

**SKRIPSI**

**IMPLEMENTASI FILTER SPASIAL LINEAR PADA VIDEO *STREAM*  
MENGUNAKAN *FPGA HARDWARE ACCELERATOR***

Disusun dan diajukan oleh

**SULAEMAN  
H131 16 002**



**PROGRAM STUDI SISTEM INFORMASI  
DEPARTEMEN MATEMATIKA  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
UNIVERSITAS HASANUDDIN  
MAKASSAR**

**2021**

**IMPLEMENTASI FILTER SPASIAL LINEAR PADA VIDEO *STREAM*  
MENGUNAKAN *FPGA HARDWARE ACCELERATOR***

**SKRIPSI**

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer pada Program Studi Sistem Informasi Departemen Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin Makassar

**SULAEMAN  
H131 16 002**

**PROGRAM STUDI SISTEM INFORMASI  
DEPARTEMEN MATEMATIKA  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
UNIVERSITAS HASANUDDIN  
MAKASSAR  
APRIL 2021**

## HALAMAN PERNYATAAN KEOTENTIKAN

Yang bertanda tangan di bawah ini:

Nama : SULAEMAN

NIM : H131 16 002

Program Studi : Sistem Informasi

Jenjang : S1

Menyatakan dengan ini bahwa karya tulisan saya berjudul

### **IMPLEMENTASI FILTER SPASIAL LINEAR PADA VIDEO STREAM MENGUNAKAN *FPGA* HARDWARE ACCELERATOR**

Adalah benar hasil karya saya sendiri bukan merupakan pengambilan alihan tulisan orang lain dan belum pernah dipublikasikan dalam bentuk apapun.

Apabila dikemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan skripsi ini merupakan hasil karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Makassar, 9 April 2021



SULAEMAN

NIM. H131 16 002

**IMPLEMENTASI FILTER SPASIAL LINEAR PADA VIDEO  
STREAM MENGGUNAKAN FPGA HARDWARE  
ACCELERATOR**

Disusun dan diajukan oleh

**SULAEMAN  
H131 16 002**

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian Studi Program Sarjana pada Program Studi Sistem Informasi Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui,

Pembimbing Utama



Dr. Eng. Armin Lawi, S.Si., M.Eng.  
NIP. 197204231995121001

Pembimbing Pertama



Supri Bin Hj Amir, S.Si., M.Eng.  
NIP. 198805042019031012

Ketua Program Studi



Dr. Muhammad Hasbi, M.Sc  
NIP. 196307201989031003





## HALAMAN PENGESAHAN

Skripsi ini diajukan oleh:

Nama : SULAEMAN  
NIM : H131 16 002  
Program Studi : Sistem Informasi  
Judul Skripsi : IMPLEMENTASI FILTER SPASIAL LINEAR PADA VIDEO STREAM MENGGUNAKAN *FPGA HARDWARE ACCELERATOR*

Telah dipertahankan di depan Tim Penguji dan diterima sebagai bagian persyaratan untuk memperoleh gelar Sarjana Komputer pada Program Studi Sistem Informasi Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin.

### DEWAN PENGUJI

Tanda Tangan

1. Ketua : Dr. Eng. Armin Lawi, S.Si., M.Eng. (.....)
2. Sekretaris : Supri Bin Hj Amir, S.Si., M.Eng. (.....)
3. Anggota : Dr. Hendra, S.Si., M.Kom. (.....)
4. Anggota : Nur Hilal A Syahrir, S.Si., M.Si. (.....)

Ditetapkan di : Makassar

Tanggal : 9 April 2021



## KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT, karena atas berkat dan rahmat-Nya penulis dapat menyelesaikan skripsi ini. Shalawat serta salam senantiasa tercurah kepada *Rasulullah* Muhammad *Shallallahu Alaihi Wasallam*, yang merupakan teladan dalam menjalankan kehidupan dunia.

Alhamdulillah, skripsi dengan judul "IMPLEMENTASI FILTER SPASIAL LINEAR PADA VIDEO *STREAM* MENGGUNAKAN *FPGA HARDWARE ACCELERATOR*" yang disusun sebagai salah satu syarat untuk mencapai gelar Sarjana pada program studi Sistem Informasi fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin ini dapat diselesaikan. Walaupun adanya kendala-kendala yang dihadapi khususnya wabah Covid-19 ketika skripsi ini dikerjakan. Tetapi dalam penulisan skripsi ini, penulis mampu menyelesaikan pada waktu yang tepat berkat bantuan dan dukungan dari berbagai pihak.

Ucapat terima kasih dan apresiasi yang tak terhingga kepada kedua orang tua penulis bapak **Sudarmin** dan ibu **Yuli Hadiyanti** yang tak kenal lelah dalam memanjatkan doa serta memberikan nasihat dan motivasi kepada penulis. Tak lupa juga kepada saudara-saudara penulis **Fitri Handayani, Tri Novianti, Jumadil Yusuf, Muhammad Fitrah, Adam Ramadhan** yang selalu menjadi motivasi bagi penulis untuk terus melangkah maju.

Penulis menyadari bahwa skripsi ini dapat terselesaikan dengan adanya bantuan, bimbingan, dukungan dan motivasi dari berbagai pihak. Oleh karena itu, penulis mengucapkan ucapan terima kasih dengan tulus kepada:

1. Rektor Universitas Hasanuddin, Ibu **Prof. Dr. Dwia Aries Tina Pulubuhu** beserta jajarannya.
2. Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam, **Dr. Eng. Amiruddin** beserta jajarannya.
3. Ketua Departemen Matematika FMIPA, **Dr. Nurdin, S.Si., M.Si**, dan juga **Dr. Muhammad Hasbi, M.Sc** sebagai ketua Program Studi Sistem Informasi Universitas Hasanuddin.
4. Bapak **Dr. Eng. Armin Lawi, S.Si., M.Eng.** sebagai pembimbing utama yang

telah banyak memberikan arahan, ide, motivasi serta dukungan kepada penulis dalam banyak hal.

5. Almarhum bapak **Dr. Diaraya, M.Ak** dan bapak **Supri Bin Hj. Amir, S.Si., M.Eng** sebagai pembimbing pertama yang senantiasa memberikan masukan kepada penulis.
6. Bapak **Dr. Hendra, S.Si., M.Kom** dan Ibu **Nur Hilal, S.Si., M.Si** sebagai tim penguji atas saran dan masukan pada penelitian yang telah dilakukan oleh penulis.
7. Seluruh Bapak dan Ibu dosen FMIPA Universitas Hasanuddin yang telah mendidik dan memberikan ilmunya sehingga penulis mampu menyelesaikan program sarjana. Serta para staf yang telah membantu dalam pengurusan berkas administrasi.
8. Saudara-saudara **Ramsis Squad (Sultan, Muh Rizaldi, Sangereng Dewa Raja, Hajrin, Badaruddin Hidayat, Hamzah Julianto Nugraha, Muh Naim)** sebagai keluarga semasa tinggal di Ramsis sejak menjadi mahasiswa baru, saling berbagi dalam banyak hal, saling membantu dan bahkan saling merepotkan.
9. Saudara-saudara **Sunu Squad dan SSC Squad (Akbar, Muh Fikri Satria A, Andi Rezki Muh Nur, Muhammad Akbar Atori, Baharuddin Kasim, Andi Yaumil Falakh, Nur Ikhwan Putra Pratama, Bagas Prasetyo, Zinedine Kahlil Gibran Zidane, Rio Mukhtarom, Marfiandhi Putra, Abdul Aziz Mubarak, Mutawally Syarawy, Fatur Rahman, Fitriadi Syawal Mustafa)** yang telah menemani penulis selama perkuliahan, saling memberi motivasi dan bantuan, meluangkan waktu dan berbagi suka-duka serta kebersamaan selama menuntut ilmu.
10. Saudari **Suci Rahmadana Anwar** dan **Sri Juliana** yang senantiasa menemani, memberi nasihat, menjadi tempat bertanya, serta dukungan untuk menyelesaikan skripsi ini.
11. Keluarga besar **Ilmu Komputer Unhas 2016** yang setia menemani dan membantu penulis selama menjalani pendidikan. Serta kakak-kakak dan adik-adik **Ilmu Komputer 2014, 2015, 2017, 2018** yang telah banyak membantu, semoga tetap semangat dalam mengejar impian.

