DIGITAL SYSTEM DESIGN FINAL PROJECT REPORT
DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITAS INDONESIA

VGA SIMULATOR

GROUP 19

Ferdyano	**2406353723**
Mohamad Rizky Alamsyah	**2406420892**
Valiant Joshua	**2406352153**
Jesaya Hamonangan	**2406409845**

PREFACE

Pada Proyek Akhir PSD ini, kami merancang sebuah sistem pengendali tampilan VGA (Video Graphics Array) yang mampu menayangkan animasi video musik "Bad Apple" pada layar monitor dengan resolusi 640x480 piksel.

Dengan memanfaatkan konsep logika sekuensial dan manajemen pewaktuan (timing) digital yang presisi, sistem ini dirancang untuk membangkitkan sinyal sinkronisasi horizontal (HSync) dan vertikal (VSync) yang akurat. Proses penayangan dimulai dengan menginisialisasi pencacah (counter) posisi piksel, menentukan area aktif video (video on), dan memetakan data warna (RGB) ke koordinat layar yang sesuai. Pendekatan ini memungkinkan proses rendering gambar yang stabil, sinkron, dan real-time. Setelah inisialisasi, sistem menghasilkan keluaran sinyal video yang merepresentasikan setiap frame animasi secara berurutan tanpa gangguan visual (flicker).

Pendekatan berbasis VHDL memberikan fleksibilitas tinggi dalam mendesain pengendali perangkat keras yang efisien dan dapat diimplementasikan pada FPGA. Dengan metode ini, pengguna dapat memahami bagaimana data citra digital diproses dan ditransmisikan menjadi sinyal analog/digital yang dapat dilihat mata, memberikan wawasan mendalam tentang antarmuka multimedia berbasis sistem digital.

Dalam laporan ini, kami akan menjelaskan secara rinci setiap tahap perancangan, mulai dari perhitungan parameter waktu (timing parameters), perancangan logika pencacah (counter), implementasi modul VHDL, hingga pengujian sistem (testbench) untuk memverifikasi keluaran sinyal video

Depok, December 6, 2025

Group 19

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Video Graphics Array (VGA) merupakan standar antarmuka teknologi tampilan yang digunakan untuk mentransmisikan data visual secara luas dan konsisten dalam sistem komputasi. Teknologi ini telah menjadi fondasi fundamental di berbagai perangkat, seperti monitor komputer, proyektor, antarmuka industri, hingga sistem multimedia embedded. Kemampuan VGA untuk menyajikan informasi grafis dalam resolusi standar menjadikannya solusi visualisasi yang krusial dalam interaksi antara pengguna dan sistem digital. Meski demikian, pembangkitan sinyal video yang stabil pada umumnya sulit dicapai secara sempurna jika hanya mengandalkan perangkat lunak berbasis prosesor tradisional, yang sering kali terkendala oleh latensi dan kurang optimal dari segi presisi waktu (timing) sinyal sinkronisasi.

Dengan berkembangnya kebutuhan akan pengolahan grafis yang cepat dan tersinkronisasi secara real-time, FPGA atau Field-Programmable Gate Array muncul sebagai pilihan unggul. FPGA menawarkan keunggulan pemrosesan paralel dan fleksibilitas manajemen memori, memungkinkan pengembangan sistem kendali video yang efisien dan sangat akurat. Dengan menggunakan VHDL atau VHSIC Hardware Description Language, pengembangan arsitektur grafis berbasis FPGA dapat dilakukan secara modular dengan kontrol timing tingkat tinggi, menjadikannya alat yang ideal untuk merancang sistem penampil animasi video seperti "Bad Apple" yang mulus dan stabil.

1.2 PROJECT DESCRIPTION

Proyek ini bertujuan untuk merancang sistem pengendali tampilan VGA (Video Graphics Array) berbasis VHDL yang mampu menayangkan animasi video "Bad Apple" pada resolusi standar 640x480 piksel. Sistem akan mengonversi data frame animasi yang tersimpan dalam memori menjadi sinyal video sinkron sesuai dengan standar timing industri, yang kemudian divisualisasikan secara berurutan pada layar monitor. Proses penayangan mencakup pembangkitan sinyal sinkronisasi horizontal (HSync) dan vertikal (VSync) yang presisi, penentuan area aktif video (video on), hingga pemetaan data warna (RGB) ke koordinat layar yang tepat.

