar untuk *fast CU size decision* dan dua *tools* tambahan yang berguna untuk meningkatkan performa pengkodean dan meningkatkan pengurangan masing-masing waktu [12]. Flowchart dari algoritma ini seperti Gambar 3. 11 berikut :



Gambar 3. 11 Flowchart algoritma untuk memproses CU

3.1.3 Analisis Sistem Yang Akan Dibangun

Analisis sistem yang akan dibangun ini dilakukan untuk mengetahui tahapan-tahapan yang terjadi pada proses pembuatan aplikasi kompresi video dimana analisis lebih ditekankan pada metode-metode yang akan diterapkan pada tahapan pembuatan aplikasi tersebut. Gambar 3. 12 berikut adalah tahapan yang dilakukan dalam proses *encoding* dan *decoding* kompresi video HEVC/H.265.



Gambar 3. 12 Langkah-langkah proses Encoding dan Decoding HEVC/H.265

a. Langkah-langkah utama pada *encoding* HEVC/H.265 yaitu :
   1. Membagi (*partition*) setiap gambar menjadi beberapa bagian CUs (*Coding Units*).
   2. Memprediksi (*prediction*) setiap unit menggunakan inter atau intra dan mengurangi prediksi dari setiap unit, kebanyakan HEVC menggunakan *intra prediction.*
   3. Mengubah (*transform)* dan mengkuantisasi (*quantization)* proses (perbedaan antara unit asli dan prediksinya).
   4. Mengubah pengkodean output (*entropy coding*). Lalu informasi prediksi diberikan header dan informasi.

b. Langkah-langkah utama pada *decoding* HEVC/H.265 yaitu :
   1. Melakukan decoding entropi dan mencari unsur-unsur dari urutan setiap kode (*entropy decode*).
   2. Melakukan pengubahan ukuran dan melakukan tahap pembalikan transformasi (*inverse transform)*
   3. Membangun gambar video yang sudah di transformasi (*reconstruct*).

3.1.3.1 Analisis Masukan

Dalam perancangan teknik kompresi pada video yang terdiri *Encoder-Decoder* (CODEC) pertama kali adalah menentukan *input* video yang ingin diolah atau dikompres. *Input* data video mempunyai spesifikasi sebagai berikut :

1. Nama file = videocontoh.mp4
2. Format video = mp4 (*MPEG-4*)
3. Frame rate = 25 fps
4. Type file = Y|U|V
5. Resolusi spatial = [480x368] piksel

Pada video digital, umumnya data video dipisahkan menjadi komponen-komponen, baik komponen warna (*crominance*) maupun komponen kecerahan (*luminance*) . Pada komponen video, tiap komponen dipisahkan dengan cara

tertentu. Sesuai dengan tipe file video yang digunakan disini pemisahan komponen disajikan dengan bentuk Y|U|V. Y menyatakan sinyal kecerahan, sedangkan U dan V menyatakan sinyal warna.

Video yang digunakan memiliki resolusi 480x368 piksel dimana video tersebut tergolong dalam format video diatas CIF. Kedalaman piksel adalah sebesar 8 bit/piksel dan laju frame-nya adalah 25 fps. Dengan demikian kita dapat menghitung jumlah bit yang dibutuhkan, yaitu : 480 x 368 piksel x 1,5 x 8 bit/piksel x 25 fps = 52.992.000 bit/s.

Jumlah bit yang diperlukan sekitar 52 Mbit/s. Ini bukanlah jumlah yang kecil. Pada format yang memiliki kedalaman bit tinggi (misal 24 bit) dan resolusi tinggi, jumlah bit yang diperlukan tentu saja jauh lebih tinggi. Oleh karena itu, maka kompresi diperlukan untuk memampatkan data seefisien mungkin.

3.1.3.2 Encoding

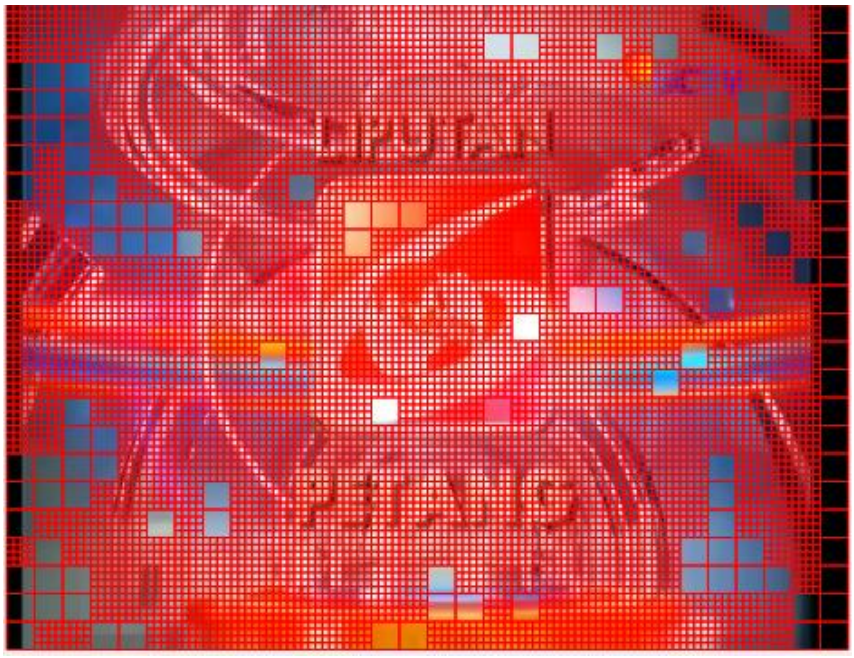
Pengkodean atau penyandian (*Encoding*) adalah proses konversi informasi dari suatu sumber (objek) menjadi data, yang selanjutnya dikirimkan ke penerima atau pengamat, seperti pada sistem pemrosesan data. Pada *encoder*, mula-mula ditentukan apakah suatu *frame* akan dikompresi secara *interframe* atau *intraframe*. Namun pada HEVC *encoder* pengkompresian dilakukan secara *intraframe*, dimana pada data video yang masuk dilakukan transformasi dengan DCT, selanjutnya dilakukan kuantisasi. Tahap-tahap dan metode yang digunakan pada *encoding* adalah sebagai berikut :

a. Partition

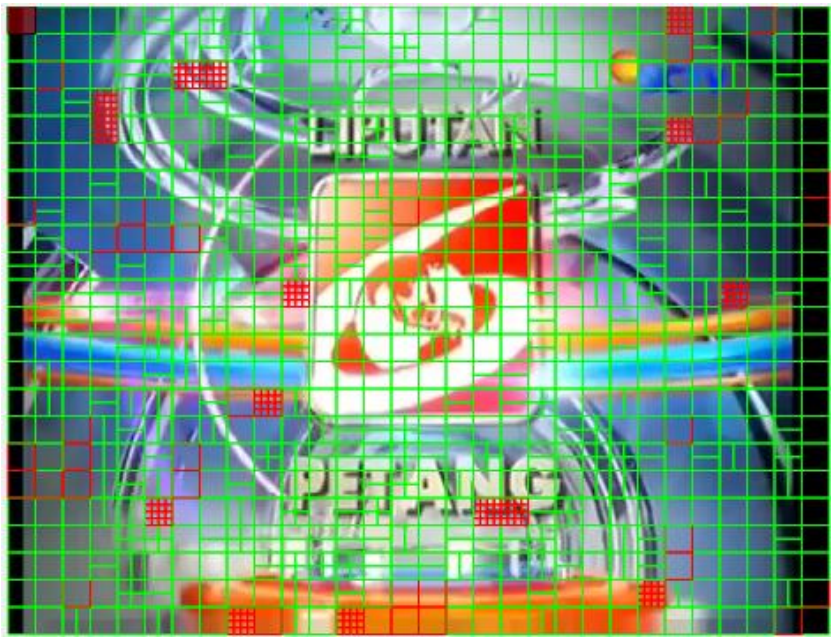
Pada proses partisi, video tersusun atas *frame-frame* yang *frame* atau gambar tersebut dibagi kedalam *Tiles* dan atau *Slices*, yang selanjutnya dibagi ke CTUs. CTU dapat dibagi menjadi persegi-persegi yang dikenal sebagai CU menggunakan struktur *quadtree*.

Format CIF yang berukuran 480x368 piksel dibagi menjadi blok-blok yang berukuran 16x16 piksel, maka dalam suatu frame akan terdapat

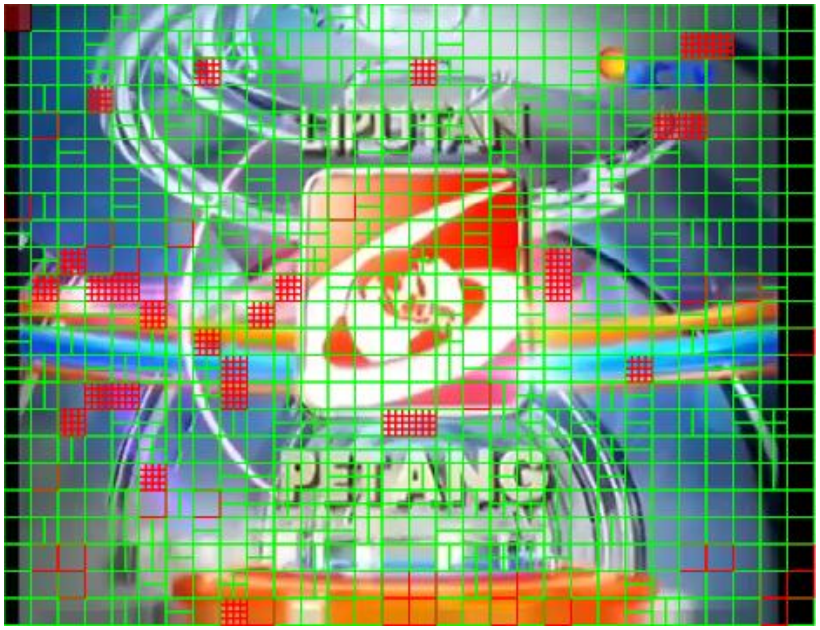
$[(\frac{480x368}{16x16} : 4]$ = ±172 buah blok. Pada tiap-tiap blok itulah diterapkan langkah-langkah kompresi intraframe. Partisi per *frame* pada video ini disajikan dalam Gambar 3. 13, Gambar 3. 14 dan Gambar 3. 15 ini, sample diambil hanya pada frame 0, frame 1, dan frame 2 :



Gambar 3. 13 Partisi video pada frame 0 (partisi awal)

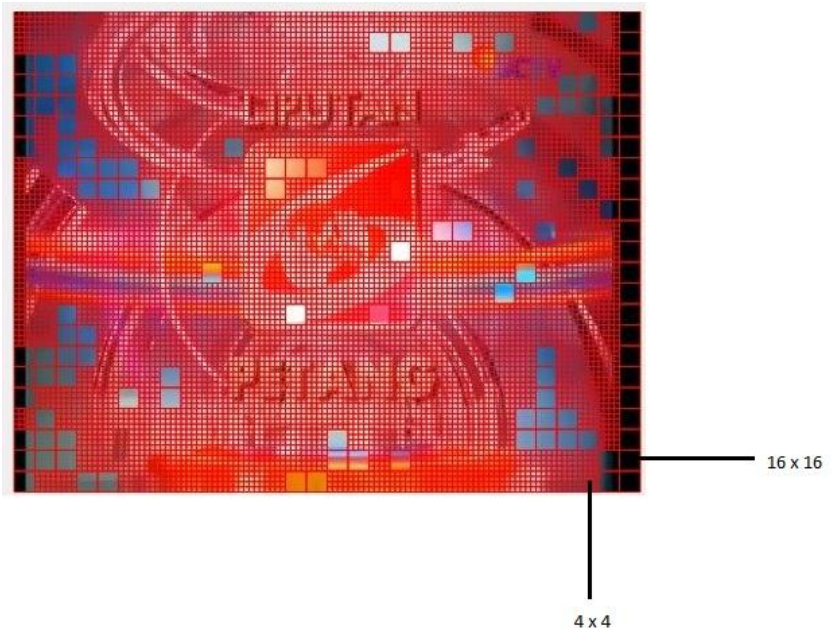


Gambar 3. 14 Partisi video pada frame 1



Gambar 3. 15 Partisi video pada frame 2

Seperti yang terlihat pada Gambar 3. 16 diketahui bahwa frame per piksel dibagi ke dalam CU ukuran 4x4 dan 16x16.



