



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

Typing Speed Analyzer

GROUP 14

Dimar Ilham Tamara	2406413804
Fernanda Raeka Yan Putra	2406415791
Gogra Friedrik Simatupang	2406487020
Naufal Rahman	2406413142

PREFACE

Puji syukur ke hadirat Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya sehingga laporan Proyek Akhir ini dapat terselesaikan dengan baik dan tepat waktu. Laporan ini disusun sebagai salah satu syarat dalam menyelesaikan praktikum perancangan sistem digital.

Adapun judul proyek yang kami kerjakan adalah “Typing Speed Analyzer”, yang bertujuan untuk mengukur dan menganalisis kecepatan pengetikan pengguna secara real-time melalui perancangan sistem digital. Melalui penyusunan laporan ini, penulis berharap dapat memberikan wawasan tambahan bagi pembaca mengenai implementasi VHDL dalam pengolahan input keystroke serta proses perhitungan Word Per Minute (WPM) pada sistem digital.

Penulis mengucapkan terima kasih kepada asisten laboratorium yang telah memberikan arahan, bimbingan, serta ilmu yang sangat bermanfaat selama pelaksanaan proyek ini. Ucapan terima kasih juga diberikan kepada seluruh pihak yang telah membantu baik dalam bentuk dukungan teknis, pemikiran, maupun motivasi.

Depok, December 7, 2025

Group 14

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

- 1.1 Background
- 1.2 Project Description
- 1.3 Objectives
- 1.4 Roles and Responsibilities

CHAPTER 2: IMPLEMENTATION

- 2.1 Equipment
- 2.2 Implementation

CHAPTER 3: TESTING AND ANALYSIS

- 3.1 Testing
- 3.2 Result
- 3.3 Analysis

CHAPTER 4: CONCLUSION

REFERENCES

APPENDICES

- Appendix A: Project Schematic
- Appendix B: Documentation

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Di era digital saat ini, kemampuan mengetik cepat dan akurat menjadi salah satu keterampilan dasar yang penting. Banyak aktivitas seperti penulisan dokumen, pengolahan data, pemrograman membutuhkan efisiensi pengetikan. Dengan kemampuan mengetik yang baik, seseorang dapat menyelesaikan tugas lebih cepat, meningkatkan produktivitas, dan mengurangi beban fisik maupun kognitif saat mengetik.

Namun, kecepatan mengetik tidak bisa dipisahkan dari akurasi, mengetik cepat tanpa presisi seringkali menghasilkan banyak kesalahan yang justru menurunkan kualitas hasil. Oleh karena itu, evaluasi keterampilan mengetik idealnya mencakup dua metrik utama, yaitu kecepatan dan akurasi.

Kecepatan mengetik 70 WPM tergolong cepat dan jauh di atas rata-rata kecepatan mengetik, yaitu 40 WPM. Bagi pemula atau pengguna biasa, kecepatan dan akurasi bisa lebih rendah, tergantung kebiasaan dan metode mengetik.

Berdasarkan kondisi ini, muncul kebutuhan akan alat atau sistem yang dapat mengukur performa mengetik secara objektif. Inilah latar belakang dikembangkannya Typing Speed Analyzer: sebuah sistem digital berbasis VHDL untuk mengukur dan menganalisis kecepatan dan akurasi mengetik secara real time.

1.2 PROJECT DESCRIPTION

Typing Speed Analyzer adalah sebuah sistem digital berbasis VHDL yang dirancang untuk mengukur performa mengetik secara real time. Sistem ini menerima input berupa karakter yang diketik oleh pengguna melalui keyboard, kemudian menganalisis durasi pengetikan, jumlah karakter yang benar dan salah berdasarkan teks referensi, serta menghitung Words Per Minute (WPM) dan akurasi pengetikan.

Sistem ini terdiri dari beberapa modul utama, seperti timer module, keystroke analyzer, text comparator, dan result formatter. Modul timer digunakan untuk menghitung waktu pengetikan dalam satuan milidetik sejak pengguna mulai mengetik hingga selesai. Keystroke analyzer mendeteksi setiap karakter yang masuk melalui sinyal dan melakukan pencatatan jumlah keystroke. Text comparator akan mengevaluasi kesesuaian antara karakter yang diketik dengan karakter referensi. Modul hasil kemudian menghitung metrik performa seperti jumlah kesalahan, akurasi, dan WPM.