Sistem ini memanfaatkan manipulasi logika pada level siklus detak (clock cycle) untuk memastikan stabilitas sinyal video yang efisien dan bebas gangguan (flicker). Setiap langkah dalam proses pembangkitan sinyal dirancang modular, memanfaatkan keunggulan FPGA dalam pemrosesan paralel untuk menangani refresh rate yang tinggi. Pendekatan ini memungkinkan penayangan animasi secara real-time dengan hasil yang presisi, serta membuktikan kapabilitas perangkat keras dalam menangani beban kerja grafis multimedia secara optimal.

Proyek ini juga mencakup pengujian komprehensif melalui simulasi test bench untuk memvalidasi integritas sinyal dan keakuratan waktu (timing) sistem. Selain itu, penerapan konsep logika sekuensial berbasis pencacah (counters) digunakan untuk mengatur alur sinkronisasi horizontal dan vertikal secara terstruktur , sehingga setiap tahap rendering piksel berjalan sinkron sesuai dengan spesifikasi resolusi yang telah ditentukan. Proyek ini diharapkan memberikan pemahaman yang lebih mendalam tentang penerapan teknologi FPGA dalam manajemen memori visual dan pengolahan sinyal video berbasis VGA.

1.3 OBJECTIVES

Tujuan dari proyek akhir ini adalah sebagai berikut:

1. Merancang sistem pengendali tampilan VGA (Video Graphics Array) yang mampu menayangkan animasi "Bad Apple" pada resolusi 640x480 piksel.
2. Menggunakan pemrograman behavioral untuk logika proses sinkronisasi dan dataflow untuk penetapan sinyal output dalam implementasi sistem.
3. Memanfaatkan process constructs dan operasi aritmatika untuk mengatur pewaktuan sinyal horizontal dan vertikal secara presisi .
4. Mengimplementasikan logika sekuensial berbasis pencacah (counters) untuk menentukan area aktif video (video on) dan pemetaan koordinat piksel secara terstruktur.
5. Mengembangkan testbench otomatis yang mampu memvalidasi sinyal timing dan merekam output menjadi file gambar digital (.ppm) untuk verifikasi visual.

1.4 ROLES AND RESPONSIBILITIES

Berikut adalah tanggung jawab dan pembagian tugas untuk setiap anggota kelompok:

Roles	Responsibilities	Person
Pengembangan Kode	Perancangan VGA Controller	Jesaya Hamonangan Gaudensius M.
	Implementasi Logika RGB	Jesaya Hamonangan Gaudensius M.
	Pengembangan Testbench	Jesaya Hamonangan Gaudensius M.
	Generasi Output Gambar (.ppm)	Jesaya Hamonangan Gaudensius M. dan Alam
Pembuatan Laporan dan Presentasi	Pembuatan laporan	Ferdyano dan Joshua
	Pembuatan ReadMe	Ferdyano dan Alam
	Pembuatan PPT	Ferdyano dan Joshua

Table 1. Roles and Responsibilities

CHAPTER 2

IMPLEMENTATION

2.1 EQUIPMENT

Tools yang digunakan selama pembuatan projek ini adalah sebagai berikut:

- Vivado 2022.22
- ModelSim
- VSCode
- GitHub

2.2 IMPLEMENTATION

Pada pengerjaan proyek ini, kami mengimplementasikan konsep-konsep Perancangan sistem digital yang terintegrasi antara perangkat lunak (Python) dan perangkat keras (VHDL). Implementasi dibagi menjadi tiga tahap utama yaitu: persiapan data, desain datapath dan kontrol VGA, dan integrasi memori.

Pemrosesan Data Video

Sebelum data diproses oleh FPGA, video “Bad Apple” dengan format awal MP4, harus dikonversi terlebih dahulu. Karena terdapat keterbatasan kapasitas memori Block RAM, video dengan resolusi tinggi tidak dapat disimpan secara langsung dan harus dilakukan resizing. Kami mencoba memodelkan ini sebagaimana sebuah video format MP4 akan dimasukan ke dalam memori. Umumnya, video dimasukkan ke dalam sebuah memori external yang bisa digunakan sebagai akses pada tiap framenya, namun karena hal itu tidak memungkinkan, kami memutuskan untuk membuat file menjadi seakan-akan file HEX sebagai objek memori.

