TESIS – EE185401

PENGURANGAN KOMPLEKSITAS KOMPUTASI PADA MULTIVIEW HEVC BERBASIS PERANGKAT FPGA

M.SUHAIRI
07111650030002

DOSEN PEMBIMBING
Dr. Ir. Wirawan, DEA
Dr. Ir. Endroyono, DEA

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK TELEKOMUNIKASI MULTIMEDIA
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

TESIS – EE185401

PENGURANGAN KOMPLEKSITAS KOMPUTASI PADA MULTIVIEW HEVC BERBASIS PERANGKAT FPGA

M.SUHAIRI
07111650030002

DOSEN PEMBIMBING
Dr. Ir. Wirawan, DEA
Dr. Ir. Endroyono, DEA

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK TELEKOMUNIKASI MULTIMEDIA
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

LEMBAR PENGESAHAN

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Teknik (M.T)
di
Institut Teknologi Sepuluh Nopember

oleh:
M.Suhairi
NRP. 07111650030002

Tanggal Ujian: 20 Desember 2018
Periode Wisuda: Maret 2019

Disetujui oleh:

1. Dr. Ir. Wirawan, DEA (Pembimbing I)
   NIP: 196311091989031011

2. Dr. Ir. Endroyono, DEA (Pembimbing II)
   NIP: 196504041991021001

3. Prof. Ir. Gamantyo Hendrantoro, M.Eng., Ph.D. (Penguji)
   NIP: 197011111993031002

4. Eko Setijadi, ST., MT., Ph.D. (Penguji)
   NIP: 197210012003121002

5. Dr. Ir. Puji Handayani, MT. (Penguji)
   NIP: 196605101992032002

Dekan Fakultas Teknologi Elektro

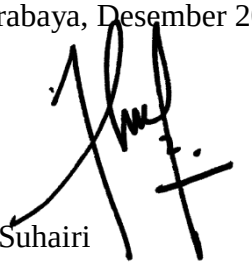
Dr. Tri Arief Sardjono, S.T., M.T.
NIP. 197002121995121001

PERNYATAAN KEASLIAN TESIS

Dengan ini saya menyatakan bahwa isi keseluruhan Tesis saya dengan judul "**PENGURANGAN KOMPLEKSITAS KOMPUTASI PADA MULTIVIEW HEVC BERBASIS PERANGKAT FPGA**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Desember 2018



M.Suhairi

NRP. 07111650030002

Halaman ini sengaja dikosongkan

PENGURANGAN KOMPLEKSITAS KOMPUTASI PADA MULTIVIEW HEVC BERBASIS PERANGKAT FPGA

Nama mahasiswa : M.Suhairi
NRP : 07111650030002
Pembimbing : 1. Dr. Ir. Wirawan, DEA
2. Dr. Ir. Endroyono, DEA

ABSTRAK

Dengan meningkatnya kualitas dan resolusi konten video, terutama video 3D, kompleksitas komputasi dalam pemrosesannya juga meningkat secara signifikan. Salah satu standar video yang populer, HEVC memiliki ekstensi yang dinamakan Multiview HEVC (MV-HEVC) dan 3D-HEVC dengan jumlah data dan resolusi yang tinggi, mengakibatkan adanya peningkatan kompleksitas komputasi. Penelitian ini bertujuan mengurangi kompleksitas komputasi dari video MV-HEVC dengan menerapkan mode decision berupa ECU, CFM, ESD, dan deblocking filter yang diujicobakan pada platform PC berbasis Linux dan platform Xilinx All Programmable SoC. Dari hasil eksperimen didapatkan pengurangan kompleksitas komputasi yang dilihat dari perbandingan dari waktu encoding. Platform Xilinx All Programmable SoC mampu memperoleh waktu encoding yang lebih cepat 35,85% daripada PC berbasis Linux. Selanjutnya untuk kualitas video yang dihasilkan antara kedua platform tersebut hampir sama dilihat dari nilai bitrate dan PSNR.

Keywords: MV-HEVC; H.265; xilinx; zynq

Halaman ini sengaja dikosongkan

COMPLEXITY REDUCTION FOR MULTIVIEW HEVC CODEC USING FPGA

By : M.Suhairi
Student Identity Number : 07111650030002
Supervisor(s) : 1. Dr. Ir. Wirawan, DEA
2. Dr. Ir. Endroyono, DEA

ABSTRACT

Due to the increasing quality and resolution of video content, especially 3D video, the computational complexity for its processing also significantly increases. One of the popular format, HEVC has extensions called Multiview HEVC (MV-HEVC) and 3D-HEVC with high amounts of data and high resolution that resulting in increased computational complexity. This study aims to reduce the computational complexity of MVHEVC videos by implementing mode decision such as ECU, CFM, ESD, and deblocking filters which are tested on Linuxbased PC platforms and the Xilinx All Programmable SoC platform. From the experimental results obtained the reduction in computational complexity can be seen from the comparison of encoding time, the Xilinx All Programmable SoC platform is able to obtain encoding times that are faster than Linux-based PCs. For the quality of the video produced between the two the platform is not significant from the bitrate and PSNR values.

Key words: MV-HEVC; H.265; xilinx; zynq

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Alhamdulillah, dengan mengucap puji syukur kehadirat Allah SWT , penulis dapat menyelesaikan buku tesis yang berjudul **"PENGURANGAN KOMPLEKSITAS KOMPUTASI PADA MULTIVIEW HEVC BERBASIS PERANGKAT FPGA".**

Tesis ini disusun untuk memenuhi salah satu syarat akademik Program Magister Jurusan Teknik Elektro, Institut Teknologi Sepuluh Nopember Surabaya. Banyak pihak yang telah memberikan bantuan selama penulis menempuh pendidikan di Jurusan Teknik Elektro ITS ini, karena itu penulis mengucapkan terima kasih kepada :

1. Kedua orang tua serta kedua saudari penulis, Lia Purnama Sari dan Rika Tri Andini atas doa, semangat, serta dukungan yang selalu mengiringi penulis dalam kondisi apapun.
2. Bapak Dr. Ir. Wirawan, DEA, Bapak Dr. Ir. Endroyono, DEA, dan Bapak Astria Nur Irfansyah, ST., M.Eng., Ph.D yang memberikan bimbingan, motivasi, serta arahan hingga penulis mampu menyelesaikan penelitian ini.
3. Bapak / ibu dosen pengajar di Program Studi Telekomunikasi Multimedia.
4. Teman – teman S2 TMM 2016 dan anggota Lab A205 yang memotivasi dan membantu penulis dalam menyelesaikan penelitian ini.
5. Semua pihak yang tidak dapat disebutkan satu per satu yang telah banyak membantu dalam menyelesaikan buku ini.

Penulis menyadari bahwa masih banyak kekurangan dalam penyusunan buku tesis ini, sehingga segala kritik dan saran yang membangun sangat penulis harapkan untuk menyempurnakan tesis ini.

Surabaya, 12 Desember 2018
Penulis

M.Suhairi
NRP. 07111650030002

Halaman ini sengaja dikosongkan

DAFTAR SINGKATAN

APU	Application Processing Unit
AU	Access Unit
CFM	Cbf Fast Mode Decision
CLB	Configurable Logic Block
CLI	Command Line Interface
CTU	Coding Tree Unit
CU	Coding Unit
ECU	Early CU Termination
EEMBC	Embedded Microprocessor Benchmark Consortium
EMIO	Extended MIO
ESD	Early Skip Detection
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit
FVT	Free-Viewpoint Television
FVV	Free-Viewpoint Video
GOP	Group of Pictures
GPIO	General Purpose Input/Output
GPU	Graphical User Interface
HEVC	High Efficiency Video Coding
HLS	High-level Syntax
IOB	Input/Output Block
JCT-VC	Joint Collaborative Team on Video Coding
LAN	Local Area Network
LUT	Lookup Tables
MIO	Multiplexed Input/Output
MMU	Memory Management Unit
MPE	Media Processing Engine

MPEG	Moving Picture Experts Groups
MVC	Multiview Video Coding
MV-HEVC	Multiview HEVC
NAL	Network Abstraction Layer
OCM	On Chip Memory
PL	Programmable Logic
PS	Processing System
PU	Prediction Unit
QP	Parameter Kuantisasi
RAM	Random Access Memory
ROM	Read Only Memory
SDK	Software Development Kit
SEI	Supplemental Enhancement Information
SHVC	Scalable HEVC
SOC	System on Chip
TU	Transform Unit
VLSI	Very Large Scale Integrated Circuits
WLAN	Wireless Local Area Network

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Dalam beberapa tahun terakhir ini, perkembangan dari teknologi 3D sangat pesat. Hal ini dipengaruhi oleh kebutuhan konsumen terhadap konten dari teknologi 3D itu sendiri yang semakin meningkat sehingga para peneliti dan perusahaan yang bergerak di bidang *broadcasting* dan lainnya yang menggunakan teknologi 3D berusaha untuk meningkatkan kualitas dan kenyamanan bagi konsumen. Teknologi 3D memiliki beberapa keunggulan yaitu menampilkan bentuk yang nyata dari sebuah objek dan menyajikan informasi yang mendalam[1]. Selain itu, teknologi 3D memberikan pengalaman visual melalui persepsi pada tampilan yang memproyeksikan video 3D dari arah yang berbeda untuk mata kiri dan mata kanan[2].

Teknologi 3D seperti holografis, sistem stereoscopic dengan menggunakan kacamata spesial, 3D dengan layar lebar, dan multiview telah dipelajari untuk memenuhi kebutuhan bagi konsumen. Selain itu, metode multiview adalah kunci dari teknologi 3D untuk berbagai aplikasi, termasuk pada *free-viewpoint video* (FVV), *free-viewpoint television* (FVT), *immersive teleconference*, dan 3DTV. Video multiview menawarkan pemandangan dengan sudut pandang yang dinamis dan memungkinkan untuk menyajikan video yang realistis kepada pengguna[3]. Teknologi 3D yang telah ada antara lain adalah holografis, stereoskopis dan multiview yang memberikan pengalaman yang berbeda bagi pengguna untuk adegan visual saat digunakan. Jenis tampilan video 3D yang umumnya digunakan adalah berupa tampilan stereoskopis yang menggunakan kacamata khusus dan tampilan autostereoskopis atau multiview yang memancarkan pixel yang bergantung kepada tampilan dan tidak memerlukan alat seperti kacamata.

Standar pengkodean video telah banyak berkembang seiring dengan kebutuhan konsumen. *High Efficiency Video Coding* (HEVC) disebut juga H.265 adalah suatu teknik pengkodean video yang telah didesain menjadi standar untuk

banyak aplikasi video dan memiliki kehandalan yang cukup signifikan dikembangkan oleh Joint Collaborative Team on Video Coding (JCT-VC) untuk meningkatkan efisiensi pengkodean video dibandingkan standar pengkodean video sebelumnya H.264/AVC dengan pengurangan bitrate yang signifikan hingga 50%[4]. Pada HEVC terdapat dua ekstensi yang mengikuti pendahulunya dan distandarisasi oleh JCT-VC yaitu Multiview HEVC (MV-HEVC) dan 3D-HEVC. Prinsip utama dari MV-HEVC adalah untuk menggunakan kembali coding tools yang mendasari pengkodean 2D [5]. Standar MV-HEVC dan 3D-HEVC memungkinkan untuk melakukan efisiensi encoding dari urutan gambar secara simultan yang ditangkap oleh banyak kamera dengan memanfaatkan korelasi antar *view* dan juga korelasi spasial dan temporal yang ada pada masing – masing view.

Bersamaan dengan dirilisnya standar pengkodean terbaru HEVC beserta dengan ekstensinya, JCT-VC mengeluarkan aplikasi yang dinamakan dengan *reference software* sebagai implementasi dari standar pengkodean terbaru tersebut. HMT-16.3 adalah aplikasi *reference software* terbaru yang digunakan untuk proses encoding video dari ekstensi HEVC yaitu MV-HEVC dan 3D-HEVC.

Meskipun ekstensi dari HEVC yaitu MV-HEVC dan 3D-HEVC memiliki pengkodean yang sangat efisien, namun disamping itu untuk melakukan proses encoding video memerlukan beban prosesor yang berat dan menjalankan beban yang paralel pada saat pengkodean data yang berisi video. *Coding tools* terbaru digunakan pada 3D-HEVC untuk mendapatkan efisiensi coding yang tinggi seperti *Coding Tree Unit* (CTU), *Coding Unit* (CU), *Transform Unit* (TU), dan *Prediction Unit* (PU), namun hal ini juga membawa peningkatan pada kompleksitas komputasi yang mengakibatkan terjadinya *bottleneck* untuk aplikasi *real-time* dari MV-HEVC dan 3D-HEVC encoder, dengan kata lain CTU, CU, TU, dan PU adalah bagian yang mengakibatkan peningkatan kompleksitas komputasi pada encoder. Kompleksitas komputasi mengarah kepada kalkulasi yang dilakukan pada proses pengkodean seluruh video atau sebagian video, jumlah komputasi ini memberikan efek kepada proses waktu encoding [6]. Akibat lain dari peningkatan kompleksitas komputasi ini adalah konsumsi *power* yang besar, ukuran dari video yang besar, serta kualitas video yang berkurang.

Beberapa penelitian sebelumnya pada [7]–[11] untuk pengurangan kompleksitas komputasi pada *coding tools* CTU, CU, TU, dan PU ini telah pernah dilakukan pada HEVC. Pada [7] menggunakan algoritma *fast* berbasis pada ukuran informasi pada *neighboring* CU untuk encoder HEVC dalam mengatasi kompleksitas komputasi. Algoritma tersebut dapat menghemat waktu encoding dengan efisien pada QP rendah dan tinggi, algoritma ini dapat dikombinasikan dengan *Early CU Termination* (ECU) dan *Cbf Fast Mode Decision* (CFM) didapatkan efisiensi waktu pengkodean sebesar 50,71% untuk video HD. Dalam [8] *motion estimation* adalah salah satu faktor yang menyebabkan peningkatan kompleksitas komputasi pada HEVC. Konfigurasi mode *fast* berdasarkan pengoptimalan proses *motion estimation* menggunakan *diamond search* dan algoritma *fast mode decision* untuk *Coding Unit* (CU) *partitioning* digunakan untuk mengurangi kompleksitas komputasi pada encoder. Hasil dari simulasi menunjukkan efisiensi waktu encoding hingga 56,75% dengan mengabaikan pengurangan kualitas video dan penurunan bitrate. Pada [9] ukuran CU yang besar dikenal juga dengan macroblock, selain dapat menghemat bitrate hal ini juga mengakibatkan tingginya kompleksitas komputasi. Menggunakan algoritma *fast* CU *decision* pada level *frame* dan level CU untuk HEVC dapat mempercepat prosedur encoding pada ukuran CU yang besar, waktu encoding dapat diefisiensikan sebesar 45%. Skema *Early Transform Unit* (TU) *Termination* berbasis deteksi *Quasi-Zero-Block* diusulkan pada [10] untuk mengurangi kompleksitas komputasi yang disebabkan oleh TU saat proses encoding. Hasil dari eksperimen menggunakan metode tersebut didapatkan pengurangan pada waktu encoding sebesar 22,82% dan 50,59% untuk pemrosesan TU dengan pengurangan kualitas video sebesar 0,04dB. Selanjutnya dalam [11] dengan mengatur *motion search range* dikombinasikan dengan *Early Termination mode search* untuk inter/intra *prediction* dan menimbang perbedaan antara *base view* dan *dependent* view pada 3D video coding / multiview, didapatkan pengurangan kompleksitas komputasi sebesar 29,64%.

Oleh karena itu, dari penelitian yang telah dilakukan sebelumnya pada HEVC, pada penelitian kali ini akan dicoba mengurangi kompleksitas komputasi pada Multiview HEVC (MV-HEVC) dengan mengadopsi beberapa *mode decision*,

mode decision yang akan diadopsi untuk pengurangan kompleksitas komputasi seperti ECU, CFM, *Early Skip Detection* (ESD), dan *deblocking filter*. *Mode decision* ini akan diterapkan pada 3 kondisi yang berbeda pada video MV-HEVC menggunakan *reference software* HTM-16.3. Fokus dari penelitian ini adalah analisis pada waktu encoding dan kualitas video dari penerapan *mode decision,* untuk kualitas video nantinya dilihat pada nilai PSNR dan bitrate pada video MV-HEVC yang dikodekan. Pada [12] diusulkan peningkatan kehandalan dalam proses encoder yang dilakukan dengan mengimplementasikan kode HEVC ke Xilinx All Programmable SoC, didapatkan hasil dari implementasi tersebut waktu pemrosesan lebih cepat sebesar 5% dari PC berbasis Linux. Maka dari itu penelitian ini nantinya juga akan dicoba mengimplementasikan pengurangan kompleksitas komputasi dengan *mode decision* yang telah disebutkan sebelumnya lalu dikonfigurasikan pada *reference software* MV-HEVC untuk *platform* PC berbasis Linux dan *platform* Xilinx All Programmable SoC. Dari implementasi ini nantinya akan dibandingkan pengurangan kompleksitas komputasi yang ditinjau dari waktu encoding dan kualitas video antara *platform* PC berbasis Linux dengan *platform* Xilinx All Programmable SoC.

1.2 Rumusan Masalah

Permasalahan yang dibahas pada penelitian ini adalah sebagai berikut :

1. Bagaimana mengurangi kompleksitas komputasi pada encoder MV-HEVC?
2. Bagaimana melakukan implementasi pengurangan kompleksitas komputasi video MV-HEVC pada *platform* PC berbasis Linux dan *platform* Xilinx All Programmable SoC?
3. Bagaimana menganalisa hasil perbandingan pengurangan kompleksitas komputasi pada *platform* PC berbasis Linux dengan *platform* Xilinx All Programmable SoC?

1.3 Tujuan

Tujuan yang diharapkan tercapai setelah penelitian ini selesai adalah sebagai berikut :

1. Untuk membuktikan pengurangan kompleksitas komputasi dengan menggunakan *mode decision*.
2. Untuk mengetahui pengurangan kompleksitas komputasi pada MV-HEVC antara *platform* PC berbasis Linux dan *platform* Xilinx All Programmable SoC.
3. Untuk mengetahui kualitas video dari pengkodean video MV-HEVC pada *platform* PC berbasis Linux dan *platform* Xilinx All Programmable SoC.

1.4 Batasan Masalah

Batasan masalah pada penelitian ini adalah sebagai berikut :

1. Menganalisis pengurangan kompleksitas komputasi pada MV-HEVC.
2. Hanya dikodekan pada bagian encoder saja.
3. Jumlah view yang digunakan terbatas untuk 3 view.
4. Menggunakan 3 kondisi yang berbeda untuk mengetahui pengurangan kompleksitas komputasi yang terdiri dari Kondisi Baseline, Kondisi 1, dan Kondisi 2.

1.5 Kontribusi

Penelitian ini diharapkan dapat memberikan kontribusi berupa pengurangan kompleksitas komputasi pada MV-HEVC, mampu meningkatkan kinerja encoder, dan mengurangi waktu encoding dalam pengkodean video MV-HEVC.

1.6 Sistematika Penulisan

Sistematika penulisan merupakan rincian penulisan laporan tesis yang berisi tentang tahapan – tahapan yang dilakukan selama penelitian. Berikut merupakan penjelasan tahapan – tahapan dari penulisan tesis :

BAB 1 PENDAHULUAN

Pada bab 1 berisi tentang latar belakang, rumusan masalah, tujuan penelitian, batasan masalah, kontribusi penulis, dan sistematika penulisan.

BAB 2 KAJIAN PUSTAKA

Pada bab 2 menelaskan terkait kajian pustaka dari penelitian yang akan dilakukan yaitu membahas mengenai kompleksitas komputasi, dasar – dasar video 3D, *High Efficiency Video Coding*, Ekstensi dari HEVC yaitu MV-HEVC dan 3D-HEVC.

BAB 3 METODOLOGI PENELITIAN

Pada bab 3 menjelaskan tentang metodologi penelitian dalam pengurangan kompleksitas komputasi pada MV-HEVC serta implementasinya pada *platform* PC berbasis Linux dan Xilinx All Programmable SoC. Dalam hal ini akan menggunakan 3 *test condition* yang nantinya akan dicobakan pada kedua *platform* tersebut. Metode pengurangan kompleksitas komputasi yang digunakan dalam pengkodean video MV-HEVC ini adalah ECU, ESD, CFM, dan *deblocking filter*. Implementasi dari metode ini bertujuan untuk mengurangi kompleksitas komputasi pada encoder MV-HEVC. Untuk mengevaluasi dari pengurangan kompleksitas komputasi tersebut dilakukan dengan perhitungan PSNR, bitrate dan waktu encoding.

BAB 4 HASIL PENELITIAN DAN PEMBAHASAN

Pada bab 4 berisi tentang hasil implementasi pengurangan kompleksitas komputasi menggunakan *reference software* HTM-16.3 pada *platform* PC berbasis Linux dan Xilinx All Programmable SoC beserta dengan analisisnya. Analisis pengurangan kompleksitas komputasi yang dibahas adalah dari perbandingan waktu encoding dan kualitas video yang dikodekan pada kedua *platform* yaitu PC berbasis Linux dan Xilinx All Programmable SoC menggunakan *reference software* HTM-16.3.

BAB 5 PENUTUP

Bab 5 menjelaskan isi kesimpulan dari penelitian yang telah dilakukan serta saran untuk pengembangan penelitian selanjutnya.

BAB 2
KAJIAN PUSTAKA

Bab 2 diawali dengan kajian penelitian terkait tentang pengurangan kompleksitas komputasi. Selanjutnya, teori dasar yang terdiri dari beberapa bahasan yaitu kompleksitas komputasi, dasar – dasar video 3D, High Efficiency Video Coding (HEVC), Multiview Video Coding, dan Xilinx All Programmable SoC. Kajian penelitian terkait ini bertujuan untuk memuat tentang hasil penelitian sebelumnya yang berupa penelitian yang dilakukan oleh orang lain. Sehingga, peneliti dapat melihat sejauh mana penelitian terkait judul tesis sudah pernah dilakukan atau dipublikasikan. Selanjutnya, bab berikutnya membahas mengenai tinjauan pustaka dari penelitian terkait.

2.1 Kajian Penelitian Terkait

Beberapa kajian penelitian terkait yang mendukung pengurangan kompleksitas komputasi pada HEVC telah dilakukan, berikut tabel 2.1 berisikan penjelasan singkat mengenai kelebihan masing – masing penelitian yang menjadi referensi dalam melakukan penelitian ini.

Tabel 2.1 Deskripsi Penelitian Sebelumnya

No	Peneliti	Judul	Deskripsi
[7]	W.-J. Hsu and H.-M. Hang (2013)	*Fast Coding Unit Decision Algorithm for HEVC*	Algoritma *fast* berbasis pada ukuran informasi pada *neighboring* CU untuk encoder HEVC dikombinasikan dengan *Early CU Termination* (ECU) dan *Cbf Fast Mode Decision*
[8]	R. Khemiri, N. Bahri, F. Belghith, F. E. Sayadi, M. Atri, and N. Masmoudi	*Fast Motion Estimation for HEVC Video Coding*	Konfigurasi mode *fast* berdasarkan pengoptimalan proses *motion estimation* menggunakan *diamond search* dan algoritma *fast mode decision* untuk *Coding Unit* (CU) *partitioning* digunakan untuk mengurangi

			kompleksitas komputasi pada encoder
[9]	J. Leng, L. Sun, T. Ikenaga, and S. Sakaida	*Content Based Hierarchical Fast Coding Unit Decision Algorithm for HEVC*	Algoritma *fast* CU *decision* pada level *frame* dan level CU untuk HEVC dapat mempercepat prosedur encoding pada ukuran CU yang besar
[10]	Y. Shi, Z. Gao, and X. Zhang	*Early TU Split Termination in HEVC Based on Quasi-Zero-Block*	Skema *Early Transform Unit* (TU) *Termination* berbasis deteksi *Quasi-Zero-Block*
[11]	H. R. Tohidypour, M. T. Pourazad, P. Nasiopoulos, and V. Leung	*A Content Adaptive Complexity Reduction Scheme for HEVC-Based 3D Video Coding*	Mengatur *motion search range* dikombinasikan dengan *Early Termination mode search* pada inter/intra *prediction* dan menimbang perbedaan antara *base view* dan *dependent* view pada 3D video coding / multiview untuk mengurangi kompleksitas komputasi

2.2 Teori Dasar

2.2.1 Kompleksitas Komputasi

Saat sekarang ini, televisi digital, komputer portabel, dan *smartphone* merupakan perangkat populer yang digunakan oleh konsumen yang mampu menerima dan menampilkan video dengan resolusi tinggi secara *real-time*. Perangkat tersebut juga mampu menangkap dan mentransmisikan video digital melalui koneksi Local Area Network (LAN) ataupun Wireless Local Area Network (WLAN). Selanjutnya, ini adalah tren pada perangkat portabel yang terpasang kamera digital dengan kemampuan untuk encoding dan decoding video beresolusi tinggi.

Terlepas dari itu semua, dalam konteks telekomunikasi dan komputasi daya yang digunakan, terbatasnya kapasitas baterai pada perangkat tersebut merupakan masalah utama pada aplikasi multimedia seperti dalam hal video encoding. Dalam beberapa kasus, *experience* dari pengguna yang menginginkan video dengan resolusi tinggi akan menjadi terbatas dikarenakan berkurangnya kapasitas daya / baterai dari perangkat tersebut. Selain itu, encoding dan decoding

video dengan resolusi tinggi juga merupakan salah satu tantangan terutama ketita mempertimbangkan kebutuhan komputasi dari standar video terbaru [6]. Ini merupakan konsekuensi dari evolusi standar video yang menggunakan *tools* efisiensi dari pemrosesan sinyal yang secara signifikan menambah jumlah dari operasi per piksel yang dibutuhkan pada standar video terbaru.

