TESIS – TE142599

PENGKODEAN VIDEO 3D PADA FPGA BERBASISKAN XILINX ZYNQ-7000

ALEXANDER VICTOR BUKIT
2215203202

DOSEN PEMBIMBING
Dr. Ir. Wirawan, DEA

PROGRAM MAGISTER
BIDANG KEAHLIAN TELEKOMUNIKASI MULTIMEDIA
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

TESIS – TE092098

PENGKODEAN VIDEO 3D PADA FPGA BERBASISKAN XILINX ZYNQ-7000

ALEXANDER VICTOR BUKIT
2215203202

DOSEN PEMBIMBING
Dr. Ir. Wirawan, DEA

PROGRAM MAGISTER
BIDANG KEAHLIAN TELEKOMUNIKASI MULTIMEDIA
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

LEMBAR PENGESAHAN

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

oleh:

Alexander Victor Bukit
NRP. 2215203202

Tanggal Ujian : 08 Juni 2017
Periode Wisuda : September 2017

Disetujui oleh:

1. Dr. Ir. Wirawan, DEA (Pembimbing)
   NIP: 196311091989031011
2. Prof. Ir. Gamantyo Hendrantoro, M.Eng, Ph.D (Penguji)
   NIP:197011111993031002
3. Dr. Ir. Endroyono, DEA (Penguji)
   NIP:196504041991021001
4. Dr. Ir. Puji Handayani, MT (Penguji)
   NIP:196605101992032002
5. Eko Setijadi, ST, MT, Ph.D (Penguji)
   NIP: 197210012003121002

Dekan Fakultas Teknologi Elektro

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

Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN TESIS

Dengan ini saya menyatakan bahwa isi keseluruhan Tesis saya dengan judul "**PENGKODEAN VIDEO 3D PADA FPGA BERBASISKAN XILINX ZYNQ-7000**" 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, 08 Juni 2017

Alexander Victor Bukit
NRP. 2215203202

Halaman ini sengaja dikosongkan

PENGKODEAN VIDEO 3D PADA FPGA BERBASISKAN XILINX ZYNQ-7000

Nama mahasiswa : Alexander Victor Bukit
NRP : 2215203202
Pembimbing : Dr. Ir. Wirawan, DEA

ABSTRAK

Kebutuhan konsumen terhadap teknologi multimedia yang baru dan lebih handal, menggiring pihak industri untuk meningkatkan pelayanan di bidang pemasaran entertainment, sehingga pada muaranya mendorong popularisasi konten video 3D, perangkat pendukung yang berkemampuan 3D, dan aplikasi-aplikasi 3D. Sebagai fenomena yang terjadi saat ini, smartphone, tablet, dan perangkat mobile lainnya sudah melampaui nilai penjualan PC. Bersamaan dengan semakin populernya video 3D dan diaplikasikan ke perangkat mobile tersebut, mengakibatkan kebutuhan akan penyimpanan, transmisi data, dan tampilan membutuhkan pengkodean yang efisien.

High Efficiency Video Coding (HEVC) adalah teknik pengkodean video yang telah didesain menjadi standar untuk banyak aplikasi video dan memiliki kehandalan yang cukup signifikan dari generasi pendahulunya seperti teknik pengkodean H.264. Meskipun HEVC memiliki pengkodean yang sangat efiesien, namun disamping itu memerlukan beban prosesor yang berat dan menjalankan beban yang paralel pada saat pengkodean data yang berisi video. Untuk meningkatkan kehandalan dalam proses encoder, salah satunya dapat dilakukan dengan mengimplementasikan kode HEVC ke Zynq 7000 AP SoC. Diaplikasikan dalam tiga desain yaitu pertama dengan mengimplementasikan kedalam Zynq PS sebagai operasi standalone. Kedua yaitu dengan mengimplementasikan HEVC encoder dalam hardware/software co-design. Dan ketiga, implementasi code HEVC ke Zynq PL, tanpa PS. Dalam implementasi ini digunakan perangkat Xilinx Vivado HLS untuk mengembangkan kode yang dibutuhkan. Nilai hasil yang akan didapatkan adalah waktu yang dibutuhkan untuk pengkodean, PSNR, dan ukuran file hasil pengkodean kemudian akan dibandingkan antara kinerja PC berbasis Linux dengan FPGA Xilinx Zynq-7000.

Kata Kunci : Xilinx, Zynq, HEVC, H.265, Vivado.

Halaman ini sengaja dikosongkan

3D VIDEO CODING BASED ON XILINX ZYNQ-7000 FPGA

By	: Alexander Victor Bukit
Student Identity Number	: 2215203202
Supervisor(s)	: Dr. Ir. Wirawan, DEA

ABSTRACT

Consumer demand for new multimedia technologies and more reliable, drove the industry to improve services in the field of entertainment marketing, so that the estuary encourage the popularization of 3D video content, supporting devices 3D capabilities, and 3D applications. As a phenomenon that occurs at this time, smartphones, tablets, and other mobile devices has surpassed the value of PC sales. Along with the growing popularity of 3D video and be applied to the mobile device, resulting in the need for storage, data transmission, and display requires an efficient coding.

High Efficiency Video Coding (HEVC) is a video coding technique that has been designed to become the standard for many video applications and has the reliability significantly from the preceding generation such as H.264 coding techniques. Although HEVC has very efiesien coding, but besides that it requires a heavy processor load and run parallel load at the time of encoding data containing the video. To improve reliability in the process of encoder, one of which can be done by implementing a code HEVC to Zynq 7000 AP SoC. Applied in three designs into Zynq first to implement PS as a standalone operation. The second is to implement HEVC encoder in hardware / software co-design. And third, the implementation code HEVC to Zynq PL, without PS. In this implementation Xilinx device is used Vivado HLS to develop the code needed. The value of the results to be obtained is the time required for video encoding, PSNR, and encoded file size that will be compared between the performance of Linux-based PCs with Xilinx Zynq-7000 FPGA.

Keywords : Xilinx, Zynq, HEVC, H.265, Vivado

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur saya ucapkan kepada Tuhan Yang Maha Esa, atas berkat dan rahmat-Nya yang dianugerahkan kepada penulis sehingga dapat menyelesaikan tesis dengan judul "Pengkodean Video 3D pada FPGA Berbasiskan Xilinx Zynq-7000". Tesis ini merupakan salah satu syarat untuk menyelesaikan dan memperoleh gelar master teknik di bidang keahlian Telekomunikasi Multimedia, Departemen Teknik Elektro - Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember.

Penulis menyampaikan terima kasih sebesar-besarnya kepada seluruh pihak yang telah memberikan dukungan terhadap penyelesaian tesis ini, yaitu antara lain :

1. Kedua orang tua yang telah senantiasa memberikan doa kepada anaknya, sehingga penulis mampu menyelesaikan tesis dan melakukan hal yang terbaik bagi agama, almamater ITS, bangsa dan negara.
2. Istri dan anak-anak yang telah mendukung dan mendoakan sehingga penulis selalu diberi kelancaran dalam penulisan tesis ini.
3. Komandan Sekolah Tinggi Teknologi Angkatan Laut (STTAL) beserta seluruh jajaran staf dan Dinas Pendidikan TNI Angkatan Laut (Disdikal) yang telah memberi dukungan dalam masa kedinasan dan menjalani pendidikan di Institut Teknologi Sepuluh Nopember (ITS).
4. Bapak Dr. Ir. Wirawan, DEA selaku dosen pembimbing tesis.
5. Bapak Prof. Gamantyo Hendrantoro, S.T, M.Eng, Ph.D, Bapak Dr. Ir. Endroyono, DEA, Ibu Dr. Ir. Puji Handayani, M.T., Bapak Eko Setijadi, S.T, M.T, Ph. D selaku dosen penguji tesis.
6. Bapak Dr. Ir. Endroyono, DEA selaku dosen wali penulis.

Semoga Tuhan Yang Maha Esa membalas segala budi baik yang telah diberikan sehingga penulis dapat menyelesaikan tesis ini dengan baik.

Surabaya, 08 Juni 2017
Penulis

Alexander Victor Bukit

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Kebutuhan konsumen terhadap teknologi multimedia yang baru dan lebih handal, menggiring pihak industri untuk meningkatkan pelayanan di bidang pemasaran entertainment, sehingga pada muaranya mendorong popularisasi konten di bidang video 3D yaitu perangkat pendukung yang berkemampuan 3D, dan aplikasi-aplikasi 3D. Sebagai fenomena yang terjadi saat ini, smartphone, tablet, dan perangkat mobile lainnya sudah melampaui nilai penjualan Personal Computer (PC) sehingga interaksi manusia dengan perangkat mobile jauh lebih besar daripada dengan PC. Bersamaan dengan semakin populernya video 3D dan diaplikasikan ke perangkat mobile tersebut, mengakibatkan kebutuhan akan penyimpanan, transmisi data, dan tampilan membutuhkan pengkodean yang efisien.

High Efficiency Video Coding (HEVC) adalah teknik pengkodean video yang telah didesain menjadi standar untuk banyak aplikasi video dan memiliki kehandalan yang cukup signifikan dari generasi pendahulunya seperti teknik pengkodean H.264 [1]. Meskipun HEVC memiliki pengkodean yang sangat efisien, namun disamping itu memerlukan beban prosesor yang berat dan menjalankan beban yang paralel pada saat pengkodean data yang berisi video. Untuk meningkatkan kehandalan dalam proses encoder, salah satunya dapat dilakukan dengan mengimplementasikan kode HEVC ke Xilinx All Programmable SoC, karena pada FPGA proses pengkodean akan dijalankan dengan paralel processing dibandingkan pada pengkodean konvensional dengan proses sekuensial. Pada penelitian ini digunakan adalah Zynq-7000 papan ZC-702 karena tipe tersebut telah mampu menangani proses pengkodean video 3D dengan HEVC dengan baik dan juga memiliki nilai ekonomis karena harganya relatif lebih murah dibanding dengan FPGA dari tipe yang sama serta mudah didapatkan di pasaran. Diaplikasikan dalam tiga desain yaitu pertama dengan mengimplementasikan kedalam Zynq PS (Processing System)[2] sebagai operasi standalone. Kedua yaitu dengan mengimplementasikan HEVC encoder dalam hardware/software co-design. Dan ketiga, implementasi code HEVC ke Zynq PL (Programmable Logic), tanpa PS.

Dalam implementasi ini digunakan perangkat Xilinx Vivado HLS untuk mengembangkan kode yang dibutuhkan. [3]

1.2 Rumusan Masalah

1. Bagaimana mengimplementasikan pengkodean video 3D melalui PS dan PL secara independen pada Xilinx All Programmable SoC.
2. Bagaimana melakukan implementasi melalui PS dan PL secara terintegrasi pada Xilinx All Programmable SoC.
3. Bagaimana menganalisa perbandingan dari hasil video coding antara PC berbasiskan Linux, PS dan PL secara independen dan melalui PS dan PL secara terintegrasi pada Xilinx All Programmable SoC.

1.3 Tujuan

Tujuan dilakukan penelitian adalah sebagai berikut :

1. Untuk mengoptimalisasi pengkodean video 3D menggunakan Xilinx All Programmable SoC melalui Processing System (PS) dan Programmable Logic (PL) secara independen.
2. Untuk mengoptimalisasi pengkodean video 3D menggunakan Xilinx All Programmable SoC melalui Processing System (PS) dan Programmable Logic (PL) secara terintegrasi.
3. Perbandingan dari hasil video coding antara PC berbasiskan Linux, PS dan PL secara independen dan melalui PS dan PL secara terintegrasi pada Xilinx All Programmable SoC. Hasil yang diperbandingkan adalah waktu yang dibutuhkan untuk menyelesaikan pengkodean, PSNR yang dihasilkan, dan ukuran file hasil pengkodean yang diperoleh.

1.4 Batasan Masalah

Batasan masalah pada penelitian ini adalah sebagai berikut:

1. Pengkodean video 3D dilakukan pada format video 3D Stereoscopy.
2. Format pengkodean yang dipakai adalah High Efficiency Video Coding (HEVC) atau H.265 sesuai dengan format Joint Collaborative

Team on 3D Video Extension Development (JCT-3V), a joint working group of ISO/IEC MPEG and ITU-T VCEG.

3. Hanya membahas visual coding dan tidak membahas audio coding.

1.5 Kontribusi

Penelitian ini diharapkan dapat memberikan kontribusi berupa perancangan dan optimalisasi dari pengkodean video 3D dengan ukuran yang dihasilkan yang lebih kecil dan tetap mempertahankan kualitas yang sama.

Halaman ini sengaja dikosongkan

BAB 2
KAJIAN PUSTAKA

2.1 Dasar-dasar Video 3D

Perkembangan video 3D telah mengalami kemajuan yang sangat pesat, dan diaplikasikan pada beragam media seperti bioskop, televisi, komputer dan perangkat mobile. 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 stereoscopy merupakan salah satu dari bentuk video 3D disamping tipe video 3D yang lain diantaranya adalah multiview 3D dan Holoscopic. Video 3D stereoscopy bertujuan untuk menciptakan ilusi kedalaman tiga dimensi dari dua buah sumber gambar dua dimensi. Penglihatan manusia, termasuk persepsi kedalaman melibatkan proses yang kompleks dalam otak, yang diawali dengan informasi visual yang diambil melalui mata, kemudian diteruskan ke otak, dan otak akan berusaha memahami informasi yang mentah tersebut. Salah satu proses yang terjadi didalam otak adalah menafsirkan informasi yang disampaikan dari mata kemudian menilai jarak relatif benda dari mata. Pengolahan perkiraan jarak relatif akan dipengaruhi oleh faktor-faktor sebagai berikut :

- Tumpang tindih satu objek dengan yang lain.
- Sudut visual dari sebuah objek ukuran diketahui
- Perspektif linear (Konvergensi tepi paralel)
- Posisi vertikal (objek yang lebih dekat ke cakrawala cenderung dianggap lebih jauh)

- Kontras dan saturasi, benda yang lebih jauh umumnya dikaitkan dengan kontras yang lebih besar dan desaturasi dan pergeseran ke arah biru.
- Perubahan detail tekstur benda.

Dasar stereoscopy video 3D adalah pengambilan gambar dengan menggunakan 2 buah kamera, yang meniru kedua mata manusia, dapat kita bandingkan dengan bila kita menutup mata kiri dan kanan secara bergantian, maka akan menghasilkan penglihatan yang berbeda. Kedua buah kamera penangkap gambar tersebut diletakkan pada jarak yang mirip dengan mata sekitar 4 cm. Streoscopy adalah hasil dari ilusi kedalaman dalam foto, film atau citra dua dimensi lainnya dengan penyajian yang berbeda, yang menambahkan isyarat stereoscopy. Dua gambar yang berbeda secara stereoscopy yang memasuki mata yang berbeda kemudian menggabungkan informasi didalam otak untuk memeberikan persepsi kedalaman. Namun masih ada beberapa kelemahan dari stereoscopy yaitu tidak semua titik di gambar memperoleh fokus yang sesuai dan hanya mengikuti kedalaman adegan asli, maka untuk efek stereoscopy akan menyebabkan persepesi yang kurang wajar.

Video 3D stereoscopy pertama kali ditemukan oleh Sir Charles Wheatsone pada tahun 1838 [4] [5] dan diperbaiki oleh Sir David Brewster yang membuat perangkat tampilan 3D portable pertama [6]. Meskipun telah masuk kategori konten video 3D, namun penyajian gambar 2D ganda sedikit berbeda daripada menampilkan gambar tiga dimensi penuh. Perbedaan yang paling penting adalah bahwa, dalam kasus pergerakan kepala pengamat dan gerakan mata tidak mengubah informasi yang diterima dari objek 3 dimensi yang dilihat. Pada saat ini konten video 3D telah memasuki format blu-ray, HDTV, dan live TV.

Pada video 3D stereoscopy, untuk mendapatkan gambar yang berbeda pada setiap mata digunakan alat bantu kaca mata. Jenis kaca mata yang digunakan pada saat menonton konten video 3D diantaranya adalah kaca mata yang aktif dan pasif seperti tampak pada Tabel 2.1.

Kacamata 3D aktif berinteraksi secara nirkabel dengan gambar pada layar sehingga tiap kaca pada kaca mata akan membuka dan menutup dengan sinkron pada gambar tampilan pada display sehingga citra yang sesuai memasuki mata yang

diinginkan. Sedangkan kacamata pasif tidak memiliki hubungan dengan kontrol display. Kacamata 3D pasif dibagi menjadi dua subkategori utama: anaglyphc dan kacamata terpolarisasi. Pada kacamata terpolarisasi, maka citra yang ditampilkan sebenarnya adalah 2 buah citra yang tumpang tindih dengan polarisasi yang berbeda yaitu horisontal dan vertikal, sehingga masing-masing citra tunggal dapat memasuki mata dengan polarisasi yang tepat, dan dapat memisahkan citra yang memasuki mata. Selanjutnya kacamata anaglyphc menggunakan dua warna yang berbeda yaitu sistem merah/hijau atau merah/biru.

Tabel 2.1 Perbandingan Kacamata 3D aktif dan pasif

	Active 3D	**Passive 3D**
Keuntungan	Resolusi Gambar Full Frame Sudut pandang lebih luas	Kaca mata ringan, murah, tidak perlu baterai
Kerugian	Kacamata relatif berat, mahal, membutuhkan baterai yang perlu sering di-*charge* atau diganti. Kacamata bersifat proprietary, berbeda setiap merek Gambar sedikit redup karena kacamata melakukan pemblokiran cahaya bergantian	Resolusi Gambar Half Frame Sudut pandang lebih terbatas

Kacamata 3D aktif

Pada sistem Active Shutter 3D, dihasilkan dua gambar yang kualitasnya bagus (HD) yang masing-masing gambar diperuntukkan untuk mata kiri dan kanan. Masing-masing gambar ditampilkan secara bergantian dengan kecepatan (kedipan) yang tinggi (min 100 kali per detik). Kedipan kedua gambar tersebut tidak akan terlihat oleh mata kita. (TV yang lama kedipannya 50 - 100 Hz, tidak terlihat oleh

mata). Sehingga untuk tiap-tiap mata memperoleh kecepatan sebesar setengah dari kecepatan frame display.

Teknologi kacamata 3D kemudian berkembang dengan menggunakan metode "Stereoscopy", dimana Kacamata 3D yang digunakannya disebut "Shutter Glasses 3D" atau SG 3D. Kacamata ini menggunakan lensa yang dilapisi Liquid Crystal (Kristal Cair) dan berfungsi memisahkan gambar dengan menutup gambar kiri dan kanan yang ditampilkan oleh 3D TV secara bergantian mengikuti kecepatan kedipan yang dihasilkan oleh TV 3D. Kedua gambar yang diterima otak ini akan menghasilkan gambar tunggal yang berkesan 3D.

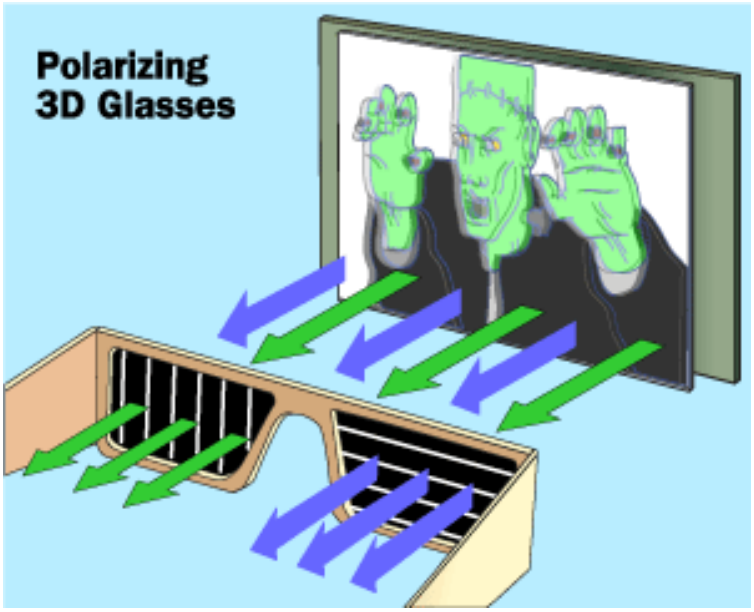
Kedipan pada kacamata ini dioperasikan secara elektronik untuk menerima perintah dari TV 3D, kapan harus terbuka dan kapan harus tertutup sehingga kedipannya selalu sinkron dengan TV 3D. Perintah dari TV 3D dipancarkan secara wireless ke kacamata tersebut. Karenanya sistem ini disebut Shutter 3D aktif, dan kacamatanya lebih mahal karena ada rangkaian elektronik beserta baterainya.

Kacamata 3D Sistem Polarisasi

Bentuk kaca mata lain yang disukai dan lebih populer adalah dengan menggunakan lensa terpolarisasi karena memungkinkan melihat warna secara jelas dan tanpa menggunakan baterai. Dua proyektor disinkronkan pada proyek dua pandangan masing-masing ke layar, masing-masing dengan polarisasi yang berbeda. Kacamata hanya mengizinkan salah satu gambar ke setiap mata karena mengandung lensa dengan polarisasi yang berbeda seperti tampak pada Gambar 2.1.