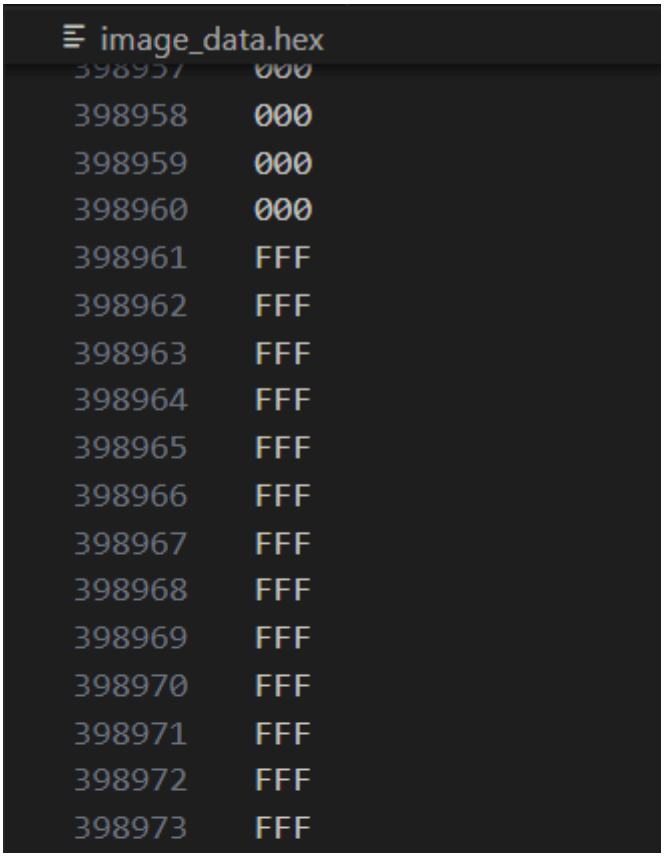
Kami menggunakan kode Python yang kami cantumkan pada repo Github kami untuk melakukan pekerjaan tersebut. Melalui skrip pada MP4toHEX.py, kami mengubah video asli menjadi resolusi dengan ukuran 64 x 48 piksel agar dapat disimpan dalam FPGA block RAM. Setiap piksel kemudian dikonversi menjadi format data HEX 12 bit menggunakan teknik thresholding. Menggunakan bantuan dari *library* OpenCV Python,

kami membaca piksel yang ada pada video dan mengkonversinya menjadi warna yang kemudian akan ditulis di file output.

```
with open(OUTPUT_FILE, 'w') as f:
    count = 0
    while count < frames_to_process:
        ret, frame = cap.read()
        if not ret:
            break

        # 1. RESIZE: Scale down to the FPGA's native grid (64x48)
        frame_resized = cv2.resize(frame, (WIDTH, HEIGHT))

        # 2. COLOR CORRECTION: Convert BGR (OpenCV default) to RGB
        frame_rgb = cv2.cvtColor(frame_resized, cv2.COLOR_BGR2RGB)

        # 3. WRITE HEX DATA
        for y in range(HEIGHT):
            for x in range(WIDTH):
                # Get the 8-bit color values (0-255)
                r, g, b = frame_rgb[y, x]

                # Downsample to 4-bit (0-15) by shifting right 4 times
                r_4bit = r >> 4
                g_4bit = g >> 4
                b_4bit = b >> 4

                # Pack into 12-bit integer: RRRR GGGG BBBB
                hex_12bit_color = f"{r_4bit:X}{g_4bit:X}{b_4bit:X}"

                # Write as 3-digit Hex (e.g., F0A)
                f.write(f"{hex_12bit_color}\n")

        count += 1
        if count % 100 == 0:
```

Gambar 1. melakukan resizing dan mengubah menjadi biner

Kode Python ini menghasilkan output berupa file dengan nama image_data.hex. File dengan format .HEX yang berisi data memori untuk setiap piksel video. Data ini ditulis secara berurutan/ sekuensial perbaris untuk seluruh frame, menjadi satu daftar panjang alamat memori.



Gambar 2. Contoh isi file output .HEX

VGA Controller

Bagian ini dapat dilihat pada kode VGA.vhd yang berada pada Repository Github. Modul ini berfungsi sebagai penghasil sinyal sinkronisasi agar monitor dapat menampilkan gambar dengan benar. Modul ini menerima input berupa clock dan sinyal reset, serta outputnya adalah sinyal sinkronisasi horizontal (h_output) dan vertikal (v_output) berdasarkan koordinat piksel saat ini (x_out, y_out) dan juga status aktif video. Lalu terdapat juga beberapa constant lain seperti H_VISIBLE, H_FRONT, H_SYNC, dan H_BACK yang didefinisikan sesuai dengan standar pada VGA, dimana total siklus untuk horizontal adalah 800 (H_TOTAL) dan 525 baris untuk total baris vertikal (V_TOTAL).