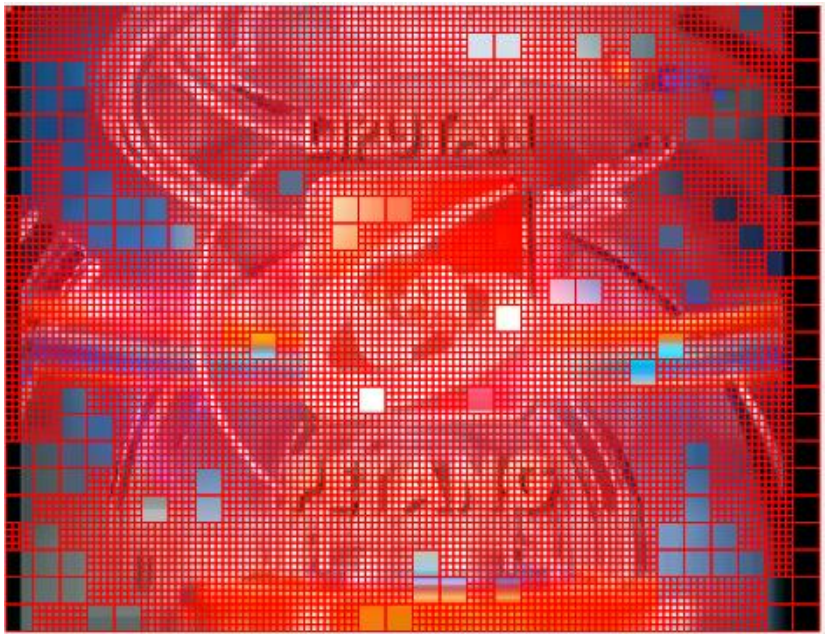
Gambar 3. 16 Ukuran CU yang digunakan pada piksel

b. Prediction

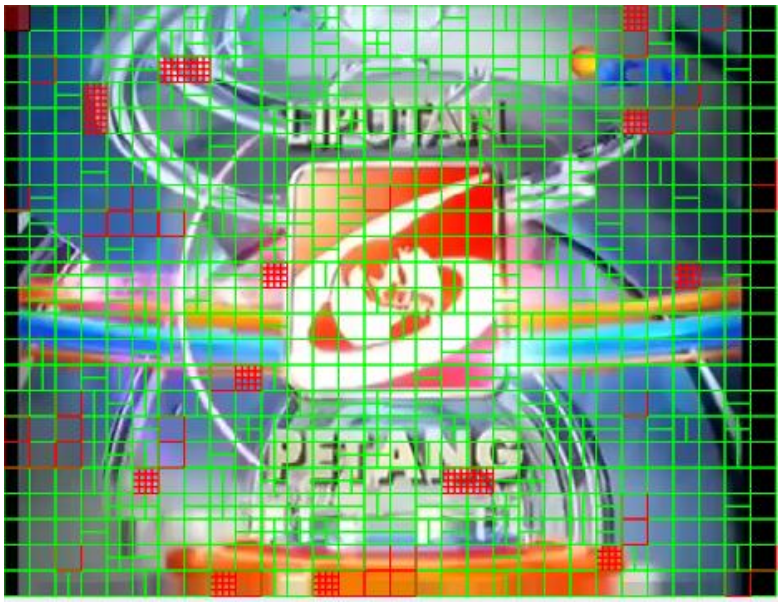
Prediksi CU pada kasus kompresi video ini menggunakan kompresi intraframe. Kompresi intraframe dilakukan dengan memanfaatkan redudansi spasial yang terdapat dalam suatu *frame*. Redudansi ini disebabkan karena adanya korelasi antara sebuah piksel disekitarnya.

Intra prediksi HEVC dilakukan pada *prediction unit* (PU)-wilayah *frame* video. Nilai-nilai piksel PU harus diprediksi paling dekat dengan nilai piksel asli sebanyak mungkin untuk meningkatkan efisiensi kompresi. Terdapat sebanyak 35 mode intra-prediksi di HEVC yaitu *Planar* (mode 0), *DC* (modus 1) dan 33 *angular modes* (2-34 mode).

Berikut Gambar 3. 17 dan Gambar 3. 18 dimana terdapat *frame* video yang sudah dalam proses prediksi, frame yang diambil yaitu frame ke 0 dan frame ke 1:



Gambar 3. 17 Frame 0 yang sudah diprediksi

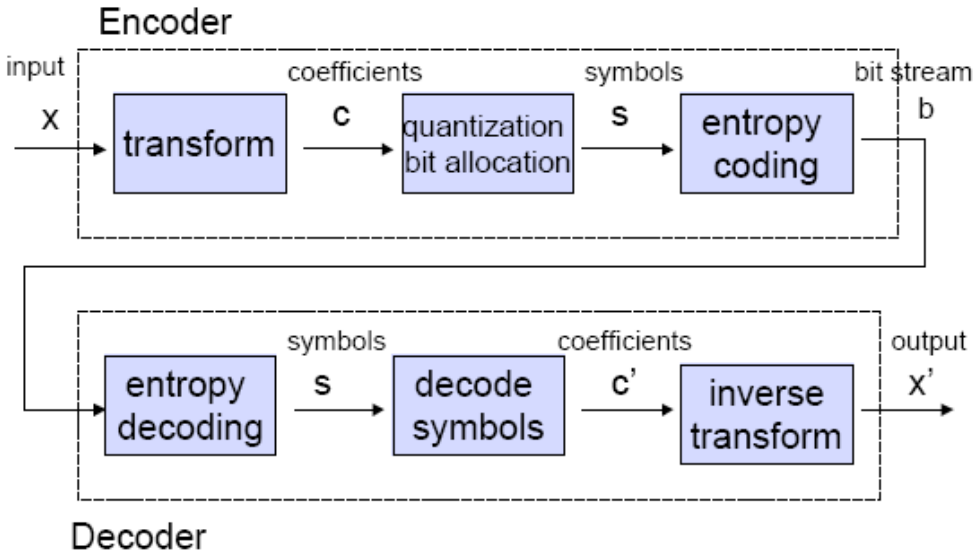


Gambar 3. 18 Frame 1 yang sudah diprediksi

c. Transformasi (*Transform*) dan Kuantisasi (*Quantization*)

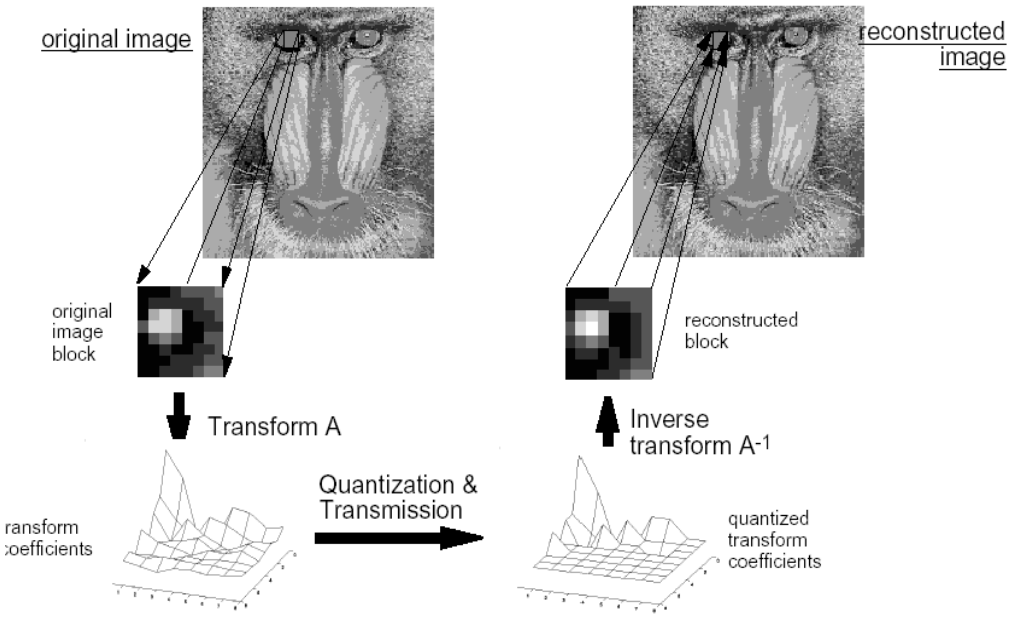
Proses transformasi dan kuantisasi juga merupakan bagian dari kompresi intraframe yang bersifat *lossy*, serta pengkodean yang bersifat *lossless*. Disini digunakan 3D-DCT untuk melakukan proses transformasi dari domain waktu ke

domain ruang. Dengan melakukan proses transformasi ini, maka data vital akan terkumpul pada frekuensi DC. Gambar 3. 19 berikut adalah blok diagram dari proses *Transform Coding* :



Gambar 3. 19 Blok Diagram Transform Coding

Skema dari *transform* seperti terlihat pada contoh Gambar 3. 20 berikut :



Gambar 3. 20 Skema Transform

d. **Entropy Coding**

Semua elemen dikodekan menggunakan CABAC (*Context Adaptive Binary Arithmetic Coding).* Operasi *entropy coding* dimulai dengan penataan kembali koefisien dalam urutan menurun dari nilai yang diharapkan. Gambar 3.21 berikut adalah tahapan dalam blok diagram *encoder* CABAC :



Gambar 3. 21 Blok diagram encoder CABAC

1. Binarization
2. Context Modelling
3. Binary Arithmetic Coding

e. Entropy Decode

Secara umum decoder CABAC bekerja dalam tiga tahap, yaitu [16]:



Gambar 3. 22 Blok diagram decoder CABAC

1. Context Modeling

CABAC memiliki kompresi yang baik, karena adanya perkiraan probabilitas yang tepat. Pada CABAC, perkiraan probabilitas ini diwakili oleh context model.

Masing-masing context model terdiri dari dua nilai, yaitu 6-bit state sebagai index probabilitas dan satu bit yang merepresentasikan MPS (Most Probable Symbol). Pada tahap modeling, model probabilitas dipilih untuk masing-masing simbol yang dikodekan. Standard menjelaskan proses pemilihan context model yang sesuai untuk tiap tahap proses decoding. Proses pemilihan context model tidak dijelaskan lebih lanjut

karena modul *Inverse* CABAC pada tugas akhir ini menerima input berupa alamat *context model* yang harus dipilih untuk setiap proses *decoding*.

a. Proses Inisialisasi Context Model

Nilai awal *context model* dapat dihitung dengan menggunakan persamaan pada gambar berikut :

```
1. σpre =
       max(1, min(126,(( μγ * SliceQP ) >> 4) + vγ))
2. if( σpre <= 63 ) {
     σ = 63 - σpre
     ϖ = 0
   } else {
     σ = σpre - 64
     ϖ = 1
   }
```

Gambar 3. 23 Prosedur untuk Inisialisasi Context Model

Dari persamaan diatas dilihat bahwa untuk membentuk nilai awal *context* model diperlukan kedua nilai parameter kuantisasi (SliceQP), dan parameter μ_γ dan v_γ. kedua parameter μ_γ dan v_γ ini telah dicantumkan pada standar.

b. Perkiraan Probabilitas

Untuk CABAC, 64 nilai representasi probabilitas, atau yang biasa disebut dengan indeks model ini diturunkan untuk LPS dari persamaan :

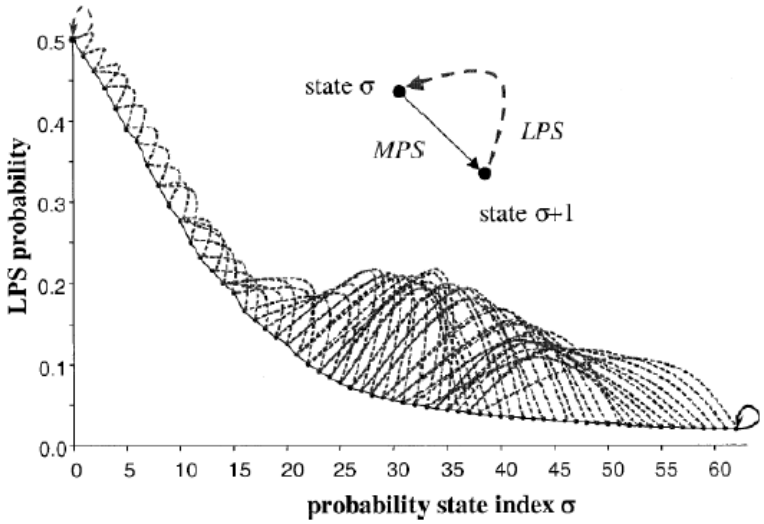
$P\sigma = \alpha . P_{\sigma-1}$, untuk $1 \leq \sigma \leq 63$

$\alpha = \left(\frac{0{,}01875}{0{,}5}\right)^{\frac{1}{63}}$ dan $p_{0=0{,}5}$

Jumlah nilai representas yang berjumlah 64 ini merupakan kompromi antara adaptasi yang cepat ($\alpha \rightarrow 0$, dan jumlahnya sedikit) dan kebutuhan akan banyaknya tabel yang cukup agar lebih akurat ($\alpha \rightarrow 1$, dan jumlahnya banyak). Perancangan ini membuat tiap context model di CABAC dapat ditentukan oleh dua parameter : nilai kemungkinan LPS

sekarang, yang diturunkan dari indeks state σ antara 0-63 dan nilai MPS ω yang dapat berupa 0 atau 1.