Kacamata terpolarisasi pasif beroperasi atas dasar yang sama seperti kacamata anaglyph, hanya saja kacamata ini lebih kepada menyaring gelombang cahaya daripada warna. Satu lagi, dua gambar yang identik dan sedikit tumpang tindih, kecuali dalam hal ini setiap gambar terpolarisasi untuk memproyeksikan cahaya yang berbeda dari yang lain. Dengan kacamata 3D terpolarisasi, setiap mata hanya memproses satu gambar. sehingga pikiran kita tertipu untuk memadukan dua gambar menjadi satu, menciptakan pengalaman menakjubkan 3D. Berbeda dengan 3D anaglyphic, yang dapat diproyeksikan dari layar manapun, 3D polarisasi bekerja

lebih baik dengan layar yang dapat menyampaikan frekuensi cahaya berbeda tanpa mengorbankan kualitas gambar.



Gambar 2.1 Bentuk cahaya terpolarisasi pada video 3D

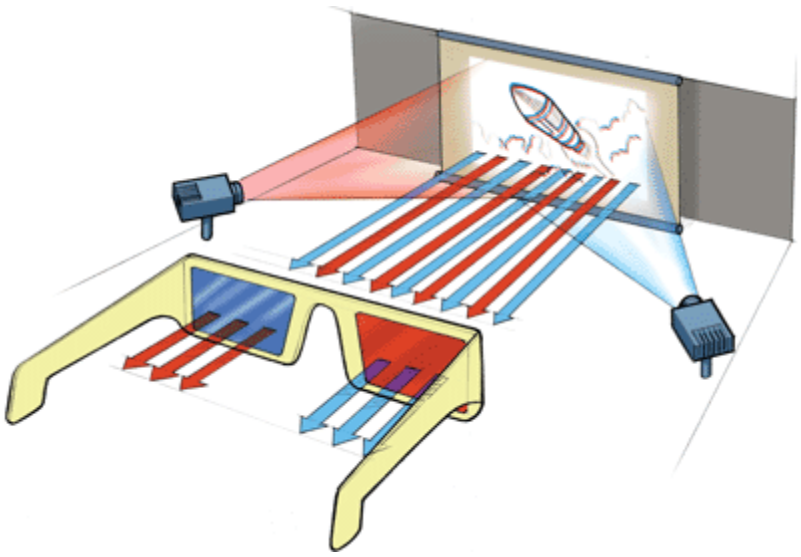
Kaca mata Sistem Warna (Anaglyph): Merah / Hijau atau Merah / Biru

Sistem merah / hijau atau merah / biru sekarang terutama digunakan untuk video efek 3D. Dalam sistem ini, dua gambar yang ditampilkan pada layar, satu merah dan lainnya dengan warna biru (atau hijau). Filter pada kacamata hanya mengizinkan satu gambar untuk masuk ke setiap mata, dan otak kita melakukan sisanya seperti tampak pada Gambar 2.2.

Di layar, dua gambar didominasi merah dan hijau/biru diproyeksikan dengan menggunakan proyektor tunggal maupun ganda. Penonton diberi kacamata 3D dengan satu lensa merah dan biru atau hijau lainnya tergantung pada warna film. Bagian merah dari gambar terhalang oleh lensa hijau dan sebaliknya seperti tampak pada Gambar 2.3. Ini memungkinkan dua retina untuk membentuk dua gambar yang berbeda dan karenanya ilusi kedalaman optik diciptakan.

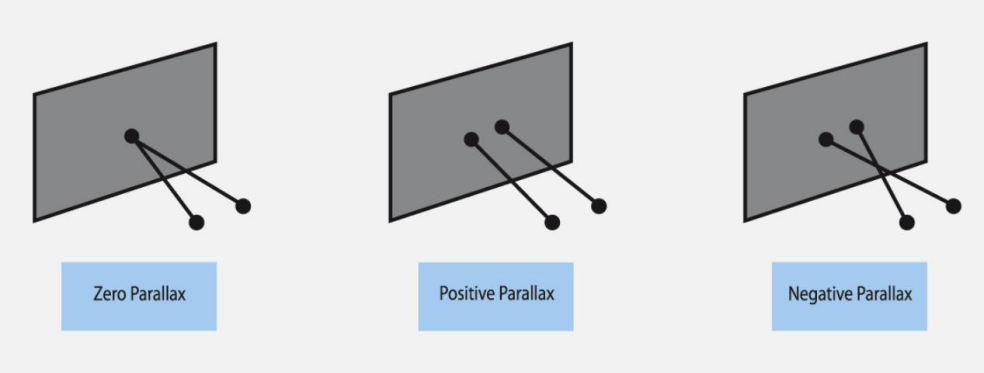
Gambar 2.2 Bentuk kacamata Anaglyph merah/biru

Namun, warna penyaringan oleh lensa terdistorsi warna akhir karena pengaruh penyaringan warna dari kacamata. Kualitas gambar juga rendah tidak sebagus sistem Polarisasi yang paling umum digunakan pada saat ini.



Gambar 2.3 Bentuk cahaya merah biru memasuki hanya di salah satu sisi kaca mata

Jika mata kiri melihat objek di layar atau monitor agak ke bagian kiri dari mata kanan, maka hasilnya adalah paralaks positif yang menyebabkan objek seakan terlihat lebih kedalam dari layar sebenarnya (latar belakang). Jika pandangan saling bersilang didepan monitor maka hasilnya adalah paralaks negatif, yang menyebabkan objek muncul di depan layar atau monitor seperti tampak pada Gambar 2.4. Video Stereo 3D dengan perkataan lain adalah bukan sebuah proyeksi ruang melainkan hanya ruang ilusi. Pada kenyataannya, gambarnya adalah gambar 2D di layar atau di monitor selanjutnya persepsi kedalamannya muncul semata-mata didalam otak kita.



Gambar 2.4 Tampilan stereoscopics yang menghasilkan pandangan berbeda dari mata yang disebabkan oleh parallax, yang dipersepsikan oleh otak menciptakan depth perception

Hal ini mengarah pada situasi persepsi yang tidak normal, mata selalu fokus pada layar, tapi perhatian pemirsa berpisah berdasarakan ketajaman dan kemampuannya, sehingga dapat menciptakan persepsi ke arah depan atau ke arah belakang layar. Untuk menyamakan stereo 3D dengan persepsi realitas Stereo 3D adalah seperti melihat dinding melalui jendela yang terbuka ke luar, atau seperti melihat sebuah akuarium. Bagaimanapun juga kenikmatan seni ini senantiasa tetap memerlukan filling in the gaps dari penonton. Seperti visualisasi yang dirasakan juga pada saat mendengarkan musik, ini adalah integrasi aktif dan perbandingan

dari pengalaman sesungguhnya dalam seni. Bentuk frame dari video 3D stereoscopics SBS (side by side) tampak seperti Gambar 2.5.

Left View Right View

Gambar 2.5 Video 3D stereoscopics Side by Side

2.2 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)[7]. 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[1]. 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.

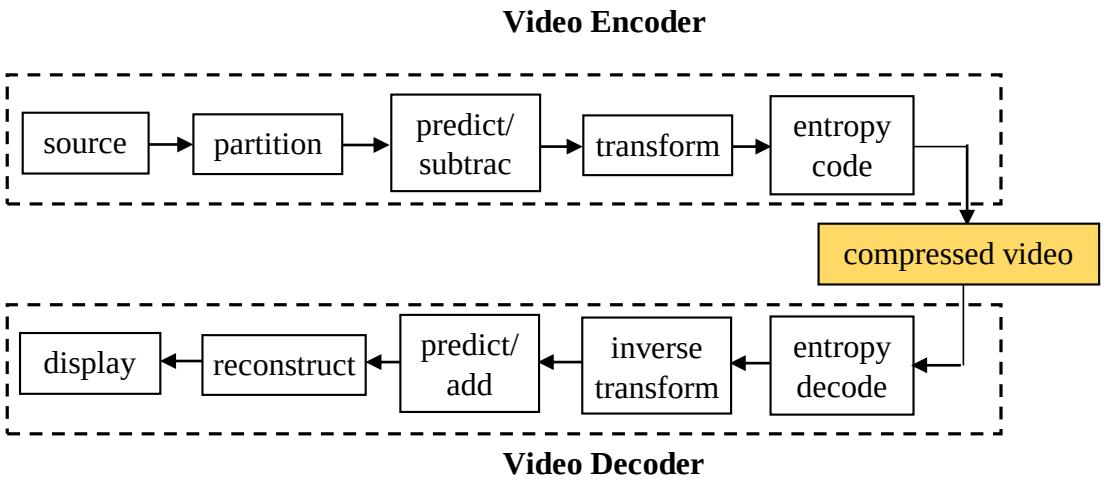
2.2.1 Metode Kompresi Video

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 2.6. Langkah umum pemrosesan video seperti pada Gambar 2.6 dapat dijelaskan sebagai berikut : Sumber video yang sama sekali belum terkompresi diproses dengan proses video encoder melalui proses partition selanjutnya predict/substract dan akan ditransform dan akhirnya entropy code untuk menghasilkan vidoe yang terkompresi. Untuk proses video decoder seluruh proses sebelumnya akan dibalik, yaitu diawali dengan entropy code, invers transform, kemudian predict/add dan

akhirnya reconstruct untuk memperoleh video seperti bentuk pada source untuk ditampilkan pada display.

Dalam pembuatan standar code, algoritma decoder harus telah didefinisikan dengan jelas, karena ruang lingkup dari standar, pada umumnya didasarkan pada decoder [7].



Gambar 2.6 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 [8]:

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.

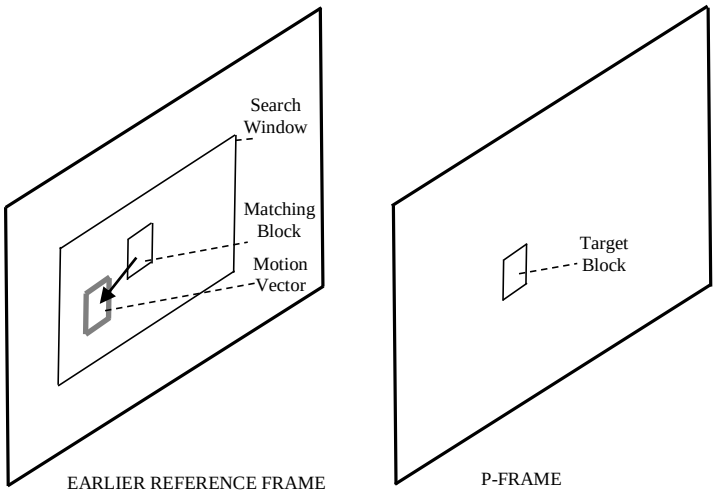
c. Mengeksploitasi hubungan temporal yang ada antara blok dengan frame berikutnya sehingga hanya perubahan antara frame yang dikodekan. Hal ini dapat dicapai dengan menggunakan vektor gerak estimasi yang memprediksi komposisi dari blok sasaran.
d. Mengidentifikasi dan meningkatkan kompresi dari setiap 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 [9]. 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 [10]. 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 [8].

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 [10]. 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.*

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 [11]. 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.



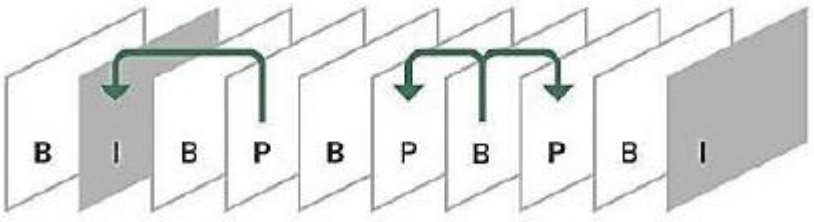
Gambar 2.7 Tahapan enkoding dalam Block-based motion compensation

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 [8] seperti tampak pada Gambar 2.7.

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 [8]. 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.

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 [12].



Gambar 2.8 Langkah kode dengan I-, B-, and P-frame

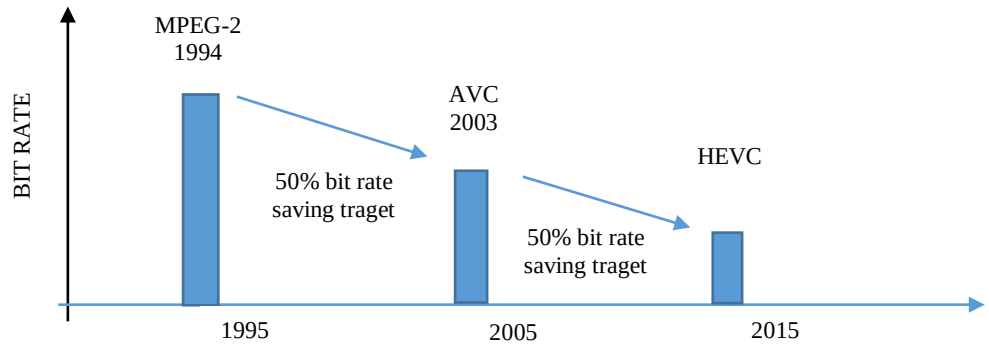
2.2.2 Pengembangan HEVC

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 [11]:

- Memberikan pengurangan bit rate sampai dengan 50% namun mempertahankan kualitas video yang tetap dibandingkan dengan H.264
- Memberikan kualitas yang lebih baik pada bit rate yang sama
- Menetapkan sintaks standar untuk menyederhanakan implementasi dan memaksimalkan interoperabilitas
- 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. Informasi lengkap mengenai pengembangan standar HEVC dapat dilihat pada [12].



Gambar 2.9 Perkembangan bit rate pada tiap-tiap standarisasi video coding

2.2.3 Dampak keuntungan implementasi HEVC [7]

Di dalam pemasaran streaming mobile, pengurangan bit rate oleh standar HEVC dengan kisaran 30 - 50% namun tetap mencapai kualitas yang sebanding dengan H.264 sehingga dapat dicapai penghematan biaya pengiriman pada seluruh jaringan. Operator seluler tidak lagi membutuhkan pengiriman data yang lebih banyak untuk tingkat kualitas yang ditentukan, sehingga juga hanya akan membutuhkan biaya yang lebih rendah dan lebih handal, dengan mengasumsikan masing-masing perangkat hardware mampu mengeksekusi sandi HEVC.

HEVC juga sekaligus mendukung perubahan pada sistem broadcasting high-resolution Ultra HD 4K dan video 8K mainstream. Dengan fitur resolusi 4K mampu menampilkan empat kali jumlah piksel seperti pada 1080p, dan efisiensi yang disediakan oleh HEVC membuat penyiaran 4K jauh lebih ekonomis dan kemampuan yang juga lebih baik.

Perusahaan-perusahaan media yang memiliki konten-konten video berukuran besar juga akan merasakan dampak positif dari untuk mengurangi kebutuhan penyimpanan bit rate. Sejalan dengan perkembangan konten video dalam permintaan user yang semakin kompleks, sehingga perusahaan-perusahaan wajib meningkatkan pula infrastruktur yang mereka miliki. Namun dengan

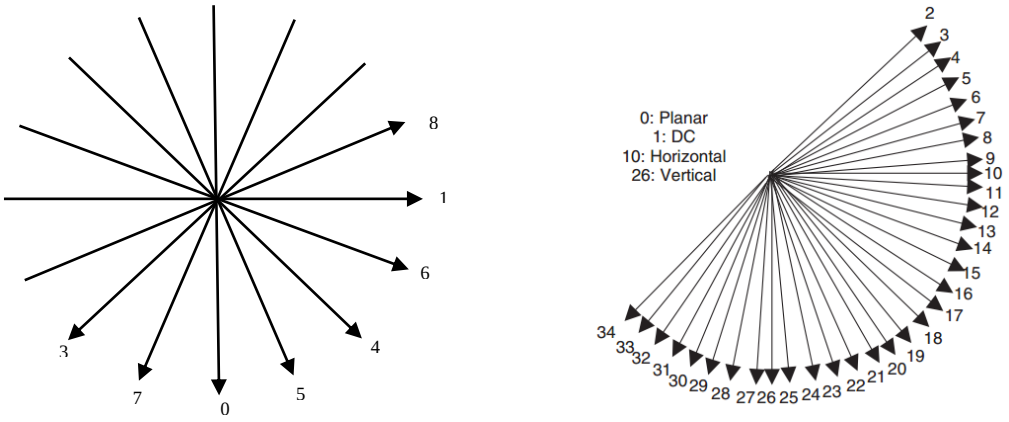
standarisasi HEVC dapat mengurangi hingga separuh dari ukuran file semula, transisi ke pengkodean yang baru ini akan memberi ruang kapasitas penyimpanan dua kali lebih banyak ke masa-masa mendatang.

2.2.4 Keunggulan-keunggulan HEVC

Tujuan utama dari standar HEVC adalah untuk mengkompresi video sehingga dalam mengirimkan bit video tersebut dapat dikirmkan dalam jumlah terkecil dari informasi yang diperlukan namun tetap mempertahankan tingkat kualitas video. Pendekatan yang mendasar untuk HEVC sangat mirip dengan generasi sebelumnya yaitu standar MPEG-2 dan H.264. Hampir dasar proses keseluruhan dari standar kompresi video tersebut adalah sama. Meskipun ada sejumlah perbedaan antara H.264 dan HEVC, antara lain : peningkatan mode untuk intra prediction dan memperkecil partisi dalam frame inter prediction.

Intra Prediction dan pengkodeannnya [8]

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 error 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 seperti tampak pada Gambar 2.10. 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.



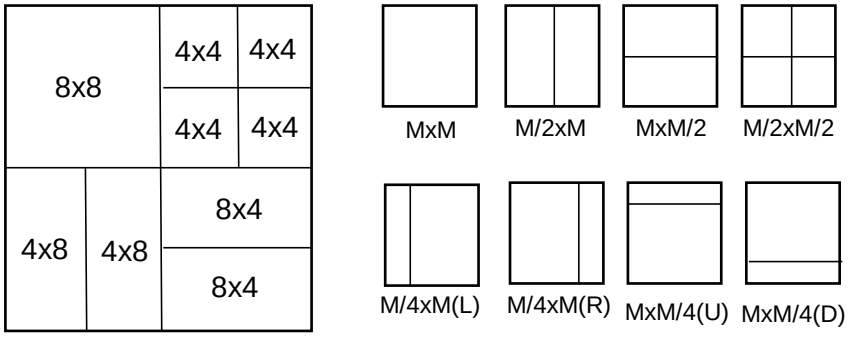
Gambar 2.10 Perbandingan mode *intra prediction pada H.264 dan* HEVC

Inter Prediction and Pengkodeannya [8]

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

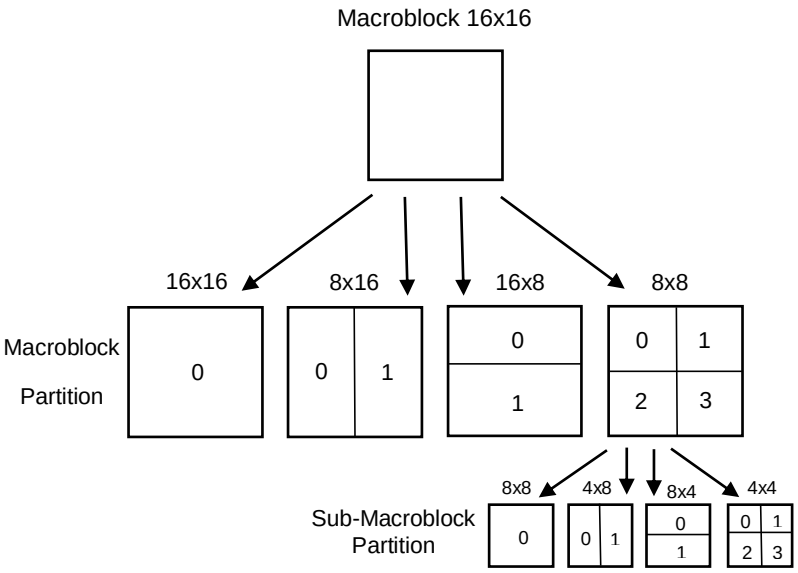
kecil pada saat prediksi yang lebih detail diperlukan, sehingga diperoleh efisiensi yang tinggi.