Secara terminologi istilah kompleksitas komputasi digunakan untuk menjelaskan jumlah kalkulasi yang dilakukan pada satu waktu. Kompleksitas komputasi mengarah kepada kalkulasi yang dilakukan pada proses pengkodean seluruh video atau sebagian video, sebagai jumlah dari komputasi yang mempengaruhi secara langsung jumlah dari waktu penggunaan prosesor [6]. Jumlah komputasi ini memberikan efek kepada proses waktu yang dibutuhkan, kompleksitas komputasi sendiri diukur sebagai waktu pemrosesan. Pengurangan kompleksitas komputasi sendiri merupakan istilah yang digunakan untuk menjelaskan metode yang digunakan dalam mengurangi kompleksitas komputasi dari suatu proses. Setelah diterapkan ke algoritma yang ditentukan, teknik pengurangan kompleksitas komputasi mampu mengurangi jumlah dari sumber komputasi yang dibutuhkan untuk menyelesaikan proses encoding ke tingkat yang tergantung pada karakteristik video. Selain itu kualitas video yang dihasilkan juga mempengaruhi dalam pengurangan kompleksitas komputasi.

Skala kompleksitas komputasi sendiri diukur dari proses penyesuaian secara adaptif untuk mencapai target kompleksitas yang diinginkan yang ditentukan oleh pengguna atau sistem. Metode skala kompleksitas ini dapat mengurangi atau menaikkan komputasi pada encoding video dengan mengatur parameter encoder hingga kompleksitas komputasinya berada pada batas yang diinginkan. Pada standar video HEVC ada beberapa metode yang digunakan untuk mengurangi kompleksitas komputasi ini, diantaranya adalah[6] :

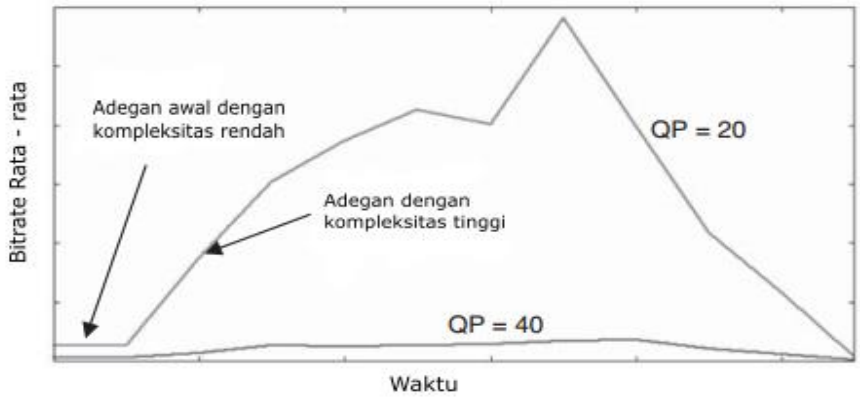
Tabel 2.2 Strategi pengurangan kompleksitas komputasi untuk HEVC

Kategori	**Pendekatan**
Ukuran / *depth* CU	Fast splitting untuk intra-coding
	Top skip + early termination
	Early termination
	Motion divergence

	Temporal correlation pada neighbouring frame
	Spatial / temporal correlation
	Skip method yang digunakan pada tree depth
Mode PU	Evaluasi kondisional
	Merge smaller PUs menjadi large PUs
	Filter out intra PU sizes
	Early SKIP mode detection
	Small intra PUs dikombinasikan menjadi larger PUs
Ukuran / depth TU	QZB-based early termination
	Nonzero coefficient-based early termination

Dari kategori dan pendekatan yang dipaparkan pada Tabel 2.2 untuk mengurangi kompleksitas komputasi pada HEVC dapat dilakukan melalui pendekatan *Early Temination* CU (ECU). Selain itu *skip mode detection* seperti *Early Skip Detection* (ESD) dan *fast splitting intra coding* seperti *Coded block flag Fast Methond* (CFM) mampu mengurangi kompleksitas komputasi pada HEVC.

Pada [13] dijelaskan efisiensi pengkodean, kompleksitas komputasi, konten video, frame rate, dan kuantisasi semua berdampak pada bitrate video. Nilai parameter kuantisasi (QP) yang lebih besar mengarah kepada encoding lossy yang mana mengurangi kualitas video dan bandwith yang dibutuhkan. Gambar 2.1 menunjukkan pada sebuah video yang dikompresi.



Gambar 2.1 Bitrate pada Video yang Dikompresi[13]

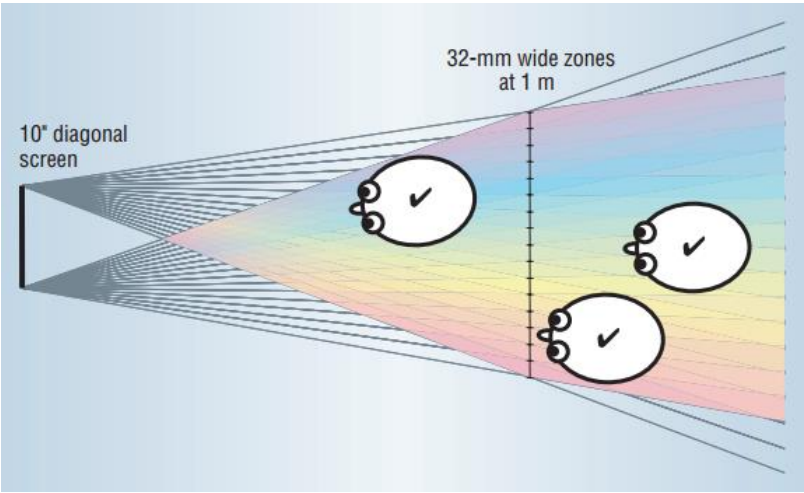
Dari Gambar 2.1, bagian awal video biasanya merupakan adegan sederhana, memungkinkan perangkat untuk mengolah video sebelum adegan kompleks tiba. Jika dilihat dari kuantisasi, level kuantisasi QP 20 dapat meningkatkan kualitas video namun kompleksitas komputasi pada video saat dikodekan menjadi lebih tinggi, selain itu video ini biasanya membutuhkan bandwidth jaringan yang lebih tinggi untuk transmisi dan waktu yang lebih lama untuk mengunduh karena lebih banyak detail warna dari video asli dipertahankan selama proses kompresi, terutama jika resolusi video tinggi. Untuk kuantisasi QP 40 bitrate pada video dapat diminimalkan sehingga mengurangi kompleksitas komputasi saat dikodekan namun banyak informasi yang dihapus dari setiap frame video dan kualitas video yang dihasilkan menjadi berkurang. Level kuantisasi memberikan efek kepada kompleksitas yang ada pada proses encoding, dimana kompleksitas dapat diturunkan jika nilai dari QP meningkat [6].

2.2.2 Dasar – dasar video 3D

Video 3D dimensi merupakan salah satu kemajuan dari teknologi video. Pengembangan dan penelitian terhadap video 3D ini menarik minat yang berkelanjutan. Perkembangan video 3D telah mengalami kemajuan yang sangat pesat, dan diaplikasikan pada beragam media seperti bioskop, televisi, komputer dan perangkat mobile. Dalam memproduksi video 3D, digunakan kamera yang berfungsi sebagai sensor fisik untuk melakukan pemrosesan berdasarkan geometri, fotometri, dan dinamika, yang memerlukan kalibrasi kamera geometris, fotometrik, dan dinamis. Karena ketepatannya untuk menentukan kualitas data video 3D yang dihasilkan, pemilihan kamera dan metode kalibrasi adalah salah satu proses terpenting dalam produksi video 3D. Selain itu, karena data video 3D dihasilkan dari data video multiview yang ditangkap oleh sekelompok kamera yang mengelilingi objek yang sedang bergerak, desain tata letak, geometri, fotometrik dan dinamik masing-masing diperlukan untuk kalibrasi kamera individual. Kamera video 3D sekarang mengalami kecenderungan semakin murah sehingga proses capture dan display video semakin umum dilakukan manusia. Namun disamping itu, sayangnya kemajuan pesat di bidang hardware yang telah mendukung video 3D, belum dapat diimbangi oleh kemajuan dibidang sisi perangkat lunak terutama

untuk konsumen media pemrosesan video 3D. Pada kebanyakan produsen video 3D menggunakan perangkat pengkodean 2D untuk 3D dalam pengkodean.

Video 3D multiview merupakan salah satu dari bentuk video 3D disamping tipe video 3D yang lain diantaranya adalah stereoscopy 3D dan Holoscopic. Video 3D multiview menggunakan menggunakan array dari banyak kamera untuk menangkap pemandangan 3D melalui beberapa aliran video independen pada satu waktu yang bersamaan dan dari posisi yang berbeda[3]. Kegunaan utama dari video 3D multiview adalah untuk mendukung aplikasi video 3D, dimana video 3D dengan persepsi depth dari visual yang didukung oleh sistem display video 3D[14], video 3D multiview dengan skenario viewpoint dan arah view yang berubah – ubah sehingga viewer dapat melihat dari arah yang berbeda selama masih berada dalam area yang dijangkau oleh kamera. Video 3D multiview sering juga disebut dengan video 3D autostereoscopy yaitu video yang tanpa menggunakan alat bantu khusus, dalam hal ini hasil implementasi terbaik menghasilkan efek yang dirasakan mirip dengan hologram cahaya putih. Untuk ilustrasi dapat dilihat pada Gambar 2.1.



Gambar 2.2 Video 3D multiview yang dilihat dari beberapa sudut pandang berbeda[15]

2.2.3 High Efficiency Video Coding (HEVC)

High Efficiency Video Coding (HEVC), yang juga dikenal dengan H.265, adalah sebuah standar terbuka yang ditetapkan oleh organisasi standarisasi telekomunikasi (ITU-T VCEG) dan industry teknologi (ISO/IEC MPEG)[16]. Pada

tiap satuan dekade, standar kompresi video mengalami peningkatan kualitas dibandingkan standar-standar sebelumnya. Sebagai contoh pada pengembangan HEVC, standar ini memiliki kemampuan yaitu dapat menurunkan cost pada transmisi dan media penyimpanan video namun disamping itu tetap dapat mempertahankan ataupun meningkatkan kualitas dari video tersebut. HEVC dapat mengurangi ukuran data video ataupun bit stream hingga 50% dibandingkan AVC/H.264 (generasi sebelumnya) atau mencapai 75% dibandingkan dengan standar MPEG-2 tanpa mengurangi kualitas video[17]. Peningkatan ini bermanfaat untuk mengurangi kebutuhan perangkat penyimpanan data video, biaya transmisi, dan juga memungkinkan untuk peningkatan kualitas video pada televisi broadcast dengan penggunaan besar ukuran kanal yang sama.

Cara yang dipakai dalam standar HEVC adalah berdasarkan pengkodean video secara hybrid. Fokus utama dalam pengkodean video secara hybrid dibuat dalam 3 bentuk : pembagian blok-blok, proses inter/intra prediction, dan proses transformasi. Pada ketiga proses tersebut, HEVC menggunakan blok partisi yang berukuran mulai dari 4x4 sampai dengan 32x32, sehingga algoritma yang diperlukan menjadi lebih kompleks daripada algoritma dalam H.264 dan MPEG-2.

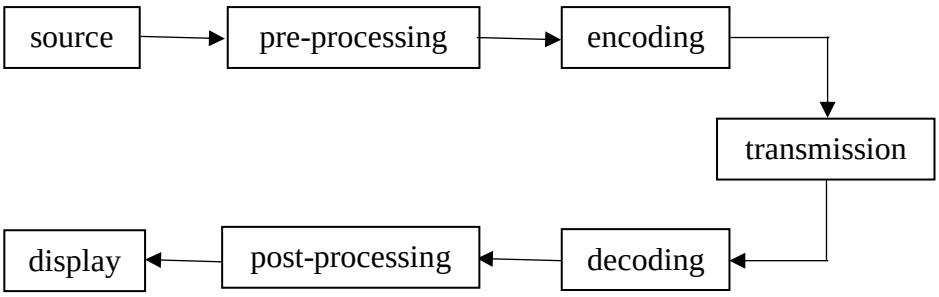
Dibutuhkan lebih banyak proses mengambil keputusan untuk melaksanakan perhitungan dalam kompresi video yang akan mengakibatkan meningkatnya beban processor pada proses video encoding. Untuk meningkatkan kemampuan kinerja dari encoder, dapat digunakan sebuah platform pengembangan untuk membuat algoritma kompresi secara paralel dan mampu mengeksekusi seluruh program dalam perangkat keras.

2.2.3.1 Metode Kompresi Video

Tujuan dari kompresi video adalah untuk menghilangkan informasi yang berlebihan (redudancy) dari streaming video sehingga video tersebut dapat dikirimkan melalui jaringan dengan cost yang seefisien mungkin. Proses encoding yang bertujuan untuk menghilangkan kelebihan informasi tersebut dilaksanakan dengan menggunakan algoritma tertentu. Encoding latency adalah jumlah waktu yang dibutuhkan sampai dengan proses encoding selesai. Sedangkan proses decoding adalah untuk membalik proses video yang tadinya terkompresi dan

mengembalikannya semirip mungkin ke keadaan video semula. Proses Kompresi dan dekompresi, bersama-sama dapat disebut sebagai pengkodean dasar. Pengkodean tesebut digunakan untuk mengurangi jumlah informasi dalam aliran bit video stream.

Diagram blok umum sistem pengkodean video dapat dilihat pada Gambar 2.2. Langkah umum pemrosesan video seperti pada Gambar 2.4 dapat dijelaskan sebagai berikut : Sumber video yang sama sekali belum terkompresi diproses di blok pra-processing menggunakan beberapa operasi seperti trimming, kompresi format warna, koreksi warna, dan denoising. Kemudian, blok encoding mengubah urutan input video ke dalam kode bitstream dan paket bitstream ke dalam format yang sesuai sebelum akan ditransmisikan melalui saluran. Dalam blok decoding, bitstream yang diterima akan direkonstruksi menjadi urutan video. Blok post-processing dapat memanfaatkan urutan video yang telah direkonstruksi untuk adaptasi pada urutan yang akan ditampilkan. Selanjutnya, urutan video siap untuk ditonton pada perangkat display. Dalam pembuatan standar kode, algoritma decoder harus telah didefinisikan dengan jelas, karena ruang lingkup dari standar, pada umumnya didasarkan pada decoder[16].



Gambar 2.3 Diagram blok umum sistem pengkodean video

Encoder dalam standar yang diberikan mungkin dapat bervariasi dari vendor ke vendor, atau bahkan dari satu produk ke produk lain dalam dari satu vendor. Variasi ini disebabkan oleh cara seorang desainer dalam mengembangkan tiap bagian tertentu dari standar dengan menggunakan alat yang spesifik. Beberapa kategori yang menjadi pertimbangan selama merancang kemampuan perangkat, faktor komersial, desain dan pengembangan lanjut, karakteristik perangkat fisik, dan fleksibilitas.

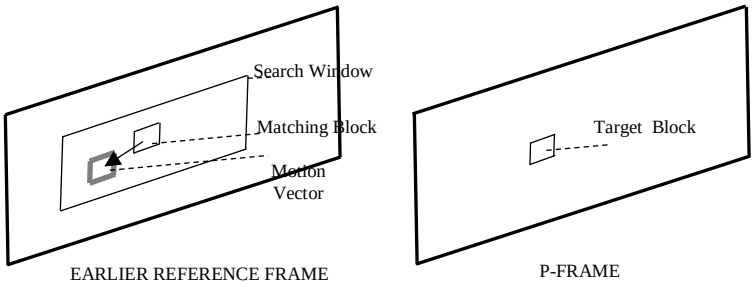
Encoder pada umumnya melaksanakan langkah-langkah seperti yang diilustrasikan sebagai berikut [18]:

a. Membagi setiap frame ke dalam beberapa blok piksel sehingga pengolahan dapat dilakukan secara paralel pada level blok.
b. Mengidentifikasi dan memanfaatkan redudansi spasial yang ada dalam frame dengan encoding beberapa blok asli melalui prediksi spasial dan teknik coding lainnya.
c. Mengeksploitasi hubungan temporal yang ada antara blok dengan frame berikutnya sehingga hanya perubahan antara frame yang dikodekan. Hal ini dapat dicapai dengan menggunakan vektor gerak estimasi yang memprediksi komposisi dari blok sasaran.
d. Mengidentifikasi dan meningkatkan kompresi dari setiap redudansi spasial tersisa yang ada dalam frame dengan pengkodean perbedaan antara frame asli dan blok prediksi dalam hal kuantisasi, transformasi, dan pengkodean entropi.

Prinsip cara kerja H.264 dan H.265 tidak terlalu berbeda[13]. Kedua standar tersebut sama-sama menggunakan *slice* dan *slice group*, dan membagi frame kedalam *region-region rectangular* yang akan diproses secara terpisah. Selama proses encoding, penggolongan jenis frame video, antara lain I-frame, P-frame, dan B-frame, sebagai target dari sebuah encoder. Ketika tipe frame yang berbeda digunakan dalam kombinasi, tingkat video bit rate dapat dikurangi dengan menemukan redudansi temporal (berbasis waktu) dan spasial antara frame yang masih memiliki informasi yang belum di encoding. Dengan cara ini, objek, atau dalam hal ini piksel atau blok piksel, yang tidak berubah dari frame satu ke frame selanjutnya atau merupakan replika dari pixel atau blok pixel di sekitarnya, dapat diproses secara tepat

Dalam implementasi algoritma motion compensation pada proses pengkodean, code ini mampu memperhitungkan kondisi bahwa sebagian besar dari frame baru dalam urutan video hampir selalu didasarkan pada frame sebelumnya. Jadi pada level blok demi blok, enkoder dapat menghitung posisi yang matching dalam frame dimana diperkirakan akan ada di frame berikutnya dengan menggunakan *vector motion*. *Vector motion* membutuhkan bit yang lebih sedikit

untuk mengkodekan keseluruhan blok dan pada muaranya akan menghemat bandwidth *code stream*.



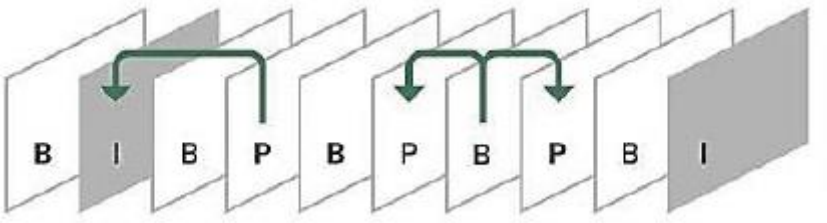
Gambar 2.4 Tahapan encoding dalam Block-based motion compensation

I-frame atau frame intra, adalah bingkai yang bersifat berdiri sendiri yang dapat dikodekan secara independen tanpa mengacu frame sebelumnya ataupun frame yang akan datang. Frame pertama selalu menjadi I-frame dalam urutan video dan akan menjadi titik awal bila aliran bit yang ditransmisikan rusak. I-frame dapat digunakan untuk proses pendeteksian pada *fast-forward*, mundur dan perubahan adegan.

Kelemahan dari sebuah I-frame adalah membutuhkan lebih banyak bit dan tidak memberikan level kompresi yang cukup memadai. Di sisi lain, I-frame tidak menghasilkan banyak artifact karena hanya akan menghasilkan sebuah gambaran yang lengkap dari sebuah frame.

P-frame, yang merupakan singkatan dari frame prediktif, yang akan menggunakan I-frame sebagai referensi dalam mengkodekan frame selanjutnya. P-frame pada umumnya memerlukan bit yang lebih sedikit daripada I-frame, namun lebih rentan terhadap kesalahan transmisi karena ketergantungan yang sangat signifikan pada frame referensi sebelumnya.

B-frame, yang merupakan singkatan dari frame bi-prediktif, adalah sebuah frame yang menggunakan referensi dua arah yaitu referensi frame sebelumnya dan frame yang selanjutnya. Sebuah P-frame akan hanya menjadikan referensi I- atau P-frame sebelumnya, sementara B-frame dapat mencari referensi baik I- atau P-frame sebelum maupun frame selanjutnya.



Gambar 2.5 Langkah kode dengan I-, B-, and P-frame

Teknik ini adalah merupakan salah satu teknik dasar dalam kompresi video. Masih ada teknik dan algoritma lain yang mengubah informasi tentang video untuk mengurangi jumlah bit yang akan ditransmisikan. Untuk informasi lebih lanjut dan lengkap dapat dilihat pada[16].

2.2.3.2 Pengembangan HEVC

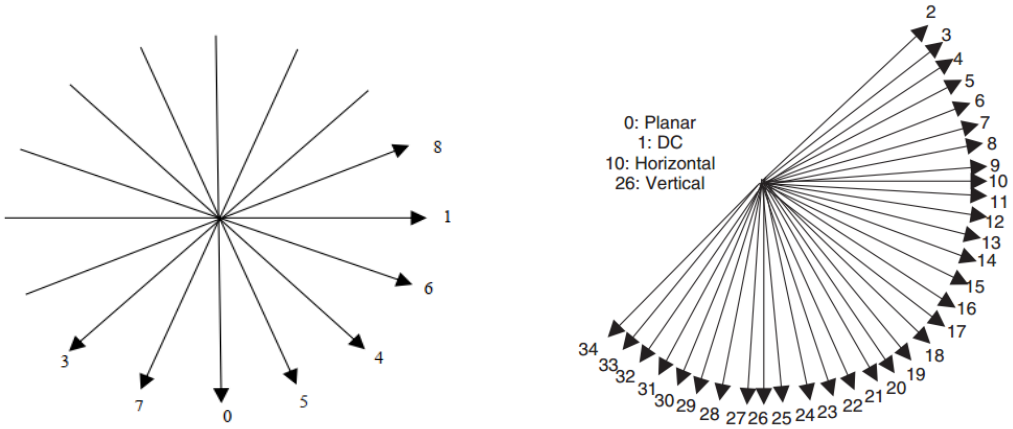
HEVC dikembangkan berdasarkan standar H.264 sebelumnya yang keduanya merupakan hasil pengembangan bersama antara ITU-T Coding Experts Group Video dan ISO / IEC Moving Picture Experts Groups (MPEG). ITU-T memfasilitasi penciptaan dan adopsi standar telekomunikasi dan ISO / IEC mengelola standar untuk industri elektronik. HEVC dirancang untuk mengembangkan kemampuan kompresi video dan mimiliki fitur sebagai berikut:

a. Memberikan pengurangan bit rate sampai dengan 50% namun mempertahankan kualitas video yang tetap dibandingkan dengan H.264
b. Memberikan kualitas yang lebih baik pada bit rate yang sama
c. Menetapkan sintaks standar untuk menyederhanakan implementasi dan memaksimalkan interoperabilitas
d. Tetap mudah diaplkasikan dalam jaringan karena masih dikemas dalam MPEG *Transport Streams*

Standar HEVC memberikan standar dasar kompresi dengan mendefinisikan 8-bit dan 10-bit 4:2:0, yang masih kompatibel dengan hampir keseluruhan format video lainnya.

2.2.3.3 Dampak Keuntungan Implementasi HEVC[19]

Dalam standar H.264 (generasi sebelumnya), sembilan mode prediksi dibuat dalam bentuk blok 4x4 untuk *intra prediction* pada tiap *frame* dan sembilan mode prediksi dibuat dalam bentuk blok level 8x8. Untuk level blok 16x16 hanya menyediakan empat mode *intra prediction. Intra prediction* akan mencari bagian blok-blok yang berseseuaian dengan arah pergerakan untuk meminimalkan eror dari estimasinya. Namun dalam HEVC, menggunakan teknik yang sama, tapi dengan jumlah kemungkinan mode sampai dengan 35 buah – yang tentunya sejalan dengan bertambahnya kompleksitas pengkodean. Hal ini menciptakan sejumlah analisis dan pengambilan keputusan yang meningkat secara dramatis lebih tinggi, karena secara spasial ada hampir dua kali lipat jumlah *intra prediction* dalam standar HEVC dibandingkan dengan H.264 dan hampir empat kali jumlah *intra prediction directions*.

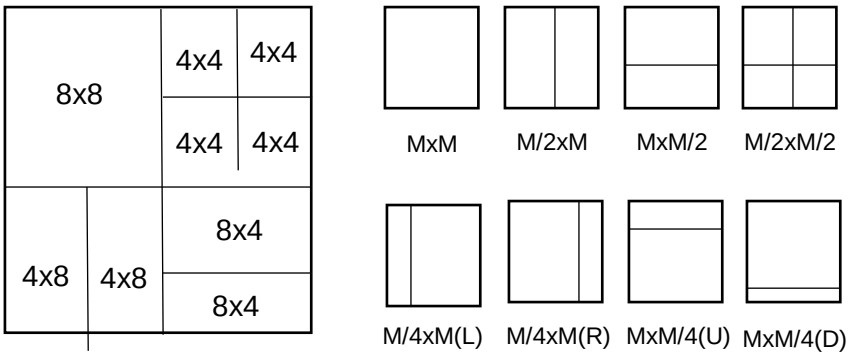


Gambar 2.6 Perbandingan mode intra prediction pada H.264 dan HEVC

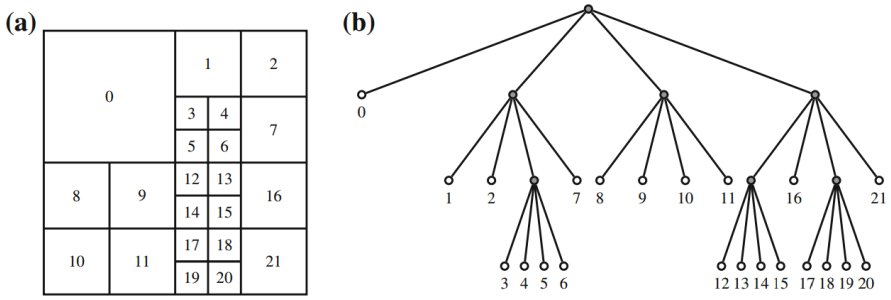
2.2.3.4 Inter Prediction dan Pengkodeannya

H.264 menggunakan block-based motion compensation dengan block size dan bentuk yang dapat disesuaikan untuk mencari *temporal redundancy* sepanjang *frame-frame* dalam sebuah *file* video. Motion compensation sering disebut sebagai bagian yang paling membutuhkan sumber daya dalam proses pengkodean. Tingkat yang dimplementasikan dalam mengambil keputusan memiliki efek yang sangat berpengaruh pada efesiensi dari pengkodean. Dan pada standar HEVC telah dikembangkan bentuk yang baru.