Gambar 3. 24 Memperlihatkan nilai probabilitas LPS p antara 0 dan 0,5 (pada sumbu y) yang berpasangan dengan nilai state index σ antara 0-63 (pada sumbu x).



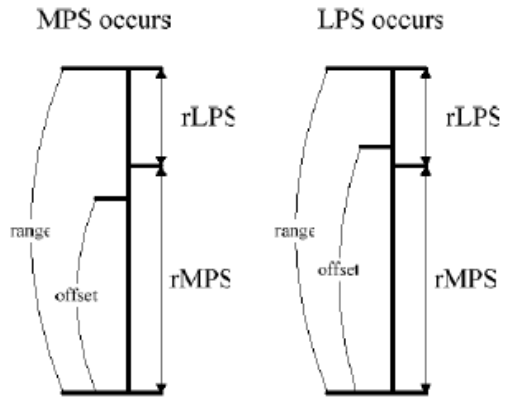
Gambar 3. 24 Nilai Probabilitas LPS dan aturan transisi

Pada gambar Gambar 3. 24 dijelaskan bahwa ketika terjadi perubahan di MPS, indeks state sebelumnya akan ditambahkan 1, kecuali sudah di 62 dimana probabilitas LPS sudah minimum atau sebaliknya MPS sudah maksimum. Bila indeks di 62, maka indeks akan tetap sampai ada perubahan LPS yang kemudian akan mengurangi indeks dengan suatu aturan tertentu seperti yang diilustrasikan oleh garis putus-putus di Gambar 3. 24. Pada kenyataannya, indeks nomor 63 digunakan oleh aritmatik decoding yang tidak adaptif. Sehingga hanya 63 model yang dapat digunakan.

2. Binary Arithmetic Coding

Untuk men-decode sebuah bin, binary arithmetic decoder membutuhkan nilai range, offset, dan context model yang bersangkutan. Nilai offset ini merupakan kriteria untuk menentukan nilai bin yang di-*decode*, dan diinisialisasi dengan mengambil 9 bit pertama dari bit stream yang telah di-encode. Gambar 3. 25 dibawah menggambarkan pemroses binary arithmatic decoder untuk satu bin. CABAC *decoding engine* akan selalu melakukan *update* terhadap dua register 9-bit : range dan offset selama proses *decoding* berlangsung. Register range memantau lebar dari interval saat ini sedangkan register offset memantau aliran bit masukan. Ketika melakukan decoding sebuah bin, range dibagi menjadi dua daerah : rLPS untuk rentang perkiraan LPS dan rMPS untuk rentang perkiraan MPS.

Ketika proses pengkodean berlangsung, nilai rLPS dibaca dari sebuah tabel 2 dimensi 256-byte, dialamatkan oleh 2-bit nilai range dan 6-bit nilai state. Subinterval tempat suatu bit input terjadi (ditandai dengan offset), menentukan bin itu MPS atau LPS.



Gambar 3. 25 Proses Decoding

Pada Gambar 3. 25, gambar kiri menunjukkan kasus ketika masukan data jatuh di MPS, dimana nilai offset kurang dari rMPS. Gambar kanan menunjukkan kasus dimana data jatuh di LPS, dimana offset lebih besar

(atau sama dengan) nilai rMPS. Maka nilai baru dari range dan offset adalah :

Jika MPS: range_baru = rMPS; offset_baru = offset Selain itu: range_baru = rLPS; offset_baru = offset - rMPS

Untuk menjaga ketelitian selama proses decoding berlangsung, range_baru dan offset_baru harus selalu di-renormalisasi untuk memastikan MSB dari range selalu 1, misalnya: range_new : 9'b001010110, offset_baru : 9'b000110010, selama proses renormalisasi, range_baru di 'geser kiri' dua bit sehingga MSB-nya 1 dan dua bit terakhir ditambahkan 2'b00. Nilai offset_baru secara bersamaan juga di 'geser kiri' dua bit dan dua bit akhir ditambahkan data masukan. Dengan cara ini, offset menerima bit dari masukan untuk menjaga jejak posisi bit masukan pada interval saat ini. Pada proses ini, probabilitas LPS (pLPS) diperkirakan oleh context model yang bersangkutan.

Untuk mode bypass, pLPS dibuat tetap 0,5 dan tidak dibutuhkan context model. Pada kasus ini, offset akan selalu dibandingkan dengan nilai range/2 untuk menentukan apakah bin itu MPS atau LPS. Pada mode ini, untuk menjaga ketelitian integer, nilai offset di 'geser kiri' satu bit dan menerima satu bit (di LSB) dari data masukan. Kemudian nilai offset baru dibandingkan dengan nilai range untuk menentukan bin itu 1 atau 0.

3. Inverse Binerisasi

Ada 5 jenis binerisasi yang digunakan pada CABAC, yaitu :

1. Unary Binarization (U)
2. Truncated Unary Binarization (TU)
3. Concatenated Unary/k-th Order Exp-Golomb Binarization (UEGk)
4. Fixed-length Binarization (FL)
5. Specified

3.1.4 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional merupakan analisis yang dibutuhkan untuk menentukan spesifikasi kebutuhan sistem. Spesifikasi ini juga meliputi elemen atau komponen-komponen apa saja yang dibutuhkan untuk sistem yang akan dibangun sampai dengan sistem tersebut diimplementasikan. Analisis kebutuhan ini juga menentukan spesifikasi masukan yang diperlukan sistem, keluaran yang akan dihasilkan sistem dan proses yang dibutuhkan untuk mengolah masukan sehingga menghasilkan suatu keluaran yang diinginkan. Pada analisis kebutuhan sistem non fungsional ini dijelaskan analisis kebutuhan perangkat keras, analisis kebutuhan perangkat lunak, dan analisis pengguna.

3.1.4.1 Analisis Kebutuhan Perangkat Keras

Perangkat keras merupakan salah satu kebutuhan yang sangat penting bagi proses kompresi video. Perangkat keras akan mempengaruhi kinerja dari proses kompresi video, semakin tinggi spesifikasi dari perangkat keras yang digunakan maka akan semakin cepat pula proses kompresi yang akan dijalankan.

Perangkat keras yang digunakan pada pembangunan aplikasi kompresi video ini yaitu seperti terlihat pada Tabel 3. 1.

Tabel 3. 1 Spesifikasi Perangkat Keras

Nama Perangkat	**Spesifikasi**
Processor	Intel(R) Core(TM) i3
RAM	2 GB
Harddisk	500 GB
VGA Card	Intel(R) HD Graphics
Mouse dan *keyboard*	Standard PS/2

Sedangkan kebutuhan perangkat keras untuk menjalankan aplikasi yang dibangun, yang harus dipenuhi yaitu seperti pada **Tabel 3. 2** .

Tabel 3. 2. Spesifikasi Perangkat Keras

Nama Perangkat	Spesifikasi
Processor	Intel(R) Core(TM) i5 2,67GHz
RAM	Min 2 GB untuk resolusi dibawah HD, dan 8 GB untuk resolusi diatas HD.
Harddisk	Min. 500 GB
VGA Card	Min. Intel(R) HD Graphics
Mouse dan *keyboard*	Standard PS/2

3.1.4.2 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak yang digunakan untuk membangun aplikasi kompresi video ini yaitu seperti pada Tabel 3. 3.

Tabel 3. 3 Spesifikasi Perangkat Lunak

Nama Perangkat	Spesifikasi
Sistem Operasi	Windows 7 Ultimate 64-bit
Bahasa Pemrograman	C#
Aplikasi pemrograman	IDE Visual Studio 2010
Library	ffmpeg
Pemodelan	UML
Video Player	*1. VLC* *2. Multimedia Player Classic*

3.1.4.3 Analisis Kebutuhan Pengguna

Analisis pengguna aplikasi ditujukan untuk seluruh user yang ingin melakukan pemampatan video yang menurut mereka ukuran video tersebut terlalu besar dan perlu dilakukan kompresi agar sesuai dengan ukuran yang mereka inginkan. Karakteristik pengguna untuk menggunakan aplikasi ini adalah :

1. Minimal dapat mengoperasikan komputer.
2. Minimal dapat mengetahui dan paham akan cara penggunaan aplikasi .

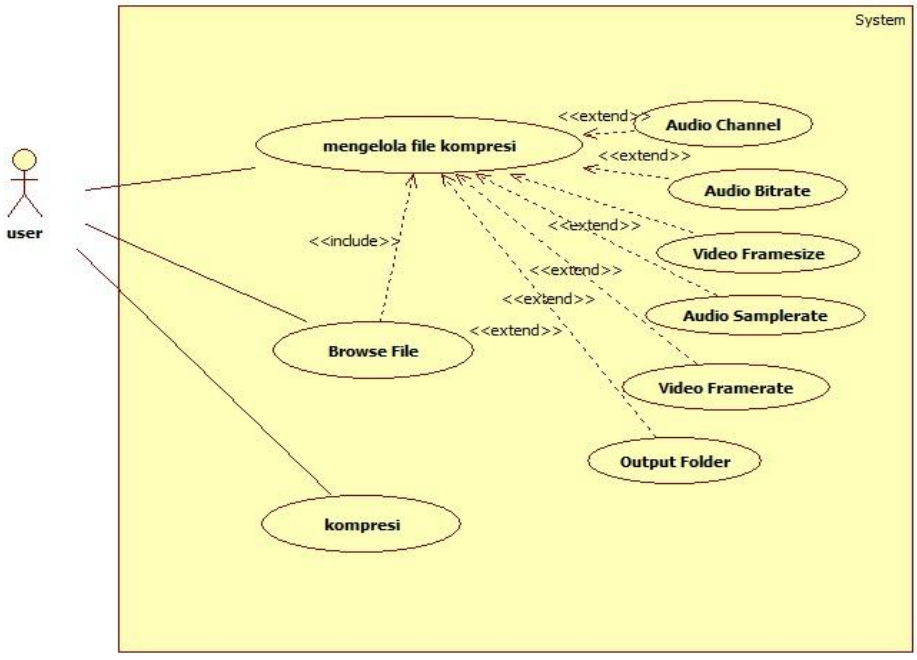
3.1.5 Analisis Kebutuhan Fungsional Perangkat Lunak

Analisis kebutuhan fungsional menggambarkan proses kegiatan yang akan diterapkan dalam sebuah sistem dan menjelaskan kebutuhan yang diperlukan sistem agar sistem dapat berjalan dengan baik.

Analisis yang dilakukan dimodelkan dengan menggunakan UML (*Unified Modeling Language*). Tahap-tahap pemodelan dalam analisis tersebut antara lain identifikasi aktor, *use case diagram, skenario use case, activity diagram, sequence diagram, class diagram.*

3.1.5.1 *Use case* Diagram

Pemodelan *use case* adalah pemodelan sistem dari perspektif pandangan pemakai aktif *(end user*). Model *use case* adalah pandangan dari luar sistem, sementara model rancangan adalah pandangan dari dalam. Model *use case* menangkap penggunaan-penggunaan sistem, sedangkan model rancangan merepresentasikan pembangunan dari sistem. Gambar 3. 26 berikut ini adalah gambar dari *use case* untuk aplikasi kompresi video.