(a) (b)

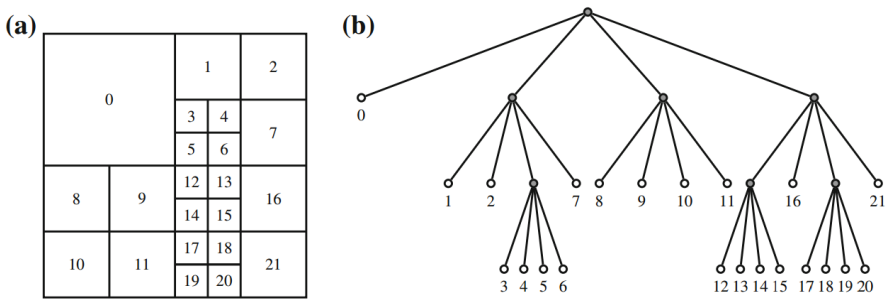
Gambar 2.11 Bentuk dari makroblok (a) 16x16 sampai dengan 4x4 standar H.264 dibandingkan dengan (b) partisi pada HEVC sampai dengan 64x64

Pada standar H.264 inter-prediction digunakan untuk memanfaatkan redundansi temporal dalam sebuah urutan video. Korelasi temporal dikurangi dengan inter prediction dengan menentukan motion estimation dan algoritma kompensasi. Sebuah gambar dibagi menjadi macroblocks; masing-masing macroblock 16x16 dipartisi lebih lanjut menjadi blok berukuran 16x16; 16x8; 8x16; 8x8 seperti tampak pada Gambar 2.12. Ukuran blok yang lebih kecil memiliki keuntungan lebih sedikit data residual, namun akan menyebabkan lebih banyak motion vector sehingga menghasilkan data yang lebih banyak.



Gambar 2.12 Partisi makroblok pada standar H.264 pada *inter prediction*

Sementara pada standar HEVC gambar dipartisi menjadi squared coding tree blocks (CTBs) sejumlah $2^n \times 2^n$ sampel, di mana $n \in \{4, 5, 6\}$ ditentukan sesuai dengan profil utama dalam HEVC. Dengan memodifikasi ukuran CTB sampai 64×64, struktur pengkodeannya bisa jadi digunakan untuk resolusi gambar yang lebih besar. CTB adalah akar dari Coding Blocks (CB) dalam pohon quadtree. CB sendiri dapat dipartisi menjadi setengah horisontal dan setengah ukuran vertikal, dan seterusnya seperti tampak pada Gambar 2.13.



Gambar 2.13 Struktur pengkodean *quad tree* pada *coding blocks.* a. Partisi spasial. b. Bentuk quadtree blok yang berkorespondensi

2.2.5 Pengolahan paralel pada HEVC [12]

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

- Tiles
- Filter deblocking in-loop
- 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.

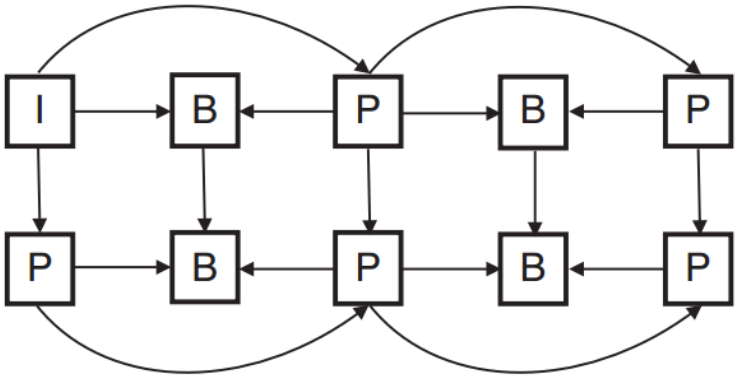
Dalam filter deblocking in-loop, ditetapkan hanya berlaku untuk tepi pada grid 8x8, untuk mengurangi tingkat interaksi antara blok dan menyederhanakan metodologi pemrosesan paralel. Arah pemrosesan ditetapkan lebih dahulu pada filtering horizontal pada tepi vertikal dan diikuti dengan filetering vertikal 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.

2.2.6 Stereoscopy Video 3D Coding [9]

Video 3D coding membutuhkan kompleksitas encoding/decoding yang lebih tinggi daripada 2D coding, karena jumlah frame dua kali lebih banyak yang akan diproses encoded/decoded secara simultan seperti tampak pada Gambar 2.14. Karena video 3D stereoscopy memiliki dua frame secara simultan yaitu kiri dan kanan, maka temporal redudancy (antar frame) dan intraview spatial redudancy (dalam sebuah frame) dan sebagai tambahan dalam video 3D adalah inter-view temporal redudancy antara frame view kiri dan kanan, yang dimanfaatkan untuk memperoleh nilai coding yang optimal. Dalam hal menurunkan kompleksitas

coding, hanya view yang nonbase yang dibuat menjadi inter-view predicted dengan menggunakan base view. Base view dikodekan sama seperti frame video 2D.



Gambar 2.14 Struktur pengkodean video 3D stereoscopy; I:Frame Intra, P:Frame Prediction, B:Frame Bi-prediction

Karena prediksi bersifat adaptif, maka predictor yang paling baik diantara referensi temporal dan inter-view dapat dipilih. Daftar frame referensi dibuat untuk setiap frame untuk dikodekan pada view yang ditentukan. Setiap daftar diinisialisasi sebagai video 2D yang mana terdiri dari frame referensi temporal untuk menentukan prediksi frame pada saat tersebut. [9]

Pengkodean video 3D dengan HEVC dikembangkan untuk format video 3D dengan depth coding, mulai dari video 3D stereo konvensional (CSV) sampai dengan video 3D multi-view dengan depth coding (MVD) dengan dua atau lebih tampilan dan ditambah dengan komponen data kedalaman per piksel. Skalabilitas formatnya dicapai dengan mengkodekan setiap tampilan video dan peta kedalaman pada koordinat yang bersesuaian. Pengkodean yang dilaksanakan adalah prediksi spasial dalam sebuah gambar, prediksi gerak temporal antara gambar pada waktu yang berbeda, dan entropy coding[9].

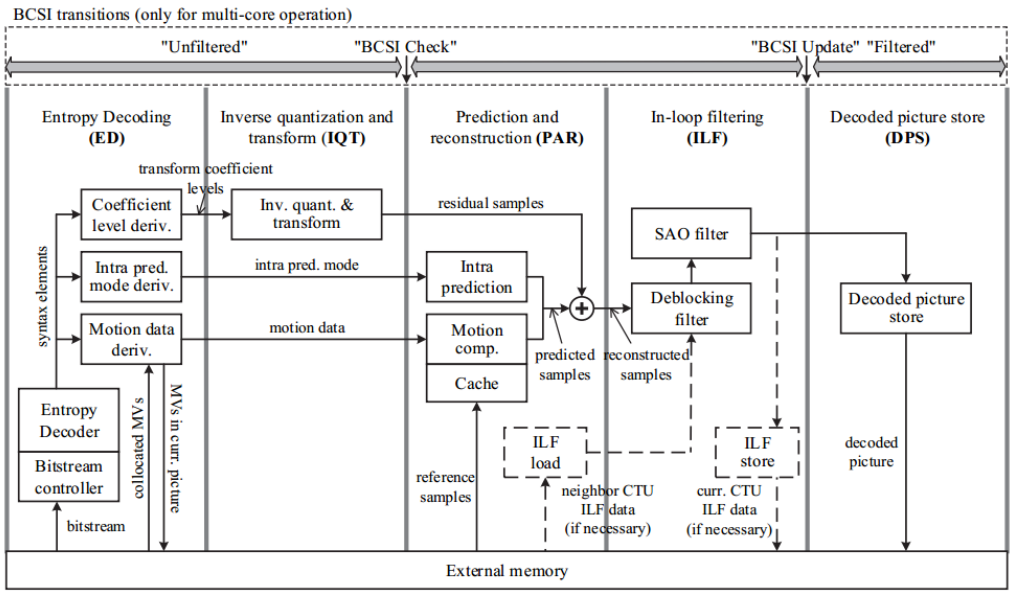
Salah satu aspek terpenting untuk efisiensi pengkodean multi-view dengan depth coding adalah reduksi redundansi diantara view yang berbeda pada saat yang bersamaan, dimana kontennya biasanya agak mirip dan hanya bervariasi dengan

sedikit berbeda. Karena pada saat pembuatan atau pengambilan gambar video 3D tersebut, posisi kamera hanya berbeda posisi sesuai dengan jarak mata manusia sehingga dapat dimanfaatkan kemiripan antara dua view tersebut untuk menambah efisiensi dari pengkodean video 3D tersebut.

2.2.7 Pengolahan HEVC dengan FPGA

Dalam penelitian ini, pengolahan HEVC akan diimplementasikan kedalam FPGA yang memiliki kemampuan eksploitasi proses paralel untuk pengkodean video dengan standar HEVC dengan memanfaatkan fitur-fiturnya yaitu Coding Tree Unit (CTU)-level pipelined architecture dengan single processor maupun multiprocessor, dengan waktu latensi yang kecil. Paralel processing pada arsitektur hardware HEVC decoder sangat efektif untuk pengkodean video UHD dimana jumlah input data yang cukup besar akan membutuhkan arsitektur pemrosesan paralel dalam berbagai aplikasi realtime. Untuk menunjukkan keefektifan arsitektur decoder, decoder HEVC diimplementasikan pada dual-core papan prototipe FPGA yang tersedia untuk demonstrasi publik. Decoder desain diperkirakan mampu melakukan real-time decoding dntuk bitstream 4K / 8K-UHD saat diimplementasikan pada a FPGA SoC [13].

Salah satu ciri paling menonjol dari HEVC adalah struktur pengkodean quadtree, dimana sebuah Coding Tree Unit sesuai dengan array 64 × 64 piksel pada struktur sintaksis hierarkis digunakan untuk mendekodekan piksel. Sebuah gambar atau satu partisi gambar dipartisi menjadi CTU. Jika unit pengolahan blok (PUB) untuk pipeline adalah sebuah Coding unit (CU), sinkronisasi untuk decoder pipeline processing menjadi sulit karena CUs didalam sebuah CTU dapat dipartisi kedalam ukuran variabel dari 8 × 8 sampai dengan 64 × 64, tergantung pada karakteristik sinyal input yang berurutan. Selanjutnya, karena unit prediksi (PU) dan transform unit (TU) di HEVC juga bisa memiliki ukuran variabel dan memiliki ketergantungan data, PU dan TU pada model pipeline akan menyebabkan PUB menjadi lebih rumit. Oleh karena itu, struktur pipelined CTU lebih banyak dipakai untuk implementasi decoder HEVC di FPGA seperti tampak pada Gambar 2.15.



Gambar 2.15 CTU-level pipeline structure untuk single-core and multicore FPGA pada perangkat keras dekoder HEVC [13]

Gambar 2.15 menggambarkan struktur CTU-level pipeline untuk single core dan multi core pada HEVC hardware encoder. Pipeline memiliki lima tahap: entropy decoding, inverse quantization dan transformasi, prediksi dan rekonstruksi, in-loop filtering, dan decode picture store. Pada gambar 2.15, unit fungsional dari proses decoding HEVC digambarkan dalam kotak dan aliran data yag diitunjukkan antara blok unit fungsional dengan tanda panah dimana input data ke tahap pipeline berasal dari tahap sebelumnya atau tahap awal. Modul motion compensation (MC) yang terdiri dari filter cache dan interpolasi memiliki ukuran maksimalnya adalah 16 × 16 sampai dengan dengan 8 × 8.

2.3 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.

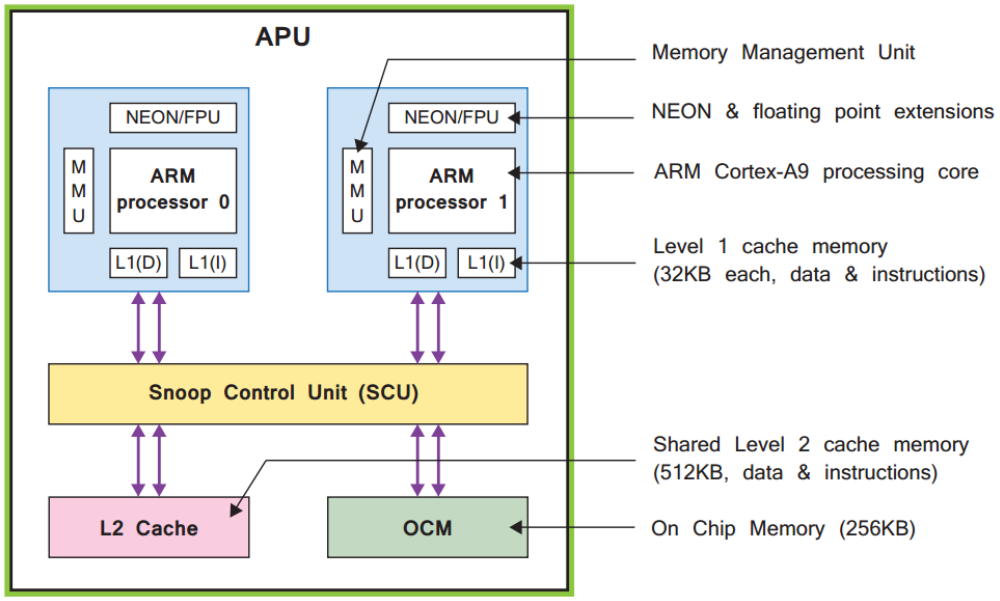
Xilinx menyebut perangkat baru mereka dengan nama Zynq, karena diharapkan perangkat ini dapat menjadi elemen pemrosesan yang dapat diterapkan untuk apa pun seperti salah satu unsur kimia Zynq logam yang bersifat fleksibel dan dapat membentuk ikatan dengan berbagai logam lain untuk membentuk paduan dengan 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-7000 Technical Reference Manual [14].

2.3.1 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 [14]. Sebuah diagram blok yang menunjukkan arsitektur PS ditunjukkan pada Gambar 2.16.

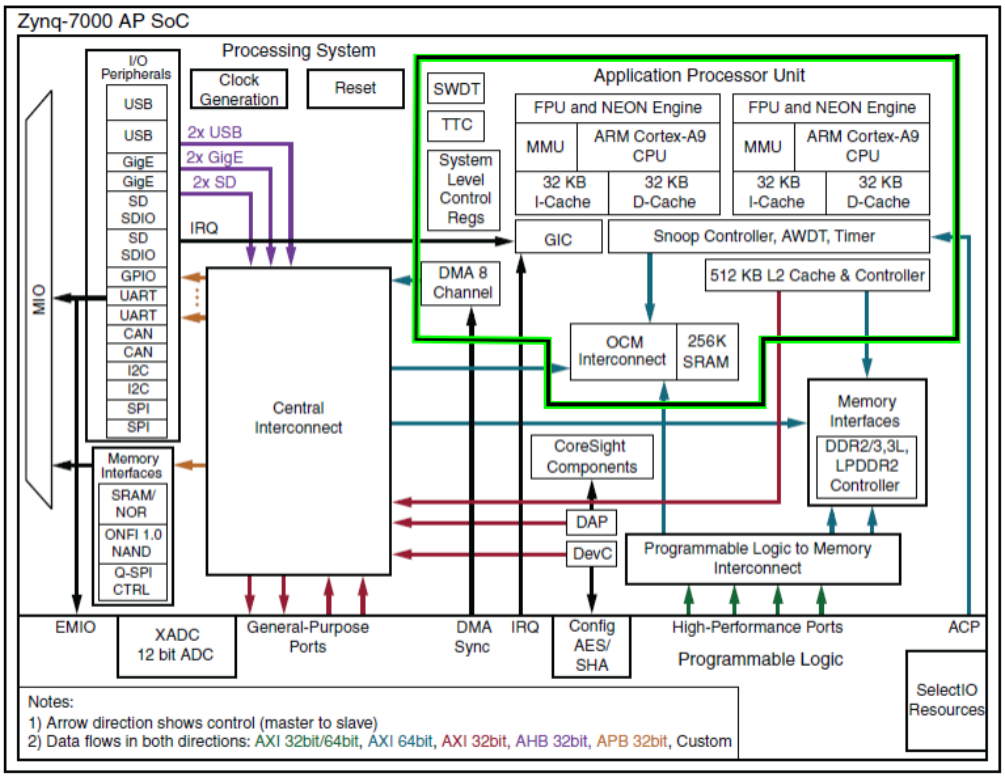


Gambar 2.16 Blok Diagram Aplication Processing Unit (APU)

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 berikut.



Gambar 2.17 Processing System Unit Zynq

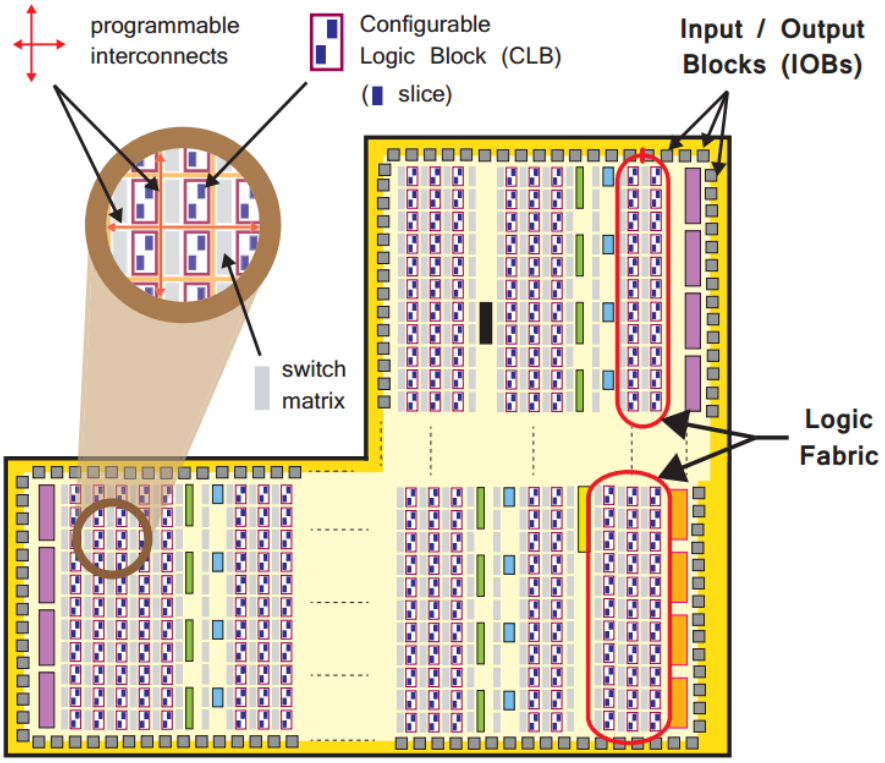
Komunikasi antara PS dan interface eksternal terhubung melalui Multiplexed Input / Output (MIO). Koneksi juga dapat dilakukan melalui Extended MIO (EMIO), yang tidak terhubung langsung dari PS ke koneksi eksternal, melainkan melewati dan I/O resources yang berbagi pakai dengan PL.

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-7000 Technical Reference Manual [14].

2.3.2 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 2.18.



Gambar 2.18 Bagan dari PL perangkat Zynq

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

- 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.

- 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.
- 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.
- Flip-flop (FF); Unit rangkaian sekuensial yang mengimplementasikan register 1-bit, dengan sebuah fungsi reset. Salah satu FFS secara opsional dapat digunakan sebagai latch.
- 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.
- 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.
- 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.

2.3.3 Processing System – Programmable Logic Interfaces

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. Informasi yang lebih lanjut dapat ditemukan di [2].

2.3.4 Perbandingan: Zynq dengan Processor Standard

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 [2]. Sesuai dengan data tersebut maka sangat memungkinkan untuk mengimplementasikan berbagai variasi arsitektur program pada perangkat Zynq tersebut.

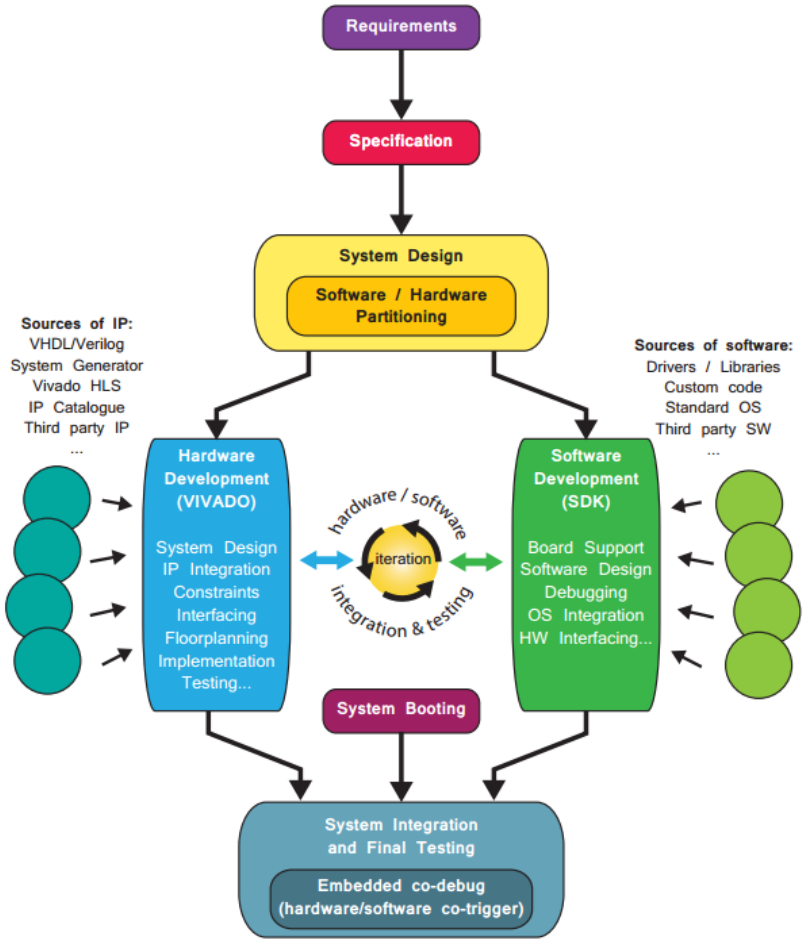
2.4 Pengembangan Zynq System-on-Chip

Pada bagian ini akan dibahas tentang penggambaran pengembangan desain Zynq yang berkonsentrasi pada pengembangan perangkat lunak secara umum. Bagian ini mengeksplorasi konsep penting partisi hardware/software, pengembangan perangkat lunak, dan profilling perangkat Zynq.