1.3 OBJECTIVES

Tujuan dari proyek ini adalah untuk:

1. Merancang sistem digital berbasis VHDL untuk menghitung dan menganalisis kecepatan mengetik secara real time berdasarkan input karakter pengguna.
2. Mengukur performa pengetikan menggunakan parameter yang ditetapkan seperti WPM, persentase akurasi, dan jumlah kesalahan.
3. Menerapkan proses perbandingan antara karakter yang diketik dengan karakter target untuk mendeteksi kesalahan pengetikan secara efektif.

1.4 ROLES AND RESPONSIBILITIES

The roles and responsibilities assigned to the group members are as follows:

Roles	Responsibilities	Person
Timer & System Configuration Engineer	<ul style="list-style-type: none">- Mengembangkan typing_speed_pkg.vhd sebagai berkas paket yang mendefinisikan parameter sistem, termasuk frekuensi clock.- Mendesain timer_module.vhd	Gogra Friedrik Simatupang

	untuk mengukur durasi pengetikan.	
Performance Analyzer & Output Handler	<ul style="list-style-type: none"> - Membuat keystroke_analyzer.vhd yang menghitung WPM, akurasi, dan jumlah kesalahan. - Membuat typing_speed_analyzer.vhd untuk mengintegrasikan modul timer dan keystroke analyzer - Membuat result_txt_logger.vhd untuk mencatat log hasil pengetikan. 	Dimar Ilham Tamara
Testbench & Simulation Data Engineer	<ul style="list-style-type: none"> - Membuat tb_typing_speed.vhd untuk testbench utama yang mensimulasikan sistem. 	Fernanda Raeka Yan Putra
Input Generator & Software Integration Engineer	<ul style="list-style-type: none"> - Membuat char_input_sim.vhd untuk simulasi input karakter yang mewakili proses pengetikan. 	Naufal Rahman

	<ul style="list-style-type: none"> - Membuat typing_game.py untuk menghubungkan atau memonitor hasil simulasi. 	
--	---	--

Table 1. Roles and Responsibilities

CHAPTER 2

IMPLEMENTATION

2.1 EQUIPMENT

Software yang digunakan dalam proyek ini adalah sebagai berikut:

- Vivado
- Visual Studio Code

2.2 IMPLEMENTATION

Implementasi sistem Typing Speed Analyzer dilakukan dengan menggunakan pendekatan desain digital berbasis VHDL yang dijalankan melalui proses simulasi. Sistem ini menerima input karakter-karakter secara real time dari program python, lalu menghitung kecepatan mengetiknya berdasarkan durasi pengetikan dan jumlah karakter yang benar atau tidak typo.

Implementasi ini dilakukan dengan melakukan beberapa tahapan utama antara lain:

- 1. Perancangan Timer Module**

Modul ini dirancang untuk menghitung durasi pengetikan dalam satuan milisecond. Timer akan aktif ketika karakter pertama diketik dan berhenti ketika seluruh target teks selesai atau pengguna menekan tombol Enter.

- 2. Perancangan Keystroke Analyzer Module**

Modul ini dirancang untuk membaca karakter, membandingkan karakter yang diketik dengan karakter target, lalu menghitung kesalahan yang terjadi.

- 3. Perancangan Typing Speed Analyzer Module**

Modul ini dirancang untuk mengolah data waktu dan jumlah karakter untuk menghasilkan nilai WPM, jumlah kesalahan, dan akurasi pengetikan.

- 4. Perancangan Testbench dan Integrasi Python**

Testbench dirancang untuk melakukan simulasi yang berinteraksi dengan python. Python akan mengirim input karakter ke dalam file realtime_input.txt, lalu testbench membacanya untuk memicu sinyal validasi input pada sistem.

- 5. Perancangan Output Logging**

Output Logging adalah proses penyimpanan hasil akhir seperti WPM, akurasi pengetikan, dan jumlah kesalahan ke dalam output report.

Dengan implementasi ini, sistem Typing Speed Analyzer bisa melakukan simulasi pengetikan secara real time dan memberikan hasil analisis yang akurat terhadap performa pengetikan pengguna.