Gambar 3. 26 Use Case Diagram

a. Identifikasi Actor

Actor adalah abstraksi dari orang dan sistem yang lain yang mengaktifkan fungsi dari target sistem. Berikut adalah aktor yang berperan dalam menjalankan aplikasi yang dibangun seperti terlihat pada Tabel 3. 4.

Tabel 3. 4 Use Case Actor

No	*Actor*	Deskripsi
A-01	*User*	Merupakan aktor dari sistem yang dibangun atau pengguna sistem yang akan menjalankan segala fungsionalitas yang ada pada aplikasi kompresi.

b. Identifikasi *Use case* Diagram

Berikut Tabel 3. 5 identifikasi *use case* yang terdapat pada aplikasi :

Tabel 3. 5 Identifikasi Use Case

No	*Use Case*	Deskripsi
UC-01	*Mengelola file kompresi*	Proses yang didalamnya mengatur segala pengaturan yang akan dan sedang berlangsung pada file video kompresi seperti menambahkan file video yang akan dikompresi, menghapus file video yang batal dikompresi, maupun mengatur segala pengaturan keluaran video yang sudah dikompresi.
UC-02	*Browse File*	Proses untuk mengambil file video dari source.
UC-03	*Kompresi*	Proses yang didalamnya terdapat fungsionalitas *Start* yang dimana fungsi tersebut terjadi proses kompresi video
UC-04	*Audio Channel*	Proses untuk mengatur *audio channel* dari video yang akan dikompresi.
UC-05	*Audio Bitrate*	Proses untuk mengatur *audio bitrate* dari video yang akan dikompresi.
UC-06	*Video Framesize*	Proses untuk mengatur *video frame size* dari video yang akan dikompresi.
UC-07	*Audio Samplerate*	Proses untuk mengatur *audio samplerate* dari video yang akan dikompresi.
UC-08	*Video Framerate*	Proses untuk mengatur *video frame rate* dari video yang akan

		dikompresi.
UC-9	*Output Folder*	Proses untuk mengatur *output folder* dari video yang akan dikompresi.

3.1.5.2 Skenario *Use Case*

Skenario *use case* mendeskripsikan urutan langkah-langkah dalam proses bisnis, baik yang dilakukan aktor terhadap sistem maupun yang dilakukan oleh sistem terhadap aktor.

a. Skenario *Use Case* Mengelola File Kompresi

Skenario *use case* ini menjelaskan interaksi antar aktor, yaitu *user* dengan *use case* mengelola file kompresi yang dijelaskan pada tabel berikut :

Tabel 3. 6 Skenario Use Case Mengelola File Kompresi

Identifikasi	
Nomor	*UC-01*
Nama *Use Case*	*Mengelola File Kompresi*
Aktor	*User*
Tujuan	File-file video yang akan dikompresi telah diatur atau ditetapkan.
Kondisi Awal	Halaman Utama

Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Menekan tombol *Add*	
	2. Menampilkan *form* pencarian file video
3. Memilih video yang akan dikompresi	
	4. Menampilkan *form output options*
5. Mengatur *output options*	
	6. Menyimpan pengaturan keluaran video, jika pengaturan selesai maka *finish*, jika belum maka :
7. Memilih file yang ingin dihapus dengan menekan tombol *Remove*	
	8. Menghapus file video terpilih, jika pengaturan selesai maka *finish*, jika belum maka :
9. Menekan tombol *Clear* untuk menghapus daftar video	
	10. Semua daftar video terhapus
Kondisi Akhir	File-file video yang akan dikompresi telah diatur atau ditetapkan.

b. Skenario Use Case Browse File

Skenario *use case* ini menjelaskan interaksi antar aktor, yaitu *user* dengan *use case* browse file kompresi yang dijelaskan pada tabel berikut :

Tabel 3. 7 Skenario Use Case Browse File

Identifikasi	
Nomor	*UC-02*
Nama *Use Case*	*Browse File*
Aktor	*User*
Tujuan	Mengambil file video dari source
Kondisi Awal	Open dialog source file
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Menekan tombol Add	
	2. Mengeluarkan open dialog source file
3. Menetapkan video yang akan dikompresi	
Kondisi Akhir	File video yang akan dikompresi akan diatur output options-nya.

c. Skenario Use Case Kompresi

Skenario *use case* ini menjelaskan interaksi antar aktor, yaitu *user* dengan *use case* kompresi yang dijelaskan pada tabel berikut :

Tabel 3. 8 Skenario Use Case Kompresi

Identifikasi	
Nomor	*UC-03*
Nama *Use Case*	*Kompresi*
Aktor	*User*
Tujuan	Proses kompresi berjalan
Kondisi Awal	Halaman T03 atau halaman dimana file video yang akan dikompresi telah dipilih
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**

1. Menekan tombol *Start*	
	2. Video mulai dibagi menjadi frame-frame (partition), 1 frame dibagi menjadi CTU, dan CTU dibagi lagi menjadi CU, 3. Frame tersebut masuk ke proses *prediction* 4. Dari proses *prediction* dilanjutkan dengan proses *transform* dan kuantisasi 5. *Entropy coding* menggunakan pengkodean CABAC yang hasilnya akan menjadi sebuah *residual coding*, 6. Proses dilanjutkan dengan *entropy decode* 7. Kemudian *inverse transform* dihasilkan, 8. Setelah itu frame-frame tersebut dibangun kembali menjadi sebuah video utuh yang telah terkompresi (*reconstruct*).
9. Video selesai dikompresi	
Kondisi Akhir	Proses kompresi selesai.

d. Skenario Use Case Audio Channel

Skenario *use case* ini menjelaskan interaksi antar aktor, yaitu *user* dengan *use case audio channel* yang dijelaskan pada tabel berikut :

Tabel 3. 9 Skenario Use Case Audio Channel

Identifikasi	
Nomor	*UC-04*
Nama *Use Case*	*Audio Channel*
Aktor	*User*
Tujuan	Mengatur *audio channel* dari video yang akan dikompresi
Kondisi Awal	Tampilan Output Options (T02)
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Pilih tipe *audio channel*	
	2. Tipe *audio channel* telah ditetapkan.
Kondisi Akhir	Tipe *audio channel* telah ditetapkan.

e. Skenario Use Case Audio Bitrate

Skenario *use case* ini menjelaskan interaksi antar aktor, yaitu *user* dengan *use case* audio bitrate yang dijelaskan pada tabel berikut :

Tabel 3. 10 Skenario Use Case Audio Bitrate

Identifikasi	
Nomor	*UC-05*
Nama *Use Case*	*Audio Bitrate*
Aktor	*User*
Tujuan	Mengatur ukuran *audio bitrate* dari video yang akan dikompresi
Kondisi Awal	Tampilan Output Options (T02)
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Pilih ukuran *audio bitrate*	
	2. Ukuran *audio bitrate* telah ditetapkan.
Kondisi Akhir	Ukuran *audio bitrate* telah ditetapkan.

f. Skenario Use Case Video Frame Size

Skenario *use case* ini menjelaskan interaksi antar aktor, yaitu *user* dengan *use case* video frame size yang dijelaskan pada tabel berikut :

Tabel 3. 11 Skenario Use Case Video Frame Size

Identifikasi	
Nomor	*UC-06*
Nama *Use Case*	*Video Frame Size*
Aktor	*User*
Tujuan	Mengatur tipe ukuran *video frame size* dari video yang akan dikompresi
Kondisi Awal	Tampilan Output Options (T02)
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Pilih ukuran *video frame size*	
	2. Tipe ukuran *video frame size* telah ditetapkan.
Kondisi Akhir	Ukuran *video frame size* telah ditetapkan.

g. Skenario Use Case Audio Sample Rate

Skenario *use case* ini menjelaskan interaksi antar aktor, yaitu *user* dengan *use case* audio sample rate yang dijelaskan pada tabel berikut :

Tabel 3. 12 Skenario Use Case Audio Sample Rate

Identifikasi	
Nomor	*UC-07*
Nama *Use Case*	*Audio Sample Rate*
Aktor	*User*
Tujuan	Mengatur ukuran *audio sample rate* dari video yang akan dikompresi
Kondisi Awal	Tampilan Output Options (T02)
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Pilih ukuran *audio sample rate*	
	2. Tipe ukuran *audio sample rate* telah ditetapkan.
Kondisi Akhir	Ukuran *audio sample rate* telah ditetapkan.

h. Skenario Use Case Video Frame Rate

Skenario *use case* ini menjelaskan interaksi antar aktor, yaitu *user* dengan *use case* frame rate yang dijelaskan pada tabel berikut :

Tabel 3. 13 Skenario Use Case Frame Rate

Identifikasi	
Nomor	*UC-08*
Nama *Use Case*	*Video Frame Rate*
Aktor	*User*
Tujuan	Mengatur ukuran *video frame rate* dari video yang akan dikompresi
Kondisi Awal	Tampilan Output Options (T02)
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Pilih ukuran *video frame rate*	
	2. Tipe ukuran *video frame rate* telah ditetapkan.
Kondisi Akhir	Ukuran *video frame rate* telah ditetapkan.

i. Skenario Use Case Output Folder

Skenario *use case* ini menjelaskan interaksi antar aktor, yaitu *user* dengan *use case* output folder yang dijelaskan pada tabel berikut :

Tabel 3. 14 Skenario Use case Output Folder

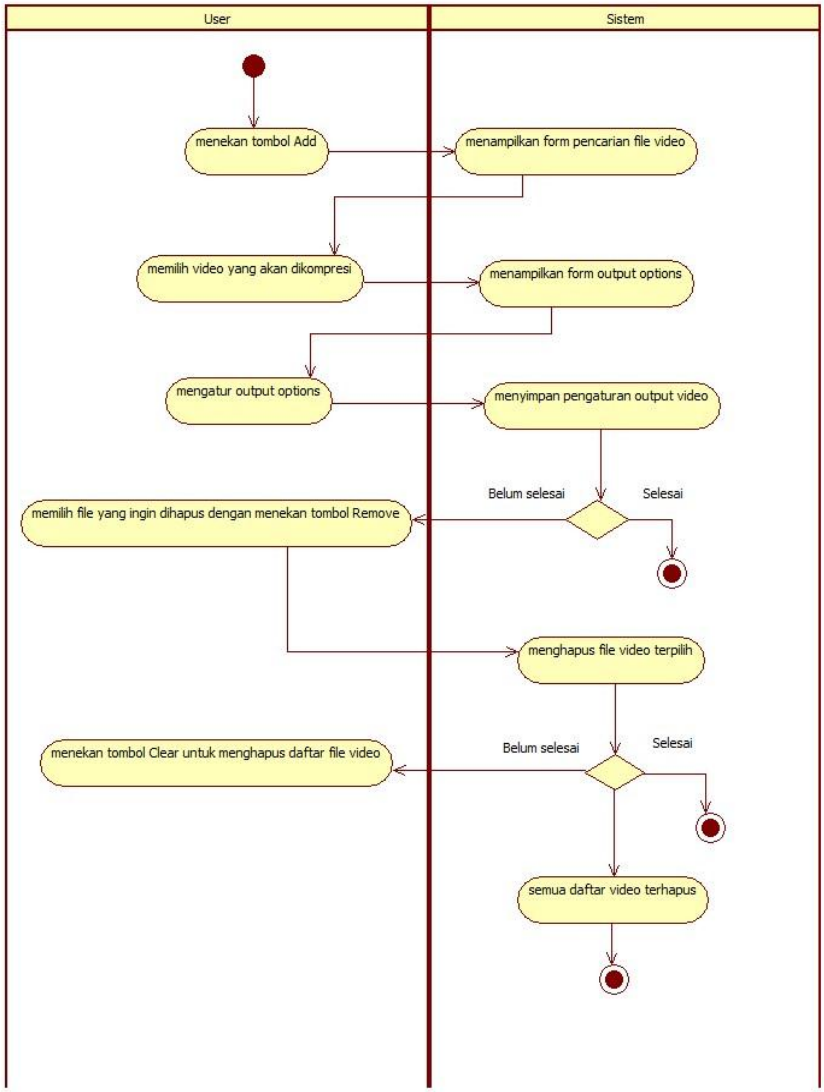
Identifikasi	
Nomor	*UC-9*
Nama *Use Case*	*Output Folder*
Aktor	*User*
Tujuan	Mengatur penyimpanan atau *output folder* dari video yang akan dikompresi
Kondisi Awal	Tampilan Output Options (T02)
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Browse tempat penyimpanan yang akan ditetapkan	
	2. *Output folder* telah ditetapkan.
Kondisi Akhir	*Output folder* telah ditetapkan.