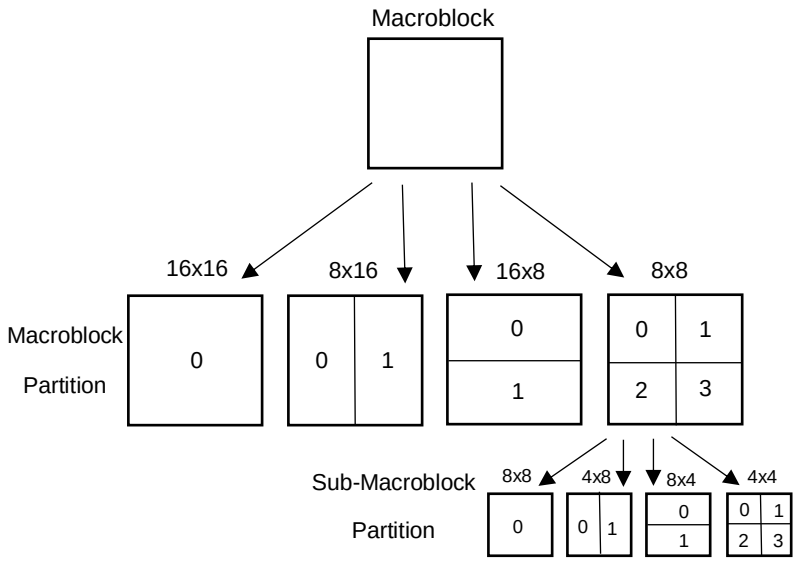
HEVC menggantikan bentuk struktur makroblok pada H.264 dengan lebih efisien, namun juga jauh lebih kompleks dengan sistem kumpulan *treeblocks*. Masing-masing *treeblock* dapat diperluas sampai dengan blok 64x64 dari blok 16x16, dan dapat secara efisien dipartisi menggunakan *quadtree*. Sistem ini dapat menjalankan pengkodean dengan fleksibilitas yang tinggi untuk menggunakan partisi yang berukuran besar pada saat prediksi antar frame cukup bagus dan sebaliknya menggunkan partisi yang lebih kecil saat prediksi yang detail diperlukan. Dengan fleksibilitas ini akan menghasilkan efisiensi yang lebih baik, karena prediksi dengan ukuran besar akan membutuhkan sumber daya yang lebih rendah, namun bila dibutuhkan beberapa bagian dari *treeblock* dapat dibuat lebih kecil pada saat prediksi yang lebih detail diperlukan, sehingga diperoleh efisiensi yang tinggi.



Gambar 2.7 Bentuk dari makroblok 16x16 standar H.264 dibandingkan dengan partisi pada HEVC



Gambar 2.8 Struktur pengkodean quad tree pada coding blocks. a. Partisi spasial. b. Bentuk quadtree blok yang berkorespondensi



Gambar 2.9 Partisi makroblok pada standar H.264 pada inter prediction

2.2.3.5 Pengolahan Paralel pada HEVC

HEVC telah dirancang sehingga memiliki kemampuan yang lebih baik dalam melaksanakan pengolahan paralel. Beberapa peningkatan proses paralel ini dapat dilihat pada :

a. Tiles
b. Filter deblocking in-loop
c. Pemrosesan paralel pada *wavefront*.

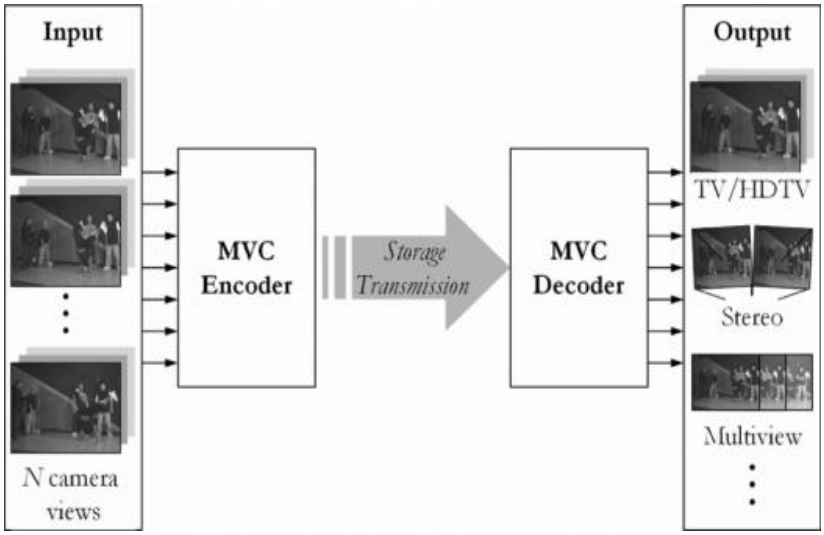
Tiles setiap frame dibagi kedalam beberapa bagian bujur sangkar sehingga tiap bagian dapat secara *independent* dikodekan dan didekodekan secara simultan. Sistem ini juga mampu untuk pengeksekusian secara *random* pada daerah tertentu pada tiap frame dalam sebuah *video stream.*

Dalam filter deblocking in-loop, ditetapkan hanya berlaku untuk tepi pada grid 8x8, untuk mengurangi tingkat interaksi antara blok dan menyederhanakan metodologi pemrosesan paralel. Arah pemrosesan ditetapkan lebih dahulu pada filtering horizontal pada tepi vertikal dan diikuti dengan filetering vertikal pada tepi horisontal. Sehingga akan mendukung independent multiple parralel dan perhitungan filter deblocking dapat dijalankan secara simultan.

Pemrosesan paralel pada *wafefront* (wavefront paralel processing/WPP) memungkinkan setiap *slice* dapat dipecal lagi menjadi kode *tree units (CTUs)* dan masing-masing unit CTU dapat didekodekan berdasarkan informasi dari CTU yang sebelumnya. Baris pertama akan diterjemahkan secara langsung namun pada baris selanjutnya membutuhkan keputusan seperti yang dibuat pada baris sebelumnya.

2.2.4 Multiview Video Coding

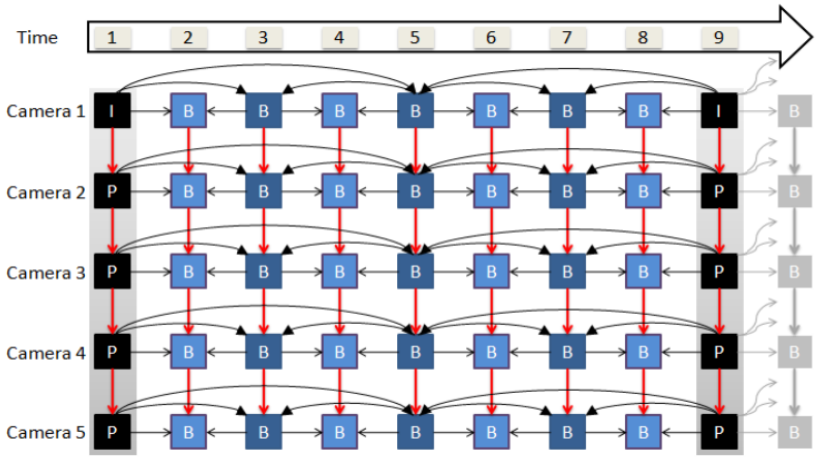
Multiview video coding (MVC) adalah sebuah proses yang mana sinyal video stereo dan multiview secara efisien dikodekan dimana setiap view pada multiview video dapat dikodekan secara independen menggunakan encoder[20]. Pada kebanyakan skema MVC digunakan tidak hanya untuk mengurangi redudansi yang terdapat diantara frame, dan juga kesamaan diantara frame dari neighboring view. MVC telah dipelajari lebih dari 20 tahun, dengan penemuan awal dilakukan pada prediksi disparity-compensated oleh Lukacs pada tahun 1986[21]. Struktur dari MVC dapat dilihat pada Gambar 2.10.



Gambar 2.10 Struktur dari Multiview Video Coding[22]

Untuk mendapatkan efisiensi dari pengkodean multiview video dibutuhkan pengurangan dari temporal redudancy diantara frame dari sebuah view dan redudancy diantara neighboring frame dari sebuah view. Perbedaan antara pengkodean video yang lama dengan MVC adalah ketersediannya dan kemampuan dari banyak kamera untuk menangkap gambar dalam waktu yang bersamaan. Dalam hal ini proses encoding dan decoding pada masing – masing view dilakukan

secara terpisah. Standar untuk MVC menggunakan hirarki struktur B-frame dengan tambahan mode inter-view prediction dari pengkodean video, pada MVC sinkroninasi view dilakukan dengan memprediksikan masing – masing view dengan memprediksikan frame dari kedua view pada view yang sama dan frame lainnya pada neighboring view. Konsep dari struktur hirarki gambar B dengan mudah dapat diimplementasikan pada urutan video multiview. Hal ini dapat mengurangi bitrate dari video tanpa harus mengurangi kualitasnya. Pada Gambar 2.11 dapat dilihat prediksi struktur dari MVC.



Gambar 2.11 Struktur Prediksi Gambar MVC[20]

Gambar 2.11 menunjukkan dengan 5 kamera dengan hirarki B-frame temporal yang di identifikasikan sebagai coding untuk stream kamera individual. Terdapat depedencies coding tambahan yang ditampilkan diantara interview prediction. Pada frame pertama untuk kamera 3 dikodekan dari frame kamera 1, kemudian ke frame pertama kamera 5. Frame pertama ini digunakan untuk memprediksi level frame B1 pada urutan individual, dan begitu seterusnya pada urutan ketiga dan kelima. Setelah Group of Pictures (GOP) selesai dibentuk, urutan kamera 2 dan kamera 4 dikodekan dengan pilihan terbaru untuk interview prediction dari frame yang telah dikodekan.

2.2.4.1 Multiview HEVC (MV-HEVC) & 3D-HEVC

Setelah distandarisasi pengkodean video HEVC oleh ITU-T sebagai standar video terbaru, komite standarisasi internasional melanjutkan pengembangan dari ekstensi HEVC untuk memenuhi kebutuhan berbagai aplikasi

yang lebih luas. Pada tahun 2013 diadakan pertemuan di Vienna untuk membahas ekstensi dari HEVC dan hasilnya terbagi kedalam 3 bagian yaitu[23] :

a. *The range extension*, yang mana memperluas *range* dari bit *depth* dan format sampling warna yang didukung oleh standar yang ada dan juga peningkatan *high-quality coding*, *lossless coding*, dan *screen-content coding*.
b. *Scalability extension*, memungkinkan untuk melakukan embedded bitstream subsets sebagai representasi pengurangan bitrate dari konten video.
c. *The 3D video extension*, memungkinkan representasi dari stereoskopis dan multiview memiliki kapabilitas untuk video 3D dalam penggunaan *depth maps* dan teknik *view-synthesis*.

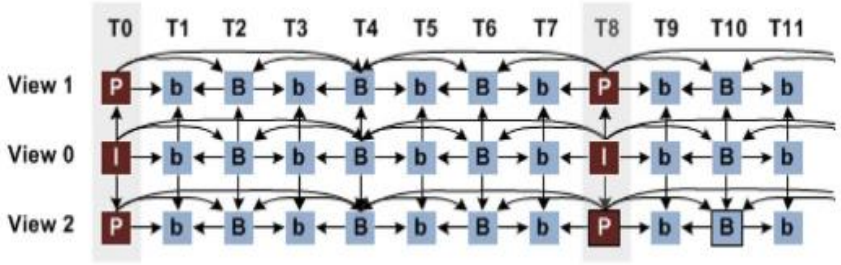
Format video 3D dan multiview mengizinkan persepsi dari kedalaman (*depth*) video pada adegan visual ketika digunakan dengan sistem *display* yang mendukung 3D. Jenis dari *display* 3D termasuk *display* stereoskopis yang menggunakan kacamata spesial untuk menampilkan *view* yang berbeda ke penglihatan *viewer*, dan autostereoskopis yang tidak memerlukan kacamata spesial namun melalui pancaran piksel.

Agar perenderan *view* mendapatkan kualitas yang diinginkan, maka dibutuhkan penggunaan *high-quality depth* maps, yang mana direpresentasikan dan dikodekan bersama tekstur dari video. *Depth maps* diestimasikan dari stereo atau multi kamera menggunakan teknik pencocokan stereo. Selain itu dapat juga menggunakan kamera spesial yang berdasarkan kepada struktur cahaya atau berdasarkan citra *time-of-light*, pada akhirnya informasi *depth* merupakan bagian integral dari pengolahan citra berbasis komputer yang populer di produsen film. Pada pengembangan HEVC, untuk versi pertama dari *reference software* yang dikontribusikan, berdasarkan platform perangkat lunak ini yang mana terdapat aplikasi untuk sintesis distorsi *view* berdasarkan optimasi *rate-distortion*.

MV-HEVC dan 3D-HEVC memiliki konsep desain yang sama dengan pendahulunya yaitu ekstensi dari H.264/MPEG-4 AVC. MV-HEVC terdiri dari penambahan *high-level syntax* (HLS) dan dapat diimplementasikan menggunakan *single-decoding cores*[24]. Kompresi yang tinggi dapat diperoleh dengan

mengeksploitasi redudansi antara *view* kamera yang berbeda pada *scene* yang sama. Selanjutnya, sejak MV-HEVC dan 3D-HEVC dikembangkan bersama dengan ekstensi *scalable* HEVC (SHVC), semua ekstensi saling berbagi desain *basic inter-layer prediction* pada HLS yang sama. Ekstensi pada *high-level syntax* itu sendiri terdiri dari *signaling* dari *prediction depedencies* antara *view* yang berbeda, identifikasi dari citra untuk setiap *view*, dan elemen sintaks untuk memfasilitasi ekstraksi dari *base view*. Keuntungan dari arsitektur ini dapat diimplementasikan tanpa adanya perubahan pada sintaks atau proses dekoding dari *single-layer* HEVC pada level *slice header*, yang memungkinkan penggunaan kembali HEVC enkoder dan dekoder tanpa perubahan yang berarti untuk aplikasi stereo dan multiview.

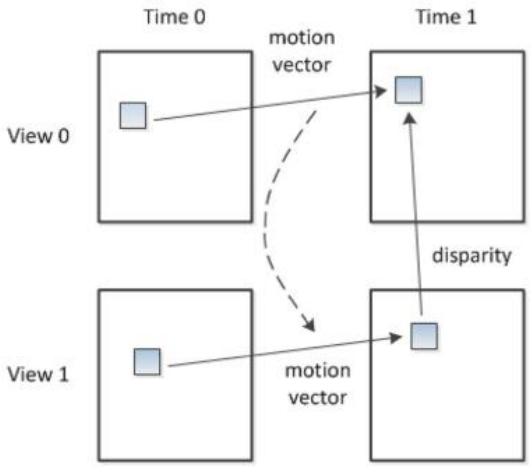
Contoh dari struktur *prediction* dapat dilihat pada Gambar 2.12. Prediksi *inter-view* diaktifkan melalui fleksibilitas kapabilitas *reference picture* dari HEVC. Citra yang dikodekan dari *view* yang lain dimasukkan kedalam *reference picture list* (RPL) pada *current view* yang digunakan untuk proses prediksi. Sebagai hasilnya, RPL yang ada pada *temporal reference pictures* dari *current view* dapat digunakan untuk memprediksi *current picture* bersama dengan *inter-view reference picture* dari *view* lainnya pada waktu bersamaan[23].



Gambar 2.12 Contoh struktur prediksi multiview untuk 3-*view*[23]

View 0 dinotasikan sebagai *base view* dan citra *non-base view* (*view* 1 atau *view* 2) dapat diprediksi dari citra pada *base view* pada waktu yang sama. Citra yang dinotasikan dengan "I" hanya digunakan pada prediksi *intra-picture*, citra yang dinotasikan dengan "P" merupakan tambahan yang digunakan untuk *uni-predictive inter-picture prediction*. Citra dengan warna yang gelap adalah *temporal random access point* dan citra dengan asosiasi "b" tidak digunakan untuk *temporal reference*.

Gambar 2.13 menunjukkan ilustrasi dari *motion prediction* antar *view*, dimana *motion vector* pada *view* 1 disimpulkan dari *motion vector* pada *view* 0 pada blok yang bersesuaian pada *time* 1.



Gambar 2.13 Ilustrasi dari *motion prediction*

2.2.4.2 Desain Coding Multilayer

MV-HEVC dan 3D-HEVC memiliki layer yang merepresentasikan tekstur, kedalaman, atau informasi tambahan dari adegan yang berhubungan dengan perspektif kamera tertentu. Semua layer termasuk kedalam perspektif kamera yang sama dan dinotasikan sebagai *view*, yang mana *layer* membawa tipe informasi yang sama dan biasanya disebut dengan komponen pada ruang lingkup video 3D.

Pada MV-HEVC dan 3D-HEVC terdapat beberapa desain coding multilayer yang digunakan pada proses dekoding dan tidak merubah dari sintaks dekoding itu sendiri yang dibutuhkan oleh HEVC, hal ini nantinya memungkinkan penggunaan kembali sintaks yang dibutuhkan pada dekode. Desain tersebut yaitu[24]:

a. MV-HEVC *inter-layer prediction*, yang memungkinkan MV-HEVC melakukan prediksi pada satu citra dengan *access unit* (AU) dan komponen yang sama tetapi berbeda *view*.
b. 3D-HEVC *inter-layer prediction*, merupakan kombinasi dari prediksi temporal dan *inter-view* pada citra dengan komponen yang sama tetapi tidak dalam AU yang sama dan *view* yang berbeda.

2.2.4.3 MV-HEVC dan 3D-HEVC High Level Syntax

High Level Syntax (HLS) adalah bagian integral dari pengkodean video. Bagian tepenting dari HLS adalah *Network Abstraction Layer* (NAL), yang menyediakan antarmuka yang umum pada pengkodean video untuk sistem / jaringan yang bermacam-macam. HEVC HLS untuk koding pada *single-layer* didesain dengan pertimbangan yang signifikan untuk mekanisme ekstenbilitas. Terdapat beberapa HLS layer untuk ekstensi HVC diantaranya :

a. *Parameter Set and Slice Segment Header Extension*
b. *Layer and Scalability Identification*
c. *Layer Sets*
d. *Profile, Tier, and Level*
e. *Reference Picture Set and Reference Picture List Construction*
f. *Random Access, Layer Switching, and Bitstream Splicing*
g. *Hybrid Codec Scalability and Multiview Support*
h. *Hypothetical Reference Decoder*
i. *Supplemental Enhancement Information (SEI) Messages*

2.2.5 Perangkat Xilinx All Programmable SoC

Selama lebih dari dua tahun, akademisi, profesional industri dan "para pengembang" di seluruh dunia telah memiliki akses ke papan pengembangan yang menggunakan Xilinx All Programmable SoC. Papan ini termasuk ZedBoard, Zc702, Zc706 dan lain-lain, telah disiapkan kemampuan sebuah sistem yang pada generasi sebelumnya belum ada, untuk membangun sistem sesuai dengan kebutuhan program system on Chip (SoC) masing-masing. Papan Zynq SoC mengintegrasikan sistem prosesor berbasis ARM® ganda Cortex®-A9 dengan Xilinx 7-series FPGA dan dengan demikian diharpakan memberikan kekuatan dan kinerja sebuah FPGA.

Xilinx menyebut perangkat baru mereka dengan nama Zynq, karena diharapkan perangkat ini dapat menjadi elemen pemrosesan yang dapat diterapkan untuk apa pun seperti salah satu unsur kimia Zynq logam yang bersifat fleksibel dan dapat membentuk ikatan dengan berbagai logam lain untuk membentuk paduan

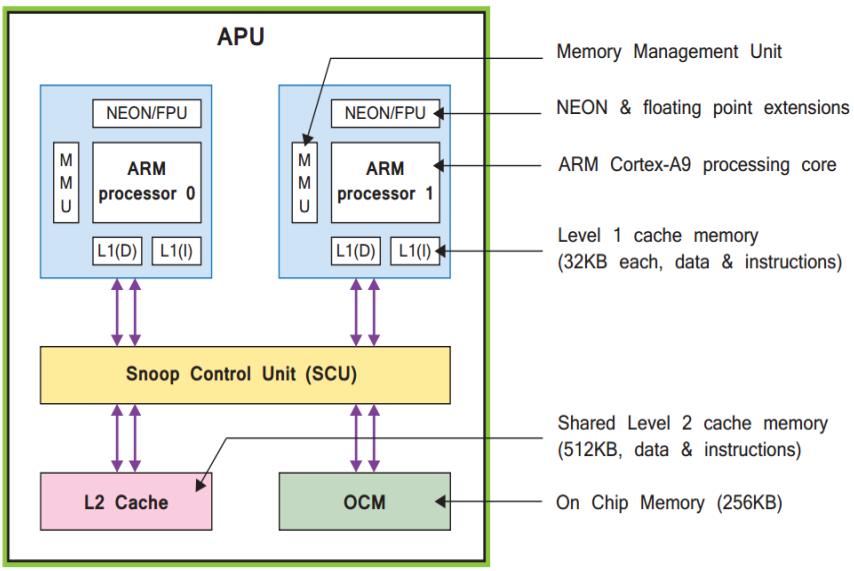
dengan sifat yang diinginkan berbeda sehingga Zynq akan menjadi sebuah platform yang menarik untuk berbagai macam aplikasi.

Fitur pada Xilinx All Programmable SoC adalah menggabungkan prosesor dual-core ARM Cortex-A9 dengan Field Programmable Gate Array (FPGA). Oleh karena itu fitur, kemampuan, dan potensi aplikasi yang sudah ditingkatkan dengan yang FPGA atau prosesor generasi sebelumnya yang dipisahkan. Dalam Zynq, ARM Cortex-A9 merupakan application prosesor, yang mampu menjalankan sistem operasi lengkap seperti Linux, disamping programmable logic berdasarkan FPGA Xilinx 7-series. Zynq System-on-Chip memiliki keunggulan dalam sistem yang sederhana yang dikemas dalam satu chip dan memiliki dimensi ukuran fisik yang semakin kecil serta harga yang semakin murah. Sistem-on-Chip (SoC) memiliki arsitektur bila ditinjau dari jumlah komponen yang terkandung didalamnya termasuk dalam desain Very Large Scale Integrated circuits (VLSI). Sistem yang kompleks yang diintegrasikan ke dalam satu chip melalui desain SoC, sehingga mampu mencapai daya yang kecil, biaya yang lebih murah, namun memiliki kecepatan lebih tinggi dari desain yang sebelumnya.

Arsitektur umum dari Zynq terdiri dari dua bagian: Processing System (PS), dan Programmable Logic (PL). Kedua bagian ini dapat digunakan secara mandiri atau bersama. Namun, model penggunaan yang paling menarik untuk Zynq adalah ketika kedua nya bagian pokok yang digunakan bersama. Informasi tentang Xilinx All Programmable SoC dapat ditemukan di Zynq ZC702 Technical Reference Manual[25].

2.2.5.1 Processing System (PS)

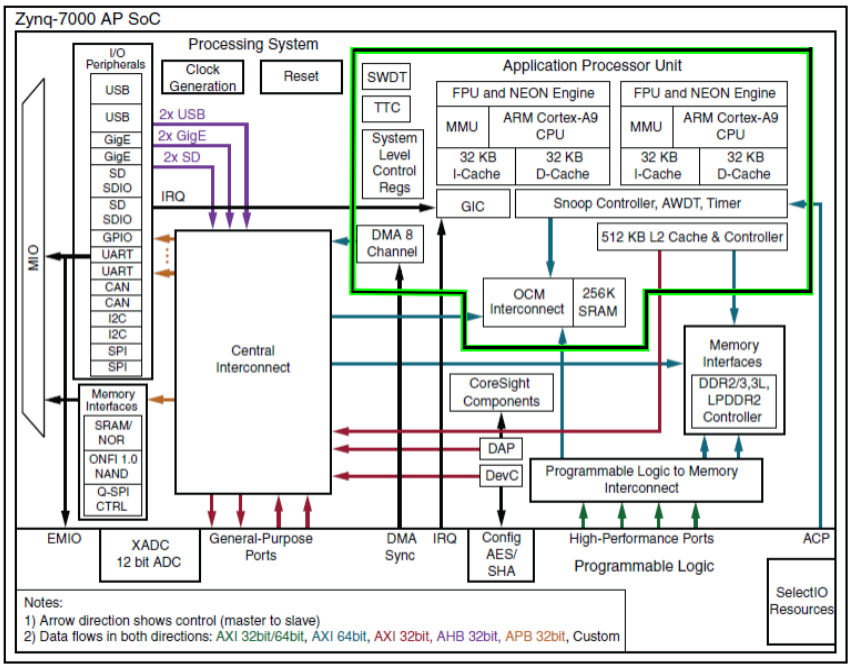
Seluruh perangkat Zynq memiliki arsitektur dasar yang sama, yaitu seluruhnya memiliki processing system (PS), yaitu dual-core prosesor ARM Cortex-A9. Namun sebagai sumber pengolahan data pada sistem Zynq bukan hanya prosesor ARM, tetapi ditambah oleh Aplication Processing Unit (APU), dan peripheral interface, memori cache, memori interface, interkoneksi, dan rangkaian pembangkit clock[25]. Sebuah diagram blok yang menunjukkan arsitektur PS ditunjukkan pada Gambar 2.13.