Tabel Implementasi File pada System:

File	Deskripsi Singkat
Package Sistem	
typing_speed_pkg.vhd	Mendefinisikan konstanta sistem, seperti frekuensi clock (100 KHz untuk simulasi).
Modul Utama	
timer_module.vhd	Menghitung durasi waktu pengetikan dalam milidetik.
keystroke_analyzer.vhd	Memproses input karakter, membandingkan dengan target, dan menghitung CPM, WPM, serta akurasi.
typing_speed_analyzer.vhd	Modul Top-Level yang menghubungkan timer dan analyzer menjadi satu sistem utuh.
Output & Logging	
result_txt_logger.vhd	Mencatat detail setiap karakter dan hasil akhir analisis ke dalam file laporan (output_report.txt).
output_report.txt	Log teknis dari sisi hardware (VHDL) yang merekam setiap karakter yang diproses dan status validasinya (OK/TYPO) secara <i>real-time</i> .
python_results.txt	Laporan akhir yang <i>user-friendly</i> , berisi ringkasan statistik seperti Waktu, CPM, WPM, Akurasi (%), dan Jumlah Typo
Simulasi & Testbench	
tb_typing_speed_file.vhd	Testbench utama yang mengatur simulasi, sinkronisasi handshake dengan Python, dan pembacaan file input.
User Interface	
typing_game.py	Script Python yang berfungsi sebagai generator soal acak dan interface input real-time bagi user.
target_soal.txt	File teks berisi soal yang dihasilkan Python untuk dibaca oleh VHDL.
realtime_input.txt	File perantara untuk mengirimkan karakter ketikan pengguna dari Python ke VHDL.