3.5.1.3 Activity Diagram

Activity Diagram merupakan *state* diagram khusus, dimana sebagian besar *state* adalah *action*. Menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decisio yang mungkin terjadi. Activity diagram dibuat berdasarkan sebuah atau beberapa use case pada use case diagram.

1. Activity Diagram Mengelola File Kompresi

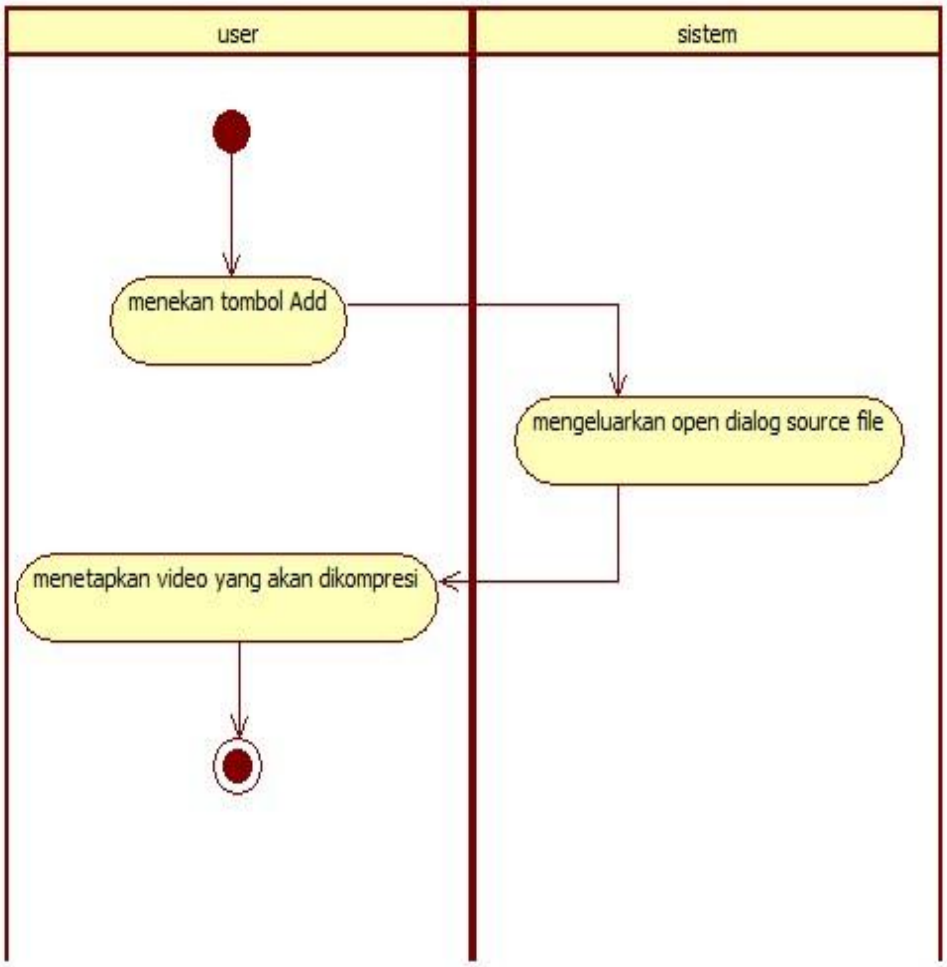
Berikut ini merupakan diagram yang menunjukkan alur pada aktivitas mengelola file kompresi yang dapat dilihat pada gambar di bawah ini.



Gambar 3. 27 Activity Diagram Mengelola Daftar Kompresi

2. Activity Diagram *Browse File*

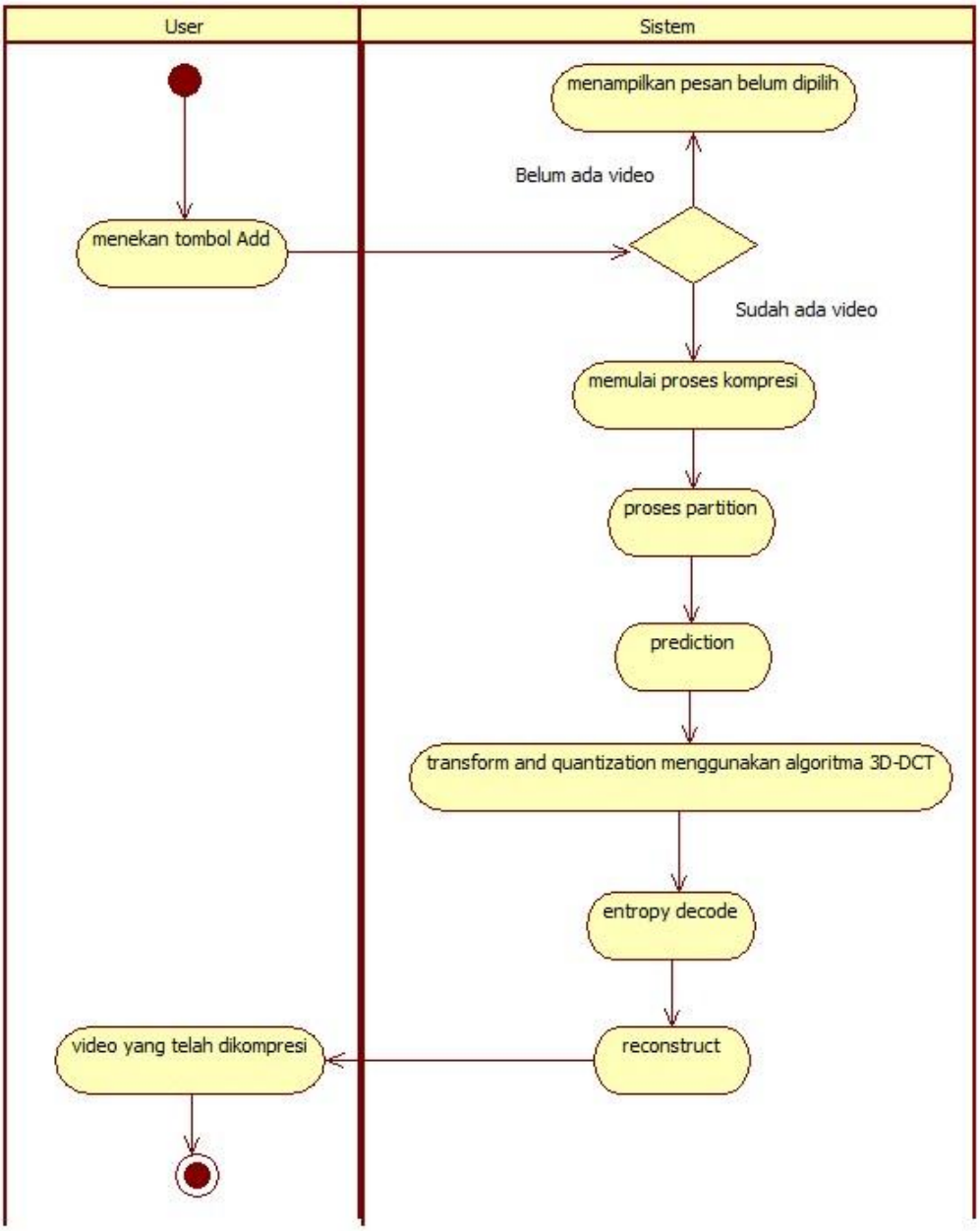
Berikut ini merupakan diagram yang menunjukkan alur pada aktivitas *Browse File* yang dapat dilihat pada gambar di bawah ini.



Gambar 3. 28 Activity Diagram Browse File

3. Activity Diagram Kompresi

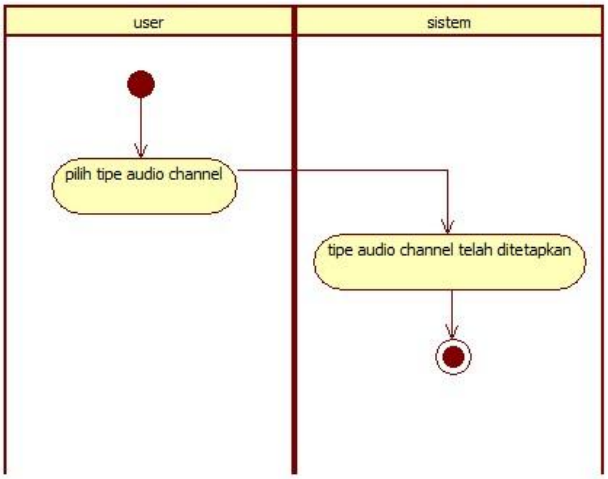
Berikut ini merupakan diagram yang menunjukkan alur pada aktivitas Kompresi yang dapat dilihat pada gambar di bawah ini.



Gambar 3. 29 Activity Diagram Kompresi

4. Activity Diagram *Audio Channel*

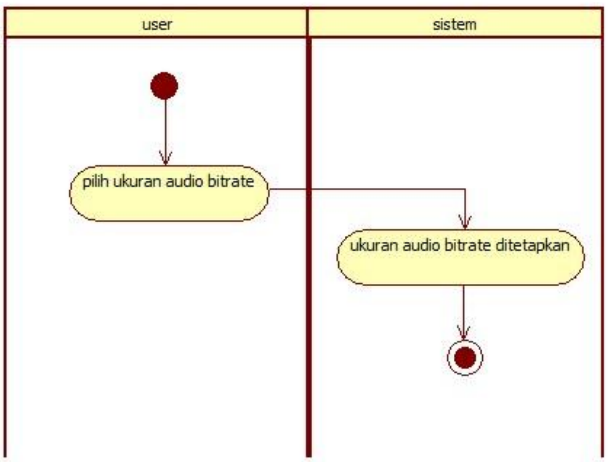
Berikut ini merupakan diagram yang menunjukkan alur pada aktivitas *Audio Channel* yang dapat dilihat pada gambar di bawah ini.



Gambar 3. 30 Activity Diagram Audio Channel

5. Activity Diagram Audio Bitrate

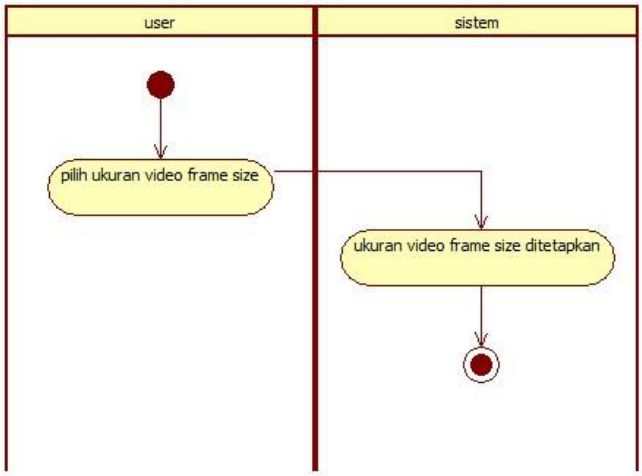
Berikut ini merupakan diagram yang menunjukkan alur pada aktivitas *Audio Bitrate* yang dapat dilihat pada gambar di bawah ini.



Gambar 3. 31 Activity Diagram Audio Bitrate

6. Activity Diagram Video Frame size

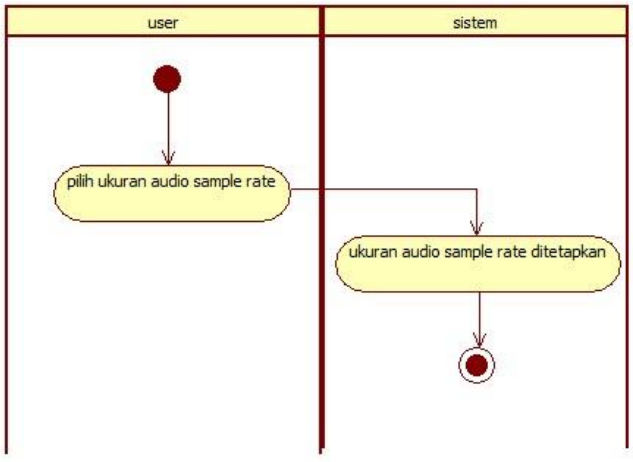
Berikut ini merupakan diagram yang menunjukkan alur pada aktivitas *Video Frame Size* yang dapat dilihat pada gambar di bawah ini.