Modul ini menjalankan proses utama yang bekerja menggunakan counter untuk coordinate generator. TErdapat dua sinyal yaitu h count untuk posisi horizontal dan v_count untuk posisi vertikal. Setiap kali terjadi rising edge atau clock naik, h_count

akan bertambah 1 (increment), dan ketika h_count mencapai batas akhir yaitu 800, v_count akan bertambah 1. Proses ini terjadi secara terus menerus, hingga seluruhnya (525 baris) selesai diproses. Setelah itu, h_count dan v_count akan kembali ke posisi awal (kiri atas) untuk memulai kembali pada frame baru.

```
-- COORDINATE GENERATOR
process(clock_in, reset_n)
begin
        if (reset_n = '0') then             --reset asserted
            h_count <= 0;                   --reset horizontal counter
            v_count <= 0;                   --reset vertical counter
        elsif rising_edge(clock_in) then
            IF(h_count < H_TOTAL - 1) THEN      --horizontal counter (pixels)
                h_count <= h_count + 1;
            ELSE
                h_count <= 0;
                IF(v_count < V_TOTAL - 1) THEN  --veritcal counter (rows)
                    v_count <= v_count + 1;
                ELSE
                    v_count <= 0;
                END IF;
            END IF;
        end if;
end process;
```

Gambar 3. Proses utama: coordinate generator

Selain itu, modul ini juga mengaktifkan sinyal sinkronisasi (sync pulse) dan status dari video. Sinyal h_output dan v_output diatur menjadi ‘0’ hanya ketika nilai dari counter berada dalam rentang waktu sinkronisasi yang sudah ditentukan sebelumnya, sementara akan bernilai ‘1’ di waktu lainnya. sinyal vidio_on disini berfungsi sebagai penanda untuk area aktif, sehingga akan menjadi ‘1’ ketika berada dalam resolusi 640x480. Jika berada pada luar area tersebut, status sinyal vidio_on akan menjadi ‘0’, tujuannya untuk memberitahu sistem agar tidak mengirimkan data warna.

Logic Unit

Modul logic_unit.vhd disini berperan sebagai top level entity, yang mengintegrasikan logika untuk timing VGA dengan Video_Controller. Dapat dilihat pada bagian entity, modul ini menghubungkan antara sinyal clock dan reset sebagai

input, serta menghasilkan output berupa sinyal RGB dan sinkronisasi (HSync, VSync). Lalu terdapat proses clock divider yang mengubah clock sistem menjadi pixel clock, dengan cara membalik nilai dari sinyal pixel_clk pada setiap rising edge, sehingga sesuai dengan standar frekuensi yang dibutuhkan untuk 640 x 480.

Komponen VGA diinstansiasi untuk menghasilkan sinyal sinkronisasi serta koordinat piksel saat ini (x_pos, y_pos) dan status aktif layar (video_on). Sinyal-sinyal koordinat ini kemudian diteruskan ke komponen Video_Controller, yang bertugas menghitung dan mengeluarkan alamat memori (mem_address_out) yang sesuai untuk *frame* saat ini. Selain itu, terdapat juga counter current_frame yang bertambah setiap kali satu frame telah selesai, yang ditandai oleh sinyal dari v_sync, sehingga memungkinkan untuk terjadi perpindahan antar frame. Hal ini memungkinkan sistem untuk menampilkan animasi dengan berpindah ke frame memory berikutnya hingga batas max frame.

```
-- Frame Counter
process(v_sync, reset)
begin
    if reset = '0' then
        current_frame <= 0;
    elsif rising_edge(v_sync) then
        if current_frame < MAX_FRAME - 1 then
            current_frame <= current_frame + 1;
        else
            current_frame <= 0;
        end if;
    end if;
end process;
```