12. Keluarga besar **HIPERMAWA Koperti Unhas** yang senantiasa memberikan naungan kekeluargaan dan dukungan.
13. Rekan-rekan **KKN Internasional Jepang Unhas Gel. 102** yang telah menjadi keluarga baru selama KKN dan menjadikan KKN sebagai momen yang berkesan.
14. Serta semua pihak yang telah banyak berpartisipasi, baik secara langsung maupun tidak langsung dalam penyusunan skripsi ini yang tidak sempat penulis sebutkan satu per satu.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna dikarenakan terbatasnya pengalaman dan pengetahuan yang dimiliki penulis. Oleh karena itu, penulis mengharapkan segala bentuk saran serta masukan bahkan kritik yang membangun dari berbagai pihak. Semoga tulisan ini memberikan manfaat kepada semua pihak yang membutuhkan dan terutama untuk penulis.

Makassar, 9 April 2021



SULAEMAN

NIM. H13116002



## **PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS**

Sebagai civitas akademik Universitas Hasanuddin, saya yang bertanda tangan di bawah ini:

Nama : SULAEMAN  
NIM : H131 16 002  
Program Studi : Sistem Informasi  
Departemen : Matematika  
Fakultas : Matematika dan Ilmu Pengetahuan Alam  
Jenis Karya : Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Hasanuddin **Hak Prediktor Royalti Noneksklusif** (*Non-exclusive Royalty-Free Right*) atas tugas akhir saya yang berjudul:

### **"IMPLEMENTASI FILTER SPASIAL LINEAR PADA VIDEO STREAM MENGUNAKAN *FPGA HARDWARE ACCELERATOR*"**

beserta perangkat yang ada (jika diperlukan). Terkait dengan hal diatas, maka pihak Universitas Hasanuddin berhak menyimpan, mengalih-media/format-kan, mengolah dalam bentuk pangkalan data (*database*), merawat dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian surat pernyataan ini saya buat dengan sebenarnya.

Dibuat di Makassar pada 9 April 2021

Yang menyatakan



(SULAEMAN)

## ABSTRAK

Berbagai macam akselerator telah dikembangkan dalam meningkatkan kinerja dan efisiensi energi untuk menangani komputasi berat, salah satu diantaranya yaitu FPGA. FPGA mampu menangani beban komputasi yang begitu berat sehingga dapat digunakan untuk *Digital Signal Processing*, *Image Processing*, *Neural Network*, dan sebagainya. Pada penelitian ini penulis mencoba mengkaji kinerja yang dimiliki ARM prosesor dan FPGA pada *FPGA Development Board* Xilinx PYNQ Z2 dalam penerapan filter spasial linear pada *video stream*. Kernel filter yang digunakan pada penelitian ini yaitu *average blur*, *gaussian blur*, *laplacian*, *sharpen*, *sobel horizontal* dan *sobel vertical*. Parameter yang digunakan untuk mengukur kinerja ARM prosesor dan FPGA yaitu waktu komputasi, *frame rate* (FPS), penggunaan CPU, penggunaan *memory*, *resident memory* (RES), *shared memory* (SHR) dan *virtual memory* (VIRT). Hasil implementasi menunjukkan rata-rata waktu komputasi yang dibutuhkan untuk menerapkan filter spasial linear pada 200 frame dengan ARM prosesor adalah 29.06 detik sedangkan dengan FPGA rata-rata hanya dibutuhkan 3.32 detik. Video hasil filter dengan ARM prosesor memperoleh rata-rata 6.95 fps sedangkan dengan FPGA rata-rata 60.37 fps. Penggunaan CPU pada FPGA lebih rendah daripada ARM prosesor. Secara umum penggunaan *memory*, *resident memory*, *shared memory* dan *virtual memory* pada ARM prosesor dan FPGA tidak jauh berbeda.

**Kata Kunci :** filter spasial linear, FPGA, ARM prosesor, *video stream*, *video processing*

## ABSTRACT

Various kinds of accelerators have been developed to improve performance and energy efficiency to handle heavy computations, one of which is FPGA. FPGA is capable of handling such a heavy computational load that it can be used for Digital Signal Processing, Image Processing, Neural Networks, etc. In this study, the authors tried to examine the performance of the ARM processor and the FPGA on the Xilin PYNQ Z2 FPGA Development Board in applying a linear spatial filter to the video stream. Kernel filters used in this study are the average blur, Gaussian blur, Laplacian, sharpen, Sobel horizontal, and Sobel vertical. The parameters used to measure the performance of ARM processors and FPGAs are runtime, frame rate (FPS), CPU usage, memory usage, resident memory (RES), shared memory (SHR), and virtual memory (VIRT). The average computation time required to apply linear spatial filters of 200 frames with an ARM processor is 29.06 seconds, while the average FPGA takes only 3.32 seconds. The filtered video with the ARM processor gets an average of 6.95 fps while the FPGA average is 60.37 fps. CPU usage on FPGAs is slightly lower than ARM processors. In general, the use of memory, resident memory, shared memory, and virtual memory on ARM processors and FPGAs is not much different.

**Keywords :** linear spatial filter, FPGA, ARM processor, video stream, video processing

## DAFTAR ISI

<b>HALAMAN JUDUL</b> . . . . .	<b>i</b>
<b>HALAMAN PERNYATAAN KEOTENTIKAN</b> . . . . .	<b>ii</b>
<b>HALAMAN PERSETUJUAN PEMBIMBING</b> . . . . .	<b>iii</b>
<b>HALAMAN PENGESAHAN</b> . . . . .	<b>iv</b>
<b>KATA PENGANTAR</b> . . . . .	<b>v</b>
<b>PERSETUJUAN PUBLIKASI KARYA ILMIAH</b> . . . . .	<b>viii</b>
<b>ABSTRAK</b> . . . . .	<b>ix</b>
<b>ABSTRACT</b> . . . . .	<b>x</b>
<b>DAFTAR ISI</b> . . . . .	<b>xi</b>
<b>DAFTAR TABEL</b> . . . . .	<b>xiv</b>
<b>DAFTAR GAMBAR</b> . . . . .	<b>xv</b>
<b>DAFTAR LAMPIRAN</b> . . . . .	<b>xvii</b>
<b>BAB I PENDAHULUAN</b> . . . . .	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Rumusan Masalah . . . . .	4
1.3 Tujuan Penelitian . . . . .	4
1.4 Batasan Masalah . . . . .	4
1.5 Manfaat Penelitian . . . . .	4
<b>BAB II TINJAUAN PUSTAKA</b> . . . . .	<b>5</b>
2.1 Landasan Teori . . . . .	5
2.1.1 Citra Digital . . . . .	5

2.1.2	Pengolahan Citra Digital . . . . .	6
2.1.3	Filter Spasial . . . . .	8
2.1.4	Kernel . . . . .	9
2.1.5	Konvolusi . . . . .	10
2.1.6	Video Stream . . . . .	12
2.1.7	FPGA . . . . .	12
2.1.8	Evaluasi Kinerja . . . . .	13
2.2	Penelitian Terkait . . . . .	18
2.2.1	Spatial Filtering Based Boundary Extraction in Underwater Images for Pipeline Detection: FPGA Implementation . . . . .	18
2.2.2	FPGA Implementation of Spatial Filtering techniques for 2D Images . . . . .	18
2.2.3	Features of Image Spatial Filters Implementation on FPGA . . . . .	19
2.2.4	An FPGA-Oriented Algorithm for Real-Time Filtering of Poisson Noise in Video Streams, with Application to X-Ray Fluoroscopy . . . . .	19
2.2.5	A real-time video denoising algorithm with FPGA implementation for Poisson-Gaussian noise . . . . .	19
<b>BAB III METODE PENELITIAN . . . . .</b>		<b>21</b>
3.1	Tahapan Penelitian . . . . .	21
3.2	Waktu dan Lokasi Penelitian . . . . .	22
3.3	Rancangan Sistem . . . . .	22
3.4	Instrumen Penelitian . . . . .	23
<b>BAB IV HASIL DAN PEMBAHASAN . . . . .</b>		<b>24</b>
4.1	Implementasi pada FPGA Development Board . . . . .	24
4.1.1	Penerapan Filter Spasial . . . . .	24
4.1.2	Penerapan Filter Spasial dengan Prosesor ARM dan FPGA . . . . .	28
4.1.3	Proses Evaluasi Kinerja . . . . .	29
4.2	Analisis Kinerja . . . . .	33
4.2.1	Waktu Komputasi . . . . .	33