2.4.1 Partisi Hardware / Software

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 2.19.



Gambar 2.19 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. Untuk informasi yang lebih detail tentang partisi perangkat keras/lunak dapat dilihat pada [15], [16].

2.4.2 Profiling [7]

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:

- Penggunaan memori
- Waktu Pelaksanaan fungsi panggilan
- Frekuensi fungsi panggilan
- 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.

2.4.3 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 [14].

Xilinx menyediakan fasilitas desain untuk pengembangan dan debugging aplikasi perangkat lunak pada Zynq-7000 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-7000 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). Untuk informasi lebih lanjut dan lengkap dapat dilihat di [17].

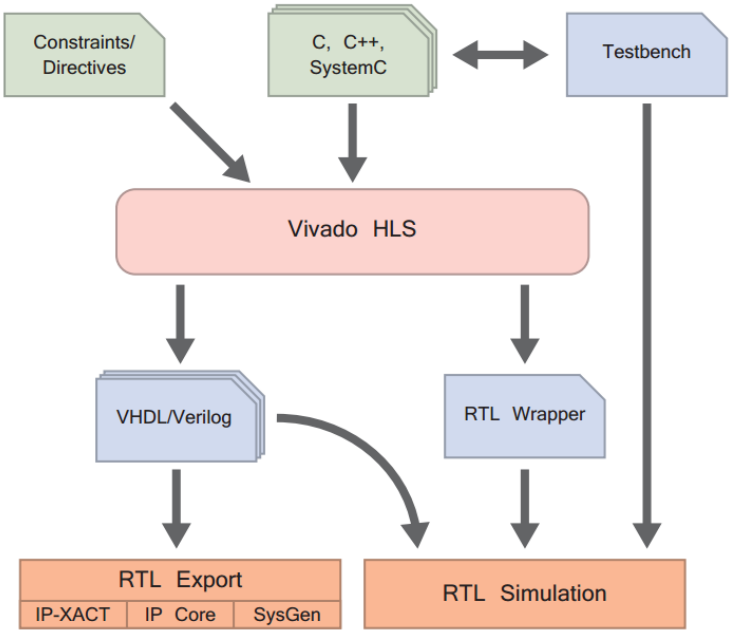
2.4.4 Desain Blok IP

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 [16]. Dalam mendesain IP cores, dapat dilakukan dengan dua jenis cara yaitu : hard IP cores dan soft IP cores. Informasi lebih lanjut dapat dilihat di [7].

2.4.5 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 2.20 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 2.20 Vivado High Level Synthesis (HLS)

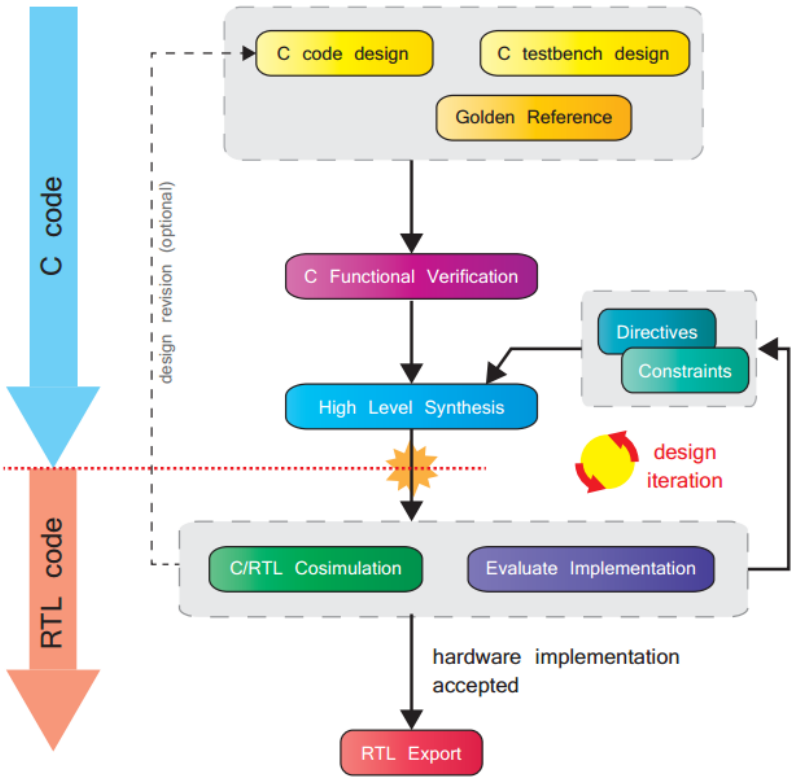
2.4.6 Vivado High Level Synthesis (HLS)

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 [14]. Dalam mengeksekusi HLS, dua aspek utama dari desain dianalisis:

- Antarmuka desain, yaitu yang koneksi top level

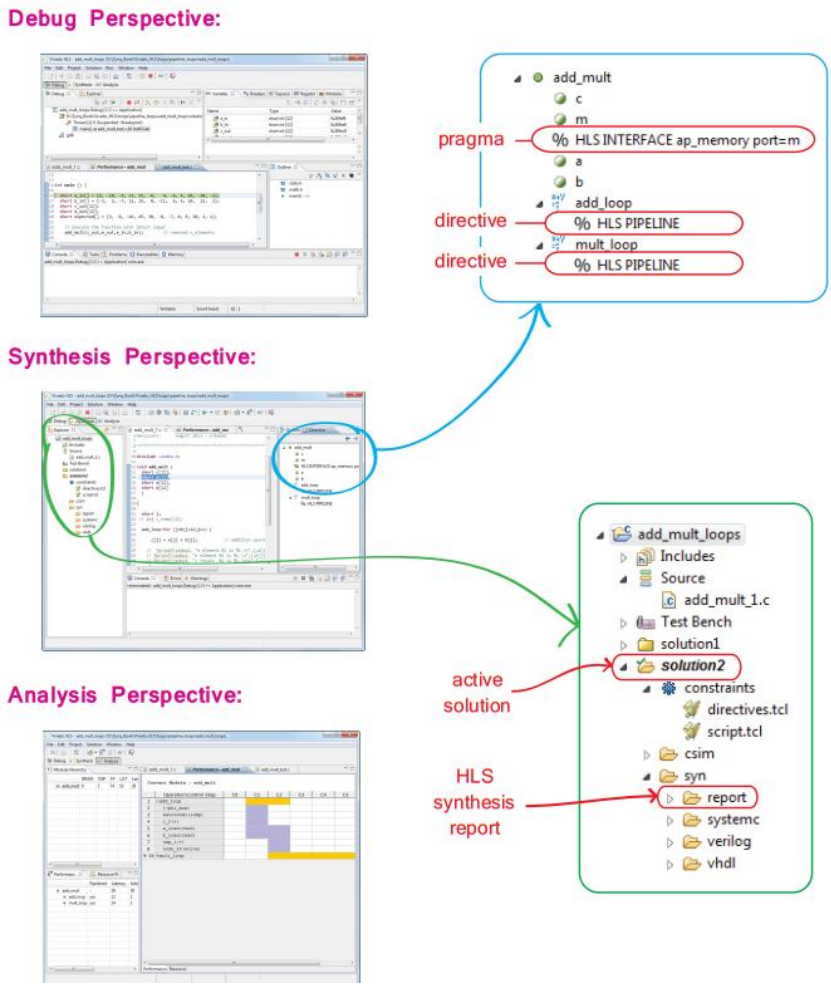
- Fungsi dari sebuah desain, yaitu algoritma yang mengimplementasikannya.

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 Interface. Untuk penjelasan singkat tentang Sintesis Algoritma dan Sintesis Antarmuka dapat dilihat di [15]. Aliran desain pada Vivado HLS seperti tampak dilihat Gambar 2.21. 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, yang masing-masing deskripsi dapat dilihat di [7].



Gambar 2.21 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 2.21 memberikan gambaran tentang Vivado HLS GUI. GUI sebenarnya menyediakan tiga perspektif yang berbeda: Debug, Sintesis, dan Analisis seperti tampak pada Gambar 2.22.



Gambar 2.22 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.

2.4.7 Petalinux

PetaLinux adalah kit pengembangan Embedded Linux System yang secara khusus diciptakan untuk desain berbasis FPGA System-on-Chip. PetaLinux membutuhkan sejumlah alat pengembangan standar dan perpustakaan yang telah terinstal di komputer yang berbasiskan Linux Anda.

Semua perpustakaan yang dibutuhkan oleh PetaLinux berukuran 32 bit. Kebutuhan minimal untuk instalasi PetaLinux Tools adalah sebagai berikut:

° 4 GB RAM (recommended minimum for Xilinx tools)

° Pentium 4 2GHz CPU clock or equivalent (minimum of 4 cores)

° 100 GB free HDD space

° Sistem operasi yang didukung:

- RHEL 6.6/6.7/7.1/7.2 (64-bit)
- CentOS 7.1 (64-bit)
- SUSE Enterprise 12.0 (64-bit)
- Ubuntu 16.04 (64 bit)

2.5 Peak Signal to Noise Ratio

Parameter kualitas video yang direkomendasikan oleh ITU-T adalah standar J.247 [9] 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). Meskipun ada tiga komponen dari perhitungan PSNR (Y,U,V), persepsi visual lebih sensitif terhadap brightness. Oleh karena itu, komponen luminance dari PSNR (Y-PSNR) merupakan parameter yang lebih sering digunakan. Nilai PSNR yang tinggi menunjukkan sinyal dengan noise yang lebih kecil. Nilai PSNR ditentukan oleh MSE dan jumlah bit per sampel dari sinyal video yang akan dikodekan (dinotasikan

sebagai B). Nilai tipikal untuk PSNR pada kompresi video lossy antara 30 dan 50 dB. Diassumsikan dua gambar video I dan K dengan resolusi MxN, maka PSNR yang diperoleh adalah :

$$PSNR = 20log_{10}\left(\frac{2^B-1}{\sqrt{MSE}}\right)\ dB$$

dan : $$MSE\ = \frac{1}{MxN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}|I(i,j)-K(i,j)|^2$$

Pada penelitian ini digunakan pengukuran PSNR average (rata-rata) dari seluruh proses pengkodean video dari awal sampai dengan akhir dari video 3D yang telah dikodekan, dan menghasilkan tiga komponen yaitu PSNR (Y,U,V). Selain parameter PSNR masih ada parameter-parameter lain yang menunjukkan kualitas dari video coding yaitu distortion measure, sum of absolute differences (SAD) dan sum of absolute transformed differences (SATD). [9]

BAB 3
IMPLEMENTASI PENGKODEAN VIDEO 3D PADA FPGA

Pada bab ini membahas mengenai pengimplementasian rancangan kedalam sistem FPGA Xilinx yang diawali dengan penentuan pengaturan awal pada sistem Xilinx, aplikasi pendukung yang dibutuhkan, menguji tingkat kualitas dari software yang berbasiskan HEVC, dalam waktu processing dan ukuran video, selanjutnya software tersebut akan dioptimasi dengan menggunakan software referensi open source sehingga dapat diketahui apabila ada bagian yang masih dapat dioptimasi, selanjutnya menyederhanakan Vivado dengan menggunakan Vivado HLS untuk mendapatkan program yang lebih cepat dan lebih ringkas. Selanjutnya membandingkan alternatif dari design-design yang lebih sesuai dengan sistem untuk memperoleh hasil yang lebih baik. Dan akhirnya implementasi software ke board Xilinx dan mengukur kinerja dari kompresi video yang akan dibandingkan dengan standar-standar generasi sebelumnya.

3.1 Pengaturan awal pada sistem Xilinx.

Dari pihak pengembang Xilinx sudah disebutkan bahwa sistem Xilinx sudah didukung untuk versi Windows dan Linux yang di update secara periodik sehingga dapat digunakan pada hampir semua versi sistem operasi Windows dan Linux. Pada penelitian ini, digunakan Linux Ubuntu 14.04 LTS sebagai sistem operasi. Spesifikasi hardware personal komputer (PC) yang digunakan untuk pembuatan desain pada penelitian ini digunakan paling rendah 4GB RAM (parameter minimal) sedangkan untuk penggunaan maksimum setidaknya memiliki hingga 12 GB RAM. Perangkat hardware yang digunakan dalam penelitian ini adalah PC dengan spesifikasi CPU core i7 Intel dengan RAM 4GB dan menggunakan sistem 64-bit yang sudah memenuhi kebutuhan karena sistem operasi 32-bit tidak mampu untuk disandingkan dengan perangkat Zynq, Perangkat komputer (PC) berkomunikasi dengan board Zynq melalui port USB di PC menuju JTAG (Joint Test Action Group) pada board Zynq, dan juga melalui UART dengan pengaturan awal switch pada papan seperti Tabel 3.1. Koneksi melalui JTAG diperlukan untuk

pemrograman Zynq sementara komunikasi melalui UART dilakukan pada saat debugging.

Tabel 3.1 Default Switch Settings

Switch	Position	Setting
SW10 (JTAG chain input select two-position DIP switch)	1	Off
	2	On
SW12 (two-position DIP switch)	1	Off
	2	Off
SW15 (two position DIP switch)	1	Off
	2	Off
SW16 (five-position DIP switch)	1	Right
	2	Right
	3	Right
	4	Right
	5	Right
SW11 (power slide switch)		Off
	1	Down

Seri board FPGA Xilinx yang digunakan dalam penelitian ini untuk prototyping dan testing hasil desain adalah Xilinx ZC702 Rev 1.0 evaluation board. Papan seri tersebut sudah diperlengkapi dengan fitur-fitur pendukung yang menjadi 1 paket dengan papan FPGA yaitu sumber daya listrik, memori eksternal 8GB high speed, interface untuk pemrograman dan komunikasi, dan juga telah dilengkapi beberapa sarana input output tombol tekan, LED dan switch, dan sejumlah perangkat antarmuka lain serta konektornya. Pada saat pembuatan rancangan sistem yang akan dibuat sampai dengan tahap debugging, desain dirancang dan disusun pada komputer (PC) berbasis sistem operasi Linux dengan menggunakan perangkat lunak pendukung Vivado dan Vivado HLS (High Level Synthesis) kemudian selanjutnya program yang sudah jadi akan didownload ke papan Zynq

menggunakan Test Joint Action Group (JTAG) atau koneksi ethernet, kemudian setelah berhasil di download maka hardware yang telah di program 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 hardware.

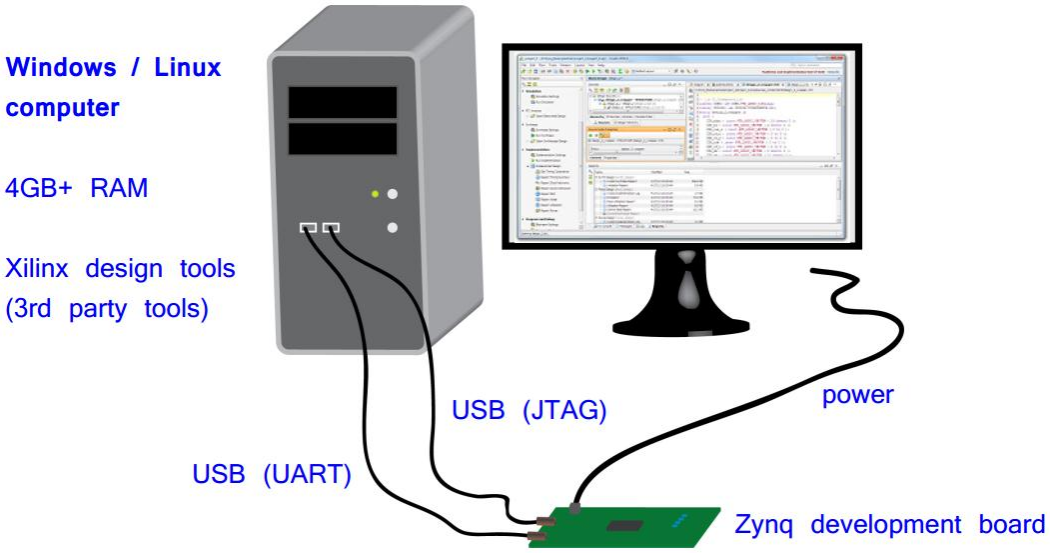
Pada Gambar 3.1 dapat dilihat pengaturan dasar dari koneksi PC dengan papan Zynq untuk memulai pengembangan program yang akan didownload ke papan Zynq.

3.2 Aplikasi Pendukung yang Dibutuhkan

Untuk mengimplementasikan sistem yang sudah dibuat pada board Zynq memerlukan beberapa aplikasi pendukung untuk kebutuhan awal agar program yang telah dibuat dapat dijalankan pada sistem Xilinx. Kebutuhan awal yang diperlukan adalah sebagai berikut :

- Vivado Design Suite (versi yang digunakan adalah 2014.1)
- Vivado HLS (High Level Syntesis) versi yang digunakan 2016.4
- License Management Tools (versi yang digunakan adalah 2014.1)
- Petalinux (versi yang digunakan adalah v2015.4)
- SDK Xilinx – ZC702 (versi v2015-final.bsp)

Selain tools tersebut masih perlu menginstal beberapa tools dari Xilinx untuk menyesuaikan setiap kebutuhan yang berbeda pada perancangan nantinya. Namun bila menggunakan Vivado, diperlukan izin penggunaan dari pihak pengembang untuk dapat digunakan secara penuh semua fitur-fitur didalamnya dalam mendesain kebutuhan board Zynq-7000 SoC.



Gambar 3.1 Penyetelan koneksi pengembangan Zynq [7]

3.3 Perangkat lunak untuk pengkodean video 3D

Perangkat lunak yang digunakan untuk melaksanakan pengkodean video 3D dalam penelitian ini adalah HEVC (High Efficiency Video Coding) atau juga dikenal sebagai standar H.265. Pemilihan perangkat lunak ini akan sangat menentukan hasil yang akan diperoleh nantinya, seperti yang dijelaskan pada bab sebelumnya. Hingga saat ini, beberapa versi perangkat lunak HEVC telah dirilis tetapi pada umumnya perangkat lunak tersebut merupakan produk komersial dimana fitur dan pengembangannya sangat dirahasiakan sehingga tidak memungkinkan untuk digunakan dalam penelitian ini. Oleh karena itu, digunakan perangkat lunak HEVC dengan versi open-source. Di antara beberapa HEVC dengan versi open source adalah HEVC test model (HM), x265, f265, dan Kvazaar. Perangkat lunak HEVC versi open source tersebut sampai saat ini tetap mengalami pengembangan aktif sehingga tetap mengalami proses *up to date*. Untuk versi HEVC test model (HM) sebagai perangkat lunak HEVC referensi memiliki kelebihan yaitu mampu mencapai efisiensi coding yang terbaik di antara perangkat lunak HEVC yang ada, tapi karena programnya diimplementasikan berbasis bahasa C++ object-base sehingga memberikan performansi yang lebih buruk dari yang lainnya. Syntax program HEVC seperti tampak pada Gambar 3.2.

```
#                                               -*- Autoconf -*-
# Process this file with autoconf to produce a configure script.

AC_PREREQ([2.68])
AC_INIT([libde265], [1.0.2], [farin@struktur.de])
AC_CONFIG_SRCDIR([libde265/de265.cc])
AC_CONFIG_HEADERS([config.h])

NUMERIC_VERSION=0x01000200 # Numeric representation of the version
(A.B.C[.D] = 0xAABBCCDD)
AC_SUBST(NUMERIC_VERSION)

LIBDE265_CURRENT=0
LIBDE265_REVISION=10
LIBDE265_AGE=0

AC_CANONICAL_SYSTEM

AC_SUBST(LIBDE265_CURRENT)
AC_SUBST(LIBDE265_REVISION)
AC_SUBST(LIBDE265_AGE)

dnl Initialize libtool
LT_INIT
AC_CONFIG_MACRO_DIR([m4])

# Checks for programs.
AM_PROG_AS
AC_PROG_CXX
AC_PROG_CC
AC_PROG_INSTALL
AC_PROG_LN_S
AC_PROG_GREP

# Initialize automake stuff
AM_INIT_AUTOMAKE

CFLAGS="$CFLAGS -std=c99"
CXXFLAGS="$CXXFLAGS -Werror=return-type -Werror=unused-result -
Werror=reorder"
AX_CXX_COMPILE_STDCXX_11()

CPPFLAGS="$OLD_CPPFLAGS"
AC_LANG_POP(C++)
```

Gambar 3.2 Syntaks program autoconf HEVC dalam bahasa C++

Oleh karena itu, perangkat lunak HEVC versi ini biasanya ditargetkan untuk penelitian dan pengujian kesesuaian hasil sebagai referensi. Versi lainnya yaitu x265 dikembangkan dengan dana komersial didasarkan pada HM C ++ yang telah ditingkatkan dengan optimasi perakitan yang luas dan multithreading yang dapat diaplikasikan secara praktis dan bersifat open-source. Sementara versi f265 adalah versi lain perangkat lunak HEVC kegunaan industri yang dikembangkan dengan mengimplementasikan dalam bahasa C. Meskipun kedua perangkat lunak tersebut dikembangkan dengan dana komersial namun penggunaannya bersifat open-source, namun disamping itu dalam penggunaannya harus menandatangani perjanjian untuk memberikan kontribusi atas hak cipta atas perangkat lunak mereka. Untuk perangkat lunak HEVC versi Kvazaar sama sekali tidak memerlukan perjanjian tambahan untuk berpartisipasi memberikan kontribusi seperti versi sebelumnya. Kvazaar adalah perangkat lunak opensource HEVC yang diperuntukkan untuk kepentingan akademik yang diprakarsai dan dikoordinasikan oleh Ultra Video Group [18].