CHAPTER 3

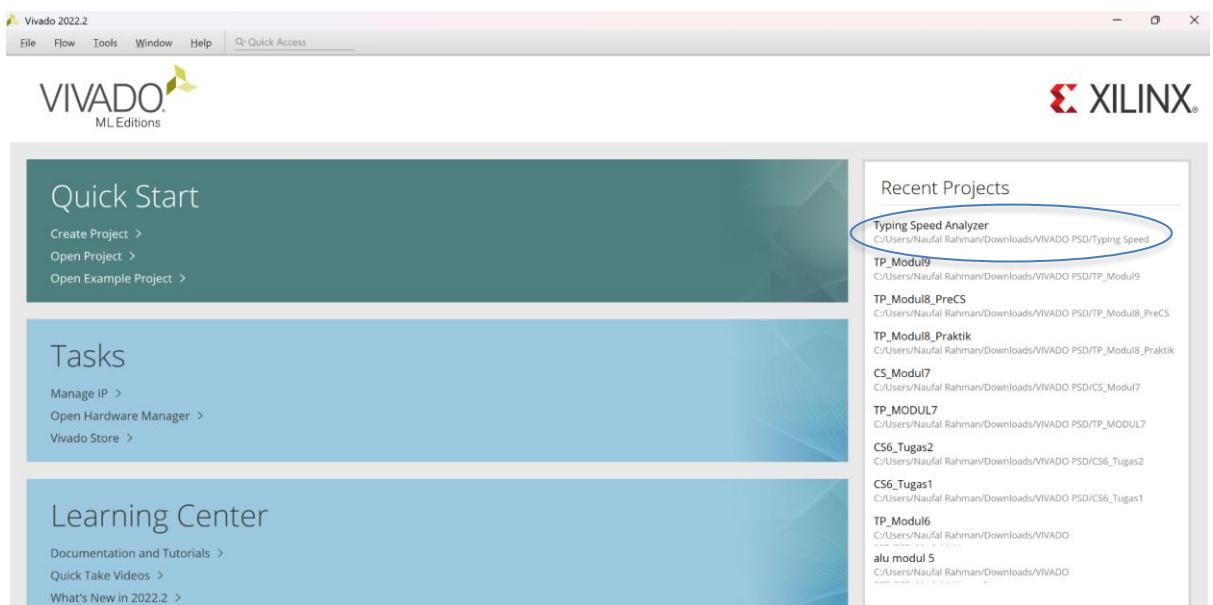
TESTING AND ANALYSIS

3.1 TESTING

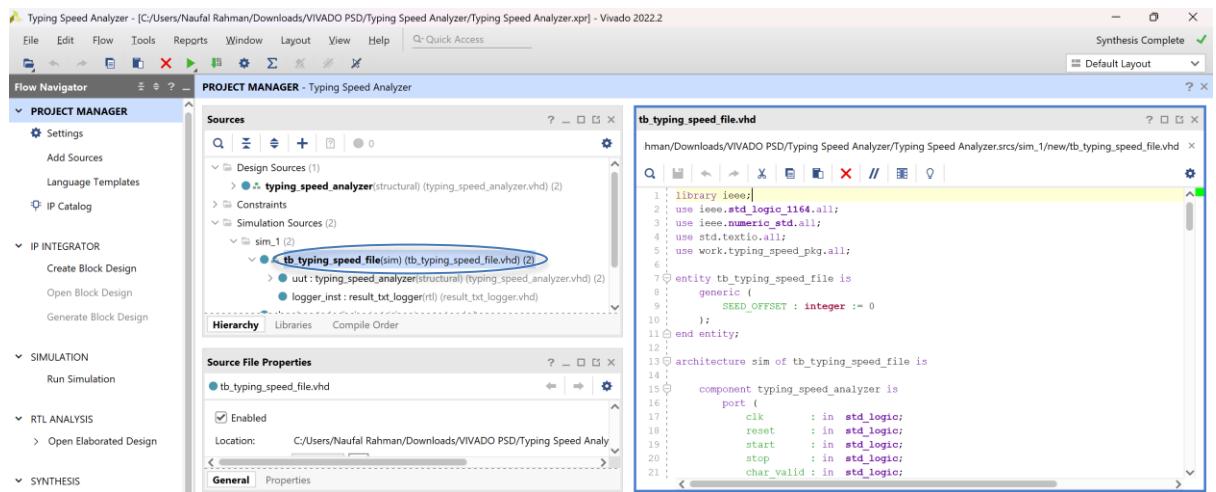
Pengujian sistem *Typing Speed Analyzer* dilakukan menggunakan metode simulasi *Hardware-in-the-Loop* (HIL) yang mengintegrasikan logika digital (VHDL) pada simulator Vivado dengan *UI (User Interface)* berbasis Python. Dalam skenario pengujian ini, *script* Python berfungsi sebagai *stimulus generator* yang menghasilkan kalimat uji acak dan menangkap input *keystroke* dari pengguna secara *real-time*. Data input tersebut kemudian dikirimkan ke *testbench* VHDL melalui mekanisme pertukaran file teks (*file I/O*) untuk diproses oleh *Device Under Test* (DUT). Validasi fungsional dilakukan dengan membandingkan metrik kinerja (CPM, WPM, dan akurasi) yang dihitung oleh modul perangkat keras VHDL terhadap hasil perhitungan perangkat lunak Python.

Langkah-langkah:

1. Buka Vivado, kemudian pilih project yang telah dibuat



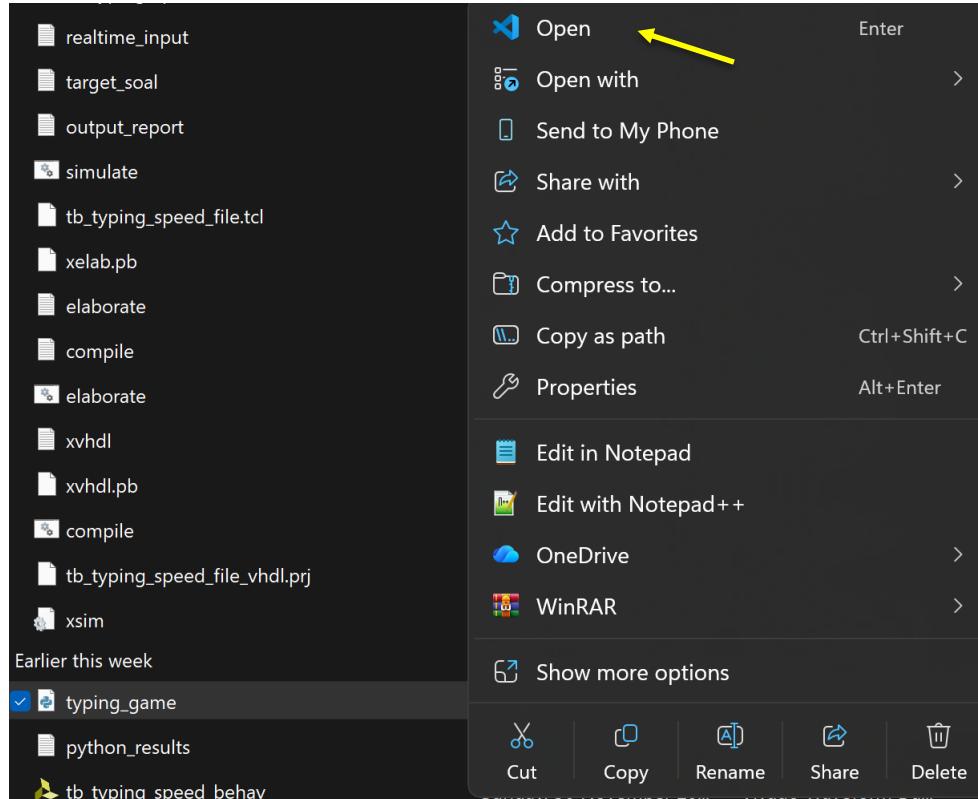
2. Klik pada tb_typing_speed_file.vhd



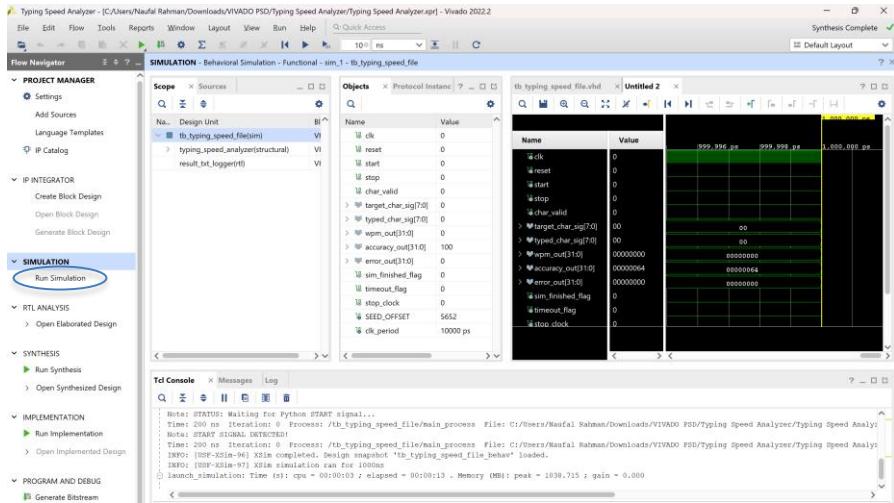
3. Buka file typing_game.py di direktori berikut ini (harus ditaruh disini filenya):

VIVADO PSD > Typing Speed Analyzer > Typing Speed Analyzer.sim > sim_1 > behav > xsim >

4. Klik kanan dan pilih Open



5. Ke Vivado lagi, klik Run Simulation, pilih Run Behavioral Simulation. Tampilannya seperti ini:



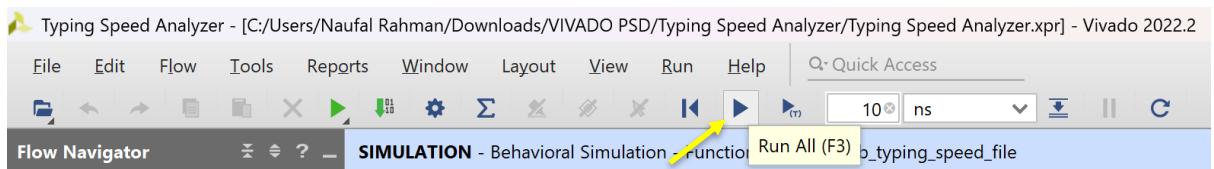
6. Buka script python di VS Code, kemudian klik Run Code

```

1 import msvrvt
2 import time
3 import os
4 import random
5
6 XSIM_FOLDER = r"C:\Users\Naufal Rahman\Downloads\VIVADO PSD\Typing Speed Analyzer\Typing Speed Analyzer.sim\sim_1\behav\xsim"
7
8 if not os.path.exists(XSIM_FOLDER):
9     print("=*60")
10    print("ERROR: Folder xsim tidak ditemukan!")
11    XSIM_FOLDER = os.getcwd()
12    print(f" Menggunakan folder saat ini: {XSIM_FOLDER}")
13
14 try:
15     os.chdir(XSIM_FOLDER)
16 except:
17     pass