Gambar 2.14 Blok Diagram Aplication Processing Unit (APU)

APU terdiri dari dual core processing ARM, masing-masing dikelompokkan dalam unit komputasi : sebuah NEON ™ Media Processing Engine (MPE) dan Floating Point Unit (FPU); Memory Manajemen Unit (MMU); dan memori cache Level 1 (dalam dua bagian untuk instruksi dan data). APU juga berisi cache memori Level 2, dan selanjutnya On Chip Memory (OCM). Snoop Control Unit (SCU) membentuk jembatan antara core ARM dan memori cache Level 2 dan memori OCM; Unit ini bertanggung jawab untuk membentuk hubungan dengan PL.

Untuk kebutuhan pemrograman, desain instruksi set untuk ARM dapat dibuat pada Xilinx Software Development Kit (SDK) yang telah mencakup semua keperluan dan akan dipakai nantinya untuk mengembangkan perangkat lunak untuk dijalankan pada prosesor ARM. Compiler yang disediakan telah mendukung ARM dan Thumb instruction set (16-bit atau 32-bit), bersama dengan 8-bit Java bytecode yang telah umum digunakan pada Java Virtual Machines. Diagram tentang Processing System Unit Zynq dapat dilihat pada Gambar 2.14.



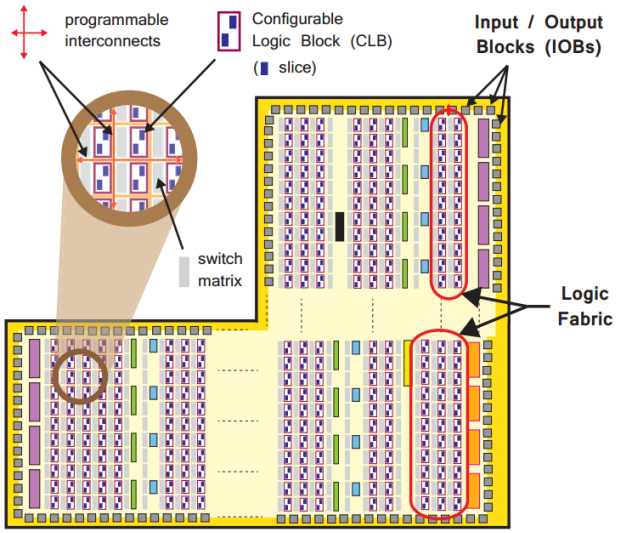
Gambar 2.15 Processing System Unit Zynq

Komunikasi antara PS dan interface eksternal terhubung melalui Multiplexed Input / Output (MIO). Koneksi juga dapat dilakukan melalui Extended MIO (EMIO), yang tidak terhubung langsung dari PS ke koneksi eksternal, melainkan melewati dan I/O resources yang berbagi pakai dengan PL.

Port I/O yang tesedia digunakan untuk interface komunikasi standar, dan General Purpose Input/Output (GPIO) yang dapat digunakan untuk bermacam-macam kegunaan diantarnya tombol sederhana, switch, dan LED. Informasi lebih lanjut tentang masing-masing antarmuka ini tersedia dalam Zynq ZC702 Technical Reference Manual.

2.2.5.2 Programmable Logic (PL)

Bagian utama yang kedua dari arsitektur Zynq adalah programmable logic. Seperti perangkat Xilinx yang lain Artix-7 dan Kintex-7 FPGA, PL terdiri dari slices Configurable Logic Blocks (CLBs), dan Input/Output Blocks (IOBs) yang digunakan untuk interfacing. Bagan dari PL perangkat Zynq ditunjukkan pada Gambar 2.15.



Gambar 2.16 Bagan dari PL perangkat Zynq

Fitur dari PL (yang ditunjukkan pada Gambar 2.14.) adalah sebagai berikut:

a. Configurable Logic Block (CLB); CLBs berukuran sangat kecil, dikelompokan secara logika dalam lokasi array dua dimensi pada PL, dan terhubung ke kelompok lain yang sejenis melalui interkoneksi yang terprogram. Setiap CLB diberi matriks saklar yang dapat diberi dua kondisi logika.
b. Slice; Sebuah sub-unit dalam CLB, yang digunakan untuk mengimplementasikan rangkaian logika secara kombinatorial dan sekuensial. Zynq slices terdiri dari 4 Tabel Lookup, 8 flip-flops, dan fungsi logika lainnya.
c. Lookup Table (LUT); Sebuah unit yang fleksibel mampu menerapkan fungsi logika diantaranya enam buah input; Read Only Memory (ROM) berukuran kecil; Random Access Memory (RAM) berukuran kecil; dan sebuah shift register. LUT dapat dikombinasikan bersama-sama untuk membentuk kelompok fungsi logika yang lebih besar, memori , ataupun shift register geser, sesuai kebutuhan.
d. Flip-flop (FF); Unit rangkaian sekuensial yang mengimplementasikan register 1-bit, dengan sebuah fungsi reset. Salah satu FFS secara opsional dapat digunakan sebagai latch.

e. Switch Matrix; Sebuah switch matriks yang berada di sebelah setiap CLB, berfungsi untuk menyediakan routing yang fleksibel dalam pembuatan koneksi antara unsur-unsur secara internal dalam sebuah CLB; dan juga dari satu CLB ke unit lainnya di PL.
f. Carry logic; Rangkaian aritmatika yang dalam operasinya membutuhkan sinyal perantara untuk disalurkan antara slices yang berdekatan. Carry logic terdiri dari rantai rute dan multiplexer untuk menjalin hubungan antara slices dalam secara vertikal dalam kolom.
g. Input/Output Blok (IOBs); IOBs adalah unit yang menghubungkan selaku interface antara unit PL, dan kelompok perangkat fisik rangkaian eksternal. Setiap IOB mampu menangani input atau output sinyal 1-bit. IOBs terletak di sekeliling perangkat Zynq.

Meskipun seorang desainer wajib memiliki pengetahuan yang mendasar dari struktur perangkat tersebut, namun dalam banyak kasus tidak perlu secara khusus menargetkan setiap unit yang akan digunakan. Perangkat Xilinx akan secara otomatis memutuskan perangkat apa saja yang akan digunakan seperti LUT, FFS, IOBs dll dari desain dan peta yang telah dibuat dalam pemrograman yang bersesuaian.

2.2.5.3 Processing System – Programmable Logic Interface

Seperti disebutkan dalam bagian sebelumnya, kelebihan dari perangkat Zynq tidak hanya terletak pada sifat unit-unit penyusunnya, seperti PS dan PL, namun juga dalam kemampuan menggunakannya secara serentak untuk membentuk sistem yang terintegrasi. Hal ini dapat dibuat dengan memberikan interkoneksi AXI dan interface sehingga membentuk jembatan antara kedua bagian tersebut. Beberapa jenis lain dari koneksi antara PS dan PL, dapat dibentuk pada EMIO tertentu.

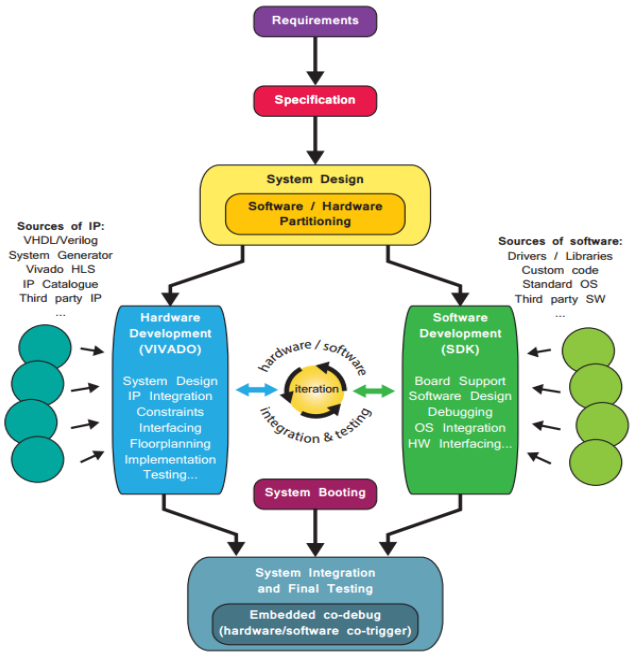
2.2.5.4 Perbandingan Zynq dengan Prosesor Standar

Berbagai macam prosesor yang tersedia di pasar dan dari fitur processor tersebut dapat dievaluasi dan dibandingkan dengan perangkat Zynq. Sesuai dengan *Embedded Microprocessor Benchmark Consortium* (EEMBC), perangkat Zynq

memiliki database skor CoreMark. Sesuai dengan data tersebut maka sangat memungkinkan untuk mengimplementasikan berbagai variasi arsitektur program pada perangkat Zynq tersebut.

2.2.5.5 Pengembangan Zynq System-on-Chip

Partisi Hardware / software, juga disebut sebagai hardware/software co-desain, merupakan tahap desain sistem embedded dan akan menghasilkan perbaikan yang signifikan dalam kinerja sistem. Proses dari partisi hardware/software secara pokok yaitu memutuskan unit mana yang akan diimplementasikan dalam perangkat keras dan yang akan diimplementasikan dalam perangkat lunak. Hal tersebut diputuskan berdasarkan dari sifat dasar FPGA programmable logic, bekerja lebih cepat karena melaksanakan pemrosesan secara paralel. Di sisi lain bila diimplementasikan pada sisi perangkat lunak (mikroprosesor), maka akan lebih lambat karena pemrosesan dilaksanakan secara sekuensial. Aliran desain untuk partisi hardware/software partisi di Zynq SoC ditunjukkan pada Gambar 2.16.



Gambar 2.17 Aliran desain untuk partisi *hardware/software* partisi di Zynq SoC

Secara konvensional dalam hal mengambil keputusan modul desain akan dilaksanakan secara hardware ataupun diwujudkan pada perangkat lunak dilakukan

secara manual oleh sistem desainer. Namun pada desain yang terbaru ini, sejumlah algoritma dan teknik telah dikembangkan yang memungkinkan otomatisasi proses pengambilan keputusan partisi untuk berbagai lingkungan desain yang berbeda. Faktor lain yang menjadi dasar untuk pertimabangan dalam mengambil keputusan apakah suatu proses harus diimplementasikan dalam perangkat keras atau perangkat lunak, adalah jumlah format yang akan digunakan.

2.2.5.6 Profiling [26]

Profiling adalah bentuk analisis program yang digunakan untuk membantu optimalisasi aplikasi perangkat lunak. Hal ini digunakan untuk mengukur jumlah properti dari kode aplikasi, antara lain:

a. Penggunaan memori
b. Waktu Pelaksanaan fungsi panggilan
c. Frekuensi fungsi panggilan
d. Penggunaan instruksi

Profiling dapat dikerjakan secara statis (tanpa mengeksekusi program software) atau secara dinamis (dilakukan saat aplikasi perangkat lunak berjalan pada prosesor fisik atau virtual). Profiling statis umumnya dilakukan dengan menganalisis kode sumber, atau kadang-kadang juga pada kode objek, sedangkan profil dinamis adalah keadaan dimana suatu proses dialihkan sementara pada pelaksanaan program karena prosesor diinterupsi untuk mengumpulkan informasi.

Penggunaan profiling memungkinkan kita untuk mengidentifikasi hambatan dalam pelaksanaan kode untuk menemukan bagian kode yang tidak efisien, atau komunikasi yang buruk antara interaksi fungsi dengan modul dalam PL ataupun pada fungsi-fungsi lain dalam perangkat lunak. Hal ini juga bisa menjadi permasalahan bahwa algoritma mungkin lebih cocok untuk diimplementasikan pada hardware. Setelah diidentifikasi, proses pelaksanaan kode dapat dioptimalkan dengan menulis ulang fungsi perangkat lunak awal atau dengan memindahkannya ke PL untuk percepatan waktu proses.

2.2.5.7 Tool Pengembangan Software

Pengembangan aplikasi perangkat lunak yang semakin berkembang pada Xilinx All Programmable SoC, yang memungkinkan pengguna untuk membuat aplikasi perangkat lunak menggunakan satu set alat Xilinx, yang telah memiliki kemampuan setara bila memanfaatkan berbagai macam alat dari vendor pihak ketiga yang memiliki kapasitas ARM prosesor Cortex-A9.

Xilinx menyediakan fasilitas desain untuk pengembangan dan debugging aplikasi perangkat lunak pada Zynq ZC702 AP SoC. Perangkat lunak yang disediakan meliputi: IDE software, compiler toolchain berbasis GNU, JTAG debugger, dan berbagai utilitas terkait lainnya.

Xilinx menyediakan dua buah tools untuk mengkonfigurasi hardware pada Zynq ZC702 AP SoC. Antara lain : Vivado IDE design Suite IP integrator dan ISE design Suite embedded development kit (EDK) Xilinx platform studio (XPS).

Perangkat lunak pengembangan lain yang disediakan oleh Xilinx adalah Software Development Kit (SDK). Xilinx SDK menyediakan tools di mana aplikasi perangkat lunak dapat dibuat, disusun, dan di-debug dan semua ini dapat dikerjakan didalam satu alat. SDK tersebut seperti compiler toolchain berbasis GNU (GCC compiler, GDB debugger, utilities, dan library), JTAG debugger, flash programmer, driver Xilinx's IP, dll. Semua fitur yang telah disebutkan dapat diakses dari dalam IDE yang berbasiskan Eclipse, dimana menggabungkan Kit Pengembangan C/C ++ (CDK).

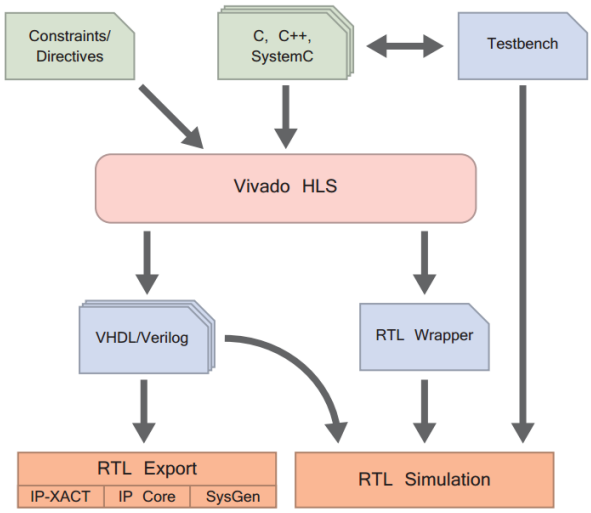
2.2.5.8 Desain Blok IP

Blok IP atau IP core adalah bagian dari hardware yang dapat digunakan untuk mengkonfigurasi perangkat logika dari sebuah FPGA atau dipergunakan untuk perangkat silikon lainnya, yang telah dibuat secara fisik dari pabrik pada sebuah IC [21]. Dalam mendesain IP cores, dapat dilakukan dengan dua jenis cara yaitu : hard IP cores dan soft IP cores.

2.2.5.9 Metode Desain IP cores

Xilinx menyediakan sejumlah tools yang mendukung dalam pembuatan blok IP yang diinginkan yang nantinya akan digunakan dalam desain sistem

embedded program yang akan dibutan. Sebagau contoh adalah HDL, Sistem Generator, HDL coder, dan Vivado High Level Synthesis. Dalam proyek ini, perangkat lunak yang digunakan adalah Vivado High Level Synthesis untuk merancang IP cores. Gambar 2.17 menunjukkan gambaran Vivado High Level Synthesis (HLS). Vivado HLS adalah alat yang disediakan oleh Xilinx, sebagai bagian dari Vivado Desain Suite, yang mampu mengkonversi desain yang berbasiskan bahasa C (C, C ++ atau SystemC) ke file desain berbasiskan RTL (VHDL / Verilog atau SystemC) untuk implementasi pada semua perangkat Xilinx.



Gambar 2.18 Vivado High Level Synthesis (HLS)

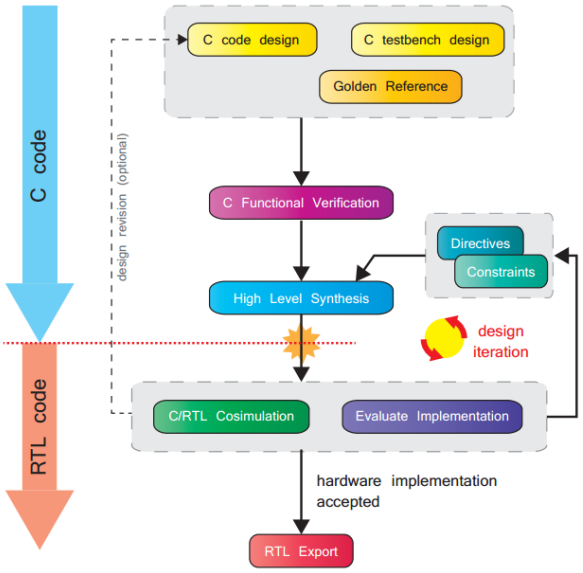
2.2.5.10 Vivado High Level Synthesis (HLS)

Pada Vivado HLS mengubah C, C ++ atau desain SystemC menjadi sistem implementasi RTL sesuai dengan kebutuhan sistem Xilinx, yang kemudian dapat disintesis dan diimplementasikan ke programmable logic dari Xilinx FPGA atau perangkat Zynq [18]. Dalam mengeksekusi HLS, dua aspek utama dari desain dianalisis:

a. Antarmuka desain, yaitu yang koneksi top level
b. Fungsi dari sebuah desain, yaitu algoritma yang mengimplementasikannya.

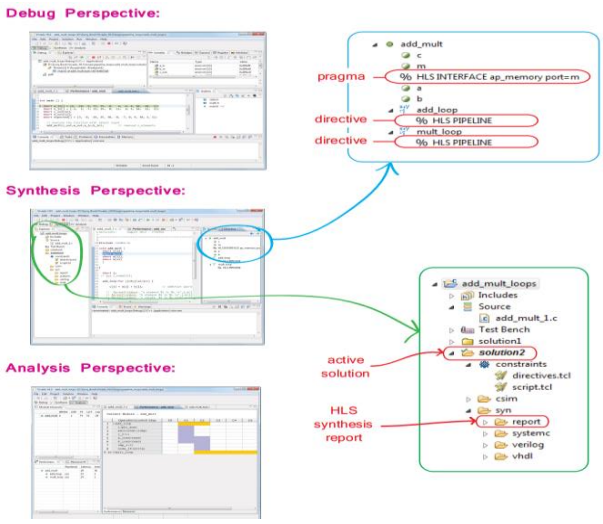
Dalam desain Vivado HLS, fungsi disintesis dari kode input melalui proses Sintesis Algoritma. Kemudian antarmuka dibuat menggunakan salah satu dari dua alternatif: secara manual, atau dapat dibuat dari menyimpulkan kode yang disebut proses sintesis antarmuka. Aliran desain pada Vivado HLS seperti tampak dilihat

Gambar 2.18. Tahapan yang digunakan dalam aliran desain mencakup input untuk proses HLS, verifikasi fungsional, sintesis top level, C/RTL co-simulation, evaluasi implementasi, iterasi desain, dan eksopr ke bentuk RTL



Gambar 2.19 Aliran desain Vivado HLS

Tools Vivado HLS menyediakan dalam bentuk Graphical User Interface (GUI) maupun dalam bentuk Command Line Interface (CLI), yang dapat digunakan secara terpisah atau bersama dengan satu sama lain. Gambar 2.18 memberikan gambaran tentang Vivado HLS GUI. GUI sebenarnya menyediakan tiga perspektif yang berbeda: Debug, Sintesis, dan Analisis seperti tampak pada Gambar 2.19.



Gambar 2.20 Vivado HLS GUI perspektif

Bila menggunakan metode desain tradisional untuk FPGA, perlu untuk menentukan tipe data dengan terperinci. Aspek ini sama pentingnya dalam Vivado HLS dibandingkan dengan metode lain seperti pengembangan HDL atau desain berbasis blok, bahkan jika tipe data pada titik masuk desain berbeda. Bahasa C, C++ dan tipe data SystemC, dan sintesisnya, merupakan syarat utama dalam mengembangkan desain yang efektif dan efisien.

Halaman ini sengaja dikosongkan

BAB 3
METODOLOGI PENELITIAN

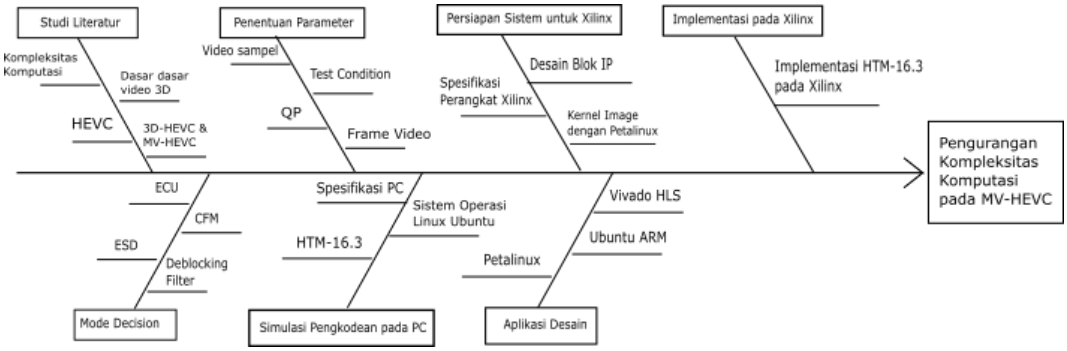
3.1 Pendahuluan

Pada bagian ini membahas beberapa sub pokok bahasan antara lain diagram fish bone penelitian, penentuan parameter *setting* untuk pengurangan kompleksitas komputasi berdasarkan *mode decision* yang telah dijelaskan sebelumnya dan nantinya diterapkan pada video MV-HEVC menggunakan *reference software* HTM-16.3, simulasi pengkodean video MV-HEVC pada *platform* PC berbasis Linux, aplikasi desain, persiapan sistem untuk pengkodean video MV-HEVC pada *platform* Xilinx All Programmable SoC, dan implementasi pada *board* Xilinx All Programmable SoC.

3.2 Diagram *Fishbone* Penelitian

Pada penelitian ini dilakukan serangkaian kegiatan mulai dari *study* literatur, penentuan parameter, simulasi pengkodean MV-HEVC pada PC berbasis Linux, persiapan sisem untuk implementasi pada Xilinx All Programmable SoC yang dimulai dengan membuat desain menggunakan aplikasi desain Vivado lalu membuat *kernel image* Ubuntu ARM menggunakan Petalinux, selanjutnya dilakukan implementasi pada Xilinx All Programmable SoC dan terakhir dilakukan analisa sehingga hasil akhir yang akan dicapai adalah mendapatkan pengurangan kompleksitas komputasi pada MV-HEVC.

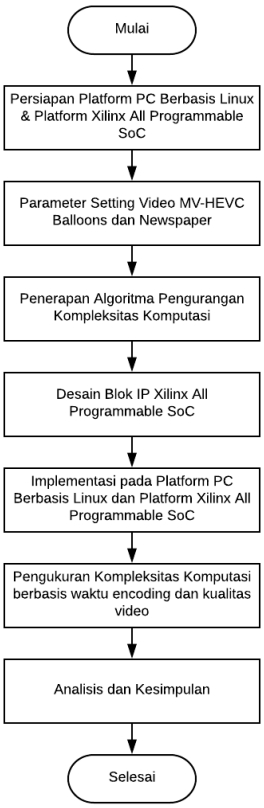
Pada tahapan studi literatur dipelajari tentang kompleksitas komputasi, dasar – dasar video 3D, HEVC, ekstensi dari HEVC yaitu 3D-HEVC dan MV-HEVC, *mode decision* untuk pengurangan kompleksitas komputasi pada MV-HEVC yang kemudian diterapkan pada *reference software* HTM-16.3. selanjutnya implementasi pengurangan kompleksitas komputasi pada *platform* PC berbasis Linux dan *platform* Xilinx All Programmable SoC. Adapun sumber – sumber yang dipergunakan adalah berasal dari buku teks serta paper – paper penelitian terkait yang sudah dilakukan sebelumnya. Parameter pengujian sistem yang dianalisis adalah waktu encoding dan kualitas video yang dilihat dari bitrate dan PSNR.



Gambar 3.1 Diagram Fishbone Penelitian

3.3 Diagram Blok Penelitian

Secara garis besar penelitian ini dapat digambarkan melalui diagram alir seperti yang ditunjukkan pada Gambar 3.2 yaitu mengenai pengurangan kompleksitas komputasi pada MV-HEVC berbasis perangkat FPGA.



Gambar 3.2 Diagram Alir Penelitian

Dari Gambar 3.2, ada 7 tahapan yang dilalui dalam penelitian ini. Penelitian ini diawali dengan persiapan *platform* yang akan digunakan pada penelitian yaitu PC berbasis Linux dan *platform* Xilinx All Programmable SoC, selanjutnya mendeskripsikan parameter *setting* video MV-HEVC Balloons dan Newspaper yang digunakan. Penerapan algoritma pengurangan kompleksitas komputasi yang digunakan adalah *reference software* HTM-16.3 dengan menggunakan *mode decision* ECU, CFM, ESD, dan *deblocking filter*. Sebelum melakukan implementasi untuk *platform* Xilinx All Programmable SoC, maka diperlukan desain blok IP yang di desain menggunakan aplikasi Xilinx Vivado Design. Setelah parameter – parameter yang diperlukan untuk penelitian ditentukan selanjutnya dilakukan implementasi pada kedua *platform.* Selanjutnya dilakukan pengukuran terhadap waktu encoding dan kualitas video dari pengkodean video yang dilakukan. Terakhir dilakukan analisis hasil pengkodean video MV-HEVC yang didapat dari implementasi *reference software* pada kedua platform setelah proses pengkodean, selanjutnya ditarik kesimpulan dari analisis tersebut.

