

Seminar I

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



Oleh
SULAEMAN
H131 16 002

Pembimbing Utama : Dr. Eng. Armin Lawi, S.Si., M.Eng.
Pembimbing Pertama : Dr. Diaraya, M.Ak.
Penguji : 1. Dr. Hendra, S.Si., M.Kom.
2. Nur Hilal A Syahrir, S.Si., M.Si.

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

2020

DAFTAR ISI

DAFTAR ISI	ii
DAFTAR GAMBAR	iii
BAB I PENDAHULUAN	1
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
BAB II TINJAUAN PUSTAKA	5
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
BAB III METODE PENELITIAN	21
3.1 Tahapan Penelitian	21
3.2 Waktu dan Lokasi Penelitian	22
3.3 Rancangan Sistem	22
3.4 Instrumen Penelitian	23
DAFTAR PUSTAKA	23

DAFTAR GAMBAR

2.1	(a) Contoh citra biner, (b) contoh citra grayscale, (c) contoh citra warna.	6
2.2	Ilustrasi konvolusi pada citra. Sumber: https://indoml.com	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

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 kordinat spasial, dan amplitudo f di titik kordinat (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 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.

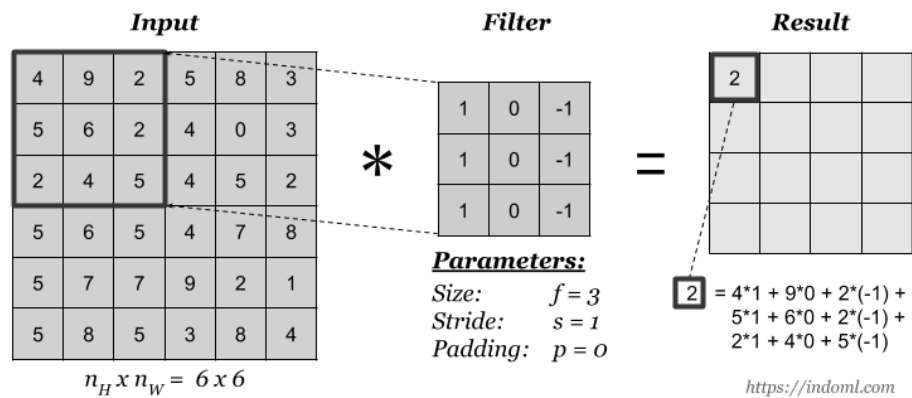
$$h(x) = f(x) * g(x) = \int_{-\infty}^{\infty} f(a)g(x-a)da \quad (2.7)$$

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).

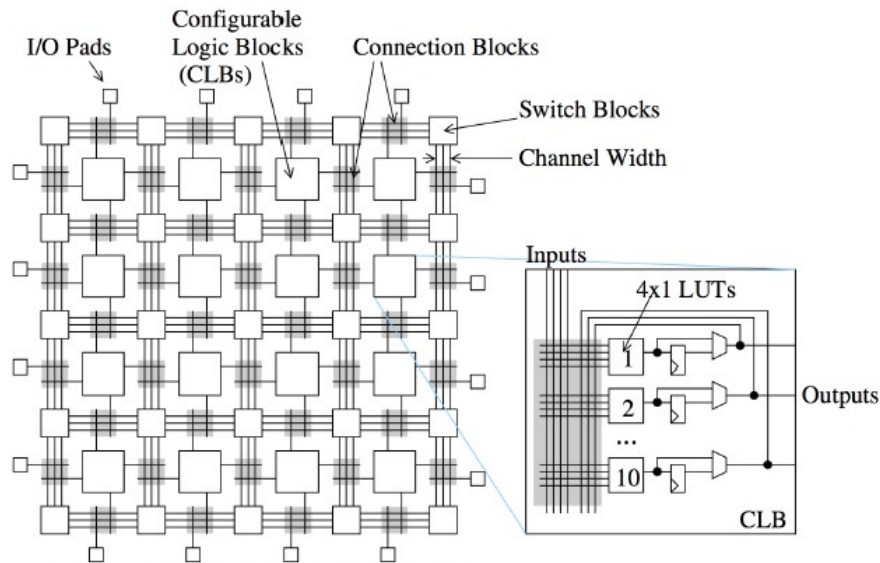
2.1.6 Video Stream

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.