```

7. Sebelum mulai mengetik di VS Code, buka lagi Vivado dan pilih Run All



8. Mulai mengetik di VS Code sesuai dengan kalimat yang muncul

```

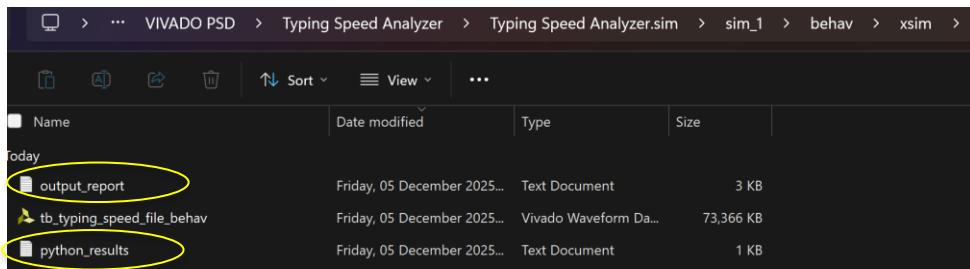
=====
VHDL TYPING SPEED TEST - CLEAN MODE
=====
Soal siap! Menunggu VHDL...
=====
SOAL : komputer generasi terbaru menggunakan teknologi quantum computing yang revolusioner
-----
KETIK: komputer generasi terbaru menggunakan teknologi quantum computing yang revolusioner
=====
```

3.2 RESULT

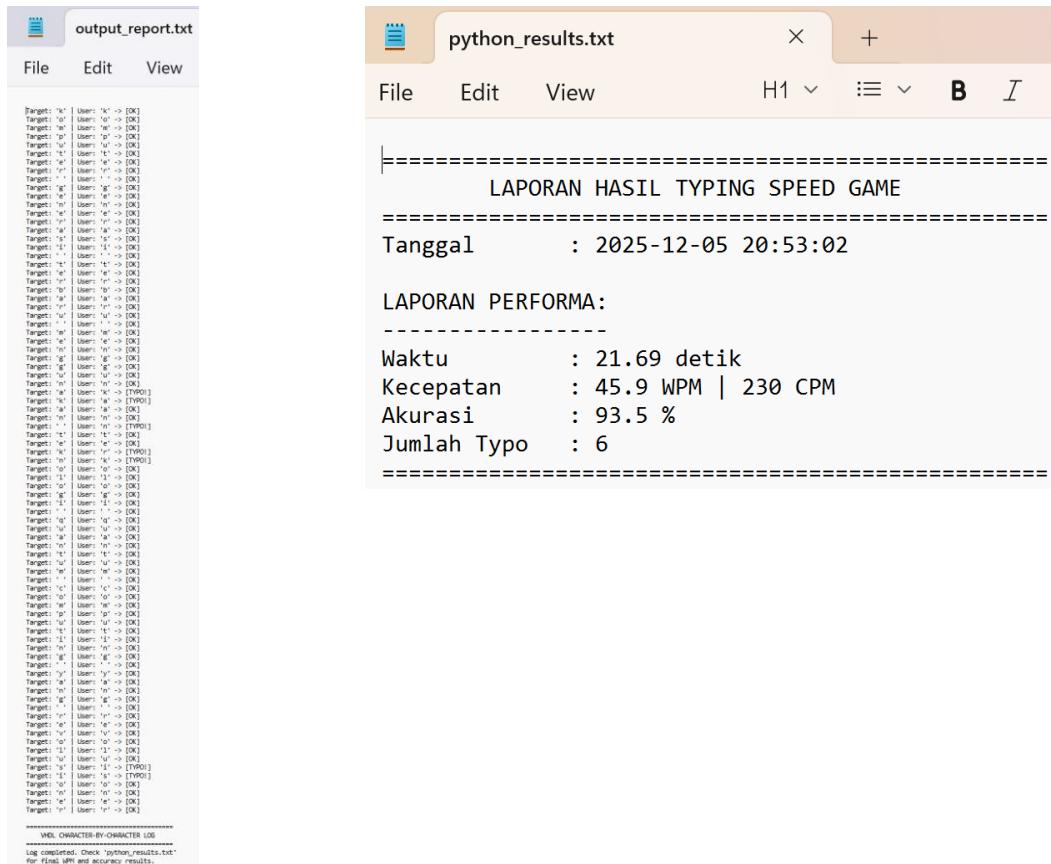
Setelah testing dilakukan, simulation di Vivado akan berhenti dan menampilkan:

```
Tcl Console x Messages Log ? - □
[ ] Note: =====
Time: 188174715 ns Iteration: 1 Process: /tb_typing_speed_file/main_process File: C:/Users/Naufal Rahman/Downloads/VIVADO PSD/Typing Speed Analyzer/Typing Speed
Note: VHDL SIMULATION COMPLETE
Time: 188174715 ns Iteration: 1 Process: /tb_typing_speed_file/main_process File: C:/Users/Naufal Rahman/Downloads/VIVADO PSD/Typing Speed Analyzer/Typing Speed
Note: Characters logged: 83
Time: 188174715 ns Iteration: 1 Process: /tb_typing_speed_file/main_process File: C:/Users/Naufal Rahman/Downloads/VIVADO PSD/Typing Speed Analyzer/Typing Speed
Note: Check 'output_report.txt' for character-by-character log
Time: 188174715 ns Iteration: 1 Process: /tb_typing_speed_file/main_process File: C:/Users/Naufal Rahman/Downloads/VIVADO PSD/Typing Speed Analyzer/Typing Speed
Note: Check 'python_results.txt' for final WPM/accuracy
Time: 188174715 ns Iteration: 1 Process: /tb_typing_speed_file/main_process File: C:/Users/Naufal Rahman/Downloads/VIVADO PSD/Typing Speed Analyzer/Typing Speed
Note: =====
Time: 188174715 ns Iteration: 1 Process: /tb_typing_speed_file/main_process File: C:/Users/Naufal Rahman/Downloads/VIVADO PSD/Typing Speed Analyzer/Typing Speed
Failure: Simulation finished
Time: 188374815 ns Iteration: 0 Process: /tb_typing_speed_file/main_process File: C:/Users/Naufal Rahman/Downloads/VIVADO PSD/Typing Speed Analyzer/Typing Speed
$finish called at time : 188374815 ns : file "C:/Users/Naufal Rahman/Downloads/VIVADO PSD/Typing Speed Analyzer/Typing Speed Analyzer.srcsim_1/new/tb_typing_speed
run: Time (s): cpu = 00:00:17 : elapsed = 00:00:21 . Memory (MB): peak = 1057.285 ; gain = 0.285
```