Gambar 3. 32 Activity Diagram Video Frame Size

7. Activity Diagram Audio Sample Rate

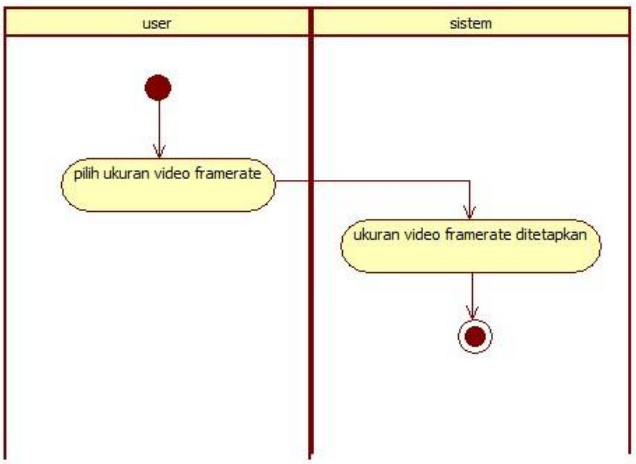
Berikut ini merupakan diagram yang menunjukkan alur pada aktivitas *Audio Sample Rate* yang dapat dilihat pada gambar di bawah ini.



Gambar 3. 33 Activity Diagram Audio Sample Rate

8. Activity Diagram Video Framerate

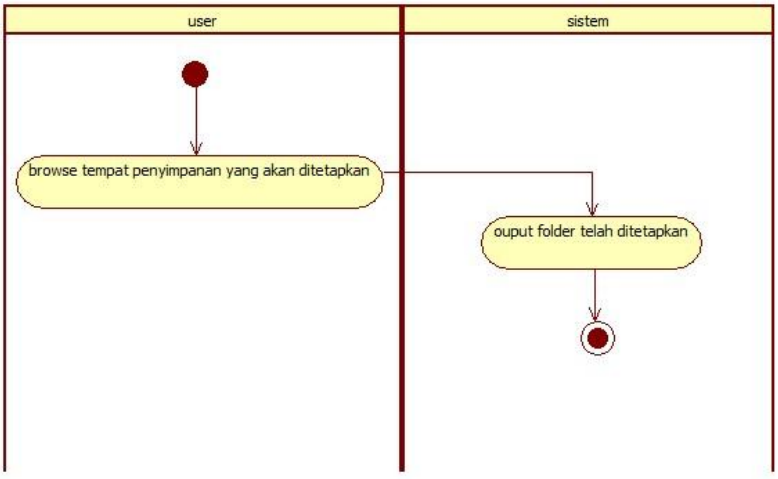
Berikut ini merupakan diagram yang menunjukkan alur pada aktivitas *Video Framerate* yang dapat dilihat pada gambar di bawah ini.



Gambar 3. 34 Activity Diagram Video Framerate

9. Activity Diagram Output Folder

Berikut ini merupakan diagram yang menunjukkan alur pada aktivitas *Output Folder* yang dapat dilihat pada gambar di bawah ini.

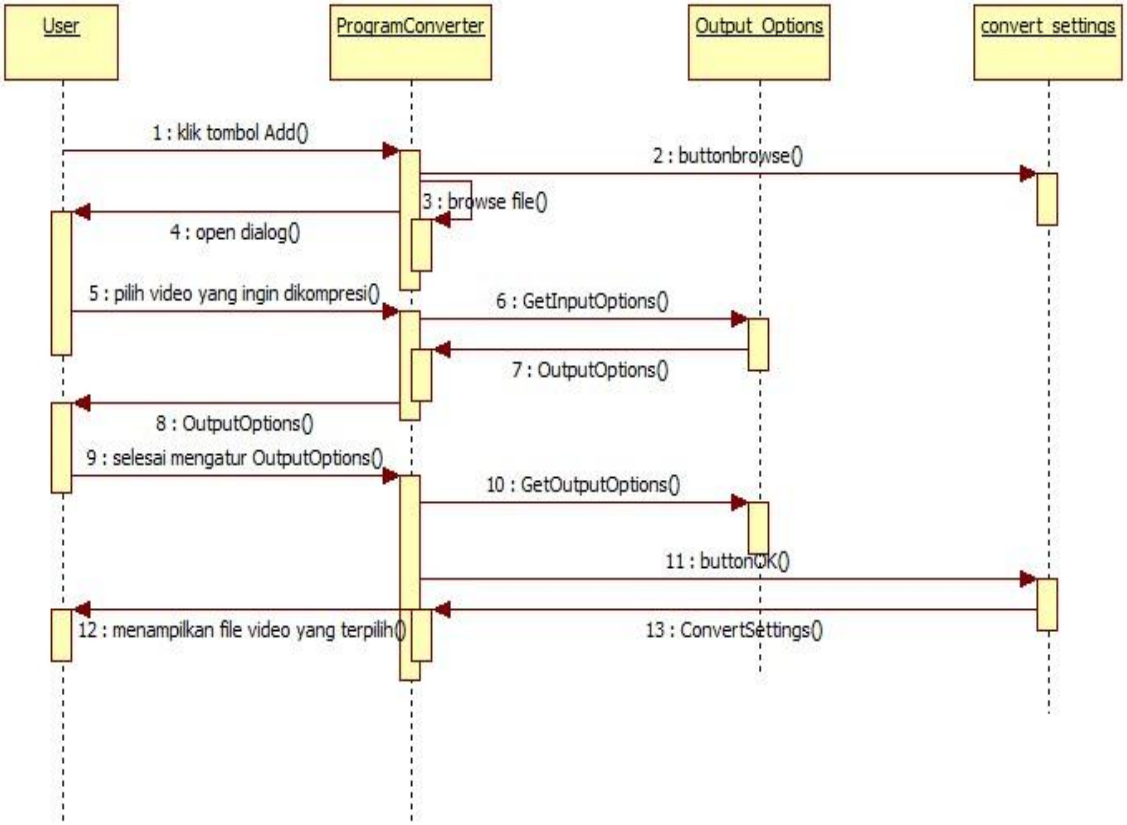


Gambar 3. 35 Activity Diagram Output Folder

3.1.5.4 Sequence Diagram

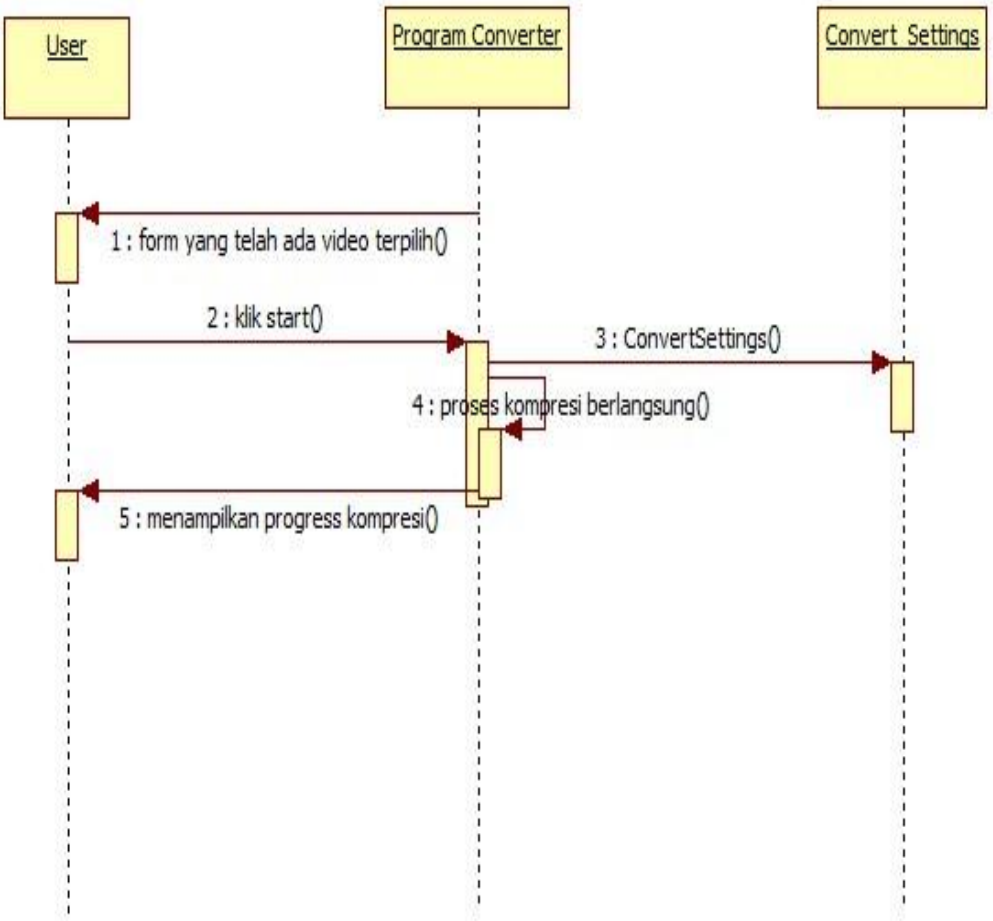
Sequence diagram menggambarkan perilaku pada sebuah skenario. Diagram ini menunjukkan sejumlah contoh obyek dan *message* (pesan) yang diletakkan diantara obyek-obyek ini di dalam *Use Case*. Komponen utama sequence diagram terdiri atas obyek yang dituliskan dengan kotak segiempat. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan *progres* vertikal. Berikut diagram *sequence* dari aplikasi kompresi video ini :

1. Diagram Sequence Menambahkan Video Yang Akan Dikompresi



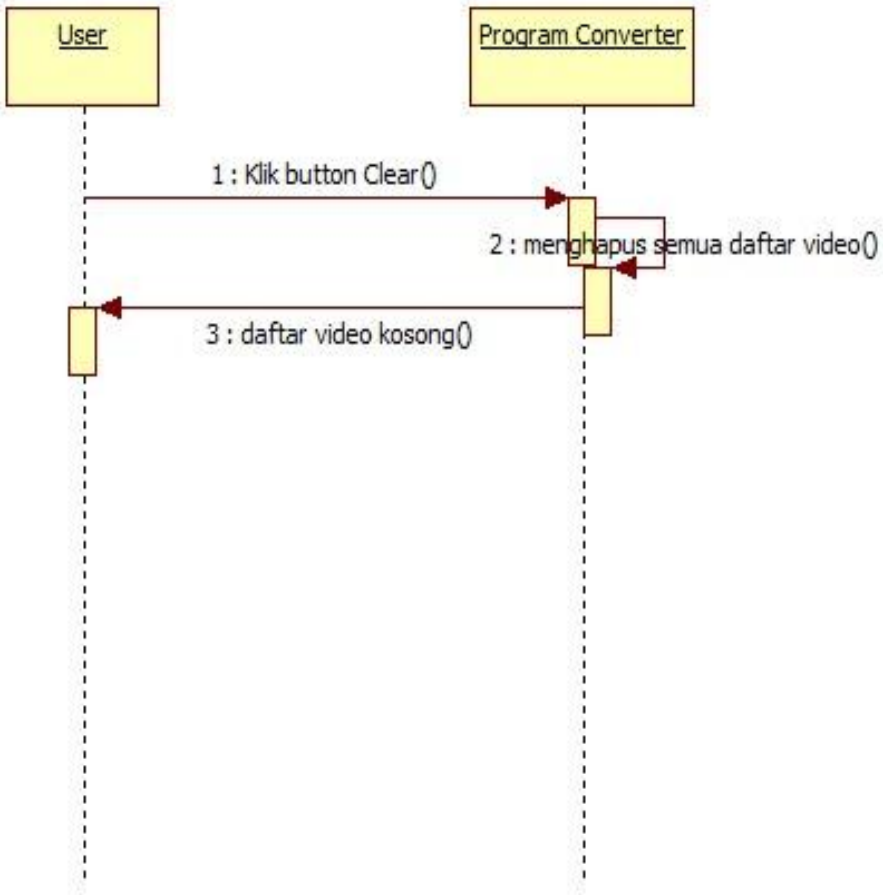
Gambar 3. 36 Sequence Diagram Proses Kompresi Video