Oleh karena hal tersebut maka perangkat lunak yang paling sesuai untuk pengembangan akademis dalam penelitian ini adalah HEVC versi Kvazaar yang dengan sifat open source sehingga terbuka untuk pengembangan lebih lanjut dan lebih fleksibel untuk diimpementasikan kedalam papan Zynq-7000 SoC.

3.3.1 Perangkat lunak HEVC versi test mode (HM)

Untuk dipergunakan sebagai referensi perbandingan dapat digunakan perangkat lunak HEVC versi test mode untuk encode dan decode video 3D yang akan digunakan. Perangkat lunak yang bersifat referensi ini akan dipakai untuk menunjukkan kemampuan standar dari pengkodean HEVC. Karena perangkat lunak ini dikembangkan berdasarkan bahasa C++ maka sebelum dapat menggunakan perangkat lunak ini, harus sudah diinstal aplikasi pendukung untuk dapat menjalankan dan mengkompilasi file proyek yang akan digunakan. Aplikasi pendukung dapat diunduh secara online untuk pengembangan environtment dalam bahasa C++. Disamping itu dibutuhkan aplikasi untuk konfigurasi dengan kebutuhan papan Zynq 7000, dimana file ini dapat diunduh secara online. Pada penelitian ini digunakan file konfigurasi *encoder_intra_main.cfg*. HM diinstal pada

perangkat PC yang berbasiskan Linux untuk digunakan sebagai perbandingan hasil pengkodean video 3D nantinya.

3.3.2 Perangkat lunak HEVC versi Kvazaar

Kvazaar pada dasarnya dikembangkan dari versi HM terutama sebagai referensi skema encoding dan implementasi algoritma, tetapi telah mengadopsi data dengan tipe yang berbeda dan benar-benar dibuat baru dalam hal struktur fungsi dan percabangannya. Kvazaar dikembangkan berbasiskan bahasa C. Oleh karena itu sangat mendukung dalam hal akselerasi, portabilitas, dan paralelisasi dan lebih hardware oriented sehingga memudahkan pengembangan program. Perangkat lunak HEVC versi Kvazaar dapat diunduh pada laman GitHub [19]. Kvazaar yang digunakan dalam penelitian ini adalah versi 0.72 lisensi dibawah LGPLv2.1

3.4 Perancangan Sistem pada Zynq ZC702

Pada sistem Zynq ZC702 dapat dikembangkan 3 (tiga) jenis desain untuk dapat mengimplementasikan HEVC pada sistem Zynq ZC702 yaitu menerapkan open-source HEVC dalam desain Zynq PS, Zynq PL secara mandiri, dan Zynq PS, Zynq PL secara co-desain. Dari setiap tipe desain yang dikembangkan akan dijelaskan tentang tahapan dan hambatan yang ditemukan dari setiap bagian yang telah dilaksanakan.

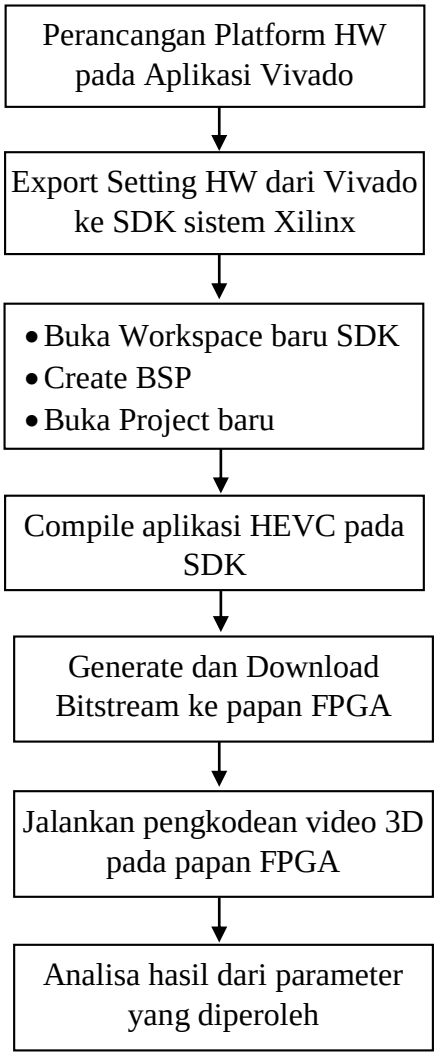
3.4.1 Implementasi Perangkat Lunak HEVC dengan mode PS

Untuk mengimplementasikan perangkat lunak HEVC digunakan tools pendukung Xilinx Vivado Desain Suite untuk nantinya dapat diunduh dan dijalankan di papan Zynq 7000 AP SoC dalam tipe desain Zynq PS (Processing System). Seperti yang dijelaskan sebelumnya, Zynq SoC terdiri dari Dual Processor ARM Cortex-A9, perangkat keras intellectual property (IP) dan programmable logic (PL). Dalam penggunaan tipe desain Zynq PS secara mandiri dapat dibuat dalam dua cara yaitu:

- Zynq SoC PS dapat digunakan dalam mode standalone, tanpa koneksi apapun tambahan dari perangkat keras IP.

- Cara yang kedua adalah perangkat keras IP dapat dipakai dalam koneksi antara Zynq PS sebagai PS+ PL.

Dalam desain tipe ini, HEVC diimplementasikan ke Zynq PS secara mandiri dengan pengembangan awal pada tools Vivado Design Suite yang digunakan untuk pembuatan sebuah proyek yang akan disiapkan sebagai sistem tertanam (embedded system). Gambar 3.3 menunjukkan diagram alir pengembangan perangkat lunak pada SDK sistem Xilinx.



Gambar 3.3 Diagram alir pengembangan aplikasi pada SDK sistem Xilinx [7]

Pada kenyataannya perangkat lunak HEVC versi HM bukanlah tipe yang terbuka untuk dikembangkan, oleh karena itu tidak dapat secara langsung dijalankan pada tools Vivado dan Xilinx SDK. Untuk dapat dijalankan pada Xilinx SDK dapat dilakukan dengan mengembangkan proyek yang ada pada Vivado. Program dikembangkan dengan Makefile dari kode sumber yang ada, selanjutnya mengedit hasil dari makefile untuk memperoleh bentuk dan hirarki sesuai dengan yang dibutuhkan. Untuk menentukan perangkat keras yang diikutsertakan dalam proyek platform, ditentukan dalam Vivado, yang ditunjukkan dalam gambaran perangkat keras yang tertanam secara software (dari sumber daya yang tersedia dan seluruh peripheral pada Zynq ZC702). Secara ringkas tahapan dalam membuat dan mengembangkan desain adalah sebagai berikut:

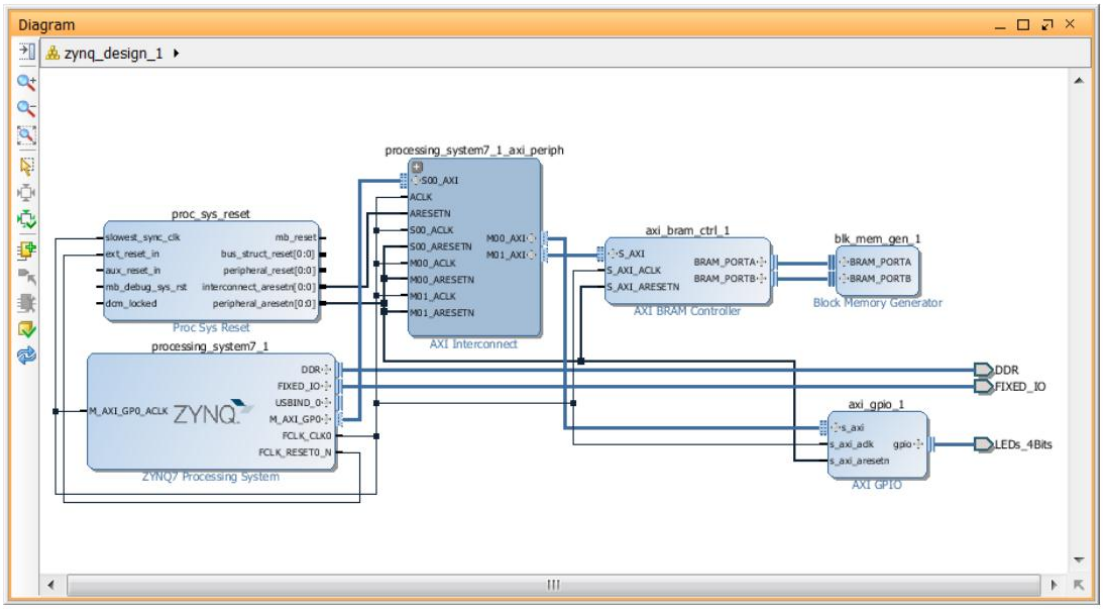
Langkah 1: Membuat proyek baru (Create New Project)

Dalam pembuatan proyek baru di dalam Vivado, akan dideklarasikan seluruh sumber daya hardware pada FPGA yang akan digunakan dan bentuk koneksi antar blok yang diinginkan. Penggunaan tiap blok dipilih sesuai dengan visual yang disediakan sehingga memudahkan perancang untuk mendapatkan bentuk yang diinginkan. Untuk menghubungkan antar blok dapat dilakukan dengan pilihan automatic ataupun ditarik manual. Gambar 3.4 menunjukkan bentuk blok hasil perancangan dari aplikasi Vivado.

Langkah 2: Membuat (Create) Proyek Embedded Processor

Untuk membuat proyek embedded proccessor pada papan ZC702 dapat digunakan fasilitas add sources wizard. Pilihan yang dapat digunakan unuk membuat (create) blok desain wizard dapat dipilih seperti pada pada Tabel 3.1. 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 sesuai dengan yang telah ditambahkan sebelumnya.



Gambar 3.4 Project baru pada Vivado

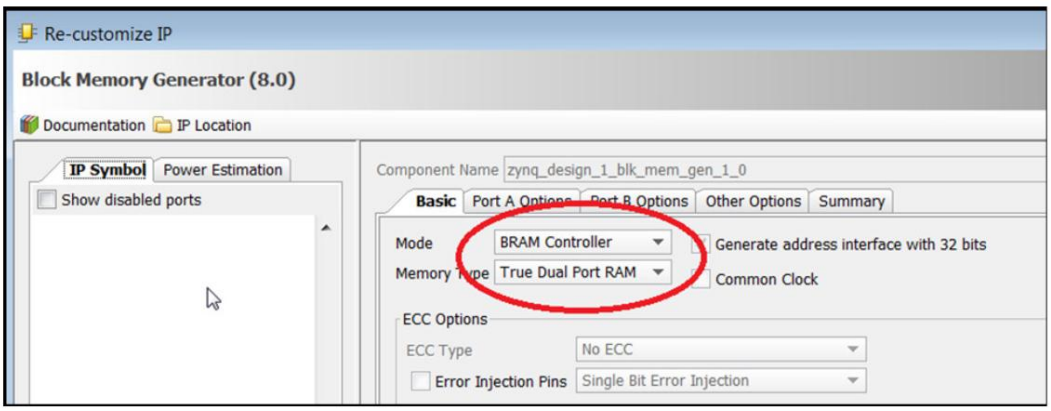
Tabel 3.2 Parameter untuk membentuk (create) block design wizard

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

Langkah 3: Pengaturan Sistem ZC702 Processing di Vivado

Setelah selesai pembuatan blok IP sesuai dengan yang dijelaskan pada langkah sebelumnya maka dilanjutkan dengan pengaturan seluruh parameter blok IP yang akan digunakan pada Zynq7 processing system. Dengan melakukan double klik pada blok Processing system Zynq7 di window blok diagram window, kotak

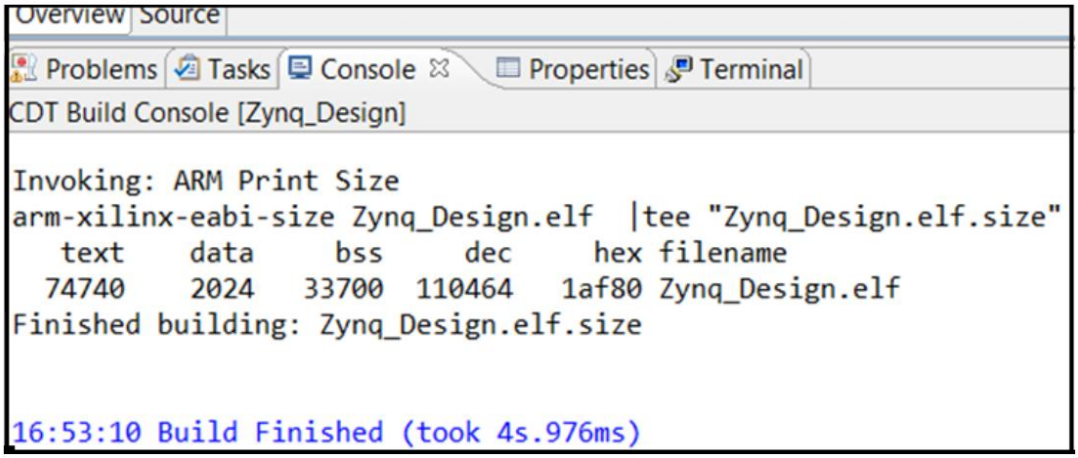
dialog re-costimize akan muncul sebagai sarana untuk mengatur parameter blok processing system seperti tampak pada Gambar 3.5. Secara default, sistem prosesor pada awalnya tidak memiliki periferal yang terhubung (masih blok tunggal secara mandiri dari tiap-tiap blok). Koneksi yang akan terbentuk dilambangkan 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.



Gambar 3.5 Seting mode dan tipe memory pada project baru Vivado

Langkah 4: Memvalidasi Desain dan Menghubungkan antar Port

Desain yang telah dibuat harus di validasi terlebih dahulu untuk memeriksa kesesuaian project yang telah dibuat dengan standar yang berlaku pada Vivado. Untuk memvalidasi (validating) 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 seperti tampak pada Gambar 3.6.



Gambar 3.6 Building file .elf pada SDK sistem Xilinx

Bila sudah tidak terdapat lagi kesalahan (seluruh kesalahan telah diatasi) 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 5: Mengelola desain, menjalankan implementasi, dan membangkitkan bitstream

Untuk dapat diekspor nantinya ke sistem xilinx maka harus dibangkitkan bitstream dari sistem yang telah kita buat. Pada diagram blok view tersedia fasilitas synthesizing the design, running implementation, and generating bitstream, untuk melaksanakan seluruh proses tersebut secara otomatis, namun untuk menyelesaikan seluruh proses tersebut akan memerlukan waktu yang cukup lama sehingga diperoleh bitstream yang dibutuhkan untuk ekspor ke SDK sistem Xilinx. Setelah generasi Bitstream selesai, maka dilakukan ekspor ke SDK sistem Xilinx dengan klik export the hardware and launch the Software Development Kit (SDK) yang secara otomatis akan membuka SDK sistem xilinx saat proses export selesai.

Langkah 6: Mengekspor ke Software Development Kit (SDK)

SDK sistem xilinx seperti dijelaskan pada bagian sebelumnya akan aktif dari perangkat lunak Vivado setelah perintah Ekspor Hardware. Untuk mengikutsertakan bitstream yang telah dibuat sebelumnya maka kotak centang sertakan bitstream harus aktif. Pada saat SDK diluncurkan, file hardware desciption file (file ekstensi .hdf) secara otomatis akan dimuat. Tab system.hdf menunjukkan alamat pemetaan seluruh Sistem Processing secara lengkap. Sampai pada tahap ini, Vivado juga akan mengekspor spesifikasi hardware untuk disesuaikan dengan lingkungan tujuan (xilinx environment). Jika pilihan <local to Project> sebelumnya aktif maka Vivado menciptakan workspace baru di folder proyek Vivado.

Workspace tersebut akan diberi nama sebagai <project_name>.sdk. Dalam penelitian ini, workspace diberi nama adalah /opt/Xilinx/Vivado/.../bin/ edt_tutorial/hal_baru.sdk. Sistem Vivado mengekspor Hardware Platform Spesification dari desain yang telah dibuat (system.hdf) ke SDK sistem xilinx. Selain system.hdf tersebut, maka vivado akan mengekspor file lainnya seperti berikut:

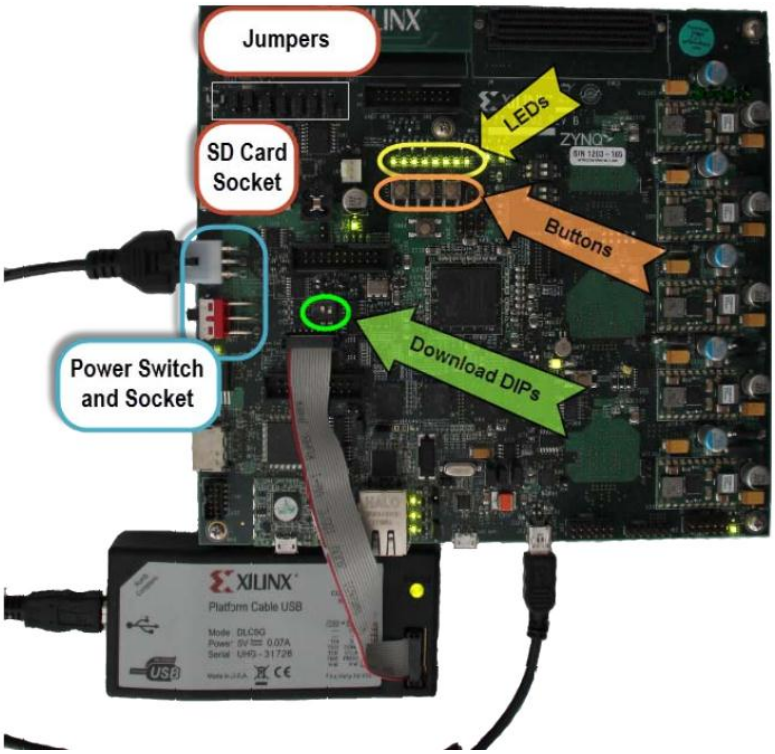
- design_1_bd.tcl
- ps7_init.c
- ps7_init.h
- ps7_init.html
- ps7_init.tcl
- ps7_init_gpl.c
- ps7_init_gpl.h

File tersebut akan dibutuhkan oleh SDK sistem xilinx sebagai kode inisialisasi untuk sistem Zynq SoC Processing system. File system.hdf selanjutnya akan terbuka secara default pada saat SDK sistem xilinx diluncurkan. File system.hdf akan memberika pemetaan alamat yang akan ditampilkan pada SDK window. Kemudian kode inisialisasi akan diterjemahkan dari file-file : ps7_init.c, ps7_init.h, ps7_init_gpl.c, dan file ps7_init_gpl.h untuk mengatur sistem awal Zynq SoC processing sistem dan pengaturan inisialisasi untuk DDR, pewaktu, Phase Locked Loops (PLLs), dan seluruh MIO yang akan digunakan. SDK menggunakan pengaturan inisialisasi ini saat mengawali mengatur sistem processor sehingga aplikasi yang kita buat nantinya dapat berjalan di atas sistem processor. Beberapa pengaturan telah ditetapkan fix pada papan ZC702. Setelah seluruh inisialisasi selesai pengaturan pengembangan sistem akan dilanjutkan berbasiskan SDK sistem Xilinx.