Gambar 4. Counter frame

Terakhir, alur data warna ditangani melalui jalur mem_data_in dan rgb_data. Modul ini menerima data piksel 12-bit (vektor rgb_data) dari memori eksternal (image_data.hex) melalui Video_Controller. Sinyal ini kemudian dipetakan langsung ke pin output fisik: 4-bit teratas ke vga_r, 4-bit tengah ke vga_g, dan 4-bit terbawah ke vga_b. Dengan demikian, logic_unit memastikan bahwa data warna yang telah diproses oleh *controller* disalurkan ke port VGA monitor secara sinkron dengan sinyal *timing*.

```
-- Video Controller
video_controller_inst : Video_Controller
port map (
    pixel_clk       => pixel_clk,
    reset           => reset,
    x_pos           => x_pos,
    y_pos           => y_pos,
    video_on        => video_on,
    current_frame   => current_frame,
    mem_address_out => mem_address_out,
    mem_data_in     => mem_data_in,
    rgb_out         => rgb_data
);

vga_r <= rgb_data(11 downto 8);
vga_g <= rgb_data(7 downto 4);
vga_b <= rgb_data(3 downto 0);
```

Gambar 5. Video_Controller & Warna Output

Video Controller

Kode video_controller.vhd ini berfungsi sebagai Unit Manajemen Memori Video atau jembatan data dalam arsitektur sistem VGA. Modul ini memiliki tanggung jawab yaitu untuk menerjemahkan koordinat dari piksel layar X, Y yang diterima dari VGA Controller untuk menjadi alamat memori, di mana data warna gambar tersimpan. Modul ini akan menerima input posisi dari modul VGA, lalu menghitung alamat untuk meminta data ke modul Block Memory (RAM/ROM), dan kemudian menerima data warna tersebut untuk diteruskan ke port output fisik rgb_out..

Modul ini juga melakukan kalkulasi alamat dan upscaling. Karena gambar sumber memiliki resolusi yang kecil yaitu 64 x 48 sementara layar output akan beresolusi besar yaitu 640 x 480, kode ini menerapkan *upscaling* sebesar 10 kali lipat. Dengan Logikanya yaitu (y_pos / 10) dan (x_pos / 10) sehingga memastikan bahwa setiap 10 piksel di layar monitor akan membaca alamat memori yang sama, sehingga satu titik pada gambar asli akan merepresentasikan sebagai blok kotak berukuran 10 x 10 piksel di layar. Lalu rumus (current_frame * (IMG_W * IMG_H)) berfungsi sebagai *offset* memori yang memungkinkan

sistem untuk berpindah ke blok memori berisi *frame* animasi berikutnya secara otomatis sesuai input current_frame.

Kode ini juga mengatur aliran data warna 12-bit (4 bit Merah, 4 bit Hijau, 4 bit Biru) melalui sinyal mem_data_in dan rgb_out. Proses dalam kode ini memastikan bahwa data warna hanya dikirim ke layar jika sinyal video_on aktif dan posisi piksel berada dalam batas area gambar yang telah di *upscale*. Jika posisi pemindaian berada di luar area gambar tersebut, modul ini akan memutus data warna (mengirim logika '0' atau hitam), sehingga gambar "Bad Apple" tampil rapi di pojok kiri atas layar dengan latar belakang hitam di sekelilingnya.

CHAPTER 3

TESTING AND ANALYSIS

3.1 TESTING

Pengujian sistem ini dilakukan menggunakan kode testbench VHDL. Pengujian dilakukan menggunakan modul testbench tb_bad_apple.vhd yang berfungsi sebagai verifikasi untuk pengujian visualnya. Untuk langkah langkah pengujiannya:

1. Sebelum memulai, memori pada FPGA diinisialisasi menggunakan file .hex yang dihasilkan oleh skrip python MP4toHEX.py dalam folder python helper di Github. File yang dihasilkan dari skrip python ini memuat video "Bad Apple" yang telah dikompresi menjadi format biner 12-bit dengan resolusi 64 x 48 piksel.
2. Testbench kemudian mengaktifkan sinyal clock sebesar 50 MHz dan memberikan reset aktif selama 100 ns awal untuk memastikan counter pada modul VGA dan logic_unit berada pada status/kondisi '0'
3. Testbench ini juga menggunakan library std.text.io sehingga bisa menghandle file i/o. Testbench memantai sinyal output horizontal synx dan vertical sync. Ketika sinyal sync menandakan area aktif (active video), testbench akan mengambil data warna r,g,b. Data warna tersebut kemudian ditulis dari baris ke baris ke dalam file gambar dengan format .ppm
4. Simulasi ini dijalankan untuk merender 1000 frame video, dan menghasilkan 1000 file gambar terpisah dengan format .ppm

Setelah test bench selesai dilakukan dan terkumpul 1000 file gambar ,ppm. Kami menggunakan bantuan kode python untuk mempermudah dalam menganalisis dan memverifikasi visualnya. Kode python ini (PPMToMP4.py di dalam folder python helper) mengambil ratusan gambar dengan format .ppm yang dihasilkan oleh kode testbench VHDL, dimana setiap file mewakili satu frame output VGA, dan menggabungkannya menjadi format video .mp4.