3.4 Spesifikasi Platform PC Berbasis Linux dan Platform Xilinx All Programmable SoC

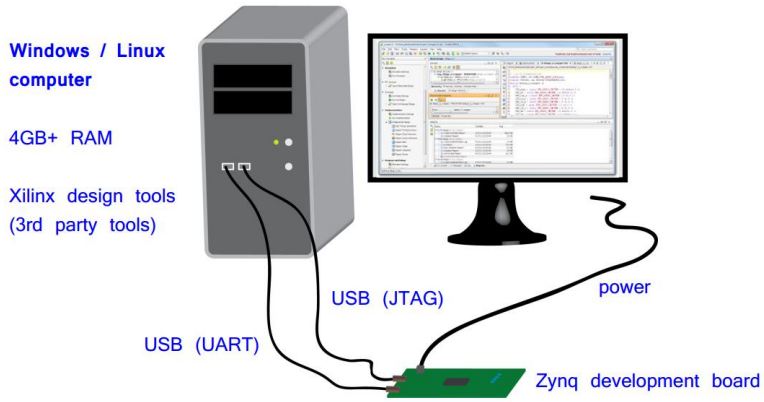
Perangkat yang digunakan pada penelitian ini terdiri dari dua *platform* yaitu *platform* PC berbasis Linux dan *platform* Xilinx All Programmable SoC. *Platform* PC berbasis Linux memiliki spesifikasi sebagai berikut yang dijelaskan pada Tabel 3.1.

Tabel 3.1 Spesifikasi *platform* PC berbasis Linux

Prosesor	Intel® Core™ i7-4770 CPU@3,4GHz
Memori (RAM)	8GB
Sistem Operasi	Linux Ubuntu 16.04 LTS 64-bit

Untuk spesifikasi minimum yang dibutuhkan dalam penelitian ini menggunakan memori sebesar 4GB dan maksimum setidaknya memiliki 12GB. Sistem operasi yang digunakan dengan arsitektur 64-bit dikarenakan arsitektur 32-bit tidak dapat berjalan dengan baik.

Selanjutnya perangkat FPGA Xilinx All Programmable SoC yang digunakan dalam penelitian ini adalah Xilinx Zynq ZC702 Rev1.0 *evaluation board*. *Board* ini telah dilengkapi dengan fitur – fitur pendukung seperti daya listrik, memori eksternal 8GB *high speed*, antarmuka untuk pemrograman dan komunikasi, dan juga dilengkapi tombol input dan output, LED, dan sejumlah perangkat antarmuka lain beserta konektornya. Perancangan sistem yang akan dibuat sampai tahap *debugging*, desain dirancang menggunakan perangkat lunak pendukung Vivado dan Vivado High Level Synthesis (HLS) kemudian program yang telah selesai dibuat akan diimplementasikan ke *board* Zynq menggunakan JTAG atau koneksi ethernet, setelah berhasil diimplementasikan maka perancangan sistem tersebut diuji menggunakan periferal dan antarmuka eksternal. Kinerja processor papan Zynq dan proses real timenya dapat dimonitor pada saat debugging melalui koneksi yang tersedia dari PC ke papan Zynq, dengan menjalankan program simulasi. Untuk pengaturan dasar koneksi dari PC ke *board* Zynq dapat dilihat pada Gambar 3.3.



Gambar 3.3 Pengaturan Koneksi PC ke *board* Zynq ZC702

Lebih lanjut spesifikasi dari *board* Zynq ZC702 dijelaskan pada Tabel 3.2 berikut.

Tabel 3.2 Spesifikasi Perangkat Xilinx Zynq ZC702

Spesifikasi	Jumlah
Lookup Table (LUT)	53200
Flip – Flop	106400
BRAM	140

I/O	484
DSP	220

3.5 Parameter *Setting* Video MV-HEVC

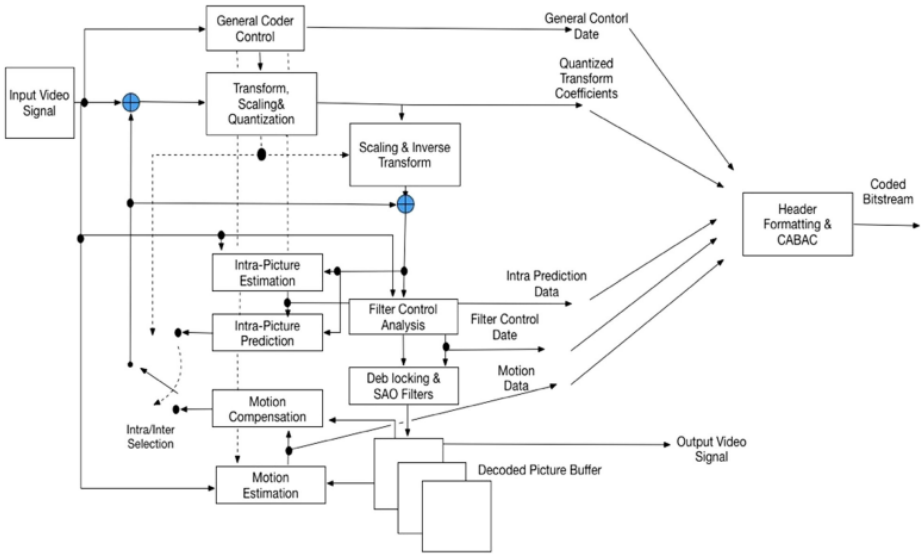
Seperti yang telah dijelaskan sebelumnya, pada penelitian ini digunakan 2 buah video sampel MV-HEVC yaitu video Balloons dan video Newspaper dalam bentuk format stereoskopis *left view* dan *right view* dapat dilihat pada Gambar 3.4 dan Gambar 3.5. Kedua video memiliki jumlah pixel yang sama yaitu 1024 x 768 dengan *frame per second* (fps) 30fps, durasi video adalah 10 detik, dari frame yang berjumlah 300, pada penelitian ini hanya akan dikodekan sebanyak 150 frame, dan jumlah view yang dikodekan adalah 3 view.

Gambar 3.4 Video MV-HEVC Balloons

Gambar 3.5 Video MV-HEVC Newspaper

3.6 Penerapan Algoritma Pengurangan Kompleksitas Komputasi

Sebelum masuk ke penerapan algoritma untuk pengurangan kompleksitas komputasi pada video MV-HEVC, terlebih dahulu dijelaskan *mode decision* yang digunakan dalam pengurangan kompleksitas komputasi untuk penelitian ini. Pada MV-HEVC terdapat encoder *hybrid* yang berfungsi untuk mengolah video menjadi video terkompresi dalam bentuk bit stream yang nantinya akan ditransmisikan. Gambar 3.6 menunjukkan encoder dari video MV-HEVC. Proses pengolahan video dimulai dengan memisahkan satu frame menjadi beberapa bagian. Bagian – bagian tersebut dinamakan dengan *Coding Tree Unit* (CTU) yang dapat dibagi lagi menjadi beberapa bagian kecil dikenal dengan *Coding Units* (CU), *Transform Units* (TU), dan *Prediction Units* (PU).



Gambar 3.6 Hybrid Video Encoder yang digunakan pada MV-HEVC [27]

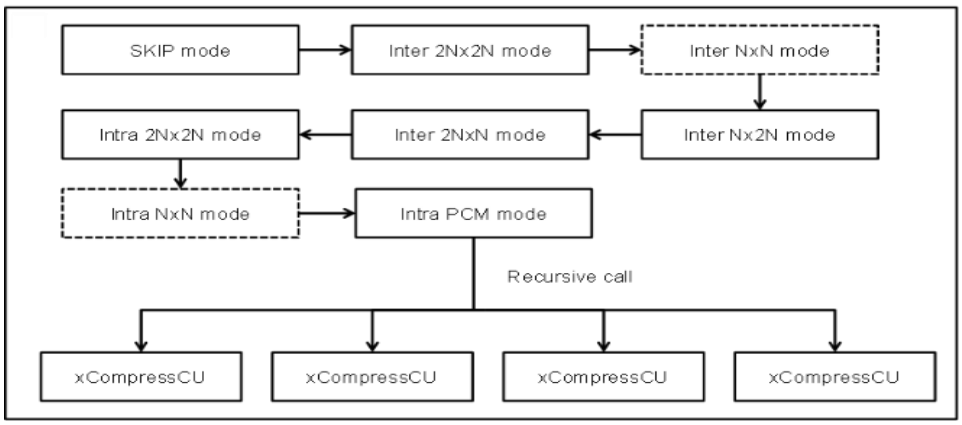
Pada encoder HEVC menggunakan dua tipe prediksi yaitu *intra prediction* dan *inter prediction*. *Intra prediction* melakukan prediksi pada PU dari *neighboring image* pada frame citra yang sama. *Inter prediction* menggunakan *motion compensated* untuk memprediksi PU dari citra sebelum dan sesudahnya. Setiap data yang tersisa setelah diprediksi dimasukkan kembali kedalam blok dengan proses *Discrete Cosine Transform* (DCT). Dari sini lalu dilanjutkan dengan proses Entropy Coding dimana video sudah dikompresi, disimpan, dan ditransimisikan.

Bagian – bagian dari CTU ini pada prosesnya di encoder MV-HEVC memberikan kontribusi dalam kompleksitas komputasi yang besar. Maka dari itu diperlukan metode yang dapat mengurangi kompleksitas komputasi ini. *Mode decision* adalah metode yang mampu mengurangi kompleksitas komputasi, adapun mode decision yang digunakan pada penelitian ini adalah ECU, CFM, ESD, dan *deblocking* filter. Penjelasan dari *mode decision* tersebut adalah sebagai berikut.

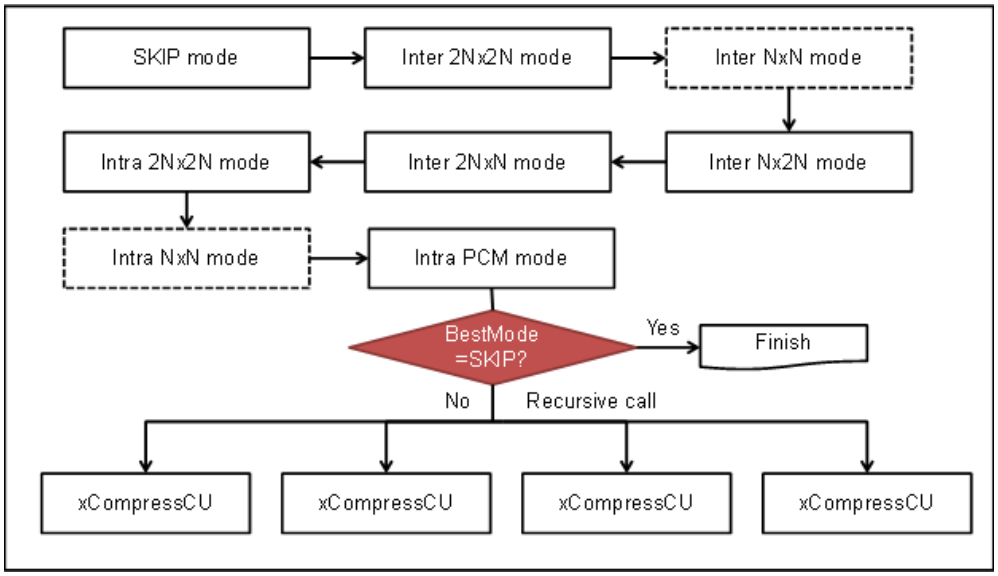
3.6.1 *Early Termination* CU (ECU)

Coding Unit (CU) pada HEVC merupakan bagian dasar pada encoder untuk pembagian wilayah pada inter / intra prediction. Konsep dari CU ini sendiri adalah memungkinkan pembagian secara rekursif sehingga menjadi empat blok yang berukuran sama yang dimulai dari *treeblock*. Dalam menentukan ukuran CU terbaik, encoder dari HEVC menguji setiap kemungkinan ukuran dari CU untuk memperkirakan kinerja pengkodean pada setiap CU yang didefinisikan dari ukuran CU pada aplikasi perangkat lunak *reference software*, ini dapat dilihat pada Gambar 3.6.

Hal ini mengakibatkan adanya kompleksitas komputasi pada proses encoding, maka dari itu pada [28] mengusulkan dilakukan metode *early termination* pada aplikasi perangkat lunak *reference software* HEVC dalam memproses CU agar mengurangi kompleksitas komputasi terutama pada waktu encoding video. Gambar 3.7 menunjukkan proses *early termination* CU pada encoder HEVC.



Gambar 3.7 Pemrosesan CU pada encoder HEVC [28]



Gambar 3.8 *Early termination* CU pada encoder HEVC [28]

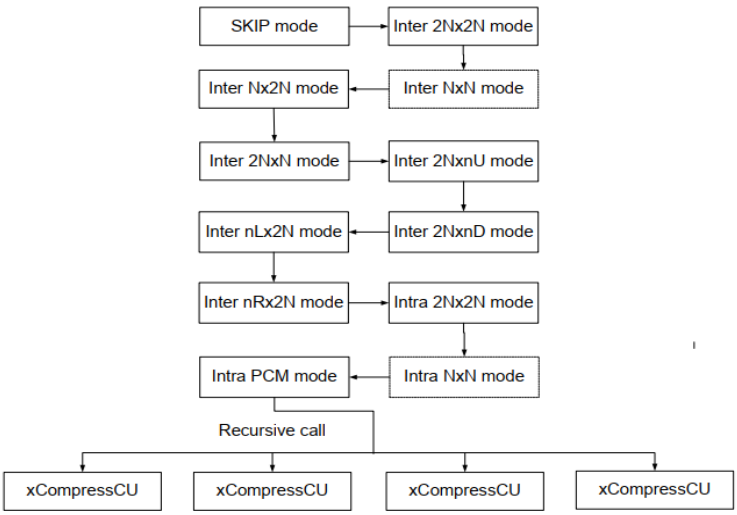
Metode ini menentukan CU *depth* lebih awal dan memastikan tidak adanya pemrosesan *sub-trees* dari CU ketika berada pada SKIP mode, sehingga SKIP mode ini dipilih sebagai *best prediction* pada CU.

3.6.2 *Cbf Fast Mode Decision* (CFM)

Kompleksitas komputasi juga mengalami peningkatan dikarenakan ada penambahan efisiensi coding *tools*. Salah satunya adalah pada PU yang dikodekan dengan 2N×2N, 2N×N, N×2N, dan N×N terlepas dari kinerja PU yang telah dikodekan sebelumnya. Dengan menggunakan informasi dari *coded_block_flag* (cbf) pada sintaks HEVC, PU dengan kinerja terbaik mampu diprediksi [29]. Cara kerja metode ini adalah jika cbf adalah nol (0) setelah pengkodean PU untuk luma dan dua chroma (*cbf_luma, cbf_u, cbf_v*), proses pengkodean PU selanjutnya dari CU diakhiri. Urutan pengkodean PU dalam CU adalah Inter 2Nx2N, Inter 2NxN, Inter Nx2N, Inter NxN, Intra 2Nx2N, Intra NxN. Namun, *early termination* pada PU Inter NxN tidak diperhitungkan. Metode ini disebut juga dengan *Cbf Fast Mode Decision* (CFM).

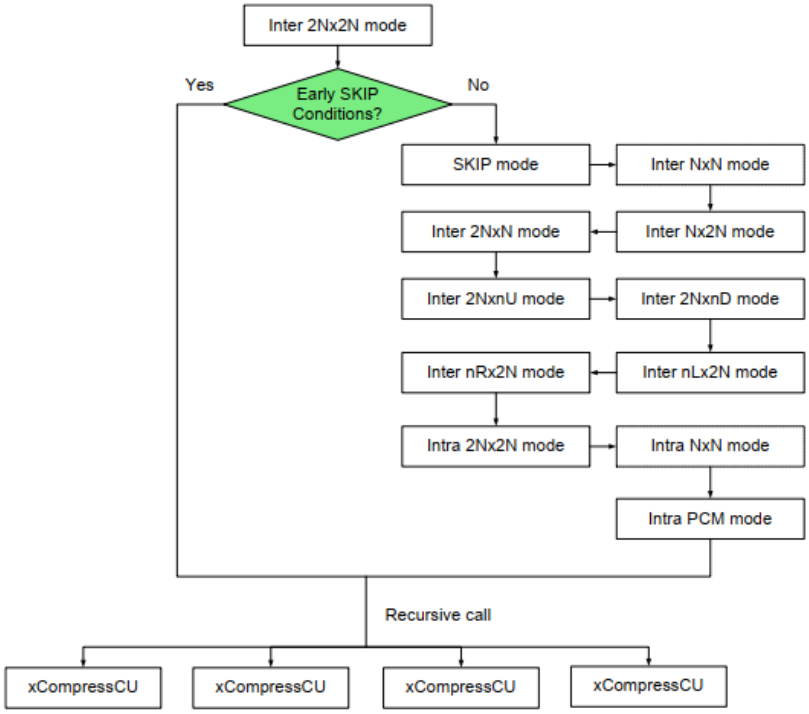
3.6.3 *Early SKIP Detection* (ESD)

Partition Unit (PU) juga merupakan salah satu bagian yang memiliki kompleksitas komputasi pada encoder HEVC. Dalam memilih mode PU yang terbaik, aplikasi perangkat lunak *reference software* untuk HEVC mengkalkulasi *Rate Distortion* (RD) *cost* dari semua mode inter PU yang memungkinkan, diperlihatkan pada Gambar 3.8.



Gambar 3.9 Mode decision PU proses pada encoder HEVC [29]

Hal ini yang mengakibatkan tingginya kompleksitas komputasi. Untuk mengurangi kompleksitas komputasi, pada [29] diperkenalkan metode *Early Skip Detection* pada encoder HEVC. Dengan metode ini, encoder HEVC melakukan pengecekan pada *motion vector* dan *Coded Block Flag* (CBF) untuk *best* inter mode 2N×2N, dapat dilihat pada Gambar 3.9.



Gambar 3.10 Mode decision PU dengan Early SKIP [29]

Mode *decision* dengan *early* SKIP yang ditunjukkan pada Gambar 3.8 melakukan pengecekan mode inter 2N×2N sebelum masuk ke mode SKIP. Setelah didapatkan mode inter 2N×2N yang terbaik dengan RD *cost* yang rendah, selanjutnya menelusuri *motion vector* dan Cbf. Jika motion vector dan Cbf dari mode inter 2N×2N sama dengan (0,0) atau nol, CU yang ada didefinisikan sebagai SKIP mode dan ridak lagi dilakukan pengecekan pada PU. Hal ini mengurangi kompleksitas komputasi tanpa mengurangi efisiensi coding.

3.6.4 *Deblocking Filter*

HEVC dengan skema hybrid coding pada encoder dengan menggunakan *block-based prediction* dan transform coding. Skema ini memungkinkan terjadinya diskontinuitas pada sinyal yang direkonstruksi pada *block-boundaries*. Sumber utama penyebab terjadinya diskontinuitas ini adalah adanya kesalahan prediksi pada *block-transform coding* yang diikuti oleh proses kuantisasi kasar. Hal ini berpengaruh kepada kualitas dari video setelah dikodekan. Tujuan dari *deblocking*

filter ini untuk meningkatkan kualitas subjektif pada saat mengurangi kompleksitas komputasi dan aspek lainnya adalah *deblocking filter* di HEVC telah dirancang dengan cara untuk mencegah ketergantungan spasial di seluruh gambar dan memungkinkan paralelisasi yang mudah pada *multi core*, lebih jelasnya dapat dilihat pada [30].

Selanjutnya masuk ke pembahasan penerapan algoritma pengurangan kompleksitas komputasi pada video MV-HEVC. Untuk menerapkan *mode decision* yang telah dijelaskan sebelumnya digunakan maka digunakan sebuah aplikasi yang dinamakan *reference software* atau *test model* pada video MV-HEVC. *Reference software* yang digunakan pada penelitian ini adalah HTM-16.3. Terlebih dahulu dilakukan penentuan konfigurasi pada *reference software* HTM-16.3 sebelum dilakukan implementasi pada *platform* penelitian, adapun konfigurasinya sebagai berikut :

- Video yang digunakan adalah video Balloons dan Newspaper seperti yang telah dijelaskan sebelumnya.
- Kondisi untuk pengujian ada 3 yaitu kondisi baseline, kondisi 1, dan kondisi 2. Kondisi baseline merupakan kondisi awal dan tidak adanya *mode decision* yang digunakan, kondisi 1 adalah kondisi dengan penerapan *mode decision* ECU dan CFM dengan tambahan , dan kondisi 2 adalah kondisi dengan penerapan *mode decision* ECU, CFM, ESD, dan *deblocking filter*.
- Pada penelitian ini dibatasi jumlah view sebanyak 3. Sehingga file konfigurasi yang digunakan adalah baseCfg_3view.
- File konfigurasi seqCfg_Balloons dan seqCfg_Newspaper digunakan untuk mengkonfigurasikan video dan *mode decision*.
- Parameter kuantisasi (QP) yang digunakan yaitu : 25, 30, 35, 40, dan 45.
- Menggunakan *fast search* dengan *search range* 64.

Dari penentuan konfigurasi diatas, selanjutnya dipindahkan ke *reference software* HTM-16.3 dengan sintaks yang dapat dilihat pada Gambar

```
#======== File I/O =====================
InputFile_0                   : ../testseq/video1.yuv
InputFile_1                   : ../testseq/video1.yuv
InputFile_2                   : ../testseq/video1.yuv
BitstreamFile                 :
../testseq/balloons_00_1024x768_common_bin_QP29_base.bin

ReconFile_0                   : ../testseq/rec_3.hevc
ReconFile_1                   : ../testseq/rec_1.hevc
ReconFile_2                   : ../testseq/rec_5.hevc
```

Gambar 3.11 Sintaks Konfigurasi Input Video MV-HEVC

```
#=========== Motion Search =============
FastSearch           : 1   # 0:Full search  1:TZ search
SearchRange          : 64  # (0: Search range is a Full frame)
BipredSearchRange    : 4   # Search range for bi-prediction refinement
HadamardME           : 1   # Use of hadamard measure for fractional ME
FEN                  : 0   # Fast encoder decision
FDM                  : 0   # Fast Decision for Merge RD cost
DispSearchRangeRestriction    : 1  # Limit Search range for vertical
component of disparity vector
VerticalDispSearchRange       : 56 # Vertical Search range in pixel

#=========== Sequences Config ===========
SourceWidth               : 1024   # input frame width
SourceHeight              : 768    # input frame height
FrameRate                 : 30     # frame rate in frames per second
FramesToBeEncoded         : 150    # number of frames to be coded
BaseViewCameraNumbers     : 3 1 5  # camera numbers of coded views (in
coding order)
ECU                   : 0
CFM                   : 0
ESD                   : 0
```

Gambar 3.12 Sintaks Konfigurasi untuk *test condition* Baseline

```
#=========== Motion Search =============

FastSearch           : 1   # 0:Full search  1:TZ search
SearchRange          : 64  # (0: Search range is a Full frame)
BipredSearchRange    : 4   # Search range for bi-prediction refinement
HadamardME           : 1   # Use of hadamard measure for fractional ME
FEN                  : 1   # Fast encoder decision
FDM                  : 1   # Fast Decision for Merge RD cost
DispSearchRangeRestriction    : 1  # Limit Search range for vertical
component of disparity vector
VerticalDispSearchRange       : 56 # Vertical Search range in pixel

#=========== Sequences Config ===========
SourceWidth               : 1024   # input frame width
SourceHeight              : 768    # input frame height
FrameRate                 : 30     # frame rate in frames per second
FramesToBeEncoded         : 150    # number of frames to be coded
BaseViewCameraNumbers     : 3 1 5  # camera numbers of coded views (in
coding order)
ECU                    : 1
CFM                    : 1
ESD                    : 0
```

Gambar 3.13 Sintaks Konfigurasi untuk *test condition* 1

```
#=========== Motion Search =============

FastSearch           : 1   # 0:Full search  1:TZ search
SearchRange          : 64  # (0: Search range is a Full frame)
BipredSearchRange    : 4   # Search range for bi-prediction refinement
HadamardME           : 1   # Use of hadamard measure for fractional ME
FEN                  : 1   # Fast encoder decision
FDM                  : 1   # Fast Decision for Merge RD cost
DispSearchRangeRestriction    : 1  # Limit Search range for vertical component
of disparity vector
VerticalDispSearchRange       : 56 # Vertical Search range in pixel

#=========== Sequences Config ===========
SourceWidth               : 1024   # input frame width
SourceHeight              : 768    # input frame height
FrameRate                 : 30     # frame rate in frames per second
FramesToBeEncoded           : 150    # number of frames to be coded
BaseViewCameraNumbers     : 3 1 5  # camera numbers of coded views (in coding
order)
ECU                    : 1
CFM                    : 1
ESD                    : 0

#=========== Deblock Filter ============
LoopFilterOffsetInPPS         : 1  # Dbl params: 0=varying params in
SliceHeader, param = base_param + GOP_offset_param; 1 (default) =constant
params in PPS, param = base_param)
LoopFilterDisable             : 0  # Disable deblocking filter (0=Filter, 1=No
Filter) (mc)
LoopFilterBetaOffset_div2     : 0  # base_param: -6 ~ 6 (default:0)
LoopFilterTcOffset_div2       : 0  # base_param: -6 ~ 6 (default:0)
DeblockingFilterMetric        : 0  # blockiness metric (automatically
configures deblocking parameters in bitstream). Applies slice-level loop filter
offsets  (LoopFilterOffsetInPPS and LoopFilterDisable must be 0)
```

Gambar 3.14 Sintaks Konfigurasi untuk *test condition* 2

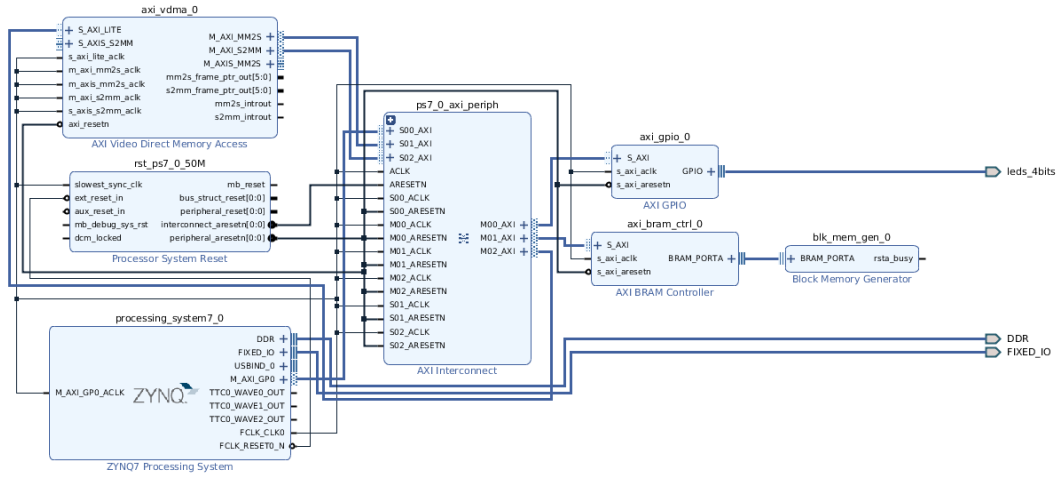
3.7 Desain Blok IP

Desain blok IP diperlukan untuk implementasi pada *platform* Xilinx All Programmable SoC. Untuk memulai desain pada *board* Xilinx All Programmable SoC dibutuhkan aplikasi yang sesuai yang dapat diunduh pada situs Xilinx. Aplikasi desain yang dibutuhkan pada penelitian ini adalah Xilinx Vivado Design Suite dan License Management Tools dengan versi aplikasi yang digunakan pada penelitian ini adalah versi 2017.4. Pembuatan proyek *embedded proccessor* pada *board* ZC702 menggunakan fasilitas add sources wizard. Pilihan yang dapat digunakan unuk membuat blok desain wizard dapat dipilih seperti pada pada Tabel 3.3. Penambahan blok IP (Intellectual Property) yang akan digunakan, dapat ditambahkan dari katalog yang disediakan. Untuk membantu mempermudah pemilihan IP yang diinginkan dapat digunakan fasilitas *search*, dapat diketik "zynq" untuk menemukan pilihan seluruh blok yang merupakan milik produk Zynq. Selanjutnya untuk memilih perangkat IP yang diinginkan klik dua kali ZYNQ7 Processing System IP untuk menambahkan secara otomatis ke Blok Desain. Maka visualisasi sistem pengolahan Zynq SoC blok IP pada blok desain akan muncul, Tabel 3.3 memperlihatkan pengaturan untuk membuat desain blok pada Vivado.