4.2.2	Frame Rate (FPS) . . . . .	34
4.2.3	Penggunaan CPU . . . . .	35
4.2.4	Penggunaan Memory . . . . .	36
4.2.5	Resident Memory (RES) . . . . .	37
4.2.6	Shared Memory (SHR) . . . . .	39
4.2.7	Virtual Memory (VIRT) . . . . .	40
<b>BAB V</b>	<b>KESIMPULAN DAN SARAN . . . . .</b>	<b>42</b>
5.1	Kesimpulan . . . . .	42
5.2	Saran . . . . .	42
	<b>DAFTAR PUSTAKA . . . . .</b>	<b>43</b>
	<b>LAMPIRAN . . . . .</b>	<b>45</b>



## **DAFTAR TABEL**

4.1	Tabel perbandingan waktu komputasi dengan menggunakan 50 frame.	33
4.2	Tabel perbandingan waktu komputasi dengan menggunakan 200 frame.	34
4.3	Tabel perbandingan FPS dengan menggunakan prosesor ARM dan FPGA. . . . .	35
4.4	Tabel perbandingan penggunaan CPU dengan menggunakan prosesor ARM dan FPGA. . . . .	36
4.5	Tabel perbandingan penggunaan memory dengan menggunakan prosesor ARM dan FPGA. . . . .	37
4.6	Tabel perbandingan penggunaan resident memory (RES) dengan menggunakan prosesor ARM dan FPGA. . . . .	38
4.7	Tabel perbandingan penggunaan shared memory (SHR) dengan menggunakan prosesor ARM dan FPGA. . . . .	39
4.8	Tabel perbandingan penggunaan virtual memory (VIRT) dengan menggunakan prosesor ARM dan FPGA. . . . .	40

## DAFTAR GAMBAR

2.1	(a) Contoh citra biner, (b) contoh citra grayscale, (c) contoh citra warna.	6
2.2	Ilustrasi konvolusi pada citra. Sumber: <a href="https://indoml.com">https://indoml.com</a>	11
2.3	Struktur FPGA.	13
2.4	FPGA Board Xilinx PYNQ-Z2.	14
2.5	Tampilan program top pada Linux.	15
2.6	Kuadran pembagian memory pada Linux.	16
3.1	Flowchart tahapan penelitian.	21
3.2	Rancangan sistem.	22
4.1	Contoh Frame Grayscale.	25
4.2	Hasil filter Average Blur.	26
4.3	Hasil filter Gaussian Blur.	26
4.4	Hasil filter Laplacian.	27
4.5	Hasil filter Sharpening.	27
4.6	Hasil filter Sobel Horizontal.	28
4.7	Hasil filter Sobel Vertical.	28
4.8	Menghitung waktu komputasi dengan library time di Python.	30
4.9	Tampilan program <b>top</b> .	30
4.10	Menampilkan PID sebuah proses dengan bahasa pemrograman Python.	31
4.11	Menjalankan program <b>top</b> kemudian menyimpan hasilnya pada file arm-laplacian1.txt.	31
4.12	Potongan isi file arm-laplacian1.txt.	32
4.13	Isi file arm-laplacian1.csv.	32
4.14	Grafik perbandingan waktu komputasi dengan 50 frame dan 200 frame	33
4.15	Grafik perbandingan waktu komputasi dengan 50 frame dan 200 frame	34
4.16	Grafik perbandingan FPS dengan menggunakan prosesor ARM dan FPGA.	35
4.17	Grafik perbandingan penggunaan CPU dengan menggunakan prosesor ARM dan FPGA.	36

4.18 Grafik perbandingan penggunaan memory dengan menggunakan prosesor ARM dan FPGA. . . . .	37
4.19 Grafik perbandingan penggunaan resident memory (RES) dengan menggunakan prosesor ARM dan FPGA. . . . .	38
4.20 Grafik perbandingan penggunaan shared memory (SHR) dengan menggunakan prosesor ARM dan FPGA. . . . .	39
4.21 Grafik perbandingan penggunaan virtual memory (VIRT) dengan menggunakan prosesor ARM dan FPGA. . . . .	40

## DAFTAR LAMPIRAN

Sourcode Program . . . . .	46
Data Hasil Percobaan Waktu Komputasi dan FPS . . . . .	51
Data Hasil Percobaan ARM Average Blur . . . . .	54
Data Hasil Percobaan ARM Gaussian Blur . . . . .	61
Data Hasil Percobaan ARM Laplacian . . . . .	64
Data Hasil Percobaan ARM Sharpening . . . . .	68
Data Hasil Percobaan ARM Sobel Horizontal . . . . .	74
Data Hasil Percobaan ARM Sobel Vertical . . . . .	78
Data Hasil Percobaan FPGA Average Blur . . . . .	81
Data Hasil Percobaan FPGA Gaussian Blur . . . . .	86
Data Hasil Percobaan FPGA Laplacian . . . . .	89
Data Hasil Percobaan FPGA Sharpening . . . . .	92
Data Hasil Percobaan FPGA Sobel Horizontal . . . . .	95
Data Hasil Percobaan FPGA Sobel Vertical . . . . .	98

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Citra digital merupakan citra yang dihasilkan dari pengolahan secara digital dengan merepresentasikan citra secara numerik dengan nilai-nilai diskrit. Suatu citra digital dapat direpresentasikan dalam bentuk matriks dengan fungsi  $f(x,y)$  yang terdiri dari M kolom dan N baris. Perpotongan antara baris dan kolom disebut sebagai piksel (Gonzalez dkk. 2001).

Pengolahan citra digital merupakan proses mengolah piksel di dalam citra secara digital untuk tujuan tertentu. Berdasarkan tingkat pemrosesannya pengolahan citra digital dikelompokkan menjadi tiga kategori, yaitu: *low-level*, *mid-level* dan pemrosesan *high-level*. Pemrosesan *low-level* dilakukan dengan operasi primitif seperti *image preprocessing* untuk mengurangi derau (*noise*), memperbaiki kontras citra dan mempertajam citra (*sharpening*). Pemrosesan *mid-level* melibatkan tugas-tugas seperti segmentasi atau mempartisi gambar menjadi beberapa bagian atau objek, deskripsi objek untuk dilakukan pemrosesan lanjutan, dan klasifikasi objek yang terdapat dalam citra digital. Pemrosesan *high-level* merupakan proses tingkat lanjut dari dua proses sebelumnya, dilakukan untuk mendapat informasi lebih yang terkandung dalam citra seperti *pattern recognition*, *template matching*, *image analysis* dan sebagainya (Gonzalez dkk. 2001).