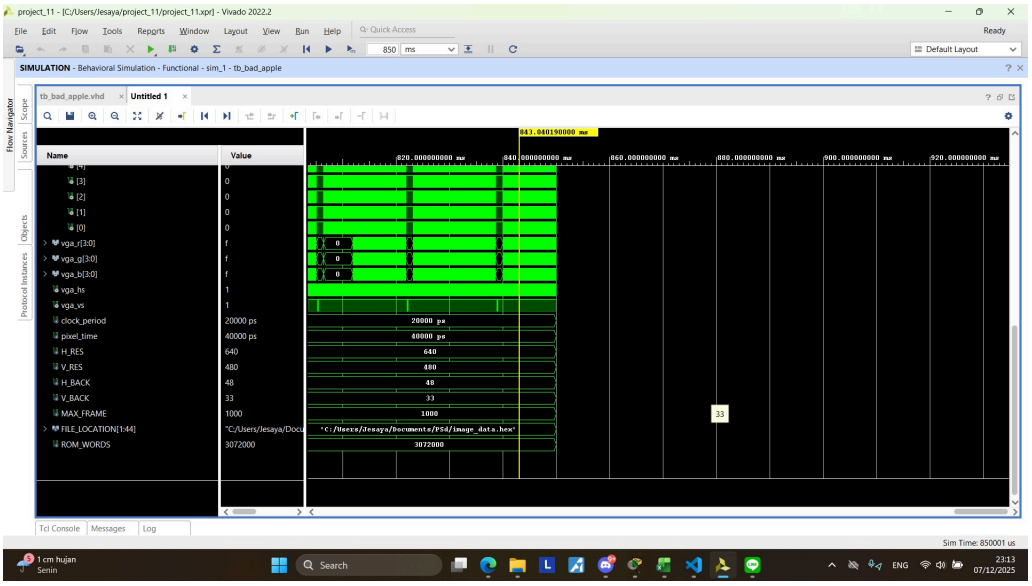
Kode python ini dimulai dengan mengurutkan semua file .ppm, kemudian membaca gambar pertama untuk mendeteksi resolusi video secara otomatis, lalu menginisialisasi objek videoWriter menggunakan codec mp4v. Akhirnya terjadi looping untuk membaca setiap file

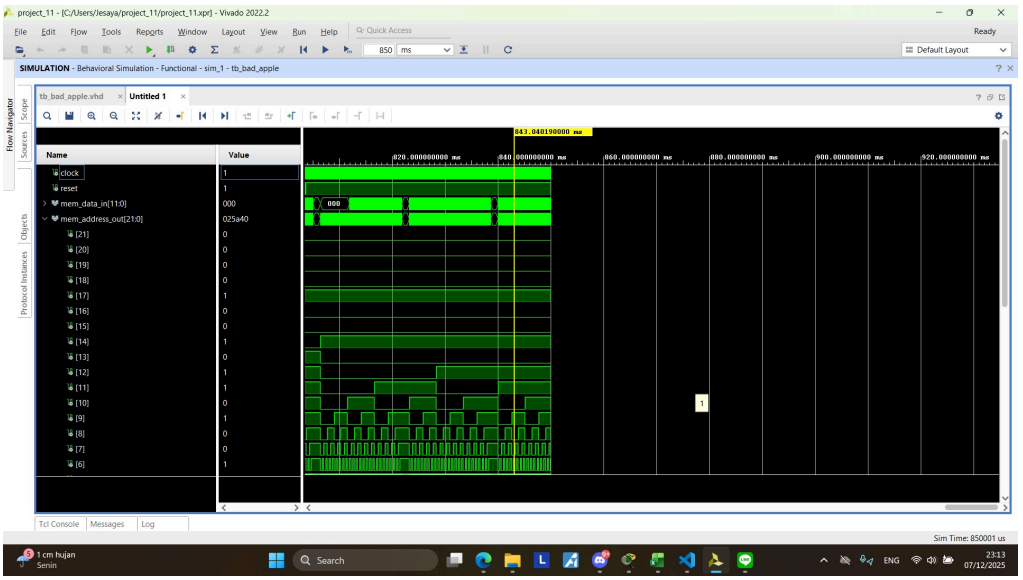
gambar satu per satu dan menuliskannya dalam kontainer video. Hasil akhirnya berupa file simulated_video.mp4.