Tabel 3.3 Parameter untuk membuat desain blok IP pada Vivado

Wizard Screen	**System Property**	**Setting or Command to Use**
Create Block Design	Design name	design_1_bd
	Directory	<Local to Project>
	Specify source set	**Design Sources**

Selanjutnya pada Gambar 3.15 diperlihatkan desain blok IP pada vivado yang telah dibuat.



Gambar 3.15 Desain Blok IP pada Vivado

Desain blok IP yang didesain menggunakan vivado sepeti pada Gambar 3.15 terdiri dari :

a. Prosesor ARM Cortex-A9
b. AXI Interconnect 32KB
c. AXI BRAM Controller
d. AXI GPIO
e. Block Memory Generator 36KB
f. Processor System Reset
g. AXI Video Direct Memory Access 64KB

Dari desain blok IP tersebut menghasilkan spesifikasi *hardware* yg digunakan untuk penelitian ini, hal ini berbeda dengan spesifikasi asli perangkat Xilinx All Programmable SoC. Spesifikasi dari desain tersebut dapat dilihat pada Tabel 3.4.

Tabel 3.4 Spesifikasi *Hardware* dari Desain Blok IP

Spesifikasi	Jumlah
Lookup Table (LUT)	5951
Flip – Flop	7063
BRAM	5
I/O	4
BUFG	1

3.8 Implementasi pada *Platform*

3.8.1 PC Berbasis Linux

Untuk menjalankan *reference software* MV-HEVC pada PC berbasis Linux telah disediakan file *makefile* dalam perangkat lunak yang telah dikemas dalam satu sistem, sehingga dapat langsung dijalankan pada PC yang menggunakan sistem operasi Linux. Dengan menjalankan file *makefile* tersebut, maka seluruh *library* yang dimiliki MV-HEVC akan langsung dijalin secara otomatis dan dapat dijalankan pada PC tersebut.

```
### enforce 32-bit build : 1=yes, 0=no
M32?= 0
HEVC_EXT?= ''
export M32
ifneq ($(HEVC_EXT), '')
ADDDEFS = -DHEVC_EXT=$(HEVC_EXT)
endif
EXTENSION_360_VIDEO?=0
export EXTENSION_360_VIDEO
all:
	$(MAKE) -C lib/TLibVideoIO	MM32=$(M32) ADDDEFS=$(ADDDEFS)
	$(MAKE) -C lib/TLibCommon  MM32=$(M32) ADDDEFS=$(ADDDEFS)
	$(MAKE) -C lib/TLibDecoder	MM32=$(M32) ADDDEFS=$(ADDDEFS)
	$(MAKE) -C lib/TLibEncoder	MM32=$(M32) ADDDEFS=$(ADDDEFS)
	$(MAKE) -C lib/TLibRenderer	MM32=$(M32) ADDDEFS=$(ADDDEFS)
	$(MAKE) -C lib/TLibExtractor	MM32=$(M32) ADDDEFS=$(ADDDEFS)
$(MAKE) -C lib/TAppCommon	MM32=$(M32) ADDDEFS=$(ADDDEFS)
	@if [ $(EXTENSION_360_VIDEO) -eq 1 ] ; then \
	  $(MAKE) -C lib/TLib360		MM32=$(M32) ; \
	  $(MAKE) -C lib/TAppEncHelper360	MM32=$(M32) ; \
	  $(MAKE) -C app/TApp360Convert	MM32=$(M32) ; \
	fi
	$(MAKE) -C app/TAppDecoder	MM32=$(M32) ADDDEFS=$(ADDDEFS)
	$(MAKE) -C app/TAppEncoder	MM32=$(M32) ADDDEFS=$(ADDDEFS)
	$(MAKE) -C app/TAppRenderer	MM32=$(M32) ADDDEFS=$(ADDDEFS)
	$(MAKE) -C app/TAppExtractor	MM32=$(M32) ADDDEFS=$(ADDDEFS)
	$(MAKE) -C utils/annexBbytecount MM32=$(M32) ADDDEFS=$(ADDDEFS)
	$(MAKE) -C utils/convert_NtoMbit_YCbCr MM32=$(M32)

ADDDEFS=$(ADDDEFS)
	$(MAKE) -C lib/TlibDecoderAnalyser MM32=$(M32)

ADDDEFS=$(ADDDEFS)
	$(MAKE) -C app/TAppDecoderAnalyser MM32=$(M32)

ADDDEFS=$(ADDDEFS)

debug:
	$(MAKE) -C lib/TLibVideoIO debug MM32=$(M32) ADDDEFS=$(ADDDEFS)
	$(MAKE) -C lib/TLibCommon  debug MM32=$(M32) ADDDEFS=$(ADDDEFS)
	$(MAKE) -C lib/TLibDecoder debug MM32=$(M32) ADDDEFS=$(ADDDEFS)
	$(MAKE) -C lib/TLibEncoder debug MM32=$(M32) ADDDEFS=$(ADDDEFS)

$(MAKE) -C lib/TLibExtractor	debug MM32=$(M32) ADDDEFS=$(ADDDEFS)
	$(MAKE) -C lib/TAppCommon	debug MM32=$(M32)
```

```
ADDDEFS=$(ADDDEFS)
       @if [ $(EXTENSION_360_VIDEO) -eq 1 ] ; then \
         $(MAKE) -C lib/TLib360                debug MM32=$(M32) ; \
         $(MAKE) -C lib/TAppEncHelper360       debug MM32=$(M32) ; \
         $(MAKE) -C app/TApp360Convert         debug MM32=$(M32) ; \
fi
       $(MAKE) -C app/TAppDecoder debug MM32=$(M32) ADDDEFS=$(ADDDEFS)
       $(MAKE) -C app/TAppEncoder debug MM32=$(M32) ADDDEFS=$(ADDDEFS)
       $(MAKE) -C app/TappRenderer debug MM32=$(M32)
ADDDEFS=$(ADDDEFS)
       $(MAKE) -C app/TappExtractor debug MM32=$(M32)
ADDDEFS=$(ADDDEFS)

clean_highbitdepth:
HIGHBITDEPTH=1
       $(MAKE) -C lib/TLibEncoder clean MM32=$(M32) HIGHBITDEPTH=1
       $(MAKE) -C lib/TLibRenderer    clean MM32=$(M32) HIGHBITDEPTH=1
       $(MAKE) -C lib/TLibExtractor   clean MM32=$(M32) HIGHBITDEPTH=1
       $(MAKE) -C lib/TAppCommon      clean MM32=$(M32) HIGHBITDEPTH=1
       @if [ $(EXTENSION_360_VIDEO) -eq 1 ] ; then \
         $(MAKE) -C lib/TLib360  clean MM32=$(M32) HIGHBITDEPTH=1 ; \
         $(MAKE) -C lib/TAppEncHelper360 clean MM32=$(M32)
HIGHBITDEPTH=1 ; \
         $(MAKE) -C app/TApp360Convert clean MM32=$(M32)
HIGHBITDEPTH=1 ; \
       fi
       $(MAKE) -C app/TAppDecoder   clean MM32=$(M32) HIGHBITDEPTH=1
       $(MAKE) -C app/TAppEncoder   clean MM32=$(M32) HIGHBITDEPTH=1
       $(MAKE) -C app/TAppRenderer  clean MM32=$(M32) HIGHBITDEPTH=1
       $(MAKE) -C app/TAppExtractor clean MM32=$(M32)HIGHBITDEPTH=1
       $(MAKE) -C lib/TlibDecoderAnalyser  clean MM32=$(M32)
HIGHBITDEPTH=1
       $(MAKE) -C app/TAppDecoderAnalyser  clean MM32=$(M32)
HIGHBITDEPTH=1

everything: all all_highbitdepth
```

Gambar 3.16 Sintaks program makefile MV-HEVC dalam bahasa pemrograman C++

Selanjutnya meng-*compile* aplikasi tersebut dengan langkah – langkah seperti yang terlihat pada Gambar 3.17.

```
// Implementasi HTM-16.3 pada platform PC Berbasis Linux
Start;
        1. Buka folder trunk >> folder build >> folder linux;
        2. if(Eksekusi makefile) {
              Lanjutkan melakukan konfigurasi file .cfg;}
              else  Ulangi eksekusi makefile;
        3.  Konfigurasi  file  config baseCfg_3view.cfg,  qpCfg_QP25.cfg,
           seqCfg_Balloons.cfg, dan seqCfg_Newspaper.cfg;
        4. Jalankan aplikasi reference software HTM-16.3;
Stop;
```

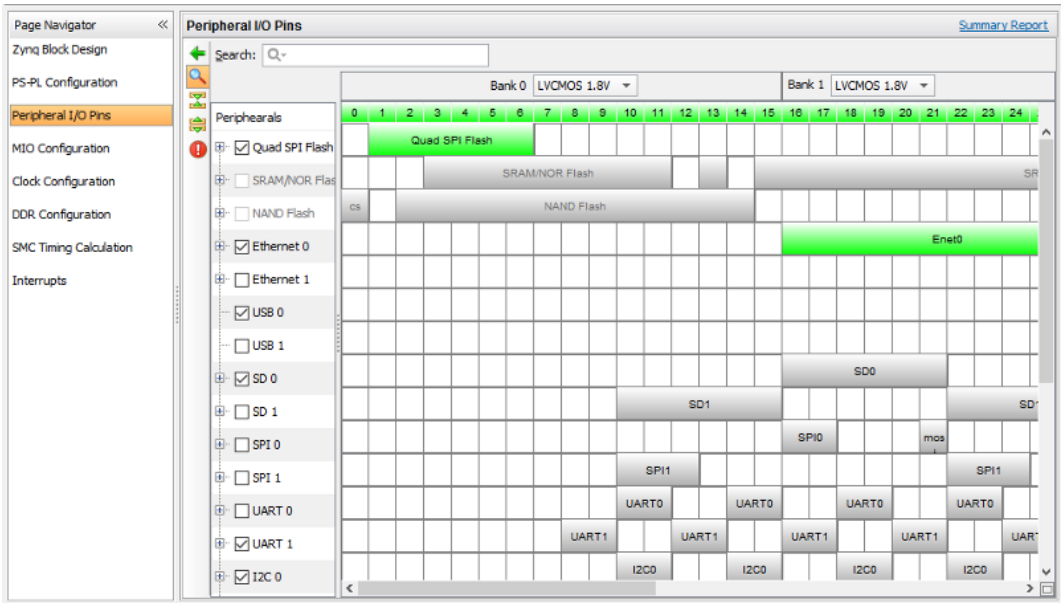
Gambar 3.17 Langkah - Langkah Menjalankan Aplikasi *Reference Software* HTM-16.3 di PC Berbasis Linux

3.8.2 Xilinx All Programmable SoC

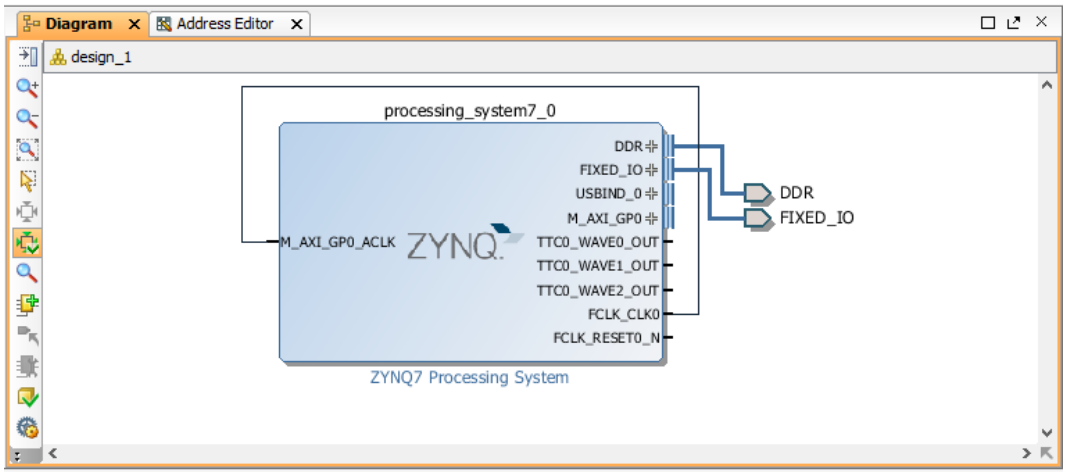
Selanjutnya, adapun tahapan – tahapan untuk implementasi pada *platform* Xilinx All Programmable SoC dilakukan sebagai berikut :

Langkah 1 : Pengaturan sistem *board* ZC702 di Vivado

Setelah blok IP selesai dibuat pada sub bab sebelumnya selanjutnya seluruh parameter blok yang akan digunakan pada Zynq7 *processing system* diatur dengan melakukan klik dua kalo pada blok IP Zynq7, kotak dialog akan muncul sebagai sarana untuk mengatur parameter blok IP Zynq7. Secara default, sistem prosesor pada awalnya tidak memiliki periferal yang terhubung (masih blok tunggal secara mandiri dari tiap-tiap blok). Koneksi yang akan terbentuk dilambangkan dengan tanda centang. Template ZC702 telah disediakan untuk dapat langsung dipakai sehingga dapat langsung diaplikasikan ke papan ZC702 dengan format data yang sudah disesuakan. Konfigurasi wizard untuk menghubungkan antar blok IP disediakan sehingga memungkinkan untuk dapat menghubungkan banyak peripheral dari Processing Sistem dengan beberapa pin MIO sekaligus secara otomatis, yang akan dihubungkan sesuai dengan tata letak papan ZC702. Misalnya bila UART1 diaktifkan dan UART0 dinonaktifkan maka pada saat papan ZC702 dijalankan, jalur UART1 nantinya akan dihubungkan ke konektor USB-UART melalui UART ke USB converter chip pada board ZC702. Pemberian cek tanda pada box disamping setiap nama peripheral di diagram perangkat blok Zynq akan menyebabkan peripheral I/O menjadi aktif dan sebaliknya bila centang dihilangkan akan membuat peripheral tersebut menjadi tidak aktif. Setelah semua koneksi diatur maka Vivado akan mengimplementasikan perubahan yang telah dibuat untuk dipersiapkan nantinya diimplementasikan pada papan ZC702. Bila kita menggunakan Run Block Automation link maka vivado akan mengatur secara otomatis koneksi antar pin secara default oleh pengaturan sistem vivado. Hal ini dapat dilihat pada Gambar 3.19. dan Gambar 3.20.



Gambar 3.18 Pengaturan koneksi port Zynq



Gambar 3.19 Automation Link pada Vivado

Langkah 2 : Memvalidasi Desain

Desain yang telah dibuat harus di validasi terlebih dahulu untuk memeriksa kesesuaian desain yang telah dibuat dengan standar yang berlaku pada Vivado. Untuk memvalidasi desain yang telah dibuat, dapat melalui tombol F6 atau klik validating. Bila terjadi kesalahan kritis akan muncul pesan critical error, salah satu penyebab adalah bila M_AXI_GP0_ACLK belum dihubungkan sesuai aturan. Pengaturan ulang koneksi M_AXI_GP0_ACLK dapat dilakukan dengan meletakkan pointer mouse pada Diagram view blok processing system Zynq7 pada

pin yang akan diubah, menunggu muncul pointer pensil dan klik port M_AXI_GP0_ACLK kemudian ditarik ke port input FCLK_CLK0 untuk membuat koneksi antara dua port tersebut dimana aliran sinyal pewaktu akan dikirimkan dari port M_AXI_GP0_ACLK ke FCLK_CLK0. Karena telah diadakan perubahan maka sistem yang telah dibuat harus divalidasi ulang untuk memastikan tidak ada lagi kesalahan lainnya. Bila sudah tidak terdapat lagi kesalahan maka selanjutnya akan dibentuk HDL wrapper File untuk mengemas seluruh rancangan sistem yang akan dipersiapkan untuk diekspor ke sistem Xilinx. Untuk membentuk HDL wrapper file disediakan pilihan Let Vivado manage wrapper, untuk menggunakan setting dari standar yang ada pada Vivado untuk mengelola wrapper dan auto-update, dan disertakan pilihan Generate output products agar nantinya kita memperoleh material yang akan kita gunakan untuk ekspor ke sistem Xilinx. Langkah ini selanjutnya akan membangun semua hal yang akan diperlukan produk output. Karena telah dipilih Vivado mengelola secara otomatis maka seluruh pilihan manual dapat dilewati pada sistem processor IP. Vivado secara otomatis menghasilkan file dengan ekstensi .XDC untuk processor sub-system sehubungan telah disertakan pilihan Generate output products. Produk output tersebut akan disimpan pada direktori yang telah ditentukan sebelumnya.

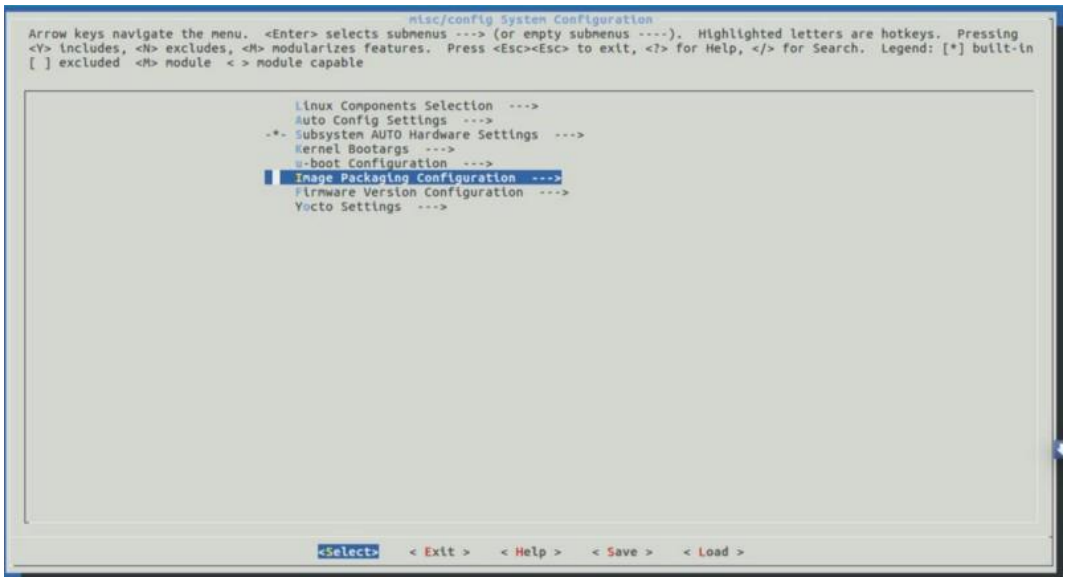
Langkah 3 : Membuat *kernel images* untuk desain *hardware* menggunakan Petalinux

Kernel images diperlukan untuk implementasi pada *board* Zynq dan juga sebagai perantara dalam membuat *environment* untuk Ubuntu ARM. Pembuatan *kernel image* itu sendiri menggunakan aplikasi Petalinux dengan versi 2017.1 yang kompatibel dengan aplikasi desain Xilinx Vivado Design Suite. Adapun cara untuk membuat *kernel images* sebagai berikut :

a. Buka terminal pada Ubuntu 16.04 LTS dan buat sebuah folder pada direktori yang diinginkan untuk membuat proyek Petalinux.
b. Buat proyek Petalinux pada direktori yang telah ditentukan tadi dengan menggunakan perintah `petalinux-create --type project --template zynq --name coba`. Disini penulis menggunakan nama folder “coba”.

c. Selanjutnya pindah ke folder “coba” dengan mengetikkan perintah `cd coba`.

d. Spesifikasikan posisi dari file .hdf yang berada pada folder .sdk dari desain yang telah dibuat dengan menggunakan Vivado tadi, lalu atur konfigurasi untuk proyek Petalinux dengan menggunakan perintah `petalinux-config -get-hw-description=/home/b304/Documents/design_1/design_1.sdk/`, akan muncul jendela untuk mengatur dari proyek yang dibuat, ini dapat dilihat pada Gambar 3.21.



Gambar 3.20 Tampilan konfigurasi Petalinux

Selanjutnya atur *boot image*, *kernel image*, dan *dtb image* menjadi *primary SD* pada Subsystem AUTO Hardware Setting. *Image Packaging Configurations* dan *root filesystem type* juga diatur menjadi SD Card.

e. *Build* proyek Petalinux dan *generate* boot file dengan perintah `petalinux-build`

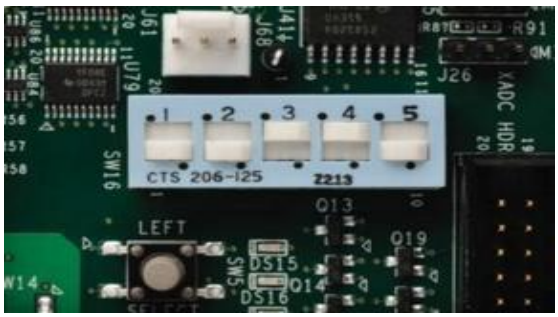
f. Setelah *build* proyek selesai, jalankan perintah lainnya untuk me-*generate* file BOOT.BIN, perintah yang dijalankan yaitu `petalinux-package -boot --format BIN -fsbl ./images/linux/zynq_fsbl.elf -fpga./home/b304/Documents/coba/design_1.bit --u-boot`.

Langkah 4 : Bangun *environment* Ubuntu ARM

Selanjutnya membuat environment Ubuntu ARM yang nantinya dijalankan pada *board* Zynq dengan tujuan untuk mempermudah pengguna dalam menggunakan Xilinx Zynq ZC702. Adapun cara membangun environment untuk Ubuntu ARM sebagai berikut :

a. Unduh Ubuntu ARM versi 16.04.2 dari situs https://rcn-ee.com/rootfs/eewiki/minfs/
b. Selanjutnya ekstraksi dari terminal file yang telah diunduh dengan menggunakan perintah `tar xf ubuntu-16.04.2-minimal-armhf-2017-06-18.tar.xz`.
c. Ekstrak dan salin Ubuntu root file system ke partisi rootfs dengan perintah
   `sudo tar xfvp ./*-*-*-armhf-*/armhf-rootfs-*.tar -C /media/rootfs/`
   Dilanjutkan dengan perintah : `sync`
   ```
   sudo chown root:root /media/rootfs/
   sudo chmod 755 /media/rootfs/
   ```

Untuk menjalankan reference software MV-HEVC yaitu HTM-16.3 terlebih dahulu menyalin isi dari folder "/images/linux/" pada folder proyek Petalinux yang telah dibuat sebelumnya ke SD Card. Selanjutnya pasangkan kabel UART ke port USB pada PC dan SD Card ke slot SD Card yang ada pada board Zynq. Atur konfigurasi switch pada board Zynq seperti pada Gambar 3.9.



Gambar 3.21 Konfigurasi switch untuk booting SD Card

Komunikasi antara *board* Zynq dan PC adalah dalam bentuk komunikasi serial. Sehingga harus disesuaikan port dan parameter yang digunakan dari PC yang berbasis Linux ke papan ZC702, untuk memastikan komunikasi telah dapat berlangsung dengan baik. Pada sistem Linux yang digunakan yaitu versi Ubuntu 16.04 LTS harus diketahui pada port mana terhubung ke papan ZC702. Untuk mengetahui port yang aktif digunakan papan ZC702, digunakan perintah : $dmesg di terminal Linux. Seluruh port-port yang sedang aktif akan ditampilkan, dan selanjutnya ditentukan dari port yang aktif tersebut yang mana dipakai untuk berkomunikasi dengan *board* ZC702, dimana port yang digunakan tidak selalu sama setiap di awal terjadi koneksi ke PC tersebut. Untuk langkah kerja lebih lanjut dapat dilihat pada [31].