Konsep filter spasial pada pengolahan citra digital berasal dari penerapan transformasi Fourier untuk pemrosesan sinyal pada domain frekuensi. Istilah filter spasial ini digunakan untuk membedakan proses ini dengan filter pada domain frekuensi. Proses filter dilakukan dengan cara menggeser filter kernel dari titik ke titik dalam citra digital. Istilah *mask*, *kernel*, *template*, dan *window* merupakan istilah yang sama dan sering digunakan dalam pengolahan citra digital. Dalam penelitian ini peneliti menggunakan istilah kernel untuk istilah tersebut. Konsep filter spasial linear mirip seperti konsep konvolusi pada domain frekuensi, dengan alasan tersebut filter spasial linear biasa disebut juga konvolusi sebuah kernel dengan citra digital (Gonzalez dkk. 2001). Proses filter dalam pengolahan citra digital dilakukan dengan

memanipulasi sebuah citra menggunakan kernel untuk menghasilkan citra yang baru, sehingga dengan kernel yang berbeda maka citra hasil yang didapat juga akan berbeda.

Video *stream* dapat dipandang sebagai serangkaian citra digital berturut-turut (Zhao 2015). Berbeda dengan format video lainnya, video *stream* ini tidak disimpan pada media penyimpanan sebagai file video melainkan setiap *frame* langsung disalurkan dari sumber (*source*) ke penerima, dalam hal ini FPGA. Dengan menganggap video *stream* adalah kumpulan citra digital (*frame*) maka dapat dilakukan metode pengolahan seperti pada citra digital, termasuk filter spasial.

Frame per second (*fps*) atau *frame rate* adalah banyaknya *frame* yang ditampilkan per detik. Semakin tinggi *fps* sebuah video maka semakin baik pula gerakan yang dapat ditampilkan karena dibentuk dari *frame* yang lebih banyak. Namun dengan jumlah *frame* yang lebih besar tentu dibutuhkan juga *resource* yang lebih besar dalam pengolahan video tersebut (Kowalczyk dkk. 2018).

Untuk meningkatkan kinerja dan efisiensi energi dari sebuah program, berbagai jenis akselerator telah dikembangkan, salah satu diantaranya yaitu FPGA (Cong dkk. 2018). *Field Programmable Gate Arrays* atau FPGA adalah perangkat semikonduktor yang berbasis *matriks configurable logic block* (CLBs) yang terhubung melalui interkoneksi yang dapat diprogram. FPGA dapat diprogram ulang dengan aplikasi atau fungsi yang diinginkan setelah *manufacturing*. Fitur ini yang membedakan FPGA dengan *Application Specific Integrated Circuits* (ASICs), yang dibuat khusus untuk tugas tertentu saja (Xilinx 2020).

FPGA telah menunjukkan kinerja yang sangat tinggi di dalam banyak aplikasi dalam pemrosesan citra. Namun CPU dan CPU terbaru memiliki potensi kinerja tinggi untuk masalah-masalah tersebut. CPU terbaru mendukung *multi-core*, dimana masing-masing *core* mendukung SIMD (*Single Instruction, Multiple Data*) yang telah dikembangkan dan dijalankan hingga 16 operasi pada 128 bit data dalam satu *clock cycle*. GPU terbaru mendukung sejumlah besar *core* yang berjalan secara paralel, dan kinerja puncaknya mengungguli CPU (Asano dkk. 2009).



Paralelisme dalam SIMD pada CPU terbatas, tetapi frekuensi operasional CPU sangatlah tinggi, dan CPU diharapkan dapat menunjukkan kinerja yang tinggi dalam aplikasi yang dimana *cache memory* berjalan dengan baik. Ukuran *cache memory* cukup besar untuk menyimpan seluruh citra di banyak aplikasi pemrosesan citra, dan CPU dapat menjalankan algoritma yang sama dengan FPGA meskipun *bandwidth memory* yang dibutuhkan tinggi (Asano dkk. 2009).

Frekuensi operasional GPU lebih cepat dibandingkan dengan FPGA, namun sedikit lebih lambat dibandingkan dengan CPU. Akan tetapi, GPU mendukung banyak *core* yang berjalan secara paralel sehingga kinerja puncaknya mengungguli CPU. Namun *core*-nya dikelompokkan, dan transfer data antara kelompok sangatlah lambat. Selain itu, ukuran *local memory* yang disediakan masing-masing kelompok sangat kecil. Karena keterbatasan ini, GPU tidak dapat menjalankan algoritma yang sama seperti FPGA dalam beberapa masalah aplikasi (Asano dkk. 2009).

Pada FPGA terdahulu tidak terdapat prosesor untuk menjalankan *software* apapun, sehingga ketika ingin mengimplementasikan aplikasi haruslah merancang sirkuit dari awal. Sebagian besar FPGA sekarang telah dirangkai dengan prosesor dalam satu *board*, sering disebut sebagai FPGA Development Board. Xilinx PYNQ-Z2 dibangun dari prosesor ARM Cortex-A9, sehingga dapat menjalankan beberapa *software* seperti *python* tanpa harus merancang sirkuitnya dari awal. Akan tetapi, kinerja yang dimiliki oleh prosesor ARM pada FPGA Development Board tentu berbeda dengan kinerja fungsi arsitektur FPGA itu sendiri sehingga dapat dikaji lebih dalam mengenai perbandingan kinerja dari keduanya.

Berdasarkan uraian permasalahan di atas, peneliti ingin melakukan penelitian dengan judul "Implementasi Filter Spasial Linear pada Video *Stream* menggunakan FPGA *Hardware Accelerator*" untuk menunjukkan kinerja dari prosesor ARM dan FPGA pada Xilinx PYNQ-Z2 FPGA Development Board.

## 1.2 Rumusan Masalah

Adapun rumusan masalah dalam penelitian ini yaitu:

1. Bagaimana cara implementasi filter spasial linear pada video *stream* menggunakan FPGA?
2. Bagaimana kinerja FPGA dalam mengimplementasikan filter spasial linear pada video *stream*?

## 1.3 Tujuan Penelitian

Adapun tujuan dari penelitian ini yaitu:

1. Mampu melakukan implementasi filter spasial linear pada video *stream* menggunakan FPGA.
2. Mengetahui kinerja FPGA dalam mengimplementasikan filter spasial linear pada video *stream*.

## 1.4 Batasan Masalah

Berikut ini merupakan beberapa batasan dalam penelitian ini.

1. Filter kernel yang digunakan berukuran 3x3.
2. Video *stream* yang digunakan dalam penelitian ini beresolusi 720p.
3. Setiap frame dari video *stream* diubah menjadi citra grayscale sebelum dilakukan penerapan filter spasial.
4. FPGA *Board* yang digunakan adalah Xilinx PYNQ-Z2 dengan processor 650MHz dual-core ARM Cortex-A9.

## 1.5 Manfaat Penelitian

Hasil dari penelitian ini diharapkan dapat memberikan pemahaman tentang penerapan filter spasial pada video *stream* dengan FPGA Development Board. Selain itu, penelitian ini juga diharapkan dapat menjadi rujukan untuk melihat kinerja FPGA PYNQ-Z2 dalam mengimplementasikan filter spasial linear pada video *stream*.

## **BAB II**

### **TINJAUAN PUSTAKA**

#### **2.1 Landasan Teori**

##### **2.1.1 Citra Digital**

Citra digital dapat didefinisikan sebagai fungsi  $f(x,y)$  berukuran M baris dan N kolom, dengan  $x$  dan  $y$  adalah koordinat spasial, dan amplitudo  $f$  di titik koordinat  $(x,y)$  dinamakan intensitas atau tingkat keabuan dari citra pada citra tersebut (Putra 2010). Pada umumnya warna dasar dalam citra RGB menggunakan penyimpanan 8 bit untuk menyimpan data warna, yang berarti setiap warna mempunyai gradasi sebanyak 255 warna. Dewasa ini, citra digital dapat menggunakan 16 bit untuk menyimpan data warna dasarnya, hal ini menyebabkan semakin banyak gradasi warnanya sehingga citra yang dihasilkan memiliki tingkat warna yang jauh lebih banyak. Namun tentu saja hal ini mengakibatkan ukuran file citra digital yang dihasilkan juga menjadi semakin besar walaupun dengan ukuran yang sama. Berdasarkan jenis warnanya citra digital dibagi menjadi 3 jenis yaitu citra biner, citra grayscale dan citra warna.