3.2 RESULT

Berdasarkan simulasi yang dilakukan menggunakan test bench. Dihasilkan 1000 file yang merepresentasikan setiap frame. Kemudian kumpulan file ini digabung dengan bantuan skrip python menjadi sebuah video dengan format .mp4 dengan resolusi 640 x 480 dan refresh rate 60 hz. Hasil video animasi dapat dilihat melalui link berikut.

Hasil akhir: https://youtu.be/_vS-kSauPwM





3.3 ANALYSIS

Satu-satunya hal yang bisa kita analisis dari waveform di sini adalah nilai-nilai yang terus berganti pada mem_data_in dan mem_address_out. Secara sederhana, image_rom akan mengeluarkan data_out yang dihubungkan langsung ke mem_data_in dari logic_unit, hal ini menghasilkan mem_data_in yang berubah-ubah terus, diisi dengan lokasi frame dari array memori saat itu yang telah ditentukan oleh logic_unit.

mem_data_out adalah sinyal yang dikeluarkan dari image_rom ke logic_unit untuk menggambarkan frame saat ini adalah apa. Maka dengan demikian, kita bisa melihat nilai dari piksel pada posisi x atau y tertentu pada frame tertentu dengan sinyal ini.

CHAPTER 4

CONCLUSION

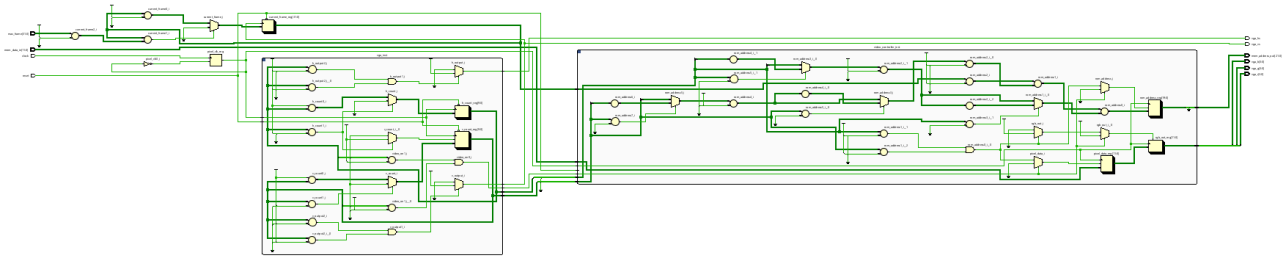
Berdasarkan perancangan, implementasi, dan pengujian yang telah dilakukan, dapat disimpulkan bahwa sistem VGA simulator berhasil dirancang, dan diimplementasikan menggunakan VHDL. Sistem yang dibuat ini mampu menampilkan animasi “Bad Apple” pada resolusi 640 x 480 piksel. Integrasi antara python dan vhdl juga terbukti efektif. Skrip python digunakan untuk mengkonversi video dari format .mp4 menjadi .HEX yang lebih efisien memori. Begitu juga dengan testbench VHDL yang berhasil mengonversi sinyal balik menjadi file gambar .ppm. Upscaling dan memory addressing yang diterapkan pada logic_unit.vhd berhasil mengatasi keterbatasan memori pada FPGA, sehingga memungkinkan penyimpanan 1000 frame tanpa butuh memori eksternal. Penggunaan/implementasi gaya pemrograman structural, yaitu menggabungkan komponen VGA, ROM, dan logic unit serta gaya behavioral pada proses counter dan logika warna, memudahkan kami untuk proses debugging dan verifikasi sistem

REFERENCES

[1]“ModelSim ® Tutorial.” Accessed: Dec. 07, 2025. [Online]. Available: https://courses.cs.washington.edu/courses/csep567/10wi/labs/modelsim_tut.pdf

APPENDICES

Appendix A: Project Schematic



Appendix B: Documentation

“maaf bang, kami tidak ada dokumentasi bersama”