Langkah 7: Menjalankan aplikasi pada papan ZC702

Aplikasi yang telah selesai dibuat seperti tampak pada Gambar 3.6. Aplikasi tersebut selanjutnya akan dijalankan pada papan ZC702, diawali dengan pengaturan papan ZC702, membuat sambungan kabel, untuk menghubungkan papan ZC702 ke

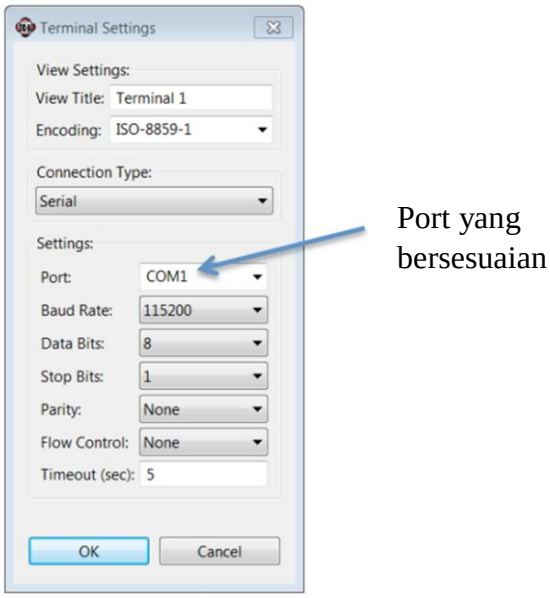
PC yang berbasiskan Linux, dan mengunduh aplikasi yang telah dibuat sebelumya pada SDK sistem Xilinx. Kabel yang disediakan pada papan ZC702 adalah menggunakan kabel Digilent untuk berkomunikasi dengan PC. Untuk mengawali penyalaan papan ZC702, pengaturan SW10 sebagai berikut : bit-1 adalah 0 (beralih terbuka), bit-2 adalah 1 (switch ditutup) untuk mengatur default koneksi papan ZC702 ke PC seperti tampak pada Gambar 3.7. Kabel daya dihubungkan ke papan, kemudian kabel USB dihubungkan ke konektor J17 pada papan target dengan host Linux mesin untuk mentransfer serial. Dan papan ZC702 siap untuk dinyalakan saat seluruhnya koneksi telah sesuai. Pada jendela SDK, dipastikan file /opt/Xilinx/Vivado/.../bin/edt_tutorial/hal_baru.sdk dalam kondisi aktif yang nantinya akan diimplementasikan ke papan ZC702.



Gambar 3.7 Tata letak fungsi pada papan ZC702

Komunikasi antara papan ZC702 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 14.04 LTE 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 papan ZC702, dimana port yang digunakan tidak selalu sama setiap di awal terjadi koneksi ke PC tersebut. Setelah dipastikan port untuk komunikasi serial antara papan ZC702 dan PC telah sesuai maka selanjutnya pengaturan parameter koneksi serial juga harus disesuaikan melalui SDK. Pada terminal SDK, diatur sedemikian rupa untuk menyesuaikan parameter komunikasi serial seperti pada Gambar 3.8.



Gambar 3.8 Tata letak fungsi pada papan ZC702

Untuk mengawali pengembangan sistem yang telah dibuat pada SDK, harus dibuat (create) proyek aplikasi baru untuk tempat mengembangkan sistem nantinya. Pada SDK sistem Xilinx dilengkapi dengan fasilitas wizard untuk memilih proyek aplikasi yang akan kita bentuk dengan pilihan yang sesuai dengan Tabel 3.3. Setelah pembuatan proyek baru selesai maka SDK akan membentuk aplikasi

HM_encoder dan HM_encoder_bsp board support package (BSP) yang tampak pada explorer proyek. Keduanya akan terus dikompilasi secara otomatis dan mebentuk file ELF yang akan dapat diimplementasikan ke papan ZC702.

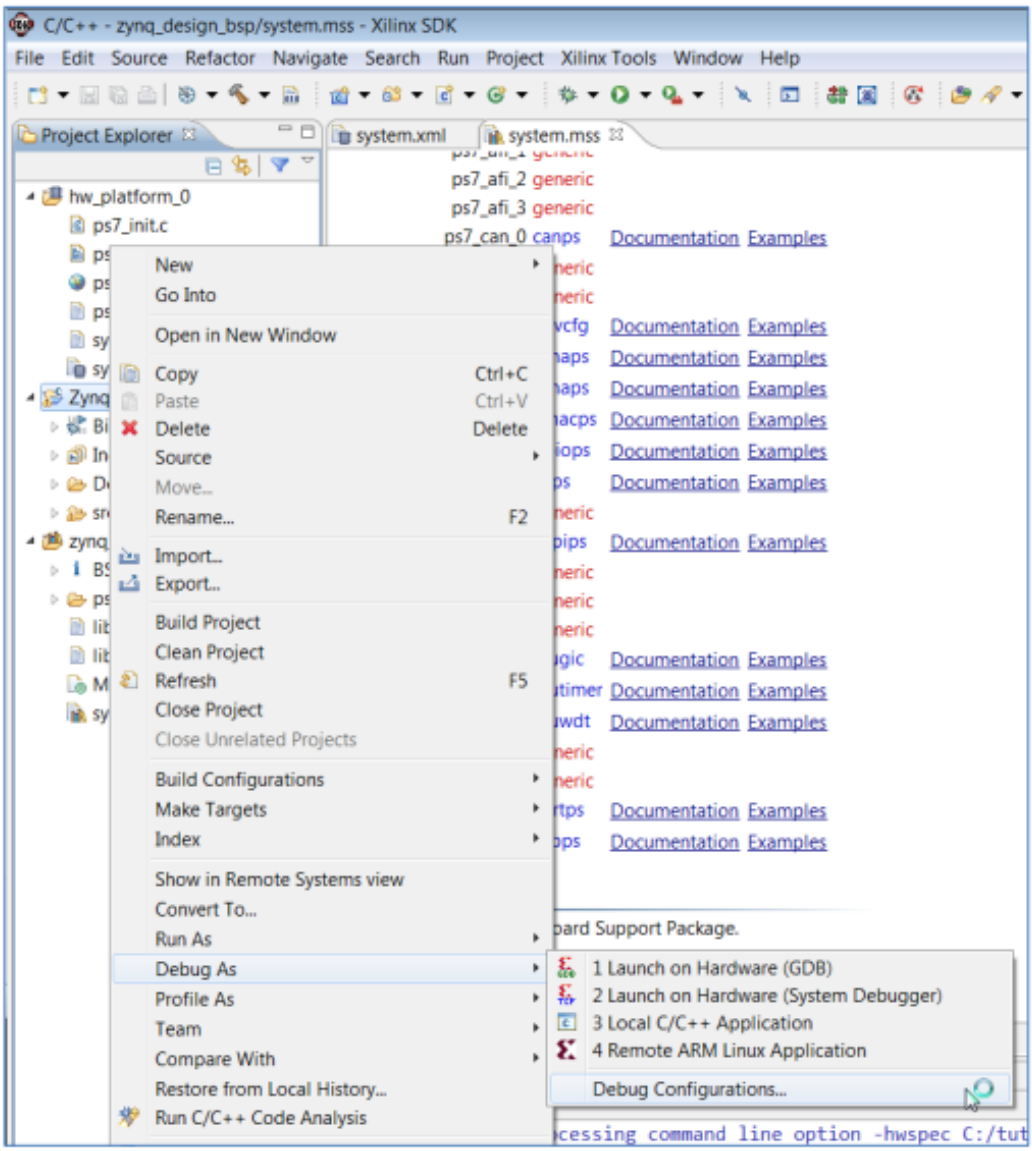
Tabel 3.3 Parameter untuk Membentuk Aplikasi Baru pada SDK Xilinx

Wizard Screen	**System Properties**	**Setting or Command to Use**
Application Project	Project Name	Hal_baru
	Use default location	**Select this option**
	Hardware platform	Hal_baru_wrapper_hw_platform_0
	Processor	**PS7_cortex9_0**
	OS Platform	**Standalone**
	Language	**C++**
	Board Support Package (BSP)	Select **Create New** and provide the name of Hal_baru_bsp
Templates	Available templates	**Empty Application**

Board Support Package (BSP) merupakan kode pendukung pada sistem hardware yang telah digunakan dalam hal inisialisasi dasar pada saat power up (awal penyalaan papan) dan membantu aplikasi perangkat lunak agar dapat berjalan dengan baik pada hardware yang telah ditentukan. BSP merupakan aplikasi yang sangat spesifik (unik) untuk setiap sistem operasi dengan bootloader dan driver perangkat yang berbeda [16].

Bila proyek aplikasi telah dibuat, maka aplikasi tersebut akan diimpor ke sistem SDK untuk disesuaikan pada sistem SDK seperti tampak pada Gambar 3.9. Karena pada dasarnya aplikasi HEVC dibuat sesuai dengan sistem operasi Linux, sehingga pada saat *compile* mengalami beberapa ketidaksesuaian dengan sistem Xilinx yang berbasiskan Petalinux. Open source HM memiliki file library yang sangat banyak yang akan dijalankan pada saat pengkodean video 3D. Bila dijalankan pada PC dengan sistem operasi Linux, install perangkat lunak HEVC dapat dilakukan dengan mudah, hanya dengan menggunakan file makefile yang telah diikutkan dalam aplikasi HEVC tersebut, maka makefile tersebut akan

menjalin seluruh file library yang ada dan selanjutnya akan berjalan secara otomatis sampai seluruh proses selesai dan perangkat lunak HEVC dapat diinstal pada PC berbasis Linux. Berbeda pada SDK sistem Xilinx, file makefile milik perangkat lunak HEVC sebelumnya harus diimpor ke SDK sistem Xilinx. Pada saat import file makefile milik perangkat lunak HEVC selalu terjadi banyak kesalahan, karena menyangkut penanganan file library milik perangkat lunak HEVC yang cukup banyak dan memiliki loop yang rumit, sehingga pesan library yang tidak valid ataupun tidak ditemukan harus ditangani secara bertahap sehingga cukup menyita waktu pengembangan aplikasi.

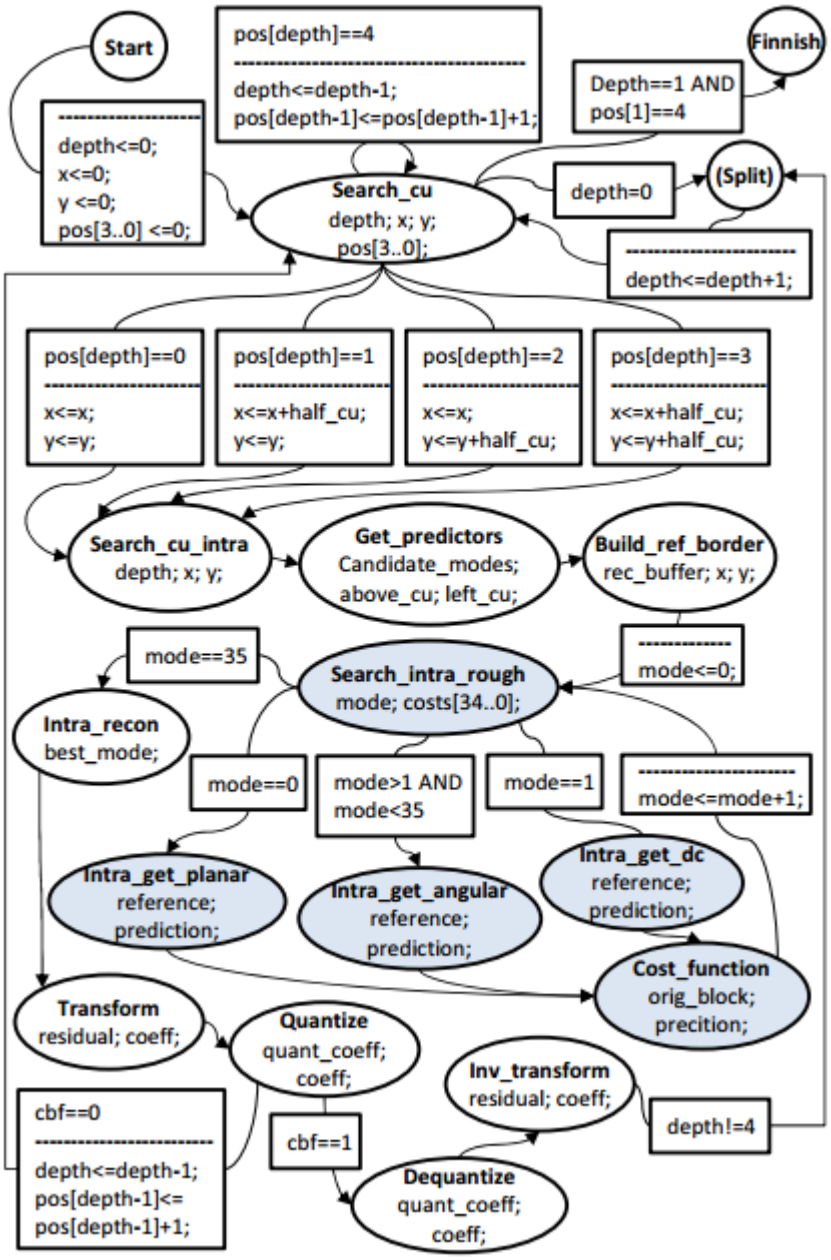


Gambar 3.9 Launch on Hardware pada SDK sistem Xilinx

Pada saat menangani impor makefile dengan masuk pada pilihan Run Configuration dan link antar library dapat disusun sehingga pesan kesalahan seluruhnya dapat diatasi. Pada saat menjalankan compiler juga memerlukan perangkat tambahan lain yaitu Xilinx toolchain yang Sourcery CodeBench Lite Edition untuk Xilinx Cortex-A9 Compiler Toolchain yang harus dipastikan telah diinstal pada perangkat SDK sistem xilinx. Bila seluruh compiler telah diselesaikan tanpa pesan kesalahan maka aplikasi perangkat lunak HM_encoder muncul di serial utilitas komunikasi pada Terminal 1 dengan 8 biner sebagai aplikasi perangkat lunak biner HEVC dan selanjutnya dapat dijalankan untuk pengkodean file video 3D sebagai HEVC referensi.

3.4.2 Implementasi Perangkat Lunak HEVC versi Kvazaar pada Hardware / Software co-design

Seperti yang dijelaskan pada bagian sebelumnya bahwa perangkat lunak HEVC versi Kvazaar adalah merupakan yang paling fleksibel untuk diimplementasikan kedalam papan Zynq 7000 AP SoC dengan faktor versi tersebut lebih mendukung pengembangan secara perangkat keras karena pada dasarnya perangkat lunak HEVC tersebut dikembangan dengan menggunakan bahasa C. Pengembangan sistem sebagai hardware / software co-design pada penelitian ini menggunakan Vivado HLS. Aplikasi Vivado HLS adalah pengembangan aplikasi vivado yang telah menggunakan bahasa tingkat tinggi sehingga mempermudah perancangan yang bersifat lebih kompleks. Pada Gambar 3.10 menunjukkan diagram alir dari perangkat lunak HEVC versi Kvazaar HEVC yang tampak memiliki komputasi yang sangat kompleks dimana bila ditemukan pengkodean yang berbeda akan diulang dari awal (bagian atas pada percabangan diagram alir) sehingga program aplikasi akan sangat membutuhkan beban perhitungan yang tinggi.



Gambar 3.10 Diagram Alir Proses Pengkodean Kvazaar HEVC [20]

Perangkat lunak HEVC versi Kvazaar mendukung semua-intra coding dari video 8-bit dengan perbandingan chroma samplin adalah 4:2:0. Parameter intra encoder dari perangkat lunak HEVC versi Kvazaar yang digunakan dalam penelitian ini adalah seperti yang tercantum pada Tabel 3.4. Pada penelitian ini digunakan Kvazaar versi 0.72 lisensi dibawah LGPLv2.1

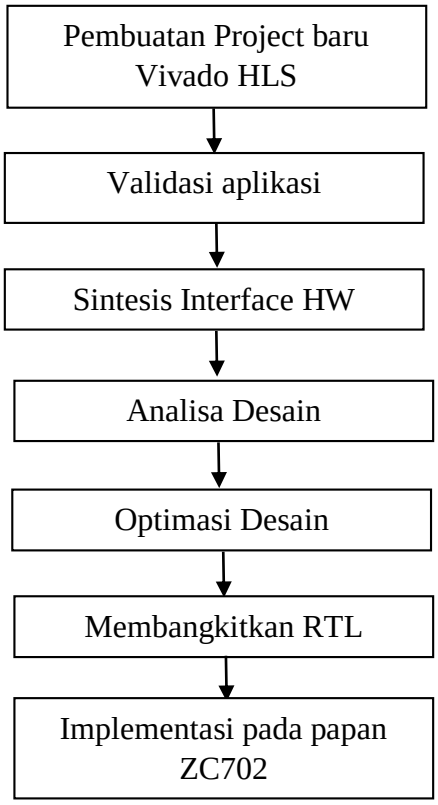
Tabel 3.4 Parameter coding Kvazaar HEVC yang Digunakan pada Penelitian ini

Features	Kvazaar HEVC intra coding
Profile	Main
Internal bit depth, color format	8, 4:2:0
Coding modes	64x64, 32x32, 16x16, 8x8
Sizes of luma coding blocks	32x32, 16x16, 8x8, 4x4
Sizes of luma prediction blocks	64x64, 32x32, 16x16, 8x8, 4x4
Intra predicition modes	DC, planar, 33 angular
Mode decision metric	SAD
RDO	Disabled
RDOQ	Disabled
Tansform	Integer DCT (integer DST for luma 4x4)
4x4 transform skip	Enabled
Loop filtering	DF, SAO

Diagram alir dari perancangan implementasi perangkat lunak HEVC versi Kvazaar ditunjukkan seperti pada Gambar 3.11. Pengukuran parameter kinerja Kvazaar pada PC dilakukan sebelum implementasi pada papan ZC702 yang ditujukan untuk membandingkan hasil yang akan diperoleh. Kompilasi pada sistem operasi Linux pada PC dangan mudah dilakukan dengan menggunakan file makefile milik Kvazaar sehingga akan menjalin seluruh file library yang ada, sampai dengan proses tersebut selesai berjalan dengan baik, dan Kvazaar dapat digunakan untuk pengkodean video 3D untuk diperoleh parameter output yang diperlukan dengan beberapa sumber dan parameter input yang berbeda. Selanjutnya Kvazaar disiapkan untuk diaplikasikan pada papan ZC702. Pada saat diuji pada file video 3D yaitu 3Dextreme Aquarium.yuv dengan pixel 1280x720p 240 urutan frame tes, diperoleh bahwa encoding yang membutuhkan waktu terbanyak adalah intra prediction, kuantisasi, dst/dct, inverse dst/dct, dan dekuantisasi. Selanjutnya pada fungsi intra prediction di Kvazaar (search_intra_rough) fungsi yang membutuhkan waktu

terbesar adalah intra_get_angular sampai dengan 35,75% dari seluruh waktu yang dibutuhkan oleh proses encoding.

Fungsi search_intra_rough akan memanggil fungsi intra_get_pred untuk menghitung prediksi pada seluruh 35 buah mode, kemudian menghitung Sum of Difference Absolute (SAD) pada masing-masing semua mode tersebut, dan akhirnya menghitung total cost untuk semua mode melalui pointer yang diteruskan pada fungsi (Gambar 3.10). Pada fungsi-fungsi ini adalah yang memiliki peluang untuk meningkatkan akselerasi sebuah hardware dalam menyelesaikan sebuah pengkodean. Dengan mengeksplorasi fungsi tersebut maka Kvazaar mendapatkan nilai yang optimum untuk cost pengkodean video dibandingkan dengan perangkat lunak HEVC versi lainnya.



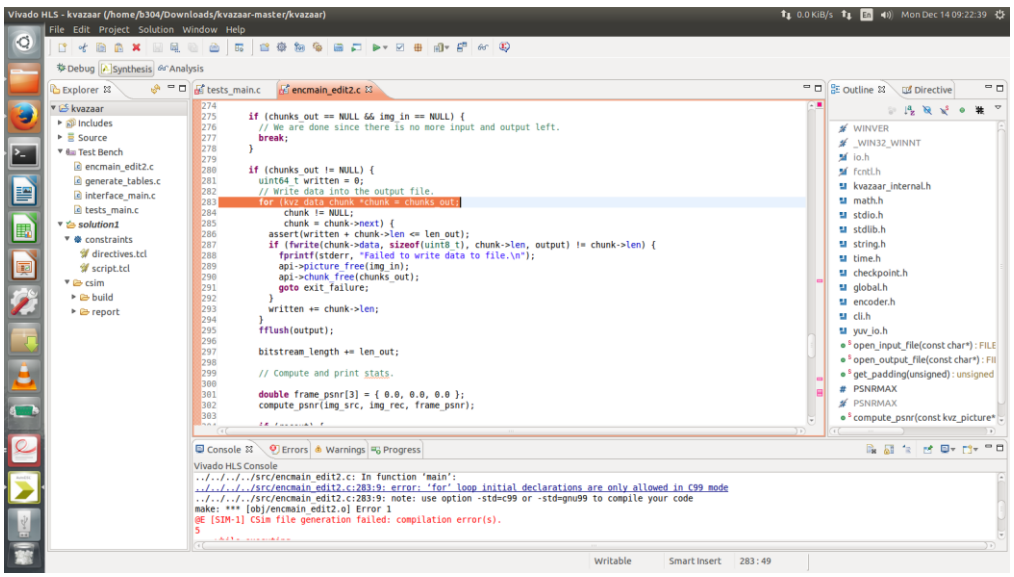
Gambar 3.11 Diagram Alir Proses pada Aplikasi Vivado HLS

Langkah 1: Buat (create) proyek baru

Secara default pada proyek Vivado HLS akan mengatur seluruh data dalam bentuk hirarki. Proyek yang dibentuk akan menggunakan informasi yang dibutuhkan seperti design source, test bench, dan solution. Dimana seperti dijelaskan sebelumnya, untuk design source yang digunakan adalah perangkat lunak HEVC versi Kvazaar, dan menyimpan file proyek pada folder Kvazaar, selanjutnya file encmain.c yang merupakan desain dengan menggunakan bahasa tingkat tinggi adalah menyimpan spesifikasi desain, dan kode C lainnya digunakan sebagai file test bench yang akan berfungsi nantinya untuk uji desain. Seluruh file header (file ektensi .h) yang berada di direktori lokal open-source Kvazaar secara otomatis akan dimasukkan kedalam proyek. Solution yang sesuai untuk sepesifikasi papan ZC702 sebelumnya harus ditentukan, karena solution tersebut akan memberikan informasi tentang teknologi target, design directive, dan hasil akhir.