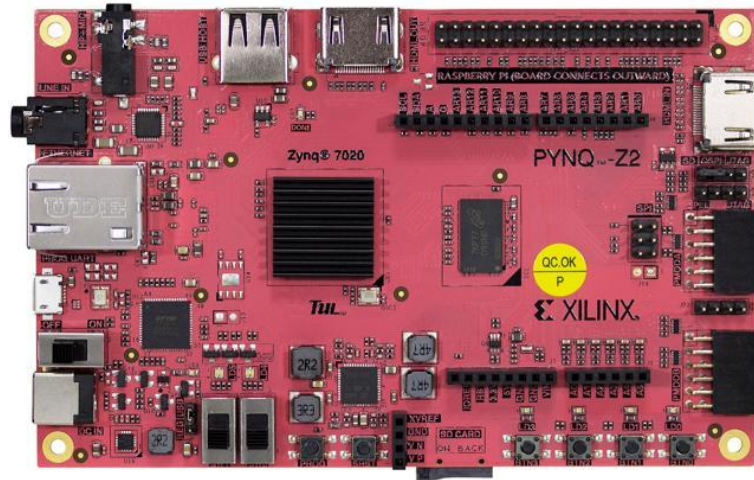
Gambar 2.3: Struktur FPGA.

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 filter spasial linear dengan prosesor ARM dan FPGA.



Gambar 2.4: FPGA Board Xilinx PYNQ-Z2.

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.

$$waktukomputasi = waktuselesai - waktumulai \quad (2.9)$$

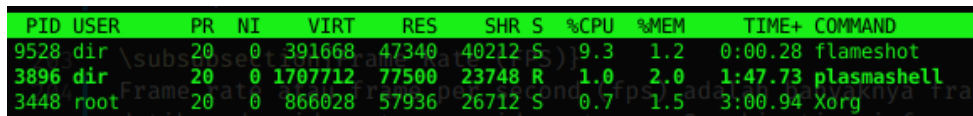
2.1.8.2 Frame Rate (FPS)

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).

$$fps = \frac{jumlahframe}{waktukomputasi} \quad (2.10)$$

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
	1	2
Anonymous	<ul style="list-style-type: none"> . stack . malloc() . brk()/sbrk() . mmap(PRIVATE, ANON) 	<ul style="list-style-type: none"> . POSIX shm* . mmap(SHARED, ANON)
File-backed	<ul style="list-style-type: none"> . mmap(PRIVATE, fd) . pgms/shared libs 	<ul style="list-style-type: none"> . mmap(SHARED, fd)
	3	4

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 *poison 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