##### **2.1.1.1 Citra Biner (monokrom)**

Citra biner adalah citra yang hanya memiliki dua warna saja yaitu hitam dan putih. Warna hitam direpresentasikan dengan 1 dan warna putih direpresentasikan dengan 0. Dibutuhkan 1 bit di memori untuk menyimpan nilai warna pada setiap piksel citra biner. Contoh citra biner dapat dilihat pada gambar 2.1(a).

##### **2.1.1.2 Citra Grayscale**

Banyaknya warna pada citra *grayscale* tergantung pada jumlah bit yang disediakan di memori untuk menampung kebutuhan warna pada citra ini. Citra *grayscale* yang 2 bit memiliki 4 gradasi warna, citra *grayscale* 3 bit memiliki 8 gradasi warna, citra *grayscale* dengan 8 bit memiliki 256 gradasi warna dan seterusnya. Semakin besar jumlah bit warna yang disediakan di memori, semakin banyak dan semakin halus gradasi warna yang terbentuk. Pada umumnya citra digital *grayscale*

menggunakan 8 bit memori dengan derajat keabuan dari 0 sampai 255. Contoh citra *grayscale* dapat dilihat pada gambar 2.1(b).

### 2.1.1.3 Citra Warna

Setiap piksel pada citra warna mewakili warna yang merupakan kombinasi dari tiga warna dasar (RGB = red, green, blue). Pada umumnya setiap warna dasar menggunakan penyimpanan 8 bit, yang berarti setiap warna mempunyai gradasi sebanyak 255 warna. Berarti setiap piksel mempunyai kombinasi warna sebanyak  $255 \times 255 \times 255 = 16$  juta warna lebih. Contoh citra warna dapat dilihat pada gambar 2.1(c).



Gambar 2.1: (a) Contoh citra biner, (b) contoh citra grayscale, (c) contoh citra warna.

### 2.1.2 Pengolahan Citra Digital

Pengolahan citra digital merupakan proses mengolah piksel-piksel di dalam citra secara digital untuk tujuan tertentu. Berdasarkan tingkat pemrosesannya pengolahan citra digital dikelompokkan menjadi tiga kategori, yaitu: *low-level*, *mid-level* dan pemrosesan *high-level*. Pemrosesan *low-level* dilakukan dengan operasi primitif seperti *image preprocessing* untuk mengurangi derau (*noise*), memperbaiki kontras citra dan mempertajam citra (*sharpening*). Karakteristik dari pemrosesan *low-level* yaitu keluaran atau hasil dari pemrosesannya berupa citra digital. Pemrosesan *mid-level* melibatkan tugas-tugas seperti segmentasi (mempartisi gambar menjadi beberapa bagian atau objek), deskripsi objek untuk dilakukan pemrosesan lanjutan, dan klasifikasi objek dalam citra digital. Karakteristik dari pemrosesan

*mid-level* yaitu keluaran atau hasilnya berupa atribut atau fitur seperti, kontur, tepi, atau objek yang terdapat dalam citra tersebut. Pemrosesan *high-level* merupakan proses tingkat lanjut dari dua proses sebelumnya, dilakukan untuk mendapat informasi lebih yang terkandung dalam citra (Gonzalez dkk. 2001).

Berdasarkan tujuannya pengolahan citra juga dapat dibagi menjadi beberapa bagian yaitu: *image enhancement*, *image restoration*, *image analysis* dan *image compression* (Silva dkk. 2005).

#### **2.1.2.1 Image Enhancement**

*Image Enhancement* adalah metode pengolahan citra digital untuk membuat citra tampak lebih baik atau dilakukan peningkatan untuk analisis tertentu. Namun hal ini dapat menyebabkan pengorbanan aspek lain dari citra tersebut. Penerapan filter, penghalusan citra, memperbaiki kontras dan morfologi citra adalah contoh *image enhancement*.

#### **2.1.2.2 Image Restoration**

*Image restoration* adalah metode pengolahan citra untuk memulihkan citra dari penurunan kualitas atau citra yang rusak karena derau (*noise*). Pada dasarnya metode ini berbeda dengan *image enhancement* yang berkaitan dengan ekstraksi fitur pada citra.

#### **2.1.2.3 Image Analysis**

*Image analysis* adalah metode pengolahan citra untuk menghitung besaran kuantitatif dari citra untuk menghasilkan deskripsinya. Metode ini dilakukan dengan mengekstraksi ciri-ciri tertentu yang membantu dalam identifikasi objek (Silva dkk. 2005).

#### **2.1.2.4 Image Compression**

Metode ini dilakukan agar citra dapat direpresentasikan dalam bentuk yang lebih kompak sehingga memerlukan memori yang lebih sedikit. Metode ini dapat

dilakukan dengan mengurangi redundansi dari data-data yang terdapat dalam citra sehingga dapat disimpan atau ditransmisikan secara efisien.

### 2.1.3 Filter Spasial

Konsep filter spasial pada pengolahan citra digital berasal dari penerapan transformasi Fourier untuk pemrosesan sinyal pada domain frekuensi. Istilah filter spasial ini digunakan untuk membedakan proses ini dengan filter pada domain frekuensi. Proses filter dilakukan dengan cara menggeser filter kernel dari titik ke titik dalam citra digital. Istilah *mask*, *kernel*, *template*, dan *window* merupakan istilah yang sama dan sering digunakan dalam pengolahan citra digital (Gonzalez dkk. 2001). Dalam penelitian ini peneliti menggunakan istilah kernel untuk istilah tersebut.

Proses filter dalam pengolahan citra digital dilakukan dengan memanipulasi sebuah citra menggunakan kernel untuk menghasilkan citra yang baru, sehingga dengan kernel yang berbeda maka citra hasil yang didapat juga akan berbeda.

#### 2.1.3.1 Operator Linear dan Non-linear

Didefinisikan H sebuah operator dengan *input* dan *output* adalah citra digital. H dikatakan operator linear jika untuk untuk sembarang gambar  $f$  dan  $g$ , dan untuk sembarang skalar  $a$  dan  $b$  berlaku,

$$H(af + bg) = aH(f) + bH(g) \quad (2.1)$$

Dengan kata lain hasil dari operator linear dengan jumlahan dua buah citra (yang telah dikali dengan konstanta  $a$  dan  $b$ ) identik dengan hasil operator linear pada masing-masing gambar, dikali dengan konstanta yang sama, kemudian hasilnya dijumlahkan. Sebagai contoh, sebuah operator dengan fungsi yang menjumlahkan  $K$  citra adalah operator linear. Operator yang menghitung nilai mutlak dari perbedaan dua gambar adalah tidak linear. Operator yang tidak memenuhi persamaan (2.1) dikatakan non-linear (Gonzalez dkk. 2001).



## 2.1.4 Kernel

### 2.1.4.1 Average Blur

*Average blur* atau biasa juga disebut *box filter* adalah salah satu filter yang digunakan untuk menghaluskan citra dan mengurangi derau. Secara sederhana nilai sebuah piksel yang baru adalah nilai rata-rata dari nilai piksel tersebut dengan nilai piksel tetangganya (Kowalczyk dkk. 2018). Berikut kernel *Average blur* yang digunakan dalam penelitian ini:

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (2.2)$$

### 2.1.4.2 Gaussian Blur

Filter ini juga digunakan untuk menghaluskan citra dan mengurangi derau. Idennya mirip seperti *Average blur*, nilai piksel yang baru dibentuk dari nilai piksel tetangganya, tetapi dengan memberikan bobot yang lebih kuat pada nilai pikselnya sendiri diikuti dengan bobot yang lebih rendah pada piksel atas, bawah dan sampingnya (Ustyukov dkk. 2019). Berikut kernel gaussian blur filter yang digunakan dalam penelitian ini:

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad (2.3)$$

### 2.1.4.3 Sobel

Filter Sobel termasuk *high-pass* filter yang umum digunakan untuk deteksi tepi pada citra. Sobel memiliki dua kernel untuk deteksi tepi yaitu kernel sobel vertikal untuk mendeteksi tepi secara vertikal dan kernel sobel horizontal yang mendeteksi tepi secara horizontal (Kowalczyk dkk. 2018). Berikut kernel Sobel vertikal dan

horizontal yang digunakan dalam penelitian ini:

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (2.4)$$

#### 2.1.4.4 Laplacian

Filter ini dapat digunakan untuk deteksi tepi pada citra karena sifatnya yang sensitif dengan perubahan intensitas yang cepat (Jingbo dkk. 2011). Tidak seperti Sobel yang menggunakan dua kernel untuk mendeteksi tepi secara vertikal dan horizontal, disini hanya digunakan sebuah kernel yang dapat digunakan untuk deteksi tepi secara vertikal dan horizontal sekaligus. Berikut kernel Laplacian yang digunakan dalam penelitian ini:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad (2.5)$$

#### 2.1.4.5 Sharpening

Sharpening filter digunakan untuk memperjelas detail halus dalam citra atau untuk meningkatkan detail pada citra yang *blur*, baik karena kesalahan ataupun karena efek dari metode akuisisi citra tertentu (Yang 2013). Berikut kernel untuk filter *sharpening* yang digunakan dalam penelitian ini:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad (2.6)$$

#### 2.1.5 Konvolusi

Konsep filter spasial linear mirip seperti konsep konvolusi pada domain frekuensi, dengan alasan tersebut filter spasial linear biasa disebut juga konvolusi sebuah kernel dengan citra digital (Gonzalez dkk. 2001). Konvolusi pada fungsi  $f(x)$

dan  $g(x)$  didefinisikan pada persamaan 2.7.

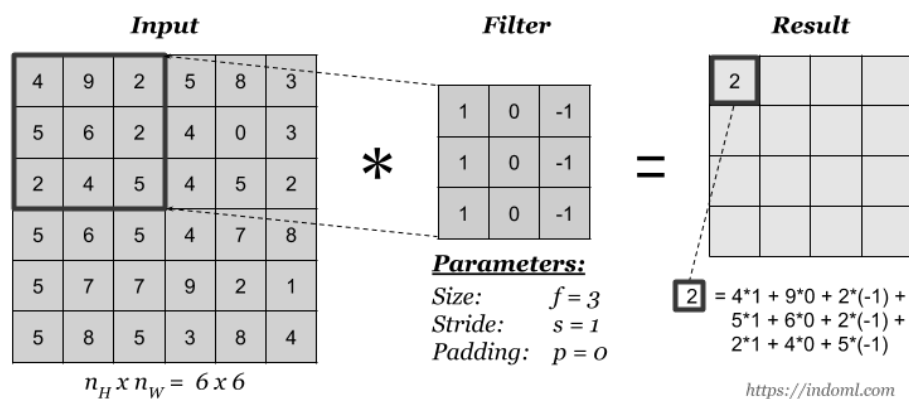
$$h(x) = f(x) * g(x) = \int_{-\infty}^{\infty} f(a)g(x - a)da \quad (2.7)$$

dimana tanda \* menyatakan operator konvolusi, dan peubah  $a$  adalah peubah bantu. Untuk fungsi diskrit, konvolusi didefinisikan pada persamaan 2.8.

$$h(x) = f(x) * g(x) = \sum_{a=-\infty}^{\infty} f(a)g(x - a) \quad (2.8)$$

Pada operasi konvolusi diatas,  $g(x)$  disebut kernel konvolusi atau filter kernel. Kernel  $g(x)$  dioperasikan secara bergeser pada sinyal masukan  $f(x)$ . Jumlah perkalian kedua fungsi pada setiap titik merupakan hasil konvolusi yang dinyatakan dengan keluaran  $h(x)$  (Rinaldi 2004).

Pada gambar (2.2) diilustrasikan bagaimana proses konvolusi pada citra digital yang direpresentasikan dalam bentuk matriks. Operasi konvolusi dilakukan pada matriks input berukuran 6x6 dengan filter berukuran 3x3. Hasil konvolusinya ditampilkan pada matriks *result*.



Gambar 2.2: Ilustrasi konvolusi pada citra. Sumber: <https://indoml.com>

Jika hasil konvolusi menghasilkan nilai piksel negatif, maka nilai tersebut dijadikan 0, sebaliknya jika hasil konvolusi menghasilkan nilai piksel yang melebihi nilai keabuan maksimum, maka nilai tersebut dijadikan ke nilai keabuan maksimum pada citra tersebut (Sutoyo dkk. 2009).

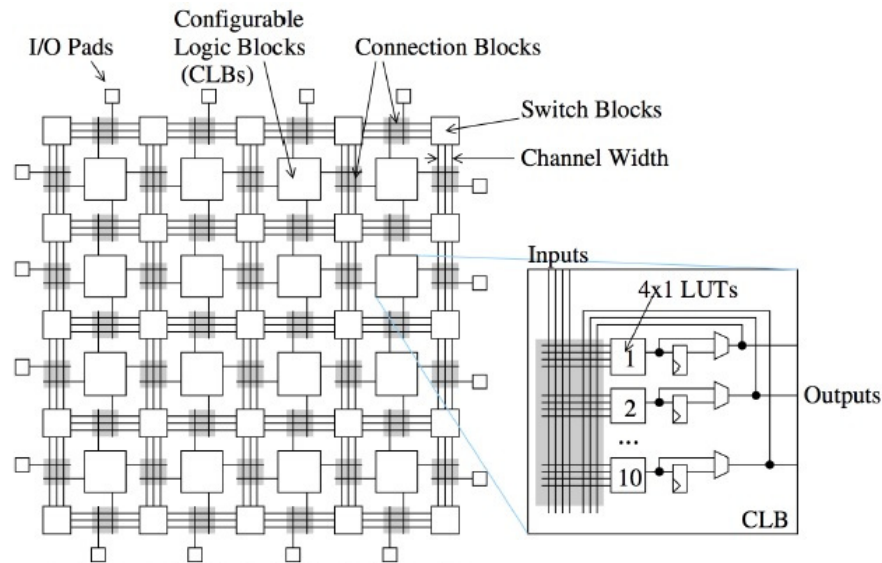
### 2.1.6 Video Stream

Video *stream* dapat dipandang sebagai serangkaian citra digital berturut-turut (Zhao 2015). Berbeda dengan format video lainnya, video *stream* ini tidak disimpan pada media penyimpanan sebagai file dengan format video melainkan langsung disalurkan setiap framenya dari sumber (*source*) ke penerima, dalam hal ini FPGA. Dengan menganggap Video *stream* adalah kumpulan citra digital (*frame*) maka dapat dilakukan metode pengolahan seperti pada citra digital, termasuk penerapan filter spasial.

### 2.1.7 FPGA

*Field Programmable Gate Arrays* atau FPGA adalah perangkat semikonduktor yang berbasis *matriks configurable logic block* (CLBs) yang terhubung melalui interkoneksi yang dapat diprogram. FPGA dapat diprogram ulang ke aplikasi atau fungsi yang diinginkan setelah *manufacturing*. Fitur ini yang membedakan FPGA dengan *Application Specific Integrated Circuits* (ASICs), yang dibuat khusus untuk tugas tertentu saja (Xilinx 2020).

Sebuah *microprocessor* menerima instruksi berupa kode 1 atau 0, kode-kode ini selanjutnya diinterpretasikan oleh komputer untuk menjalankan perintah yang diberikan. *Microprocessor* ini membutuhkan intruksi berupa kode secara terus menerus untuk menjalankan fungsinya. Sedangkan pada FPGA hanya dibutuhkan sekali konfigurasi *chip* setiap kali dinyalakan. Membuat atau mengunduh *bitstream* yang menentukan fungsi logika dilakukan oleh *logic elements* (LEs), sebuah sirkuit dapat dibuat dengan mengabungkan beberapa LEs menjadi satu kesatuan. Setelah *bitstream* dipasang, FPGA tidak perlu lagi membaca instruksi berupa 1 dan 0, berbeda dengan *microprocessor* yang selalu membutuhkan instruksi (Cheung 2019). Secara tradisional, untuk membuat sebuah desain FPGA, aplikasi dideskripsikan menggunakan *Hardware Description Language* (HDL) seperti Verilog atau VHDL sehingga menghasilkan sebuah *bitstream* FPGA.