Langkah 2: Validasi aplikasi

Setelah program yang menggunakan bahasa C tersebut kita bentuk dalam proyek baru HLS, maka selanjutnya harus divalidasi untuk mengetahui apakah masih mengandung kesalahan. Pada proses validasi akan membandingkan tes bench dengan data output dari fungsi encmain.c dengan nilai yang standar yang ada seperti tampak pada Gambar 3.12.



Gambar 3.12 Tampilan dari Kode Testbench

File test bench, yang merupakan fungsi top-level atau fungsi main (), akan memanggil fungsi yang akan disintesis (encmain). Proses selanjutnya adalah melaksanakan proses simulasi C untuk nantinya akan dikompilasi dan dibangun aplikasi yang diinginkan. Namun hasil yang diperoleh pada tahap ini belum sesuai karena seluruh library dan file header Kvazaar tidak dapat dijalin dengan sempurna. Beberapa pesan kesalahan ditemukan yaitu file tidak ditemukan sampai dengan file tidak sesuai, sehingga proses kompilasi tidak dapat dilanjutkan. Selanjutnya untuk mengatasi masalah tersebut, digunakan aplikasi pendukung yaitu Petalinux yang dijalankan pada PC. Seluruh library dan file header yang dimiliki Kvazaar diletakkan dalam folder aplikasi Petalinux, dan dideklarasikan dalam program utama dengan mengadopsi file makefile milik Kvazaar sendiri, sehingga proses kompilasi dapat meniru proses yang seharusnya dilaksanakan pada saat memanggil makefile milik Kvazaar seperti tampak pada Gambar 3.13 .

Hasil kompilasi dan building dari Petalinux selanjutnya dapat diimplementasikan pada papan ZC702 yang secara otomatis akan dimuati pada saat booting, sehingga seluruh aplikasi tersebut sudah dikenal saat pelaksanaan pengkodean file video 3D yang telah ditentukan, untuk selanjutnya dapat dianalisa sesuai kebutuhan.

3.4.3 HEVC di Zynq Programmable Logic

Sebagai bentuk kerja terakhir dari FPGA papan ZC702 adalah sebagai Zynq Programmable Logic yang Independen. Sebelumnya telah diimplementasikan dalam bentuk Zynq PS. Pada saat mencoba mengimplementasi coding tersebut, ditarik kesimpulan pada saat tersebut HEVC belum dapat diimplementasikan secara penuh pada papan ZC702 untuk menjalankan HEVC. Jadi hanya akan diimplementasikan bagian dari HM yaitu IDCT untuk dijalankan di Zynq PL.

```
INFO: Checking component...
INFO: Generating make files and build linux
INFO: Generating make files for the subcomponents of linux
INFO: Building linux
[INFO ] pre-build linux/rootfs/fwupgrade
[INFO ] pre-build linux/rootfs/peekpoke
[INFO ] pre-build linux/rootfs/uWeb
[INFO ] build system.dtb
[INFO ] build linux/kernel
[INFO ] update linux/u-boot source
[INFO ] generate linux/u-boot configuration files
[INFO ] build linux/u-boot
[INFO ] Setting up stage config
[INFO ] Setting up rootfs config
[INFO ] Updating for armv7a-vfp-neon
[INFO ] Updating package manager
[INFO ] Expanding stagefs
[INFO ] build linux/rootfs/fwupgrade
[INFO ] build linux/rootfs/peekpoke
[INFO ] build linux/rootfs/uWeb
[INFO ] build kernel in-tree modules
[INFO ] modules linux/kernel
[INFO ] post-build linux/rootfs/fwupgrade
[INFO ] post-build linux/rootfs/peekpoke
[INFO ] post-build linux/rootfs/uWeb
[INFO ] pre-install linux/rootfs/fwupgrade
[INFO ] pre-install linux/rootfs/peekpoke
[INFO ] pre-install linux/rootfs/uWeb
[INFO ] install linux/kernel
[INFO ] install linux/u-boot
[INFO ] Setting up rootfs config
[INFO ] Setting up stage config
[INFO ] Updating for armv7a-vfp-neon
[INFO ] Updating package manager
[INFO ] Expanding rootfs
[INFO ] install sys_init
[INFO ] install linux/rootfs/fwupgrade
[INFO ] install linux/rootfs/peekpoke
[INFO ] install linux/rootfs/uWeb
[INFO ] install kernel in-tree modules
[INFO ] modules_install linux/kernel
[INFO ] post-install linux/rootfs/fwupgrade
[INFO ] post-install linux/rootfs/peekpoke
[INFO ] post-install linux/rootfs/uWeb
[INFO ] package rootfs.cpio to /home/user/ZC702/images/linux
[INFO ] Update and install vmlinux image
[INFO ] vmlinux linux/kernel
[INFO ] install linux/kernel
[INFO ] package zImage
[INFO ] zImage linux/kernel
[INFO ] install linux/kernel
[INFO ] package FIT image
```

Gambar 3.13 Output Progress Kompilasi

Untuk menjalankan bagian HEVC yaitu Inverse DCT akan digunakan tools pendukung Vivado HLS. Pada dasarnya HEVC bagian IDCT melaksanakan operasi dengan perkalian matriks, oleh karena itu adalah dapat dijalankan dan

kompatibel dengan vivado HLS. Algoritma HEVC bagian IDCT memiliki komputasi algoritma yang cukup kompleks dibandingkan dengan pemrosesan vivado HLS pada pengolahan citra lain maupun kompresi video. Parameter input dari IDCT dipilih tergantung pada ukuran operasi yang akan dilaksanakan pada IDCT antara lain yaitu 4x4, 8x8, 16x16 atau 32x32. Pada proses awal, kolom IDCT dihitung, dan hasil koefisien yang diperoleh akan mengalami proses clipped. Kemudian, baris IDCT dihitung menggunakan transpose dari matriks yang sebagai input, dan koefisien yang dihasilkan juga diproses dengan clipped.

Seperti di Gambar 3.10, desain ini menggunakan desain langkah yang sama untuk Vivado HLS. Vivado HLS memiliki beberapa pilihan optimasi seperti pipelining, loop unrolling, dan loop merging. Sehingga memungkinkan untuk menambahkan blok DSP yang diinginkan seperti antara lain multiplier, divder, atau square unit. Selanjutnya dapat memilih opsi I/O port seperti bus, memori, FIFO atau tipe ACK sehingga dapat menambah secara signifikan kecepatan proses dari bus AXI-4 untuk transfer data. Sebuah bagian dari kode C yang telah dikembangkan pada masukan Vivado HLS tampak pada Gambar 3.14.

3.5 Simulasi Pengkodean HEVC pada PC

Pengkodean Video 3D dengan standar HEVC yang dijalankan pada PC dibuat sebagai simulasi yang akan diambil parameter dari hasil pengkodean dari awal sampai dengan akhir selesainya pengkodean, dan akan dibuat sebagai perbandingan parameter yang diperoleh dari hasil pengkodean HEVC pada papan ZC702.

Untuk menjalankan HEVC pada PC telah disediakan file pendukung (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 HEVC akan langsung dijalin secara otomatis dan dapat dijalankan pada PC tersebut.

```
void COL_partialButterflyInverse8(
int15 resid[DCT_8], int7 coeff[31], int7 coef8[16],
int16 *Y1, int16 *Y2, int16 *Y3, int16 *Y4,
int16 *Y5, int16 *Y6, int16 *Y7, int16 *Y8)

{
char j,l,k = 0;
int26 E[4], O[4], EE[2], EO[2];
for(l=0; l<4; l++)

    #pragma HLS unroll factor=2
    {O[l] = coef8[l*4]*resid[1] + coef8[l*4+1]*resid[3] +
    coef8[l*4+2]*resid[5] + coef8[l*4+3]*resid[7];
    };

EO[0] = coeff[1]*resid[2] + coeff[2]*resid[6];
EO[1] = coeff[2]*resid[2] - coeff[1]*resid[6];
EE[0] = coeff[0]*resid[0] + coeff[0]*resid[4];
EE[1] = coeff[0]*resid[0] - coeff[0]*resid[4];

#pragma HLS pipeline
E[0] = EE[0] + EO[0];
E[1] = EE[0] - EO[0];
E[2] = EE[1] + EO[1];
E[3] = EE[1] - EO[1];
*Y1 = Clip3(-32768, 32767, (E[0] + O[0] + 64) >> 7);
*Y2 = Clip3(-32768, 32767, (E[1] + O[1] + 64) >> 7);
*Y3 = Clip3(-32768, 32767, (E[2] - O[2] + 64) >> 7);
*Y4 = Clip3(-32768, 32767, (E[3] - O[3] + 64) >> 7);
*Y5 = Clip3(-32768, 32767, (E[3] + O[3] + 64) >> 7);
*Y6 = Clip3(-32768, 32767, (E[2] + O[2] + 64) >> 7);
*Y7 = Clip3(-32768, 32767, (E[1] - O[1] + 64) >> 7);
*Y8 = Clip3(-32768, 32767, (E[0] - O[0] + 64) >> 7);
}
```

Gambar 3.14 Cuplikan Kode Program pada bahasa C

Untuk menjalankan simulasi tersebut digunakan dua buah tipe PC, yaitu dengan spesifikasi yang mendekati papan ZC702 yaitu menggunakan procesor dengan kecepatan dual core 1,4 GHz dan menggunakan PC dengan processor quad core i7 yang sama dengan yang digunakan untuk download perangkat lunak kedalam sistem papan ZC702. Parameter yang akan diperoleh akan dibandingkan

dengan nilai yang diperoleh dengan menggunakan papan ZC702 yaitu membandingkan nilai kecepatan proses, ukuran file yang diperoleh dari hasil pengkodean video, PSNR average dan MSE (mean square error).

3.6 Cara Pengukuran Kecepatan Pengkodean, PSNR Average dan MSE

Untuk melakukan pengukuran kecepatan pengkodean, PSNR average, maupun MSE telah disediakan dalam perangkat lunak HEVC. Pada saat selesai melaksanakan sebuah pengkodean dari 3D video akan ditampilkan nilai hasil pengkodean, termasuk kecepatan pengkodean (waktu yang dibutuhkan untuk menyelesaikan seluruh pengkodean), PSNR untuk sistem video masing-masing YUV, serta MSE dari hasil pengkodean 3D video. Untuk hasil yang diperoleh memiliki optional dapat ditampilkan maupun tidak ditampilkan pada akhri dari proses pengkodean.

BAB 4
HASIL DAN PEMBAHASAN

Pada bab ini akan dibandingkan antara hasil implementasi perangkat lunak HEVC yang dijalankan berbasis Zynq 7000 AP SoC dan PC dengan sistem operasi Linux. Hasil yang diperoleh secara program penuh hanya pada bentuk kerja Zynq PS secara independent karena kompleksitas coding yang sangat tinggi yang mencapai 69k lines sehingga belum dapat memperoleh data implementasi pada Zynq PS/PL co-desain (terintegrasi) maupun PL secara independen karena keterbatasan waktu dan sumber daya pada penelitian tesis ini.

4.1 Kinerja di PC dengan sistem operasi Linux dan Zynq 7000 AP SoC

Tujuan utama dari pengujian ini adalah untuk merencanakan papan ZC702 sebagai embedded system, sehingga sistem ini memadai untuk pengkodean video standalone dengan biaya yang ekonomis. Pada bagian ini, akan dibandingkan dua tipe hasil yang diperoleh dari pengkodean video 3D dengan menggunakan perangkat lunak HEVC versi Kvazaar yang dijalankan pada Zynq 7000 AP SoC dan PC dengan sistem operasi Linux. Nilai yang akan dibandingkan adalah meliputi perbandingan ukuran video hasil pengkodean HEVC, waktu total yang dibutuhkan dengan ukuran file dan parameter yang ditentukan, dan PSNR average dari hasil pengkodean video serta MSE pada masing-masing proses. Pada penelitian ini digunakan sampel 2 buah video 3D dalam bentuk format stereoscopy SBS yaitu seperti pada Tabel 4.1.

Tabel 4.1 Sampel video 3D stereoscopy SBS

No	Input Video	Resolution	Number of frame	fps
1.	3DextremeAquarium.yuv	1280x720	1719 frame	30
2.	Dinosaur3D.yuv	1920x1080	657 frame	30

Hasil dari proses pengkodean video dapat diatur dengan mengubah coding modes untuk menghasilkan ukuran file dan MSE yang berbeda dan selanjutnya data tersebut diperbandingkan antara kedua variabel tersebut. Coding modes yang digunakan berbanding terbalik dengan ukuran file yang diperoleh, dengan semakin besar ukuran coding modes maka ukuran yang diperoleh akan semakin kecil (tingkat kompresi semakin besar)

Left View **Right View**

Gambar 4.1 Hasil tampilan Dinosaur3D.h265 dengan coding modes 64x64

Ukuran file = 28,0 MB; MSE = 1,269115

Gambar 4.2 Hasil tampilan Dinosaur3D.h265 dengan coding modes 32x32

Ukuran file = 28,2 MB; MSE = 1,217118

Hasil dari pengkodean dengan HEVC dengan coding modes yang berbeda-beda tampak seperti pada Gambar 4.1 dengan coding modes 64x64, Gambar 4.2 dengan coding modes 32x32, Gambar 4.3 dengan coding modes 16x16 dan Gambar 4.4 dengan coding modes 8x8.

Gambar 4.3 Hasil tampilan Dinosaur3D.h265 dengan coding modes 16x16

Ukuran file = 36,2 MB; MSE = 1,21667

Gambar 4.4 Hasil tampilan Dinosaur3D.h265 dengan coding modes 8x8

Ukuran file = 45 MB; MSE = 1,211998

Tabel 4.2 Hasil Pengujian HEVC pada PC berbasis Linux

Input Video	Original size	Number of frame	Encoding time (s)		Encoding size
			μP=1.4GHz	μP=core i7	
Video1	2.4 GB	1719 frame	11560,6	830,43	26,4 MB
Video2	2 GB	657 frame	9378,8	627,37	23,7 MB

Tabel 4.3 Nilai PSNR Hasil Pengkodean pada PC berbasis Linux

Input Video	Y-PSNR	U-PSNR	V-PSNR	MSE[21]
Video1	44,850	48,837	49,750	1,078026
Video2	42,861	48,780	48,96	1,201976

Untuk perbandingan kinerja PC berbasis Linux dan FPGA tampak pada Tabel 4.1, 4.2, 4.3, 4.4 dengan menggunakan ukuran coding modes otomatis oleh sistem HEVC. Nilai hasil dari pengkodean yang diperoleh dari tabel tersebut sangat dipengaruhi oleh jumlah frame video yang digunakan, pixel dari input. Waktu yang dibutuhkan sebanding dengan durasi dan jumlah pixel dari video input. Perangkat lunak HEVC versi Kvazaar mampu mengurangi ukuran video hasil walaupun persentasi penurunan ukuran kompresi bervariasi dari setiap input video yang berbeda karena sangat dipengaruhi dari bentuk pergerakan konten video dan tipe perubahan dari frame ke frame dalam domain waktu. Dengan mengacu dari panduan HEVC bahwa hasil penelitian ini bukan merupakan pembuktian kemampuan kompresi HEVC yang memiliki nilai kompresi sampai dengan 50% dari ukuran video inputnya. Namun penelitian ini lebih fokus kepada kemampuan Zynq 7000 AP SoC papan ZC702 sebagai embedded system dan dapat melaksanakan proses pengkodean video 3D.

Tabel 4.4 Hasil Pengujian HEVC pada Zynq 7000 AP SoC

Input Video	Original size	Number of frame	Encoding time (s)	Encoding size
Video1	2.4 GB	1720 frame	9759,709	26,4 MB
Video2	2 GB	657 frame	8377,041	23,7 MB

Tabel 4.5 Nilai PSNR Hasil Pengkodean pada Zynq 7000 AP SoC

Input Video	Y-PSNR	U-PSNR	V-PSNR	MSE[21]
Video1	44,7937	48,970	48,6065	1,123076
Video2	42,651	48,892	48,850	1,211609

Tabel 4.6 Perbandingan Hasil Pengujian HEVC pada PC berbasis Linux dengan Zynq 7000 AP SoC

Input Video	**Encoding time (s) pada PC**		**Encoding time (s) pada Zynq 7000**
	μP=1.4GHz	μP=core i7	
Video1	11560,6	830,43	9759,709
Video2	9378,8	627,37	8377,041

Dari Tabel 4.6, waktu pengolahan pengkodean video 3D dengan Zynq PS secara rata-rata 5% lebih cepat dari PC dengan sistem operasi Linux pada tipe yang sama. Menjalankan aplikasi pada Zynq PS tidak dapat mencapai jumlah waktu yang optimal untuk menyelesaikan proses, hal ini disebabkan karena pembuatan pemilihan sebagai Zynq PS bukan dimaksudkan untuk mencapai kecepatan pemrosesan. Disamping itu, algoritma pada Kvazaar mengandung perhitungan yang cukup kompleks yang akan sangat mempengaruhi beban pengolahan yang tinggi processor untuk encoding video 3D.

Gambar 4.5 Hasil Tampilan Dinosaur3D dalam format H.264 dibandingkan dengan H.265

Selanjutnya bila dibandingkan kualitas visual, yang tampak pada Gambar 4.5 diperbandingkan tampilan visual antara Dinosaur3D dalam standar kompresi video .h264 dan hasil kompresi Dinosaur3D.h265. Hasil cuplikan video ditampilkan bersamaan dengan layar besar adalah dalam standar H.265 dan layar kecil adalah dalam standar H.264. Walaupun HEVC menurunkan ukuran video hasil pengkodean hingga 50% namun secara visualisasi hampir tidak ada perbedaan yang signifikan antara standar H.264 dan dalam standar H.265. Kualitas visual cuplikan video 3D stereoscopy dari Dinosaur3D.h265 tidak memberikan perbedaan yang signifikan dibandingkan dengan pengkodean menggunakan standar h264. Hal ini dapat dibandingkan dari nilai PSNR dan MSE, dimana nilai PSNR dan MSE yang diperoleh cukup kecil sehingga tidak terlalu signifikan dalam perubahan kualitas visual hasil kompresi standar h265.

4.2 Profiling

Profiling adalah proses penghitungan untuk menentukan pada tiap bagian pengolahan pengkodean Kvazaar HEVC sehingga dapat diketahui kemungkinan untuk peningkatan efektivitas dari seluruh proses pengkodean video yang mana pada penelitian ini difokuskan pada persentase waktu yang dibutuhkan untuk pengkodean. Tabel 4.6 menunjukkan hasil profiling video 3D Dinosaur3D 1080p dengan menggunakan HEVC Kvazaar yang dijalankan menggunakan papan ZC702.

Tabel 4.7 Hasil Profiling dari Kvazaar

Function	Percentage time consuming for encoding
Intra prediction	67,74 %
Quantitazation	8,54 %
Dst/dct	4,69 %
Inverse dst/dct	3,78 %
Dequantization	0,95 %

Bagian proses yang membutuhkan waktu terbesar adalah bagian intra prediction sehingga bila akan ditingkatkan efektivitas dari proses pengkodean video

3D dapat dilakukan dengan melaksanakan proses tersebut dengan Znyq PL sehingga beban processor lebih tidak terbebani dalam menangani jumlah perhitungan yang cukup kompleks tersebut.

4.3 Zynq PL

Pada sub bab ini, menjalankan HEVC pada bagian IDCT dengan dukungan tools Vivado HLS yang dijalankan pada Zynq PL. Hasil dari pelaksanaan HEVC IDCT menggunakan Vivado HLS dapat diperoleh nilai penggunaan hardware dari papan ZC702 seperti pada Tabel 4.7.

Tabel 4.8 Parameter penggunaan hardware pada papan ZC702

TU	LUTs	DFFs	Slices	BRAMs	I/O
4x4	663	373	212	1	134
8x8	2834	2010	919	1	262
16x16	5000	4090	1601	1	518
32x32	40764	28772	12605	13	1030
All	50566	34955	14944	13	1045

Dari data tersebut masih terkandung beberapa data yang dihasilkan oleh Vivado HLS namun karena keterbatasan waktu dan sumber daya maka analisis lebih lanjut belum dapat dilakukan secara mendetail. Dengan menggunakan aplikasi Vivado HLS sangat membantu dalam menyederhanakan rancangan untuk mengimplementasikan HEVC pada FPGA yang pada penelitian ini digunakan papan ZC702.