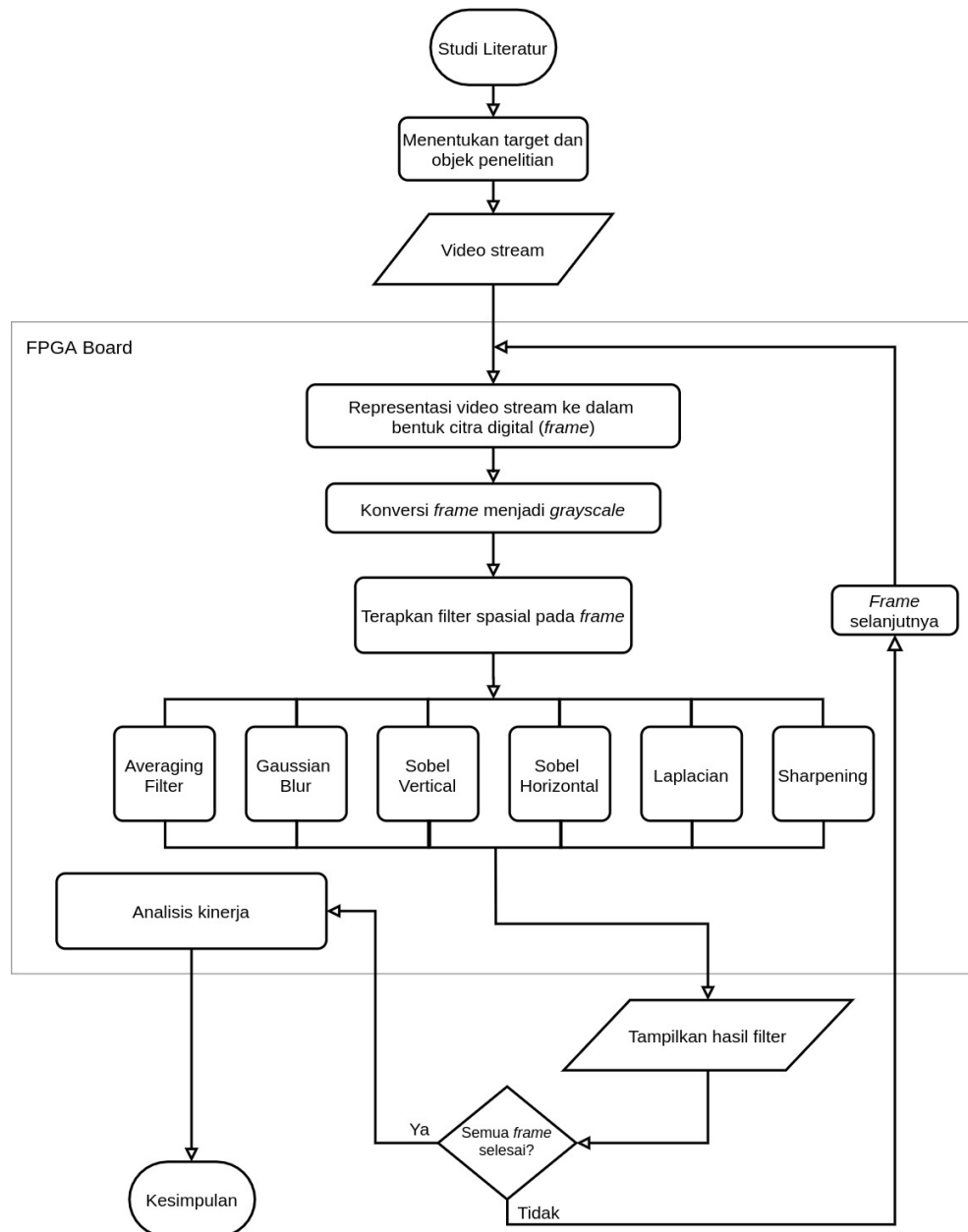
beberapa algoritma *state-of-art* yang tidak diimplementasikan pada FPGA tetapi memiliki performa yang baik pada personal komputer. Hasil eksperimen pada kedua simulasi video berderau dan video yang ditangkap pada pencahayaan yang kurang menunjukkan tingkat keefektifan pada algoritma ini, terkhusus pada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

BAB III

METODE PENELITIAN

3.1 Tahapan Penelitian

Tahapan dalam penelitian ini dapat dilihat pada gambar 3.1.



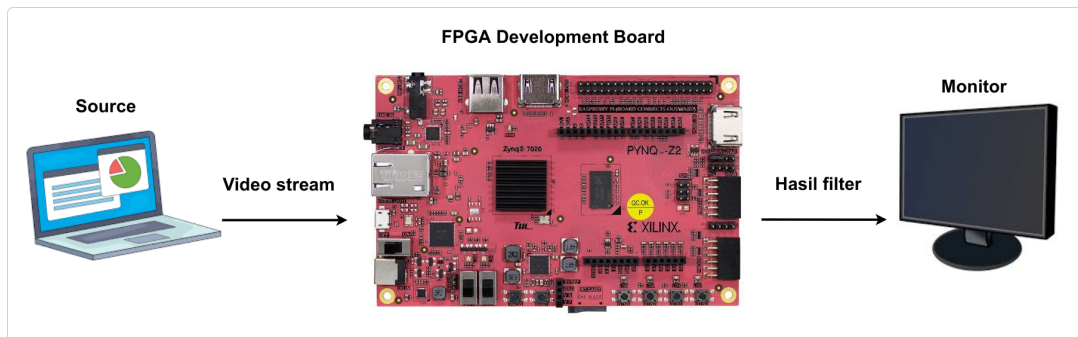
Gambar 3.1: Flowchart tahapan penelitian.