2. Diagram Sequence Memulai Proses Kompresi



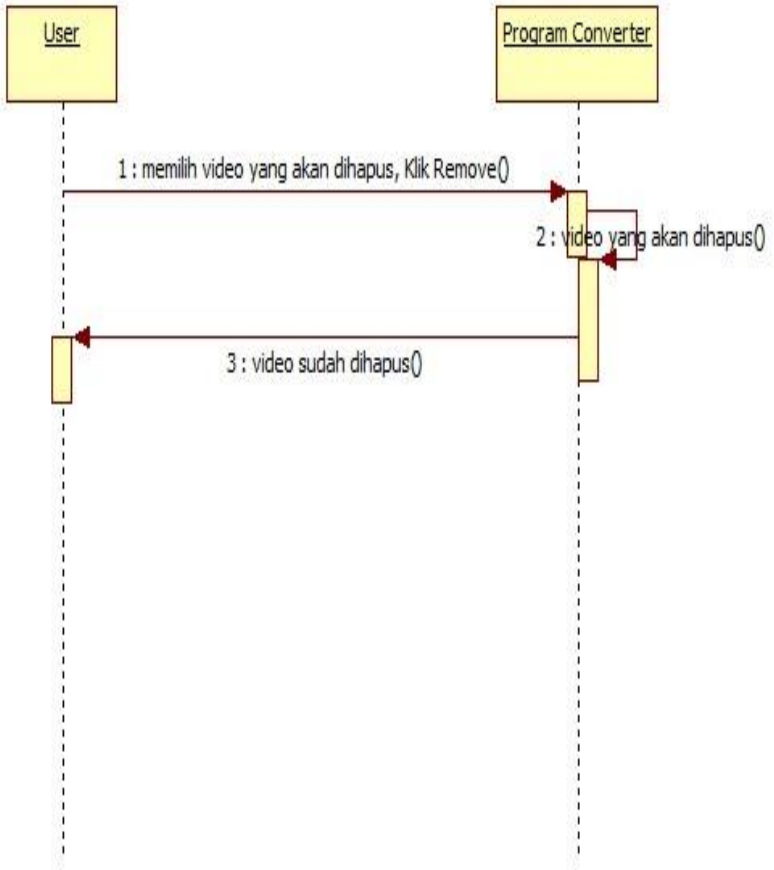
Gambar 3. 37 Diagram Sequence Memulai Proses Kompresi

3. Diagram Sequence Menghapus Daftar Video



Gambar 3. 38 Diagram Sequence Menghapus Daftar Video

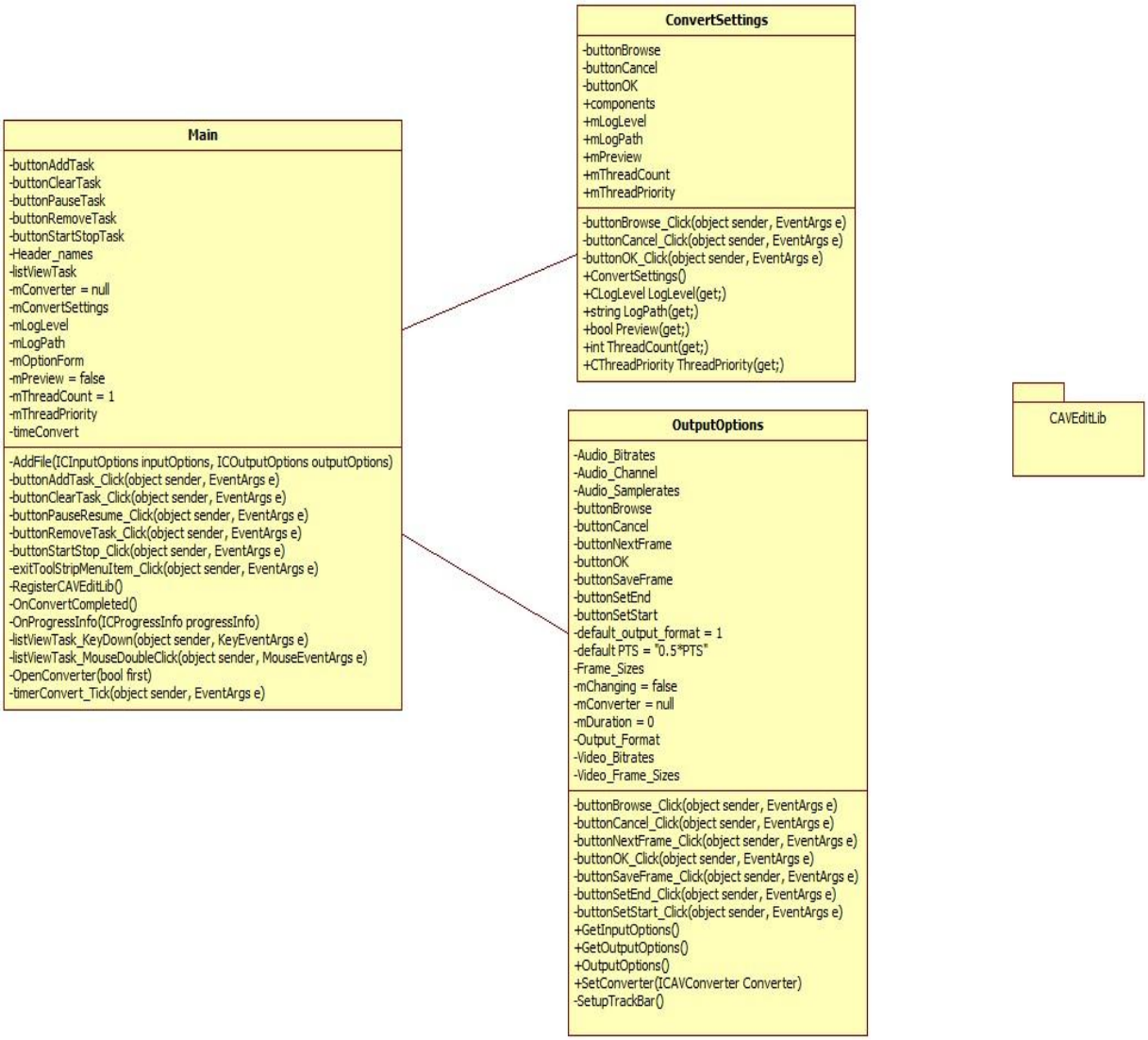
4. Diagram Sequence Menghapus Video Yang Akan Dikompresi



Gambar 3. 39 Diagram Sequence Menghapus Video Yang Akan Dikompresi

3.1.5.5 *Class* Diagram

Class diagram merupakan inti dari pemrograman berbasis objek karena diagram ini memberikan petaan terhadap kelas-kelas yang digunakan oleh suatu aplikasi. *Class diagram* menggunakan struktur dan hubungan antar objek-objek yang ada pada sistem. Struktur ini meliputi atribut-atribut dan metode-metode yang ada pada masing-masing *class*. Adapun aplikasi kompresi video ini memiliki *class diagram* sebagai berikut :



Gambar 3. 40 Class Diagram

3.2 Perancangan Sistem

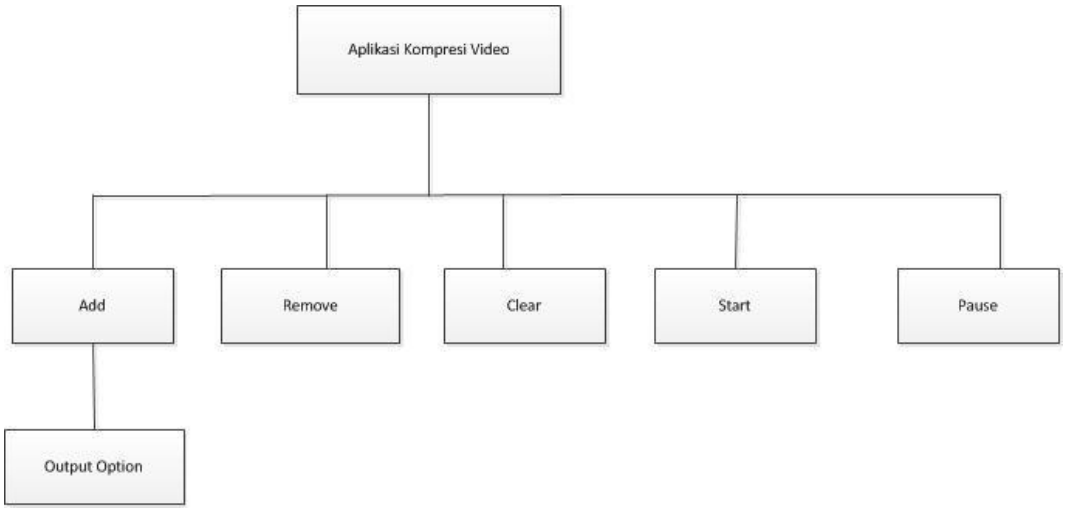
Perancangan sistem merupakan suatu proses yang menggambarkan bagaimana suatu sistem dibangun untuk memenuhi kebutuhan pada fase analisis. Tahap perancangan sistem terdiri dari perancangan arsitektur.

3.2.1 Perancangan Arsitektur

Perancangan arsitektur terdiri dari perancangan struktur menu, perancangan antarmuka, perancangan pesan dan perancangan jaringan semantik.

3.2.1.1 Perancangan Struktur Menu

Perancangan menu dilakukan untuk mempermudah interaksi antara sistem dengan pengguna. Perancangan struktur menu pada aplikasi ini dapat dilihat pada Gambar 3. 41 berikut :