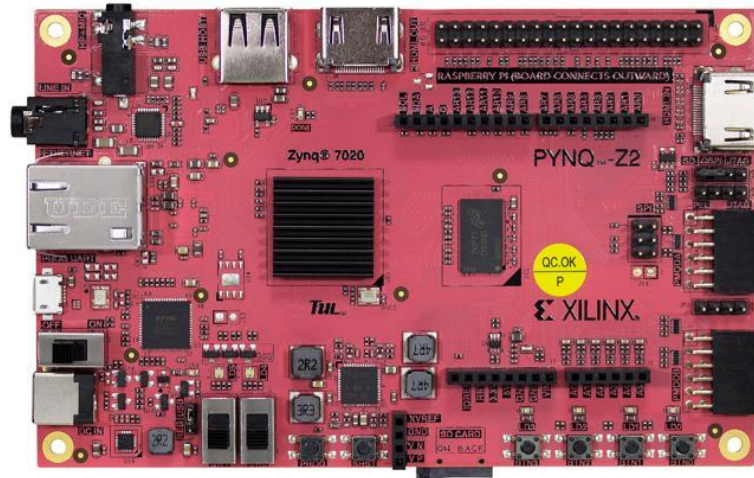
Gambar 2.3: Struktur FPGA.

### 2.1.7.1 FPGA Development Board

Pada FPGA terdahulu tidak terdapat *processor* (CPU) untuk menjalankan software apapun, sehingga ketika ingin mengimplementasikan aplikasi haruslah merancang sirkuit dari awal, seperti mengonfigurasi FPGA sesederhana gerbang logika OR atau serumit *multi-core processor* (Biswas 2019). Dewasa ini telah dikembangkan *FPGA Development Board* atau biasa disebut juga *FPGA Board* yaitu teknologi FPGA yang dirangkai dalam sebuah *board* dan dilengkapi dengan *microprocessor* dan beberapa *interface IO* untuk menjalankan tugas tertentu. Umumnya *FPGA Board* telah dilengkapi dengan interface untuk mengakses dan menerapkan desain sirkuitnya. Xilinx, Altera dan Intel adalah produsen *FPGA Board* yang terkenal. *FPGA Board* yang digunakan dalam penelitian ini yaitu Xilinx PYNQ-Z2 dengan Jupyter Notebook sebagai *interface* untuk mengakses dan menjalankan program pada penelitian ini. Bentuk *FPGA Board* Xilinx PYNQ-Z2 dapat dilihat pada gambar (2.4)

### 2.1.8 Evaluasi Kinerja

Pada penelitian ini peneliti menggunakan waktu komputasi, *frame rate* (FPS), penggunaan CPU, penggunaan memory, penggunaan resident memory (RES), shared memory (SHR), dan virtual memory (VIRT) untuk mengukur kinerja pada penerapan



Gambar 2.4: FPGA Board Xilinx PYNQ-Z2.

filter spasial linear dengan prosesor ARM dan FPGA.

### 2.1.8.1 Waktu Komputasi

Waktu komputasi yang dimaksud oleh peneliti adalah durasi yang dibutuhkan sebuah kernel untuk melakukan filter spasial linear terhadap beberapa *frame* input. Waktu komputasi ini diperoleh dengan cara menghitung selisih waktu selesai dengan waktu dimulai penerapan filter spasial linear pada *frame* input.

$$\text{waktu komputasi} = \text{waktu selesai} - \text{waktu mulai} \quad (2.9)$$

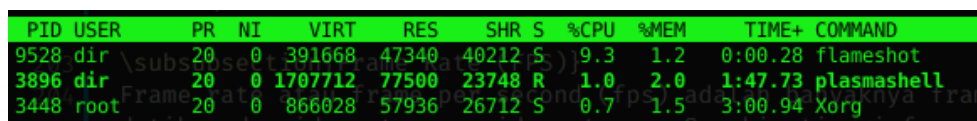
### 2.1.8.2 Frame Rate (FPS)

*Frame rate* atau *frame per second* (fps) adalah banyaknya *frame* yang ditampilkan per detik pada video ataupun video *stream*. Semakin tinggi fps sebuah video maka semakin halus pula gerakan yang dihasilkan. Sebaliknya video dengan fps rendah akan menghasilkan gerakan yang kurang baik. *Frame rate* atau fps dapat dihitung dengan cara membagi jumlah *frame* dengan waktu komputasinya seperti pada persamaan 2.10 (Madhusudana dkk. 2020).

$$\text{fps} = \frac{\text{jumlah frame}}{\text{waktu komputasi}} \quad (2.10)$$

### 2.1.8.3 Penggunaan CPU

Pada penelitian ini peneliti menggunakan fitur yang tersedia pada sistem operasi Linux yang berjalan di FPGA Development Board untuk melihat persentase penggunaan CPU pada proses penerapan filter spasial linear. Tampilan dari program ini dapat dilihat pada gambar 2.5. Program ini menampilkan informasi tentang proses-proses yang berjalan pada sistem operasi seperti ID sebuah proses, user yang menjalankan proses tersebut, *memory* yang digunakan, status sebuah proses, persentase CPU yang digunakan dan lainnya.



PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
9528	dir	20	0	391668	47340	40212	S	9.3	1.2	0:00.28	flameshot
3896	dir	20	0	1707712	77500	23748	R	1.0	2.0	1:47.73	plasmashell
3448	root	20	0	866028	57936	26712	S	0.7	1.5	3:00.94	Xorg

Gambar 2.5: Tampilan program *top* pada Linux.

Status sebuah proses pada program ini dinyatakan dengan beberapa singkatan, diantaranya yaitu:

- I = *idle*
- R = *running*
- S = *sleeping*
- Z = *zombie*
- D = *uninterruptible sleep*
- T = *stopped by job control signal*
- t = *stopped by debugger during trace*

Pada CPU yang multi-core sebuah proses akan ditampilkan persentase penggunaan CPUnya berdasarkan core yang digunakan oleh proses tersebut. Jika mode Irix pada program *top* dimatikan, maka program akan berjalan pada mode Solaris di mana penggunaan CPU sebuah proses yang ditampilkan akan dibagi dengan jumlah total core yang ada pada CPU (Kerrisk 2020).

### 2.1.8.4 Penggunaan Memory

Pada sistem operasi linux *memory* dibagi menjadi tiga jenis (Kerrisk 2020). Pertama yaitu *memory* fisik, sumber daya terbatas di mana kode dan data harus berada

saat dijalankan atau direferensikan. Berikutnya adalah *memory swap*, yaitu *memory* yang berguna untuk membantu kerja *memory* fisik, data dari *memory* fisik akan disimpan pada *swap* dan kemudian diambil kembali jika terlalu banyak permintaan pada *memory* fisik. Ketiga yaitu *virtual memory*, sumber daya yang hampir tidak terbatas yang digunakan untuk tujuan berikut (Silbershatz dkk. 2009):

- *abstraction*, bebas dari alamat / batas *memory* fisik
- *isolation*, setiap proses dalam ruang alamat terpisah
- *sharing*, pemetaan tunggal dapat memenuhi banyak kebutuhan
- *flexibility*, menetapkan alamat virtual ke data

Terlepas dari bentuk *memory* mana yang mungkin digunakan, semua dikelola sebagai *pages* (biasanya 4096 byte). Penggunaan *memory* berhubungan dengan *memory* fisik dan *swap* untuk sistem secara keseluruhan. Untuk setiap proses yang berjalan, setiap *memory* page dibatasi ke satu kuadran seperti pada gambar 2.6. Baik *memory* fisik dan *memory* virtual dapat menyertakan salah satu dari empat kuadran, sementara file *swap* hanya mencakup kuadran 1 sampai 3. Memori di kuadran 4, bertindak sebagai file *swap* khusus ketika dimodifikasi (Kerrisk 2020).