Setelah koneksi semua aman, terlebih dahulu lengkapi *depedencies* dan *library* yang dibutuhkan agar tidak terjadi *error* pada saat menjalankan *reference software* yang terdiri dari *build-essentials* dan gcc-4.8. *Compile* aplikasi HTM-16.3 dengan mengetikkan perintah `make -f makefile` pada terminal Ubuntu ARM di *board* Zynq.

3.9 Analisis Pengurangan Kompleksitas Komputasi

Analisis pada penelitian pengurangan kompleksitas komputasi ini adalah waktu encoding dan kualitas video yang dilihat dari bitrate dan PSNR. Untuk perbandingan waktu encoding yang dibutuhkan digunakan persamaan sebagai berikut.

$$\Delta TimeEnc = \frac{TimeEnc_{baseline} - TimeEnc_{HTM}}{TimeEnc_{baseline}} \times 100\% \quad (2.1)$$

Dimana :

ΔTimeEnc = Perbedaan waktu seluruh pengkodean antara kondisi baseline dengan kondisi yang menerapkan *mode decision* (kondisi 1 dan kondisi 2).

$TimeEnc_{baseline}$ = Waktu encoding pada kondisi baseline.

$TimeEnc_{HTM}$ = Waktu encoding pada kondisi 1 dan kondisi 2.

Selanjutnya parameter pengujian sistem adalah pada kualitas video yang dilihat dari bitrate dan PSNR. Untuk menentukan kualitas video yang ditentukan oleh ITU-T adalah standar J.247 [13] yaitu Peak Signal to Noise Ratio (PSNR). PSNR membandingkan nilai maksimum sampel sinyal yang diinginkan dengan perbedaan nilai sampel dari video asli yang belum dikompresi dan yang telah dikompresi (sinyal noise). Nilai PSNR khas berkisar dari 30 hingga 50 dB, di mana nilai yang lebih tinggi menunjukkan kualitas video yang lebih baik [13]. Untuk membandingkan nilai PSNR dan bitrate yang didapatkan dari proses encoding, maka digunakan persamaan dari [7]–[11] yaitu :

$$\Delta Bits(\%) = \frac{Bits_{baseline} - Bits_{HTM}}{Bits_{baseline}} \times 100\% \quad (2.2)$$

Dimana :

ΔBits = Perbedaan nilai bitrate seluruh pengkodean antara kondisi baseline dengan kondisi yang menerapkan *mode decision* (kondisi 1 dan kondisi 2).

$Bits_{baseline}$ = Nilai bitrate pada kondisi baseline

$Bits_{HTM}$ = Nilai bitrate pada kondisi 1 dan kondisi 2

$$\Delta PSNR = PSNR_{baseline} - PSNR_{HTM} \quad (2.3)$$

Dimana :

ΔPSNR = Perbedaan nilai PSNR antara kondisi baseline dengan kondisi yang menerapkan *mode decision* (kondisi 1 dan kondisi 2).

$PSNR_{baseline}$ = Nilai bitrate pada kondisi baseline

$PSNR_{HTM}$ = Nilai bitrate pada kondisi 1 dan kondisi 2

BAB 4
HASIL DAN PEMBAHASAN

4.1 Pendahuluan

Pada bagian ini, terdapat pembahasan mengenai hasil implementasi dari pengurangan kompleksitas komputasi video MV-HEVC pada *platform* PC berbasis Linux dan *platform* Xilinx All Programmable SoC. Tujuan dari implementasi ini adalah untuk mengurangi kompleksitas komputasi pada video MV-HEVC dilihat dari perbandingan waktu encoding video dan kualitas video yang dihasilkan dari pengkodean pada *platform* PC berbasis Linux dan *platform* Xilinx All Programmable SoC. Seterusnya analisis dari hasil pada kedua *platform* tersebut akan dibahas pada bab ini.

4.2 Waktu Encoding

Pada sub bab ini ditunjukkan hasil pengurangan kompleksitas komputasi pada MV-HEVC untuk waktu encoding berdasarkan algoritma yang telah dijelaskan sebelumnya dengan 3 kondisi yang berbeda, dimana ketiga kondisi tersebut adalah kondisi baseline, kondisi 1, dan kondisi 2. *Mode decision* yang digunakan pada penelitian ini adalah ECU, CFM, ESD, dan *deblocking filter*.

Teori pada bab 2 yang menjelaskan tentang kompleksitas komputasi mengarah kepada kalkulasi yang dilakukan pada proses pengkodean seluruh video atau sebagian video yang mana proses ini sebagai jumlah komputasi dari waktu penggunaan prosesor. Jumlah komputasi ini memberikan efek kepada proses waktu yang dibutuhkan.

Dari pengkodean video MV-HEVC pada *platform* PC berbasis Linux dan *platform* Xilinx All Programmable SoC yaitu video Balloons dan video Newspaper menggunakan *reference software* HTM-16.3 dengan konfigurasi file baseCfg_3view yang telah diterapkan *mode decision* untuk kondisi 1 dan kondisi 2 dan kondisi baseline tanpa penerapan *mode decision* didapatkan hasil dalam satuan detik (s) sebagaimana dijelaskan pada Tabel 4.1 dan Tabel 4.2.

Tabel 4.1 Waktu Encoding Video MV-HEVC Balloons dan Newspaper pada *platform* PC berbasis Linux.

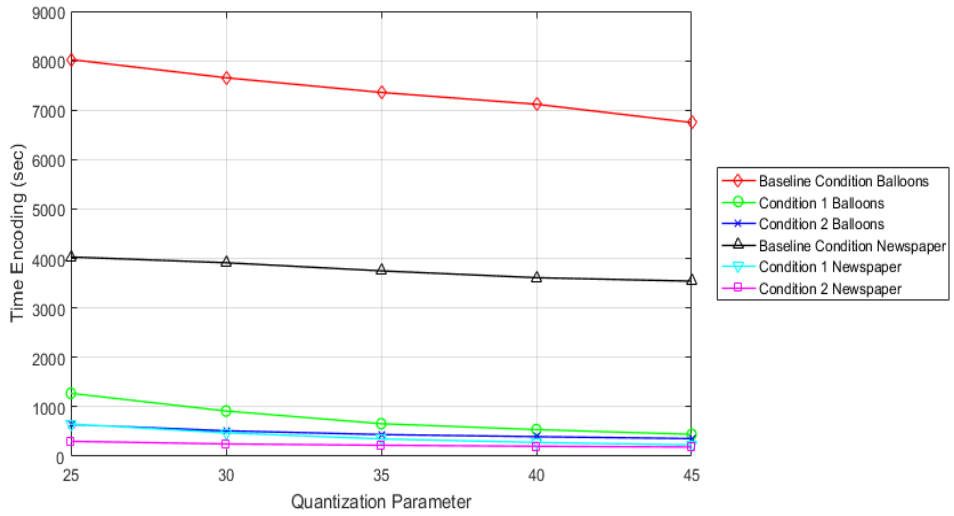
QP	Waktu Encoding					
	Balloons			Newspaper		
	Kondisi Baseline (detik)	Kondisi 1 (detik)	Kondisi 2 (detik)	Kondisi Baseline (detik)	Kondisi 1 (detik)	Kondisi 2 (detik)
25	8023,211	1273,935	642,678	4030,233	652,820	302,757
30	7654,20	915,527	515,975	3914,141	474,807	249,845
35	7357,73	659,52	439,578	3749,888	351,567	223,369
40	7118,592	540,146	396,854	3611,582	279,795	203,012
45	6749,19	444,108	358,059	3541,826	230,681	186,506

Tabel 4.2 Waktu Encoding Video MV-HEVC Balloons dan Newspaper pada *platform* Xilinx All Programmable SoC.

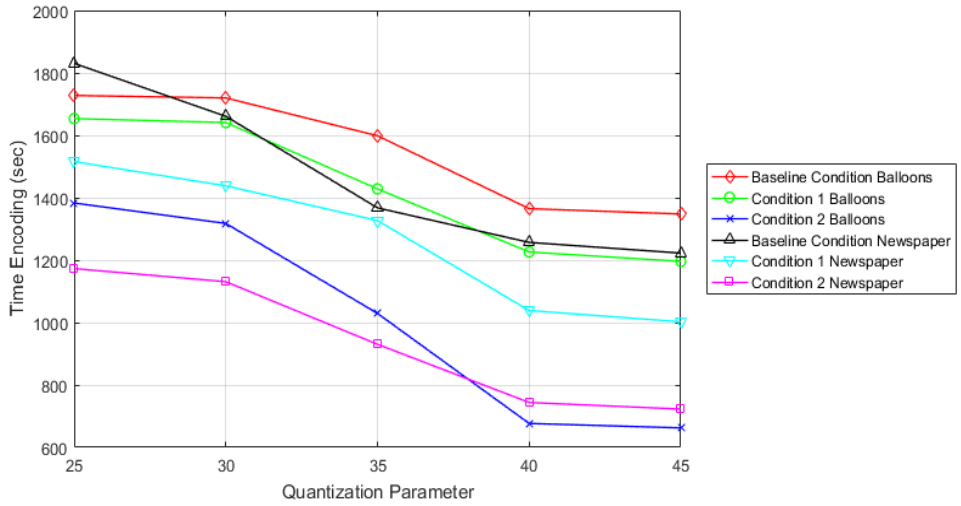
QP	Waktu Encoding					
	Balloons			Newspaper		
	Kondisi Baseline (detik)	Kondisi 1 (detik)	Kondisi 2 (detik)	Kondisi Baseline (detik)	Kondisi 1 (detik)	Kondisi 2 (detik)
25	1727,655	1653,56	1383,1	1830,687	1515,74	1172,99
30	1719,844	1640,95	1317,67	1660,989	1438,25	1130,88
35	1598,512	1428,35	1029,34	1367,485	1326,8	930,317
40	1364,97	1226,1	676,697	1256,86	1038,57	743,91
45	1347,885	1196,07	662,527	1222,552	1002,46	722,762

Dari video Balloons dan Newspaper dengan pixel yang sama yaitu 1024 x 768, *frame per second* (fps) 30fps, durasi video 10 detik, frame yang dikodekan berjumlah 150, dan jumlah view yang dikodekan adalah 3 view. Tabel 4.1 untuk waktu encoding pada *platform* PC berbasis Linux dan Tabel 4.2 untuk waktu encoding pada *platform* Xilinx All Programmable SoC memperlihatkan bahwa, untuk video Balloons dan Newspaper pada QP yang sama untuk kondisi baseline, kondisi 1, dan kondisi 2 terlihat adanya perbedaan waktu yang besar dikarenakan pengaruh dari *mode decision* dan kuantisasi pada proses pengkodeannya. Ini menandakan bahwa adanya pengurangan kompleksitas yang terjadi dari kondisi baseline ke kondisi 1 dan ke kondisi 2 yang berarti pengkodean CU, TU, dan PU pada video tersebut dapat dipercepat. Dari Tabel 4.1 dan Tabel 4.2 waktu encoding

untuk kedua *platform* lebih lanjut ditampilkan dalam bentuk grafik pada Gambar 4.1. dan Gambar 4.2.



Gambar 4.1 Grafik Waktu Encoding Video MV-HEVC pada *Platform* PC Berbasis Linux.



Gambar 4.2 Grafik Waktu Encoding Video MV-HEVC pada *Platform* Xilinx All Programmable SoC.

Pada Gambar 4.1 terlihat penurunan waktu encoding untuk kedua video yang dikodekan pada *platform* PC berbasis Linux dari parameter kuantisasi (QP) 25 ke QP 35. Kondisi 2 yang menerapkan *mode decision* lebih banyak dari kondisi 1 mampu memperoleh waktu encoding yang lebih cepat. Untuk kondisi baseline sendiri membutuhkan waktu encoding yang lebih lama dalam pengkodean video

dibandingkan kondisi 1 dan kondisi 2. Dapat disimpulkan bahwa waktu encoding kedua video MV-HEVC pada *platform* PC berbasis Linux untuk kondisi 2 lebih cepat dibandingkan kondisi 1 dan kondisi baseline.

Selanjutnya pada Gambar 4.2, pengkodean kedua video MV-HEVC pada *platform* Xilinx All Programmable SoC menggunakan *reference software* HTM-16.3 terjadi penurunan waktu encoding yang besar dari QP 30 ke QP 40 untuk ketiga kondisi pada video Balloons. Kondisi 2 mampu mencapai waktu encoding yang lebih cepat dari kondisi 1 dikarenakan penurunan waktu encoding yang sangat signifikan. Kondisi baseline dan kondisi 1 membutuhkan waktu encoding yang lebih lama.

Untuk video Newspaper pada pengkodean menggunakan *platform* Xilinx All Programmable SoC, kondisi baseline mengalami penurunan waktu encoding yang drastis dari QP 25 ke QP 35. Sedangkan pada kondisi 1 dan 2 penurunan waktu encoding terjadi pada QP 30 dan QP 40. Kondisi 2 pada pengkodean video Newspaper ini memperoleh waktu encoding lebih cepat dari kondisi 1. Disimpulkan pada pengkodean vide MV-HEVC menggunakan *platform* Xilinx All Programmable SoC, kondisi 2 untuk video Balloons dan Newspaper memiliki waktu encoding yang lebih cepat. Dari pengkodean pada kedua *platform* ini dapat disimpulkan kompleksitas komputasi pada waktu encoding pada video MV-HEVC dapat dikurangi seiring dengan peningkatan nilai QP.

Mode decision dalam pengurangan kompleksitas komputasi yang telah disebutkan sebelumnya dan implementasi pada kondisi 1 dan kondisi 2, maka dibandingkan kedua kondisi tersebut berdasarkan perbedaan waktu encoding untuk kedua *platform.* Perbedaan waktu encoding kondisi baseline dengan kondisi 1 dan waktu encoding kondisi baseline dengan kondisi 2 dianalisis menggunakan persamaan (2.1) untuk video MV-HEVC Balloons dan Newspaper dapat dilihat pada Tabel 4.3 dan Tabel 4.4.

Tabel 4.3 ΔWaktu Encoding Video MV-HEVC Balloons dan Newspaper pada *Platform* PC Berbasis Linux

QP	Balloons		Newspaper	
	Kondisi Baseline – Kondisi 1 (%)	**Kondisi Baseline – Kondisi 2 (%)**	**Kondisi Baseline – Kondisi 1 (%)**	**Kondisi Baseline – Kondisi 2 (%)**
25	84,14	91,98	83,80	92,48
30	88,03	93,25	87,86	93,61
35	91,03	94,02	90,62	94,04
40	92,41	94,42	92,25	94,37
45	93,41	94,69	93,48	94,73

Tabel 4.4 ΔWaktu Encoding Video MV-HEVC Balloons dan Newspaper pada *Platform* Xilinx All Programmable SoC

QP	Balloons		Newspaper	
	Kondisi Baseline – Kondisi 1 (%)	**Kondisi Baseline – Kondisi 2 (%)**	**Kondisi Baseline – Kondisi 1 (%)**	**Kondisi Baseline – Kondisi 2 (%)**
25	4,28	19,94	17,20	35,92
30	4,58	23,38	13,40	31,91
35	10,64	35,60	2,97	31,96
40	10,17	50,42	17,36	40,81
45	11,26	50,84	18	40,88

Untuk video yang sama dengan nilai QP yang sama, waktu encoding pada kondisi baseline – kondisi 1 video Balloons dan Newspaper untuk *platform* PC berbasis Linux nilai beda waktu encoding lebih besar daripada kondisi baseline – kondisi 1 pada video Balloons dan Newspaper untuk *platform* Xilinx All Programmable SoC. Hal ini juga berlaku pada kondisi baseline – kondisi 2. Ini menandakan *platform* PC berbasis Linux dalam melakukan pengkodean video membutuhkan waktu yang lebih lama dan kompleksitas komputasi yang besar daripada *platform* Xilinx All Programmable SoC. Desain blok IP pada *platform* Xilinx All Programmable SoC mempengaruhi dalam pemrosesan CU, TU, dan PU sehingga waktu encoding menjadi lebih cepat. Untuk peningkatan nilai QP mengindikasikan semakin tinggi persentase kompleksitas komputasi yang dapat dikurangi pada kedua video MV-HEVC.

4.3 Kualitas Video

Pada sub bab ini akan dibahas hasil pengurangan kompleksitas komputasi pada MV-HEVC dilihat dari kualitas video berdasarkan bitrate dan PSNR dari pengkodean video menggunakan *reference software* HTM-16.3. Bitrate dan PSNR untuk video Balloons dan Newspaper pada *platform* PC berbasis Linux dapat dilihat pada Tabel 4.5, Tabel 4.6, Tabel 4.7, Tabel 4.8, Tabel 4.9, dan Tabel 4.10.

Tabel 4.5 Bitrate dan PSNR untuk Kondisi Baseline Video MV-HEVC Balloons pada *platform* PC berbasis Linux

QP	Bitrate	YUV-PSNR
25	1611,1792	44,1823
30	900,76	40,5493
35	442,5424	37,3865
40	243,0464	35,217
45	124,0848	32,7313

Tabel 4.6 Bitrate dan PSNR untuk Kondisi 1 Video MV-HEVC Balloons pada *platform* PC berbasis Linux

QP	Bitrate	YUV-PSNR
25	1566,3008	43,6629
30	851,8432	39,9495
35	423,0096	37,0496
40	235,4688	34,9615
45	121,0016	32,5386

Tabel 4.7 Bitrate dan PSNR untuk Kondisi 2 Video MV-HEVC Balloons pada *platform* PC berbasis Linux

QP	Bitrate	YUV-PSNR
25	2073,6608	42,3549
30	980,2512	39,1894
35	459,8976	36,5549
40	249,0208	34,4978
45	125,52	32,0954

Tabel 4.8 Bitrate dan PSNR untuk Kondisi Baseline Video MV-HEVC Newspaper pada *platform* PC berbasis Linux

QP	Bitrate	YUV-PSNR
25	1241,128	44,3551
30	754,5136	40,862

35	414,9808	37,6262
40	242,6256	35,0463
45	124,5568	31,9673

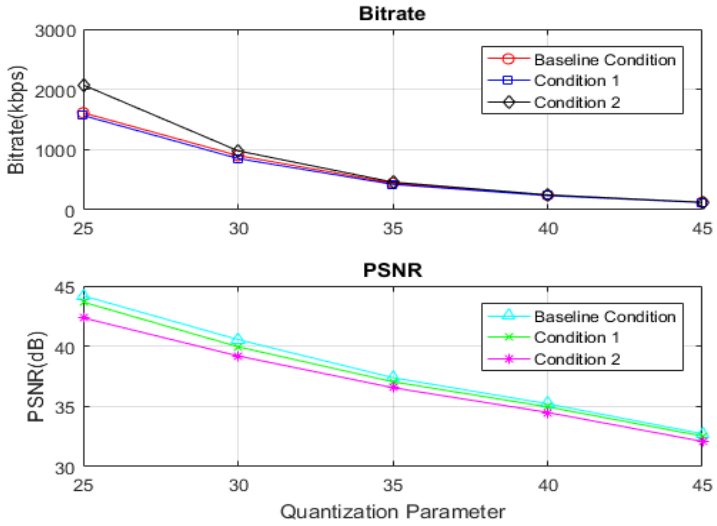
Tabel 4.9 Bitrate dan PSNR untuk Kondisi 1 Video MV-HEVC Newspaper pada *platform* PC berbasis Linux

QP	Bitrate	YUV-PSNR
25	1215,5872	44,0823
30	725,7776	40,4687
35	403,6064	37,3458
40	237,3504	34,7895
45	121,8064	31,7457

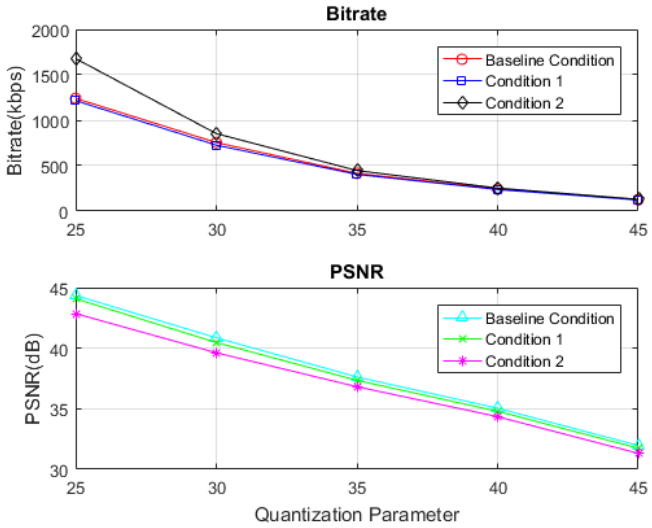
Tabel 4.10 Bitrate dan PSNR untuk Kondisi 2 Video MV-HEVC Newspaper pada *platform* PC berbasis Linux

QP	Bitrate	YUV-PSNR
25	1678,6592	42,8546
30	853,1027	39,621
35	445,9456	36,8189
40	253,0624	34,3465
45	129,3296	31,3153

Tabel diatas direpresentasikan kedalam grafik yang dapat dilihat pada Gambar 4.3 dan Gambar 4.4.



Gambar 4.3 Grafik Bitrate dan PSNR video Balloons pada *platform* PC berbasis Linux



Gambar 4.4 Grafik Bitrate dan PSNR video Newspaper pada *platform* PC berbasis Linux

Selanjutnya, Bitrate dan PSNR untuk video Balloons dan Newspaper pada *platform* Xilinx All Programmable SoC dapat dilihat pada Tabel 4.11, Tabel 4.12, Tabel 4.13, Tabel 4.14, Tabel 4.15, dan Tabel 4.16.

Tabel 4.11 Bitrate dan PSNR untuk Kondisi Baseline Video MV-HEVC Balloons pada *platform* Xilinx All Programmable SoC

QP	Bitrate	YUV-PSNR
25	1612,1568	44,1823
30	901,8304	40,5503
35	443,6112	37,3873
40	244,024	35,217
45	125,0624	32,7313

Tabel 4.12 Bitrate dan PSNR untuk Kondisi 1 Video MV-HEVC Balloons pada *platform* Xilinx All Programmable SoC.

QP	Bitrate	YUV-PSNR
25	1567,2784	43,6629
30	852,8208	39,9495
35	423,9872	37,0496
40	236,4464	34,9615
45	121,9792	32,5386

Tabel 4.13 Bitrate dan PSNR untuk Kondisi 2 Video MV-HEVC Balloons pada *platform* Xilinx All Programmable SoC.

QP	Bitrate	YUV-PSNR
25	2042,584	42,0468
30	953,8624	39,0659
35	448,0208	36,6064
40	244,1792	34,5535
45	124,3632	32,2733

Tabel 4.14 Bitrate dan PSNR untuk Kondisi Baseline Video MV-HEVC Newspaper pada *platform* Xilinx All Programmable SoC.

QP	Bitrate	YUV-PSNR
25	1245,0064	44,4019
30	756,2688	40,8812
35	415,4928	37,6318
40	243,04	35,0509
45	124,5408	31,9714

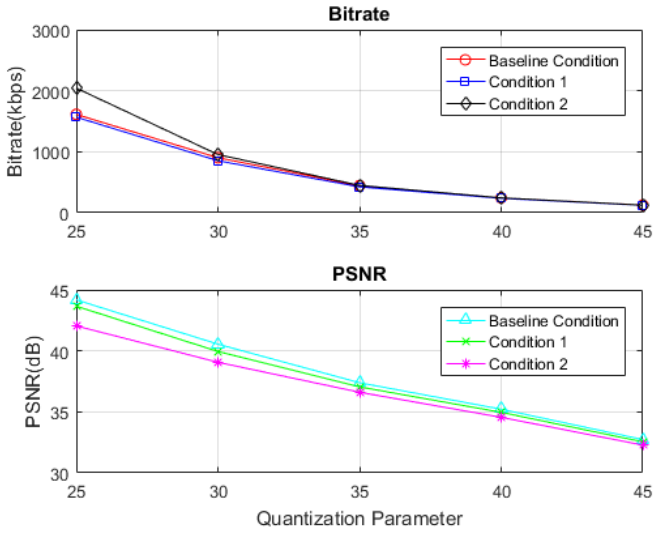
Tabel 4.15 Bitrate dan PSNR untuk Kondisi 1 Video MV-HEVC Newspaper pada *platform* Xilinx All Programmable SoC

QP	Bitrate	YUV-PSNR
25	1215,5872	44,0823
30	752,7776	40,4687
35	403,6064	37,3458
40	237,3504	34,7895
45	121,8064	31,7457

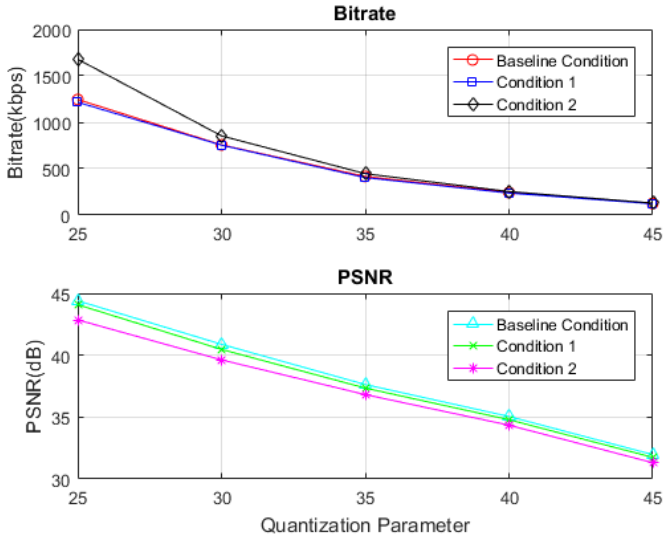
Tabel 4.16 Bitrate dan PSNR untuk Kondisi 2 Video MV-HEVC Newspaper pada *platform* Xilinx All Programmable SoC.