3.2 Waktu dan Lokasi Penelitian

Penelitian ini dilaksanakan dari bulan Juni 2020 sampai dengan bulan Agustus 2020. Lokasi penelitian dilakukan di Laboratorium Rekayasa Perangkat Lunak Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Hasanuddin Makassar.

3.3 Rancangan Sistem

Pada penelitian ini akan dibangun suatu sistem untuk mengimplementasikan filter spasial linear pada FPGA Development Board, dapat dilihat pada gambar 3.2.



Gambar 3.2: Rancangan sistem.

Video *stream* dari *source* disalurkan melalui port HDMI Input pada FPGA Development Board, kemudian video *stream* tersebut akan diolah dengan menerapkan filter spasial linear pada setiap *frame*-nya. Setiap *frame* yang telah diterapkan filter spasial akan dialirkan ke monitor untuk kemudian ditampilkan. Selanjutnya dilakukan analisis kinerja pada FPGA. FPGA Development Board yang digunakan dalam penelitian ini dapat diakses dengan *ssh* pada port 22 atau dengan *Jupyter Notebook* melalui *web browser*.

3.4 Instrumen Penelitian

1. Kebutuhan perangkat lunak:
 - a. Linux Ubuntu 18, sebagai OS pada FPGA Development Board.
 - b. Python 3.6, dengan library OpenCV, Numpy, Pynq 5.2, dan Xilinx xfOpenCV.
 - c. Jupyter Notebook pada FPGA Development Board.
 - d. Web Browser untuk mengakses Jupyter Notebook pada FPGA Development Board.
2. Kebutuhan perangkat keras:
 - a. FPGA Development Board.
 - b. Micro SD Card 16 GB, sebagai media penyimpanan OS pada FPGA Development Board.
 - c. Monitor Eksternal, untuk menampilkan hasil penerapan filter spasial pada FPGA Development Board.
 - d. Laptop Lenovo Ideapad 320 (sebagai *source video stream*).

Spesifikasi FPGA Development Board yang digunakan:

- Model : Xilinx PYNQ-Z2.
- Processor : Dual-Core ARM Cortex A9, 650 MHz
- FPGA : 1,3M reconfigurable gates
- Memory : 512 MB DDR3 / Flash
- Storage : Micro SD card slot
- Power : DC 7V-15V
- Dimension : 3,44" x 5,39" (87mm x 137mm)

DAFTAR PUSTAKA

- Asano, Shuichi, Tsutomu Maruyama, and Yoshiki Yamaguchi (2009). “Performance comparison of FPGA, GPU and CPU in image processing”. In: *2009 International Conference on Field Programmable Logic and Applications*, pp. 126–131. doi: 10.1109/FPL.2009.5272532.
- Biswas, Priyabrata (2019). *Introduction to FPGA and its Architecture*. <https://towardsdatascience.com/introduction-to-fpga-and-its-architecture-20a62c14421c>. Accessed on 2020-06-18.
- Castellano, G. dkk. (Jan. 2019). “An FPGA-Oriented Algorithm for Real-Time Filtering of Poisson Noise in Video Streams, with Application to X-Ray Fluoroscopy”. In: *Circuits, Systems, and Signal Processing*. doi: 10.1007/s00034-018-01020-x.
- Cheung, Peter (2019). *Introduction to FPGAs*. http://www.ee.ic.ac.uk/pcheung/teaching/ee2_digital/Lecture2-IntroductiontoFPGAs.pdf. Accessed on 2020-04-19.
- Cong, Jason dkk. (2018). “Understanding Performance Differences of FPGAs and GPUs”. In: *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. FPGA '18. Monterey, CALIFORNIA, USA: Association for Computing Machinery, p. 288. ISBN: 9781450356145. doi: 10.1145/3174243.3174970. URL: <https://doi.org/10.1145/3174243.3174970>.
- Gonzalez, Rafael C. and Richard E. Woods (2001). *Digital Image Processing*. 2nd. ISBN-13: 978-0201180756. Upper Saddle River, New Jersey 07458: Prentice Hall.
- Jingbo, Xu dkk. (Aug. 2011). “A New Method for Realizing LOG Filter in Image Edge Detection”. In: *The 6th International Forum on Strategic Technology*. doi: 10.1109/IFOST.2011.6021127.
- Kerrisk, Michael (2020). *(Top) Linux Manual Page*. <https://www.man7.org/linux/man-pages/man1/top.1.html>. Accessed on 2021-02-2.
- Kowalczyk, Marcin, Dominika Przewlocka, and Tomasz Krvjak (Oct. 2018). “Real-Time Implementation of Contextual Image Processing Operations for 4K Video Stream in Zynq UltraScale+ MPSoC”. In: *2018 Conference on Design and*