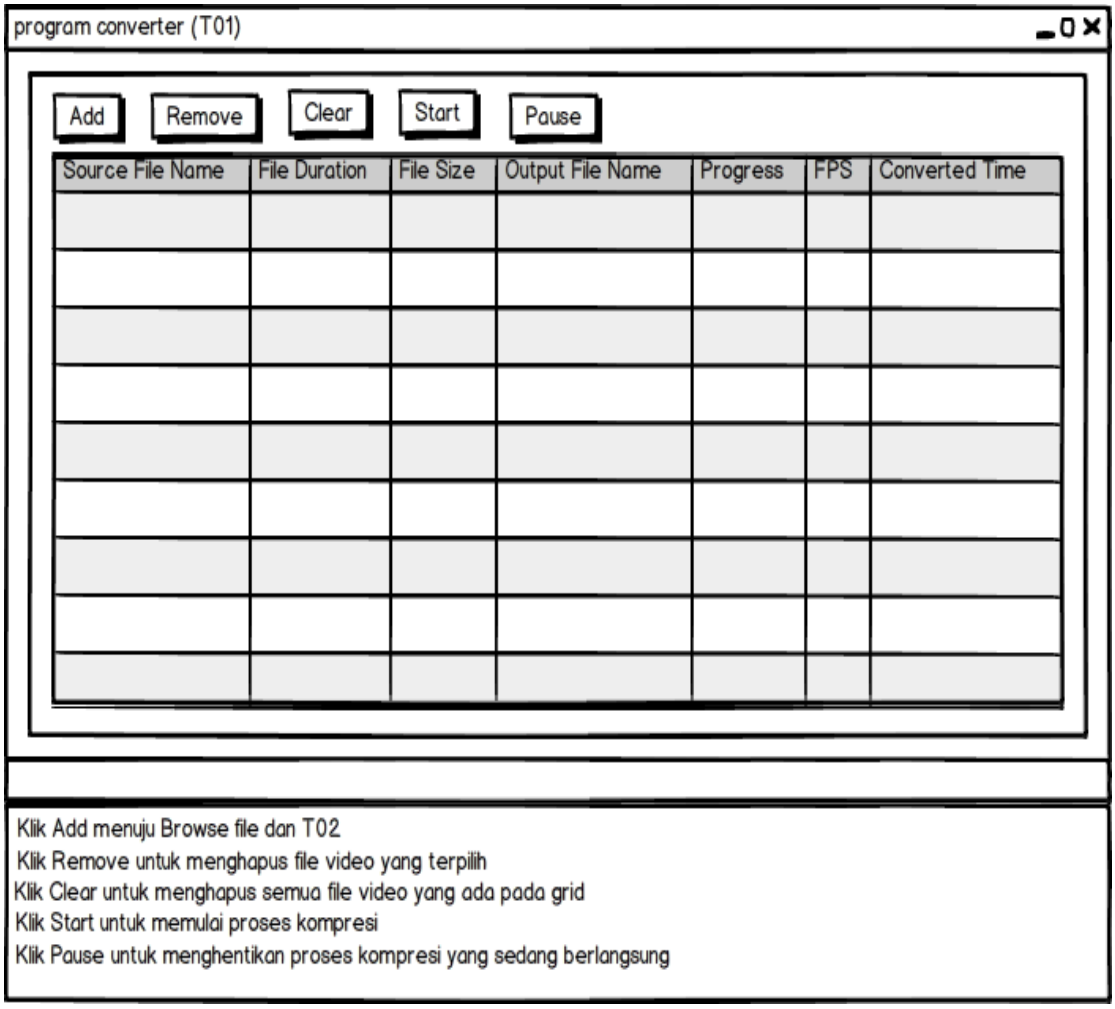


Gambar 3. 41 Perancangan Struktur Menu

3.2.1.2 Perancangan Antarmuka

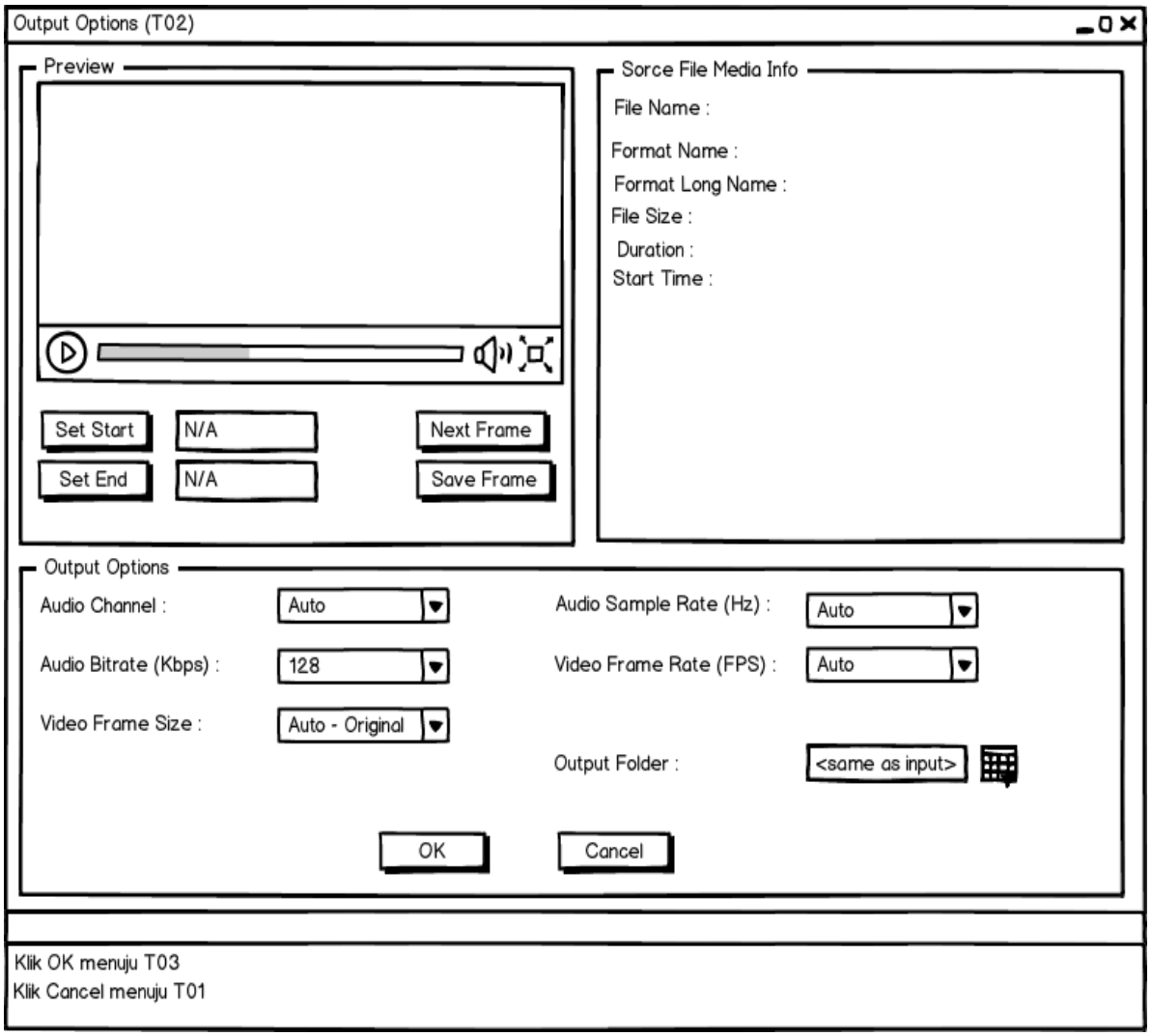
Tahap perancangan antarmuka bertujuan untuk memberikan gambaran tentang aplikasi yang akan dibangun, sehingga akan mempermudah dalam mengimplementasikan aplikasi serta akan memudahkan pembuatan aplikasi. Adapun perancangan antarmuka aplikasi kompresi video pada gambar-gambar ini adalah sebagai berikut :

1. **Perancangan Antarmuka Halaman Utama (T01)**



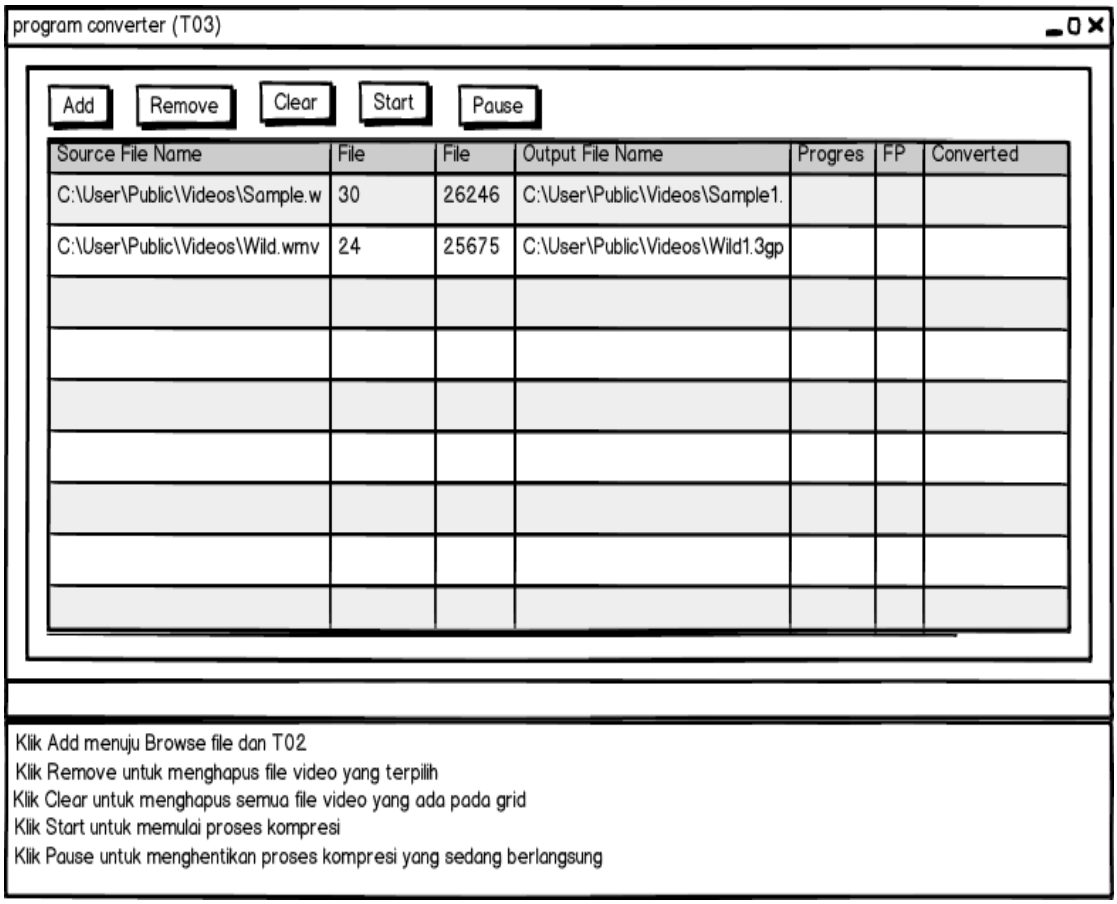
Gambar 3. 42 Perancangan Antarmuka Halaman Utama

2. Perancangan Antarmuka Output Options (T02)



Gambar 3. 43 Perancangan Antarmuka Output Options (T02)

3. Perancangan Antarmuka Halaman Setelah di Input Video (T03)



Gambar 3. 44 Perancangan Antarmuka Halaman Setelah di Input Video

3.2.1.3 Perancangan Pesan

Pesan merupakan tampilan dari suatu perangkat lunak yang berfungsi menyampaikan notifikasi dan informasi kepada pengguna agar perangkat lunak lebih interaktif. Terdapat beberapa perancangan pesan pada pembangunan aplikasi kompresi video ini, diantaranya :

1. **Perancangan pesan jika belum memilih video (M01)**

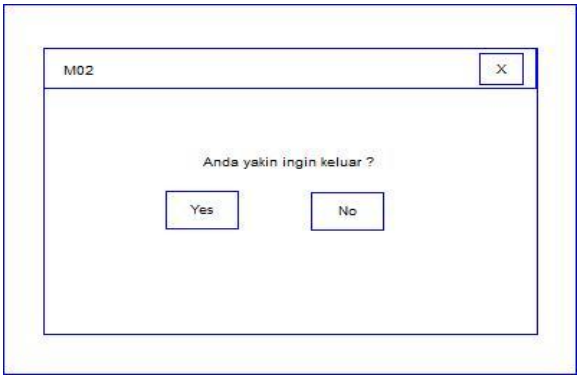
   Pesan ini muncul ketika pengguna menekan tombol *start* namun belum memasukkan *source* atau video terlebih dahulu. Perancangan pesannya sebagai berikut :



Gambar 3. 45 Perancangan Pesan Jika Belum Memilih Video

2. **Perancangan pesan ketika ingin keluar aplikasi (M02)**

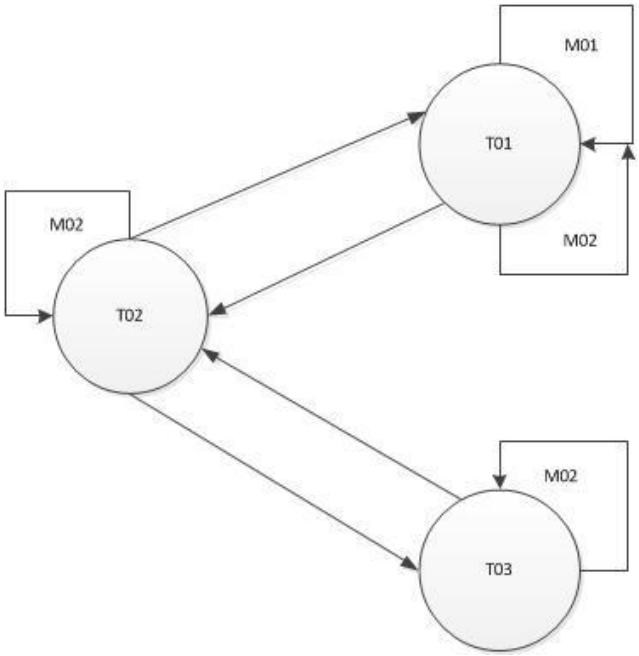
   Pesan ini muncul jika pengguna menekan menu Exit atau tombol keluar aplikasi.



Gambar 3. 46 Perancangan pesan ketika ingin keluar aplikasi

3.2.1.4 Perancangan Jaringan Semantik

Jaringan semantik adalah diagram yang menggambarkan aliran-aliran menu dan pesan dalam sebuah program. Jaringan semantik dari aplikasi kompresi video yang dibangun yaitu sebagai berikut :



Gambar 3. 47 Jaringan Semantik