QP	Bitrate	YUV-PSNR
25	1678,6592	42,8546
30	853,1027	39,621
35	445,9456	36,8189
40	253,0624	34,3465
45	129,3296	31,3153

Tabel diatas direpresentasikan kedalam grafik yang dapat dilihat pada Gambar 4.5 dan Gambar 4.6.



Gambar 4.5 Grafik Bitrate dan PSNR video Balloons pada *platform* Xilinx All Programmable SoC



Gambar 4.6 Grafik Bitrate dan PSNR video Newspaper pada *platform* Xilinx All Programmable SoC

Teori pada bab 2 telah dijelaskan yaitu efisiensi pengkodean, kompleksitas komputasi, konten video, frame rate, dan kuantisasi berdampak kepada bitrate dan PSNR video. Selain itu, nilai QP yang lebih besar mengarah kepada encoding lossy yang mengurangi kualitas video. Kualitas video Balloons dan Newspaper yang dihasilkan dari pengkodean ini didasarkan pada penilaian objektif dilihat dari nilai

bitrate dan PSNR pada kedua *platform*. Untuk video Balloons dan Newspaper menunjukkan tidak ada perbedaan kualitas video yang besar pada kedua *platform* antara tinggi dan rendahnya nilai bitrate dan PSNR. Namun jika dikaitkan dengan QP, maka nilai QP rendah menghasilkan kualitas video yang bagus dibandingkan dengan QP tinggi yang menghasilkan kualitas video yang kurang bagus. Kualitas video yang bagus diikuti dengan kompleksitas komputasi tinggi dan begitu pula sebaliknya.

Tabel 4.11, Tabel 4.12, Tabel 4.13, Tabel 4.14, Table 4.13, Tabel 4.14, Table 4.15, dan Tabel 4.16 menunjukkan nilai bitrate dan PSNR berdasarkan kondisi baseline, kondisi 1, dan kondisi 2 untuk kedua *platform* dianalisis menggunakan persamaan (2.2) dan (2.3). Dari tabel tersebut hubungan kompleksitas komputasi dengan nilai bitrate dan PSNR mengarah kepada efisiensi pengkodean. ΔBitrate pada Tabel 4.17 dan Tabel 4.19 untuk kedua *platform*, pada kondisi baseline – kondisi 1 menunjukkan bitrate dari kondisi 1 lebih rendah dari bitrate kondisi baseline. Sedangkan pada kondisi baseline – kondisi 2, bitrate untuk kondisi 2 lebih besar dibandingkan dengan kondisi baseline. Dari ΔPSNR pada Tabel 4.18 dan Tabel 4.20 untuk *platform* PC berbasis Linux dan Xilinx All Programmable SoC menunjukkan jika nilai QP besar maka nilai ΔPSNR menjadi semakin kecil, begitu juga sebaliknya.

Tabel 4.17 ΔBitrate Video MV-HEVC Balloons dan Newspaper pada *Platform* PC Berbasis Linux

QP	**ΔBitrate (%)**			
	Balloons		**Newspaper**	
	Kondisi Baseline – Kondisi 1 (%)	**Kondisi Baseline – Kondisi 2 (%)**	**Kondisi Baseline – Kondisi 1 (%)**	**Kondisi Baseline – Kondisi 2 (%)**
25	2,78	-28,70	2,05	-35,25
30	5,43	-8,82	3,80	-13,06
35	4,41	-3,92	2,74	-7,46
40	3,11	-2,45	2,17	-4,30
45	2,48	-1,15	2,20	-3,83

Tabel 4.18 ΔPSNR Video MV-HEVC Balloons dan Newspaper pada *Platform* PC Berbasis Linux

QP	ΔPSNR (dB)			
	Balloons		Newspaper	
	Kondisi Baseline – Kondisi 1 (dB)	Kondisi Baseline – Kondisi 2 (dB)	Kondisi Baseline – Kondisi 1 (dB)	Kondisi Baseline – Kondisi 2 (dB)
25	0,5194	1,8274	0,2728	1,5005
30	0,5998	1,3599	0,3933	1,241
35	0,3369	0,8316	0,2804	0,8073
40	0,2555	0,7192	0,2568	0,6998
45	0,1927	0,6359	0,2216	0,652

Tabel 4.19 ΔBitrate Video MV-HEVC Balloons dan Newspaper pada *Platform* Xilinx All Programmable SoC

QP	ΔBitrate (%)			
	Balloons		Newspaper	
	Kondisi Baseline – Kondisi 1 (%)	Kondisi Baseline – Kondisi 2 (%)	Kondisi Baseline – Kondisi 1 (%)	Kondisi Baseline – Kondisi 2 (%)
25	2,78	-26,69	2,36	-34,83
30	5,43	-5,76	0,46	-12,80
35	4,42	-0,99	2,86	-7,32
40	3,10	-0,063	2,34	-4,12
45	2,46	0,55	2,19	-3,84

Tabel 4.20 ΔPSNR Video MV-HEVC Balloons dan Newspaper pada *Platform* Xilinx All Programmable SoC

QP	ΔPSNR (dB)			
	Balloons		Newspaper	
	Kondisi Baseline – Kondisi 1 (dB)	Kondisi Baseline – Kondisi 2 (dB)	Kondisi Baseline – Kondisi 1 (dB)	Kondisi Baseline – Kondisi 2 (dB)
25	0,5194	1,8274	0,2728	1,5005
30	0,5998	1,3599	0,3933	1,241
35	0,3369	0,8316	0,2804	0,8073
40	0,2555	0,7192	0,2568	0,6998
45	0,1927	0,6359	0,2216	0,652

4.4 Perbandingan Hasil Pengkodean Video pada *platform* PC Berbasis Linux dengan *platform* Xilinx All Programmable SoC

Dari Tabel 4.3 dan Tabel 4.4 didapatkan rata – rata untuk mengetahui besar pengurangan kompleksitas komputasi dilihat dari waktu encoding pada video MV-HEVC Balloons dan Newspaper yang menerapkan *mode decision* terlihat pada Tabel 4.21.

Tabel 4.21 Pengurangan Kompleksitas Komputasi dari Waktu Encoding untuk masing - masing *Platform*

Platform	**Video MV-HEVC**			
	Balloons		**Newspaper**	
	Kondisi 1 (%)	**Kondisi 2 (%)**	**Kondisi 1 (%)**	**Kondisi 2 (%)**
PC berbasis Linux	89,804	93,672	89,602	93,846
Xilinx All Programmable SoC	8,186	36,036	13,786	36,296

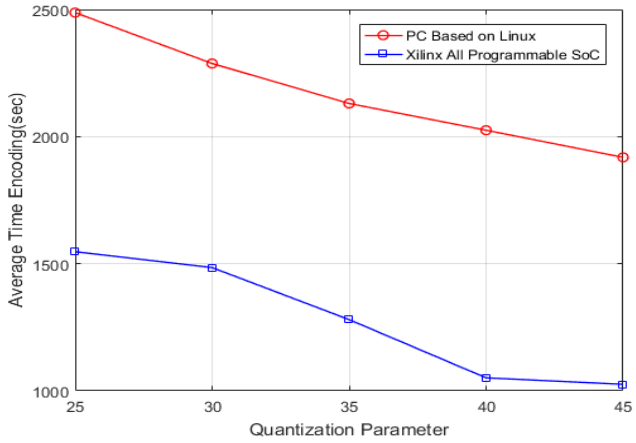
Pengurangan kompleksitas komputasi pada Tabel 4.21 pada *platform* PC berbasis Linux menunjukkan *mode decision* yang diterapkan pada kondisi 2 mampu mengurangi kompleksitas lebih besar pada video Balloons dengan persentase 93,672% dibandingkan *mode decision* yang diterapkan pada kondisi 1 dengan persentase 89,804% dengan referensi kondisi baseline. Untuk video Newspaper pada *platform* yang sama, kondisi 2 juga mampu mengurangi kompleksitas komputasi lebih besar dengan persentase 93,846% dibandingkan dengan kondisi 1 sebesar 89,602% dengan referensi kondisi baseline. Pada *platform* Xilinx All Programmable SoC, pengurangan kompleksitas komputasi untuk video Balloons dengan penerapan *mode decision* pada kondisi 2 mampu mengurangi kompleksitas komputasi sebesar 36,036% dan kondisi 1 sebesar 8,186% dari kondisi baseline. Untuk video Newspaper pada kondisi 2 sebesar 36,296% dan kondisi 1 sebesar 13,786% dengan referensi kondisi baseline.

Kedua sampel video MV-HEVC dijalankan pada *platform* PC berbasis Linux dan *platform* Xilinx Zynq ZC702. Untuk *platform* PC berbasis Linux yang memiliki spesifikasi Intel®Core™ i7-4770 CPU@3,4GHz dan RAM 8GB dan *platform* Xilinx All Programmable SoC yang memiliki spesifikasi berdasarkan desain blok IP yaitu LUT sejumlah 5951, flip – flop sejumlah 7063, BRAM

sejumlah 5, I/O sejumlah 4, dan BUFG sejumlah 1. Didapatkan rata – rata dari kedua platform tersebut mengacu kepada Gambar 4.1 dan Gambar 4.2 terlihat pada Tabel 4.22 dan direpresentasikan pada Gambar 4.7.

Tabel 4.22 Rata - Rata Waktu Encoding dalam Detik untuk *Platform* PC Berbasis Linux dan *Platform* Xilinx All Programmable SoC.

QP	PC Berbasis Linux (detik)	Xilinx All Programmable SoC (detik)
25	2487,606	1547,288
30	2287,416	1484,766
35	2130,275	1280,134
40	2024,997	1051,185
45	1918,395	1025,708



Gambar 4.7 Grafik Rata - Rata Waktu Encoding untuk *Platform* PC Berbasis Linux dan *Platform* Xilinx All Programmable.

Dari Gambar 4.1 dan Gambar 4.2 yang diambil rata – rata keseluruhan untuk masing – masing *platform* tersebut didapatkan selisih waktu encoding berdasarkan Gambar seperti yang terlihat pada Tabel 4.23. Hasil selisih tersebut dijumlahkan secara keseluruhan kemudian dirata-ratakan dan didapatkan nilai sebesar 891,921 detik. Sehingga pengurangan kompleksitas komputasi untuk waktu encoding pada *platform* Xilinx All Programmable SoC dipersentasekan sebesar 35,85% yang lebih cepat dibandingkan dengan *platform* PC Berbasis Linux.

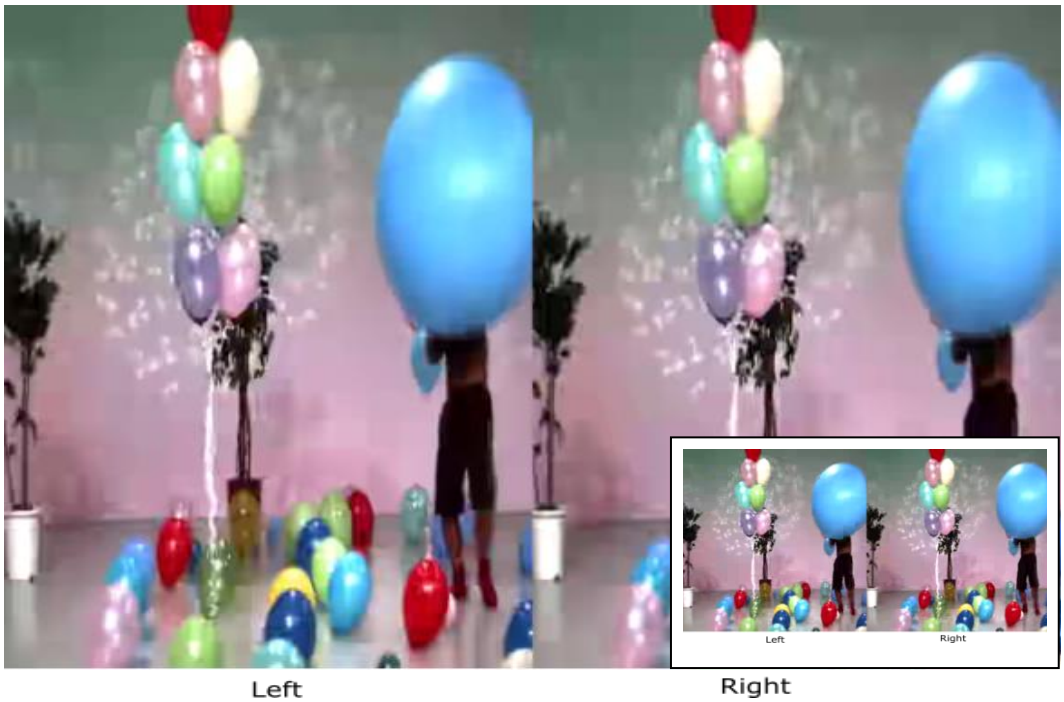
Tabel 4.23 Rata – Rata Selisih Waktu Encoding untuk Kedua *Platform*

QP	Rata – Rata Selisih Waktu Encoding (detik)
25	940,318
30	802,65
35	850,141
40	973,812
45	892,687
Avg	891,921

Visualisasi video MV-HEVC yang dibandingkan berdasarkan kedua *platform* yang digunakan dalam pengkodean dipengaruhi oleh jumlah frame video per detik, resolusi, durasi, banyak frame yang dikodekan, dan jumlah view. Dari kualitas video berdasarkan nilai bitrate dan PSNR yang telah dibahas sebelumnya, hasil cuplikan video Balloons dan Newspaper ditampilkan bersamaan dengan layar besar adalah dalam *platform* PC berbasis Linux dan layar kecil adalah dalam *platform* Xilinx All Programmable SoC. Kualitas visual cuplikan video Balloons dan Newspaper tampaknya sangat mirip untuk *platform* PC berbasis Linux dan *platform* Xilinx All Programmable SoC.

Gambar 4.8 Perbandingan Cuplikan Video Balloons *Platform* PC Berbasis Linux dengan *Platform* Xilinx All Programmable SoC untuk Kondisi Baseline.

Gambar 4.9 Perbandingan Cuplikan Video Balloons *Platform* PC Berbasis Linux dengan *Platform* Xilinx All Programmable SoC untuk Kondisi 1.



Gambar 4.10 Perbandingan Cuplikan Video Balloons *Platform* PC Berbasis Linux dengan *Platform* Xilinx All Programmable SoC untuk Kondisi 2.

Gambar 4.11 Perbandingan Cuplikan Video Newspaper *Platform* PC Berbasis Linux dengan *Platform* Xilinx All Programmable SoC untuk Kondisi Baseline.

Gambar 4.12 Perbandingan Cuplikan Video Newspaper Platform PC Berbasis Linux dengan Platform Xilinx All Programmable SoC untuk Kondisi 1.

Gambar 4.13 Perbandingan Cuplikan Video Balloons *Platform* PC Berbasis Linux dengan *Platform* Xilinx All Programmable SoC untuk Kondisi 2.

BAB 5
PENUTUP

5.1 Kesimpulan

Berdasarkan tujuan, hasil, dan analisa dari penelitian yang telah dilakukan, disimpulkan bahwa :

1. *Mode decision* yang digunakan dalam penelitian ini terbukti mampu mengurangi kompleksitas komputasi pada video MV-HEVC Balloons dan Newspaper dari waktu encoding.
2. Hasil pengurangan kompleksitas komputasi video MV-HEVC antara *platform* PC berbasis Linux dengan *platform* Xilinx All Programmable SoC dengan penerapan *mode decision*, pada *platform* PC berbasis Linux pengurangan kompleksitas komputasi untuk video Balloons yang lebih besar terjadi pada kondisi 2 yaitu sebesar 93,672% sedangkan pada kondisi 1 sebesar 89,804%. Hal yang sama juga terjadi pada video Newspaper dimana pengurangan kompleksitas komputasi pada kondisi 2 sebesar 93,846% dibandingkan dengan kondisi 1 sebesar 89,602% dengan referensi untuk kedua video adalah kondisi baseline. Untuk *platform* Xilinx All Programmable SoC, pada video Balloons kondisi 2 juga mencapai pengurangan kompleksitas komputasi yang besar yaitu sebesar 36,036% dan kondisi 1 sebesar 8,186% dari kondisi baseline. Untuk video Newspaper pada kondisi 2 sebesar 36,296% dan kondisi 1 sebesar 13,786% dengan referensi kedua video adalah kondisi baseline. Perbandingan pengurangan kompleksitas komputasi untuk kedua *platform* dalam hal percepatan waktu encoding didapatkan *platform* Xilinx All Programmable SoC memperoleh waktu encoding yang lebih cepat sebesar 35,85% dari *platform* PC berbasis Linux.
3. Berdasarkan nilai bitrate dan PSNR yang didapatkan terhadap kuantisasi dari pengkodean video MV-HEVC Balloons dan Newspaper bahwa, pada *platform* PC berbasis Linux dan *platform* Xilinx All Programmable SoC kualitas video MV-HEVC Balloons dan Newspaper tidak jauh berbeda. Jika

dilihat dari perbedaan antara kondisi baseline, kondisi 1, dan kondisi 2 dengan penilaian yang objektif, maka kondisi baseline sendiri memiliki kualitas video yang bagus sedangkan pada kondisi yang menerapkan *mode decision* memiliki kualitas video kurang bagus. Selanjutnya jika ditinjau dari nilai QP, nilai QP yang rendah menghasilkan kualitas video yang bagus dengan kompleksitas tinggi dibandingkan dengan QP tinggi yang menghasilkan kualitas video yang kurang bagus dengan kompleksitas rendah. Kondisi baseline sendiri memiliki kualitas video yang bagus sedangkan pada kondisi yang menerapkan *mode decision* kualitas video kurang bagus.

5.2 Saran

Pada penelitian selanjutnya terkait pengurangan kompleksitas komputasi pada video MV-HEVC, dapat dilakukan pengembangan atau penambahan sebagai berikut :

1. Diperlukan pengembangan desain blok IP untuk *platform* Xilinx All Programmable SoC.
2. Penambahan untuk jumlah view lebih dari 3 dan jumlah frame yang dikodekan.
3. Pengembangan dari *mode decision* dalam pengurangan kompleksitas komputasi untuk video dengan resolusi tinggi.

DAFTAR PUSTAKA

[1] Z. Li, X. S. Ye, H. Zhang, L. Lu, C. Lu, and L. C. Cheng, "Real-time 3D Video System Based on FPGA," *3rd Int. Conf. Consum. Electron. Commun. Netw. CECNet*, pp. 469–472, 2013.

[2] K. Muller *et al.*, "3D High-Efficiency Video Coding for Multi-View Video and Depth Data," *IEEE Trans. Image Process.*, vol. 22, no. 9, pp. 3366–3378, Sep. 2013.

[3] Y.-S. Ho and K.-J. Oh, "Overview of Multi-view Video Coding," *14th Int. Workshop Syst. Signals Image Process. 2007 6th EURASIP Conf. Focus. Speech Image Process. Multimed. Commun. Serv.*, pp. 5–12, 2007.

[4] M. Jamali and S. Coulombe, "Fast HEVC Intra Mode Decision Based on RDO Cost Prediction," *IEEE Trans. Broadcast.*, pp. 1–14, 2018.

[5] M. Ahmadi, A. Wali, A. Walha, and A. M. Alimi, "A new Motion Estimation technique for Video Coding," in *2015 15th International Conference on Intelligent Systems Design and Applications (ISDA)*, Marrakech, 2015, pp. 110–115.

[6] G. Corrêa, P. Assunção, L. Agostini, and L. A. da S. Cruz, *Complexity-aware High Efficiency Video Coding*. Springer, 2016.

[7] W.-J. Hsu and H.-M. Hang, "Fast coding unit decision algorithm for HEVC," in *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, Kaohsiung, Taiwan, 2013, pp. 1–5.

[8] R. Khemiri, N. Bahri, F. Belghith, F. E. Sayadi, M. Atri, and N. Masmoudi, "Fast motion estimation for HEVC video coding," in *2016 International Image Processing, Applications and Systems (IPAS)*, Hammamet, Tunisia, 2016, pp. 1–4.

[9] J. Leng, L. Sun, T. Ikenaga, and S. Sakaida, "Content Based Hierarchical Fast Coding Unit Decision Algorithm for HEVC," in *2011 International Conference on Multimedia and Signal Processing*, Guilin, China, 2011, pp. 56–59.

[10] Y. Shi, Z. Gao, and X. Zhang, "Early TU Split Termination in HEVC Based on Quasi-Zero-Block," in *Proceedings of the 3rd International Conference on Electric and Electronics*, Hong Kong, China, 2013.

[11] H. R. Tohidypour, M. T. Pourazad, P. Nasiopoulos, and V. Leung, "A content adaptive complexity reduction scheme for HEVC-based 3D video coding," in *2013 18th International Conference on Digital Signal Processing (DSP)*, Fira, Santorini, Greece, 2013, pp. 1–5.

[12] A. V. Bukit and Wirawan, "3D video coding development based on FPGA platform Xilinx Zynq-7000," in *2017 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, Surabaya, 2017, pp. 29–34.

[13] B. Bing, *Next-generation video coding and streaming*. Hoboken, New Jersey: Wiley, 2015.

[14] A. Vetro, T. Wiegand, and G. J. Sullivan, "Overview of the Stereo and Multiview Video Coding Extensions of the H.264/MPEG-4 AVC Standard," *Proc. IEEE*, vol. 99, no. 4, pp. 626–642, Apr. 2011.

[15] N. A. Dodgson, “Autostereoscopic 3D displays,” *Computer*, vol. 38, no. 8, pp. 31–36, Aug. 2005.
[16] S. Vivienne, B. Madhukar, and J. Gary, “High Efficiency Video Coding (HEVC): Algorithms and Architectures,” *Springer*, 2014.
[17] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, “Overview of the High Efficiency Video Coding (HEVC) Standard,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.
[18] K. R. Rao, D. N. Kim, and J. J. Hwang, *Video coding standards: AVS China, H.264/MPEG-4 PART 10, HEVC, VP6, DIRAC and VC-1*. Dordrecht: Springer, 2014.
[19] M. Wien, *High efficiency video coding: coding tools and specification*. Heidelberg: Springer, 2015.
[20] Y. Aksehir, K. Erdayandi, T. Z. Ozcan, and I. Hamzaoglu, “A low energy adaptive motion estimation hardware for H.264 multiview video coding,” *J. Real-Time Image Process.*, vol. 15, no. 1, pp. 3–12, Jun. 2018.
[21] M. Lukacs, “Predictive coding of multi-viewpoint image sets,” in *ICASSP ’86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Tokyo, Japan, 1986, vol. 11, pp. 521–524.
[22] P. Merkle, A. Smolic, K. Muller, and T. Wiegand, “Efficient Prediction Structures for Multiview Video Coding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 11, pp. 1461–1473, Nov. 2007.
[23] G. J. Sullivan, J. M. Boyce, Y. Chen, J.-R. Ohm, C. A. Segall, and A. Vetro, “Standardized Extensions of High Efficiency Video Coding (HEVC),” *IEEE J. Sel. Top. Signal Process.*, vol. 7, no. 6, pp. 1001–1016, Dec. 2013.
[24] G. Tech, Y. Chen, K. Muller, J.-R. Ohm, A. Vetro, and Y.-K. Wang, “Overview of the Multiview and 3D Extensions of High Efficiency Video Coding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 35–49, Jan. 2016.
[25] “Zynq-7000 All Programmable SoC Technical Reference Manual (UG585),” p. 1863, 2016.
[26] L. H. Crockett, R. A. Elliot, M. A. Enderwitz, and R. W. Stewart, “The Zynq Book.” Strathclyde Academic Media, 2014.
[27] W. Liu, J. Li, and Y. B. Cho, “A novel architecture for parallel multi-view HEVC decoder on mobile device,” *EURASIP J. Image Video Process.*, vol. 2017, no. 1, Dec. 2017.
[28] K. Choi, S.-H. Park, and E. S. Jang, “Coding tree Pruning Based CU Early Termination,” *JCT-VC Doc. JCTVC-F092*, 2011.
[29] R. Gweon and Y.-L. Lee, “Early Termination of CU Encoding to Reduce HEVC Complexity,” *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E95.A, no. 7, pp. 1215–1218, 2012.
[30] A. Norkin *et al.*, “HEVC Deblocking Filter,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1746–1754, Dec. 2012.
[31] C. Niroshan, “Installing Ubuntu on Xilinx ZYNQ-7000 AP SoC Using PetaLinux,” *Medium*, 21-Jun-2017. .

BIODATA PENULIS

Penulis telah menyelesaikan pendidikan jenjang S1 di Universitas Negeri Padang (UNP) pada jurusan Pendidikan Teknik Elektronika, lulus pada bulan September tahun 2014 dan terdaftar sebagai mahasiswa Program Pasca Sarjana bulan September tahun 2016 pada bidang studi Telekomunikasi Multimedia di Teknik Elektro Institut Teknologi Sepuluh Nopember (ITS).

Penulis telah mengikuti Seminar Tesis dengan judul **"Pengurangan Kompleksitas Komputasi pada Multiview HEVC Berbasis Perangkat FPGA**" pada tanggal 5 Juli 2017 dan Ujian Tesis pada tanggal 20 Desember 2018 sebagai salah satu persyaratan untuk memperoleh gelar Magister Teknik (M.T).

Nama : M.Suhairi

Tempat, Tanggal Lahit : Koto Baru Solok Sumatera Barat, 11 Februari 1992

Email : kaitoaokage@gmail.com / suhairi1102@gmail.com

Riwayat Pendidikan

1. SDS Pertiwi (1997-2003)
2. SMP N 4 Kota Solok (2003-2006)
3. SMA N 1 Gunung Talang (2006-2009)
4. Universitas Negeri Padang (2009-2014)
5. Institut Teknologi Sepuluh Nopember (2016-2018)