4.4 Analisa Operasi Processor pada Zynq PS

Sumber daya processor yang sudah tetap, dan terbatas pada processor dual core, dengan frekwensi kerja pada clock yang telah ditentukan. Perhitungan dari efisiensi kinerja sebuah processor adalah diukur dari segi siklus waktu (eksekusi), yang tentu saja pada sebuah operasi processor menggunakan beberapa periode siklus untuk mengerjakan sebuah proses, yang sangat dipengaruhi oleh frekwensi pewaktu yang diinginkan; dan kompleksitas dari proses yang akan dikerjakan.

Seperti penjelasan sebelumnya bahwa kompleksitas perhitungan pengkodean video 3D menggunakan perangkat lunak HEVC versi Kvazaar sangat tinggi sehingga processor memiliki beban kerja yang cukup tinggi juga.

Sistem kerja sebuah processor juga bukanlah mengerjakan suatu proses secara konsisten, namun berdasarkan multi tasking, sehingga tidak ada sebuah proses menduduki timeslot processor selamanya, sehingga kinerja sebuah processor harus berbagi dengan sistem operasinya sendiri sampai dengan menyelesaikan sebuah tugas yang diberikan padanya.

4.5 Rate Distortion Optimisasi

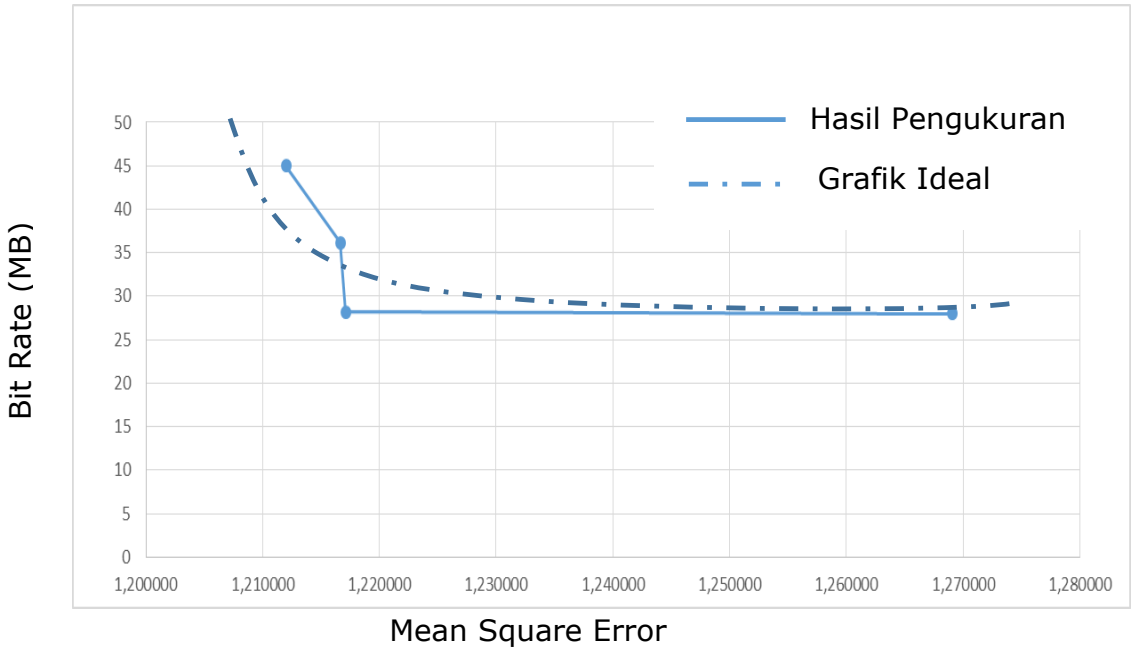
Bit rate dari pengkodean video dapat diturunkan dalam range distorsi yang masih dalam batas dapat diterima (dalam penelitian ini digunakan nilai MSE). Nilai RDO dapat berbeda dari setiap standar yang berbeda. Algoritma RDO sangat penting untuk memilih mode pengkodean yang optimal untuk memperoleh nilai terbaik dari pemilihan distorsi yang rendah ataukah nilai bit rate yang rendah (trade off antara nilai bit rate dan MSE). Dari Gambar 4.1, 4.2, 4.3, 4.4 dapat diperoleh data seperti pada Tabel 4.8.

Tabel 4.9 Hasil MSE yang diperoleh dari bit rate Video2.

Bit Rate (MB)	**MSE**
45	1,211998
36,2	1,216670
28,2	1,217118
28	1,269115

Dari pengujian tampak bahwa bit rate dapat diturunkan sampai dengan batas minimal dengan mengorbankan nilai MSE. Bila nilai bit rate dipilih semakin kecil maka akan mempengaruhi nilai MSE menjadi naik sehingga akan menurunkan kualitas dari gambar yang dihasilkan. Sebaliknya nilai maksimal dari bit rate diperoleh pada saat lossless coding namun dengan nilai MSE=0 namun dengan

konsekuensinya akan menghasilkan nilai kompresi yang rendah. Hasil dari grafik rate distortion pada pengkodean Video2 tampak seperti pada Gambar 4.6.



Gambar 4.6 Diagram Rate-distorsion

Halaman ini sengaja dikosongkan

BAB 5
KESIMPULAN

Pada bagian ini akan disimpulkan hasil dari pengujian implementasi perangkat lunak HEVC pada Zynq 7000 AP SoC. Tujuan utama dari pengujian ini adalah untuk mensimulasikan papan ZC702 sebagai embedded system yang dapat berjalan standalone dan memiliki nilai ekonomis yang bagus sebagai pengolah pengkodean video 3D yang pada penelitian ini digunakan tipe stereoscopy side by side (SBS).

5.1 Kesimpulan

Dari pengujian yang telah dilakukan, dapat diambil kesimpulan bahwa

1. HEVC diimplementasikan pada papan ZC702 dengan menggunakan software Vivado dan Petalinux, yang mampu menangani kompleksitas pengkodean HEVC dan library yang dimiliki sehingga sistem dapat berjalan dengan baik.
2. Perbandingan saat dijalankan pada Zynq PS, waktu yang dibutuhkan rata-rata lebih cepat 5% daripada pada PC berbasis Linux.
3. Untuk mengoptimalkan proses pengkodean video 3D dapat dilakukan profiling pada penggunaan perangkat lunak HEVC yang sedang dijalankan pada papan ZC702. Pada penelitian ini bagian proses dari HEVC yang membutuhkan persentase waktu terbesar adalah bagian intra prediction, sehingga pemrosesan pada tahap bagian ini dapat dioptimalkan dengan memindahkan proses tersebut ke zynq PL sehingga mengurangi beban processor dan mempercepat proses secara keseluruhan.
4. Aplikasi Petalinux dapat digunakan sebagai pendukung tools kompilasi pada saat mengkompilasi dan build aplikasi Kvazaar yang akan diimplementasikan ke papan ZC702. Aplikasi Petalinux mampu menjalin seluruh file library dan header dengan meniru proses dari file makefile milik Kvazaar sendiri, sampai dengan proses dapat diselesaikan dengan baik dan siap untuk diimplementasikan ke papan ZC702.

Berikut beberapa keuntungan dan kerugian dalam penggunaan aplikasi Vivado HLS. Beberapa keuntungannya adalah:

- Vivado HLS telah menyediakan pengaturan yang dibuat otomatis, yang sangat membantu perancang dalam menyelesaikan tugas aplikasi, sehingga lebih sederhana dibandingkan dengan menggunakan Vivado.
- Aplikasi tersebut menggunakan bahasa C dan penggunaan fungsi tidak membutuhkan penjelasan rinci, dibandingkan VHDL (Verilog Hardware Description Language), masih harus dikodekan secara manual oleh pengguna.
- Verifikasi kode dapat dilakukan dengan lebih mudah dengan satu langkah telah menyelesaikan verifikasi kode sampai dengan pembangkitan RTL dan pengaturan yang secara otomatis oleh Vivado HLS.

5.2 Saran untuk pengembangan pada masa yang akan datang

Untuk penelitian di masa yang akan datang, untuk disarankan melanjutkan penelitian ini yaitu mengaplikasikan papan Zynq 7000 AP SoC sebagai embedded system yang standalone. Dalam pengembangan lebih lanjut fungsi dari papan ZC702 diimplementasikan sebagai real-time 3D UHD/4K. Karena pada FPGA Xilinx Zinq-7000 telah disediakan port-port untuk pengkodean real time. Hal ini akan berimbas kepada pergeseran perangkat ke FPGA dibandingkan dengan berbasiskan perangkat lunak untuk mendukung kompatibilitas pada level video 3D real time karena kemampuannya mengeksekusi program secara pralel. Dengan fitur yang dimiliki FPGA untuk melaksanakan proses algoritma dengan komputasi paralel maka akan diperoleh kinerja yang lebih baik dan mampu menangani perhitungan yang jauh lebih kompleks.

DAFTAR PUSTAKA

[1] Gary J. Sullivan, J.R. Ohm, W.J. Han, T. Wiegand, “Overview of the High Efficiency Video Coding (HEVC) Standard,” IEEE Trans. on Circuit and Systems for Video Technology, vol.22, no.12, December 2012.

[2] Xilinx, Inc., “Zynq-7000 All Programmable SoC Software Developers Guide”, UG821, v8.0, April 2014.

[3] L. H. Crockett, R. A. Elliot, M. A. Enderwitz, R. W. Stewart, The Zynq Book: Embedded Processing with the ARM Cortex-A9 on the Xilinx Zynq 7000 All Programmable SoC, First Edition, Strathclyde Academic Media, 2014.

[4] Contributions to the Physiology of Vision.—Part the First. On some remarkable, and hitherto unobserved, Phenomena of Binocular Vision. By CHARLES WHEATSTONE, F.R.S., Professor of Experimental Philosophy in King's College, London.

[5] Welling, William. Photography in America, page 23

[6] International Stereoscopic Union, 2006, "Stereoscopy", Numbers 65-72, p.18

[7] M. Wien, High Efficiency Video Coding: Coding Tools and Specification, Springer, 2015.

[8] K.R. Rao, D.N. Kim, J.J. Hwang, Video Coding Standards: AVS China, H.264/MPEG-4 PART 10, HEVC, VP6, DIRAC and VC-1, Springer 2014.

[9] Benny Bing,”Next-Generation Video Coding and Streaming”, John Wiley & Sons, Inc., Hoboken, New Jersey, 2015

[10] www.elemental.com

[11] http://people.irisa.fr/Olivier.Le_Meur/teaching/HEVC_CAV_ESIR3_2011_2012.pdf

[12] V.Sze, M.Budagavi, G.J. Sullivan, High Efficiency Video Coding (HEVC): Algorithms and Architectures, Springer, 2014.

[13] Seunghyun Cho, Hyunmi Kim, Kyungjin Byun and Nak-Woong Eum, "HEVC Hardware Decoder Implementation for UHD Video Applications", Electronics and Telecommunications Research Institute, Daejeon, Korea

[14] Xilinx, Inc., “Zynq-7000 Technical Reference Manual”, UG585, v1.7, February 2014.

[15] Xilinx, Inc., “UG871 - Vivado Design Suite Tutorial: High Level Synthesis”, v2014.1, May 2014.

[16] Xilinx, Inc, “UG902 - Vivado Design Suite User Guide: High-Level Synthesis”, v2014.1, May 2014.

[17] Xilinx, Inc., “Xilinx Software Development Kit (SDK)” product webpage. http://www.xilinx.com/tools/sdk.htm

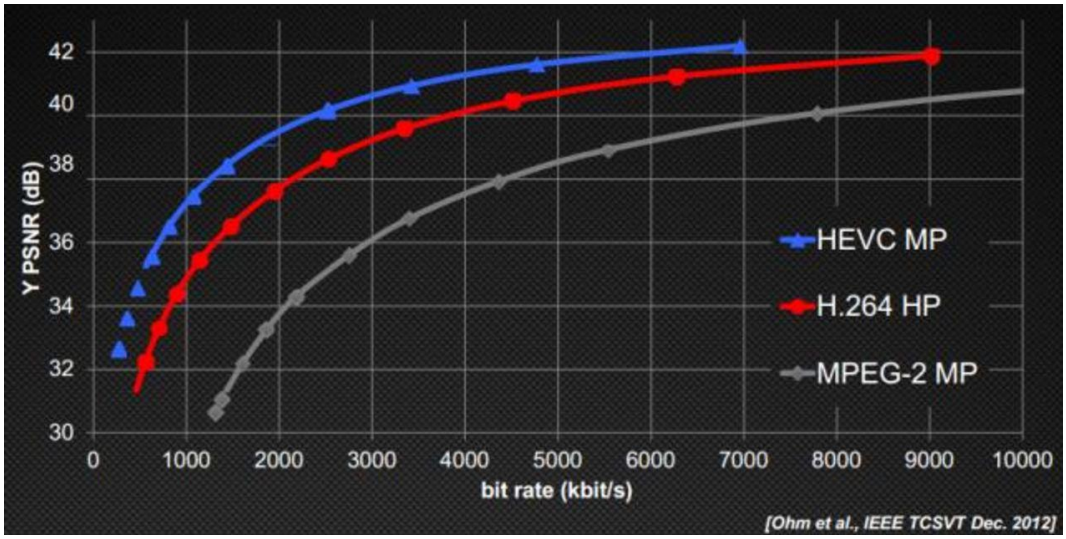
[18] Kvazaar HEVC encoder [Online]. Available : https://github.com/ultravideo/kvazaar

[19] L. Daoud, D. Zydek, and H. Selvaraj, “A survey of high level synthesis languages, tools, and compilers for reconfigurable high performance computing,” in Advances in Systems Science, Springer, 2014, pp. 483-492.

[20] Panu Sjövall, Janne Virtanen, Jarno Vanne, Timo D. Hämäläinen, "High-Level Synthesis Design Flow for HEVC Intra Encoder on SoC-FPGA", 2015 Euromicro Conference on Digital System Design.

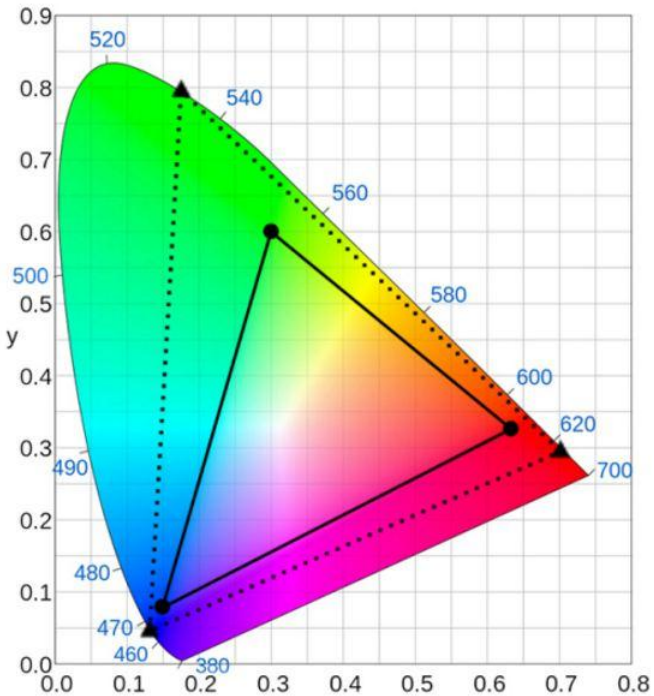
[21] P. Sjovall, J. Virtanen, J. Vanne, T. D. Hamalainen, “High-Level Synthesis Design Flow for HEVC Intra Encoder on SoC-FPGA,” Euromicro Conference on Digital System Design, 2015.

LAMPIRAN



HEVC Encoding Efficiency

25% to 35% bit rates at equivalent quality (HD)



ITU-R recommendations for H.265 codecs at 4K and 8K resolutions

COMPONENT	MPEG-2	H.264	HEVC/H.265
General	Motion compensated predictive, residual, tranformed, entropy coded	Same basics as MPEG-2	Same basics as MPEG-2
Intra prediction	DC only	Multi-direction, multi-pattern, 9 intra modes fir 4x4, 9 for 8x8, 4 for 16x16	35 modes for intra prediction, 32x32, 16x16, 8x8, and 4x4 prediction size
Coded image types	I, B, P	I, B, P, SI, SP	I, P, B
Transform	8x8 DCT	8x8 and 4x4 DCT like integer transform	32xx32, 16x16, 8x8 and 4x4 DCT-like integer transform
Motion estimation blocks	16x16	16x16, 16x8, 8x16, 8x8, 8x4, 4x8, 4x4	64x64 and hierarchical quad tree partitioning down to 32x32, 16x16, 8x8 Each size can be partitioned once more in up to 8 ways
Entropy coding	Multiple VLC tables	Context adaptive binary arithmetic coding (CABAC) and context adaptive VLC tables (CAVLC)	Context adaptive binary arithmetic coding (CABAC)
Frame distance for prediction	1 past and 1 futuure reference frame	Up to 16 past and/or future reference frames, including longterm references	Up to 15 past and/or future reference frames, including longterm references
Fractional motion estimation	½ pixel bilinear interpolation	½ pixel 6-tap filter, ¼ pixel linear interpolation	¼ pixel 8-tap filter
In-loop filter	None	Adaptive deblocking filter	Adaptive deblocking filter and sample adaptive offset filter

Comparison of MPEG-2 : H.264 : HEVC

Hasil tampilan 3DextremeAquarium.h265 dengan coding modes 64x64

Ukuran file = 31,17 MB; MSE = 1,27005

Hasil tampilan 3DextremeAquarium.h265 dengan coding modes 32x32

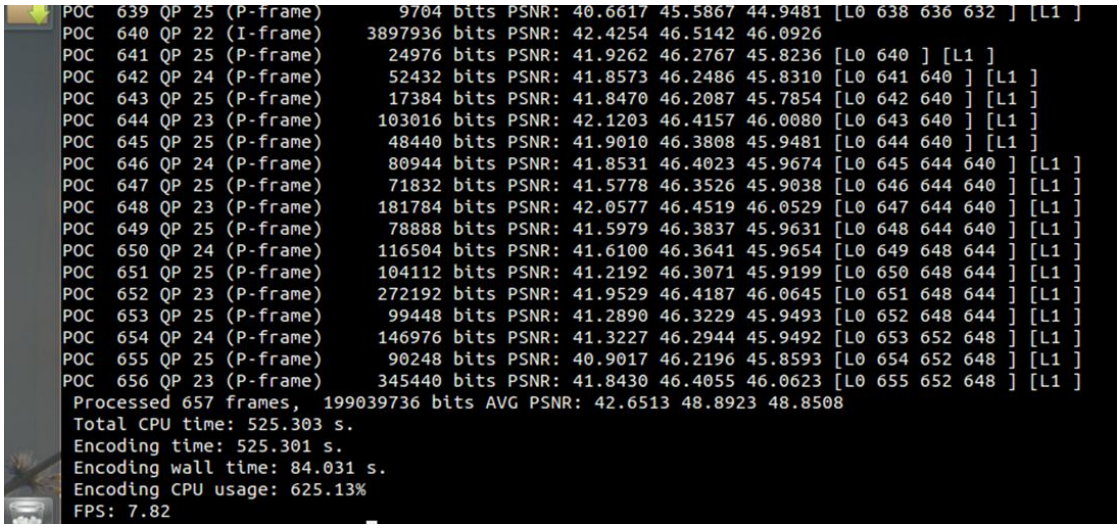
Ukuran file = 31,3 MB; MSE = 1,22243

Hasil tampilan 3DextremeAquarium.h265 dengan coding modes 16x16

Ukuran file = 40,4 MB; MSE = 1,21884

Hasil tampilan 3DextremeAquarium.h265 dengan coding modes 8x8

Ukuran file = 50,2 MB; MSE = 1,21245



Proses menjalankan HEVC Kvazaar

Halaman ini sengaja dikosongkan

DAFTAR RIWAYAT HIDUP

Alexander Victor Bukit, ST, lahir di Tigabinanga, Sumatera Utara pada tanggal 20 Juni 1978 adalah staf Program Studi S-1 Teknik Elektro Sekolah Tinggi Teknologi Angkatan Laut (STTAL). Penulis merupakan alumni Jurusan Teknik Elektro Universitas Sumatera Utara dan terdaftar sebagai mahasiswa Program Pasca Sarjana Program Strata Dua (S2) Tahun 2015 pada Bidang Studi Telekomunikasi Multimedia di Institut Teknologi Sepuluh Nopember (ITS). Penulis memiliki motto : "My life is my hobby". Penulis menerima kritik dan saran yang membangun pada email: alex.v.bukit@gmail.com. Demikian yang dapat penulis sampaikan, semoga tulisan ini dapat bermanfaat bagi kita semua khususnya yang berkeinginan mengembangkan teknologi multimedia. *Jalesveva Jayamahe.*