- Architectures for Signal and Image Processing (DASIP)*. DOI: 10.1109/DASIP.2018.8597105.
- Madhusudana, Pavan C. dkk. (2020). “Capturing Video Frame Rate Variations via Entropic Differencing”. In: *IEEE Signal Processing Letters* 27, pp. 1809–1813. DOI: 10.1109/LSP.2020.3028687.
- Putra, Darma (2010). *Pengolahan Citra Digital*. ISBN-13: 978-979-29-1443-6. Jl. Beo 38-40, Yogyakarta 55281: Penerbit Andi.
- Raj, S.M. Alex, Rita Maria Abraham, and M.H. Supriya (Sept. 2016). “Spatial Filtering Based Boundary Extraction in Underwater Images for Pipeline Detection: FPGA Implementation”. In: *International Journal of Computer Science and Information Security (IJCSIS)*. Vol. 14, No. 9.
- Rinaldi, Munir (2004). *Pengolahan Citra Digital dengan Pendekatan Algoritmik*. ISBN: 979-3338296. Bandung: Penerbit Informatika.
- S, Lars dkk. (2020). *The Linux System Administrator's Guide Chapter 6. Memory Management*. <https://tldp.org/LDP/sag/html/vm-intro.html>. Accessed on 2021-02-22.
- Sadangi, Sushant dkk. (May 2017). “FPGA Implementation of Spatial Filtering techniques for 2D Images”. In: *IEEE International Conference On Recent Trends in Electronics Information & Communication Technology (RTEICT)*.
- Silbershatz, Avi, Peter Baer Galvin, and Greg Gagne (2009). *Operationg System Concepts*. ISBN: 978-0-470-12872-5. John Wiley and Sons, Inc.
- Silva, Eduardo A.B. da and Gelson V. Mendonca (2005). “4 - Digital Image Processing”. In: *The Electrical Engineering Handbook*. Ed. by Wai-Kai Chen. Burlington: Academic Press, pp. 891–910. ISBN: 978-0-12-170960-0. DOI: <https://doi.org/10.1016/B978-012170960-0/50064-5>.
- Sutoyo, T. dkk. (2009). *Teori Pengolahan Citra Digital*. ISBN-13: 978-979-29-0974-6. Jl. Beo 38-40, Yogyakarta 55281: Penerbit Andi.
- Tan, Xin dkk. (Feb. 2014). “A Real-time Video Denoising Algorithm with FPGA Implementation for Poisson-Gaussian Noise”. In: *J Real-Time Image Proc*. DOI: 10.1007/s11554-014-0405-2.

- Ustyukov, Dmitry I., Alex I. Efimov, and Dmitry A. Kolchaev (June 2019). “Features of Image Spatial Filters Implementation on FPGA”. In: *Mediterranean Conference On Embedded Computing (Meco)*.
- Xilinx (2020). *Field Programmable Gate Array (FPGA)*. <https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html>. Accessed on 2020-04-17.
- Yang, Ching-Chung (Sept. 2013). “Finest Image Sharpening by Sse of the Modified Mask Filter Dealing with Highest Spatial Frequencies”. In: *OPTIK*. DOI: 10.1016/j.ijleo.2013.09.070.
- Zhao, Jin (Apr. 2015). “Video/Image Processing on FPGA”. Master thesis. Worcester Polytechnic Institute.