	Private	Shared
<b>Anonymous</b>	1	2
. stack		
. malloc()		
. brk()/sbrk()		. POSIX shm*
. mmap(PRIVATE, ANON)		. mmap(SHARED, ANON)
<b>File-backed</b>	3	4
. mmap(PRIVATE, fd)		. mmap(SHARED, fd)
. pgms/shared libs		

Gambar 2.6: Kuadran pembagian memory pada Linux.

### 2.1.8.5 Virtual Memory (VIRT)

*Virtual memory* menggunakan disk sebagai perpanjangan dari RAM sehingga ukuran efektif *memory* yang dapat digunakan bertambah secara bersamaan. Kernel akan menulis konten dari blok *memory* yang saat ini tidak digunakan ke hard disk sehingga *memory* dapat digunakan untuk tujuan lain. Ketika konten asli dibutuhkan lagi, mereka dibaca kembali ke dalam *memory*. Ini semua dibuat transparan



sepenuhnya bagi pengguna. Program yang berjalan di Linux hanya melihat jumlah *memory* yang tersedia lebih besar dan tidak memperhatikan bahwa sebagian dari program tersebut berada di disk dari waktu ke waktu. Tentu saja, membaca dan menulis hard disk lebih lambat daripada menggunakan *memory* fisik, sehingga program tidak berjalan secepat itu. Bagian dari hard disk yang digunakan sebagai *memory* virtual disebut ruang swap (S dkk. 2020).

Kolom VIRT pada program **top** menunjukkan jumlah total *memory* virtual yang digunakan oleh proses. Ini mencakup semua kode, data dan *shared libraries* ditambah dengan *pages* yang telah ditukar dan *pages* yang telah dipetakan tetapi tidak digunakan (Kerrisk 2020).

#### **2.1.8.6 Resident Memory (RES)**

*Resident memory* adalah bagian dari ruang alamat virtual (VIRT) yang mewakili *memory* fisik yang tidak ditukar yang sedang digunakan tugas. *Resident memory* ini juga merupakan penjumlahan dari RSan, RSfd dan Bidang RSsh. Ini dapat mencakup private anonymous *pages*, halaman pribadi yang dipetakan ke file (termasuk *program images* dan *shared libraries*) ditambah *shared anonymous pages*. Semua *memory* tersebut didukung oleh file *swap* yang direpresentasikan secara terpisah pada SWAP. *Resident memory* ini juga dapat menyertakan *pages* yang didukung *shared file-backed* yang apabila dimodifikasi, maka akan bertindak sebagai file *swap* khusus dan karenanya tidak akan pernah memengaruhi SWAP (Kerrisk 2020).

#### **2.1.8.7 Shared Memory (SHR)**

*Shared memory* adalah bagian dari *resident memory* (RES) yang dapat digunakan oleh proses lain. Termasuk *anonymous pages* dan *shared file-backed pages*. Ini juga termasuk private *pages* dipetakan ke file yang mewakili *program images* dan *shared libraries* (Kerrisk 2020).

## **2.2 Penelitian Terkait**

### **2.2.1 Spatial Filtering Based Boundary Extraction in Underwater Images for Pipeline Detection: FPGA Implementation**

Pipa bawah air diletakkan di dasar laut untuk tujuan pengangkutan minyak bumi dan gas menyebrangi lautan. Pipa perlu terus dipantau untuk menghindari gangguan dalam proses transportasi. Gambar dasar laut dapat diperoleh dengan menggunakan kamera dan dengan memproses gambar yang diperoleh dapat membantu dalam mendeteksi pipa. Penelitian ini membahas tentang metode pemrosesan citra untuk deteksi pipa bawah laut dari gambar bawah laut yang diambil oleh kendaraan bawah laut yang dapat digunakan sebagai langkah awal untuk melacak saluran pipa. Implementasinya berhasil dilakukan pada *Field Programmable Gate Array (FPGA)* berbasis *development board* (Raj dkk. 2016).

### **2.2.2 FPGA Implementation of Spatial Filtering techniques for 2D Images**

Berbagai teknik filter telah menjadi inti dari pemrosesan citra sejak awal teknik peningkatan citra (*image enhancement*). Filter spasial pada pemrosesan citra digital digunakan dalam banyak kepentingan seperti mempertajam citra, menghaluskan citra, menghilangkan derau dan sebagainya. Fleksibilitas dari metode filter spasial sering dibandingkan dengan domain transformasi karena dapat digunakan untuk filter linear dan filter non-linear. Penghalusan citra dilakukan dengan langsung memanipulasi nilai intensitas dari citra asli dengan sebuah kernel filter. Hasilnya yaitu berkurangnya detail kecil dan derau pada citra. Penelitian ini tentang penerapan berbagai macam operator filter spasial. Hasilnya didasarkan pada konsumsi perangkat keras, kecepatan desain masing-masing arsitektur. Ukuran kualitas citra didapat dengan membandingkan output dari Matlab dan output dari Xilinx FPGA dan dengan menghitung MSE (Sadangi dkk. 2017).

### **2.2.3 Features of Image Spatial Filters Implementation on FPGA**

Penelitian ini menyajikan fitur-fitur implementasi filter spasial pada citra dengan *Programmable Logic Integrated Circuits* (FPGA). Solusi yang disajikan memungkinkan untuk membuat arsitektur kristal dengan performa tinggi untuk algoritma filter spasial. Hasilnya menunjukkan kelebihan menggunakan *programmable logic* dalam tugas pemrosesan citra digital (Ustyukov dkk. 2019).

### **2.2.4 An FPGA-Oriented Algorithm for Real-Time Filtering of Poisson Noise in Video Streams, with Application to X-Ray Fluoroscopy**

Pada penelitian ini dibahas tentang algoritma baru untuk *real-time filtering* pada video yang rusak karena *poisson noise*. Algoritma yang disajikan efektif dalam penanganan derau, dan ini secara ideal cocok dengan implementasi *hardware*, dan dapat diimplementasikan pada FPGA kecil yang memiliki sumber daya *hardware* yang terbatas. Pada penelitian ini penerapan algoritma menggunakan hasil *X-ray fluoroscopy* sebagai studi kasus. Hasil implementasi menggunakan yang StratixIV FPGA menunjukkan bahwa sistem hanya menggunakan, paling banyak, 22% dari sumber daya perangkat, dalam implementasi *real-time filtering* pada video *stream* 1024x1024 @49fps. Sebagai perbandingan, implementasi filter berbasis FIR pada FPGA yang sama dan dengan video *stream* yang serupa, dibutuhkan 80% *resource logic* pada FPGA (Castellano dkk. 2019).

### **2.2.5 A real-time video denoising algorithm with FPGA implementation for Poisson-Gaussian noise**

Pada penggunaan umum metode denoising yaitu *Pixel Similarity Weighted Frame Average* (PSWFA). Pada penelitian ini, dilakukan peningkatan kemampuan denoising dari PSWFA menggunakan pre-filter yang mengandung operator *downsampling* dan small Gaussian filter. Transformasi citra dapat mengalami gangguan oleh derau Gaussian. Untuk memasang algoritma ini pada perangkat keras, sebelumnya diimplementasikan algoritma ini pada Spartan-6 FPGA untuk evaluasi. Dilakukan juga perbandingan dengan beberapa metode denoising yang sudah ada. Evaluasi selanjutnya untuk kemampuan denoising, algoritma ini dibandingkan dengan

beberapa algoritma *state-of-art* yang tidak diimplementasikan pada FPGA tetapi memiliki performa yang baik pada personal komputer. Hasil eksperimen pada kedua simulasi video berderau dan video yang ditangkap pada pencahayaan yang kurang menunjukkan tingkat keefektifan pada algoritma ini, terkhusus pada pemrosesan derau berskala besar (Tan dkk. 2014).