Lakukan pengecekan hasil pada output report.txt dan python results.txt:



Hasil testing pada output_report.txt dan python_results.txt:



3.3 ANALYSIS

Berdasarkan hasil pengujian integrasi sistem *Hardware-in-the-Loop* (HIL), modul VHDL *Typing Speed Analyzer* berhasil melakukan sinkronisasi data secara *real-time* dengan antarmuka input Python. Mekanisme *handshake* dan *polling* file yang diterapkan terbukti efektif dalam mencegah *data loss* dan *race condition* antara kedua domain waktu yang berbeda. Sistem mampu mendeteksi karakter input dengan presisi tinggi, termasuk penanganan koreksi input (seperti tombol *Backspace*) tanpa menyebabkan desinkronisasi pada logika pengecekan karakter. Setiap karakter yang diketik pengguna diproses secara akurat oleh VHDL, menghasilkan perhitungan performa (CPM, WPM, dan Akurasi) yang konsisten dan valid.

CHAPTER 4

CONCLUSION

Berdasarkan perancangan, implementasi, dan pengujian yang telah dilakukan pada sistem *Typing Speed Analyzer*, dapat ditarik beberapa kesimpulan sebagai berikut:

1. **Keberhasilan Integrasi Hardware-Software:** Sistem berhasil mengimplementasikan konsep *Hardware-in-the-Loop* (HIL) dengan mengintegrasikan modul logika digital VHDL pada simulator Vivado dengan *user interface* berbasis Python. Mekanisme pertukaran data melalui file teks (target_soal.txt dan realtime_input.txt) terbukti dapat menjembatani komunikasi antara perangkat lunak input dan simulasi perangkat keras secara *real-time*.
2. **Akurasi Pengukuran Performa:** *Keystroke Analyzer* yang dirancang terbukti mampu menghitung parameter kinerja pengetikan secara presisi. Pengujian menunjukkan bahwa sistem dapat mengukur kecepatan (WPM/CPM), akurasi (%), dan jumlah kesalahan (*typo*) dengan hasil yang konsisten dan valid serta dapat dilihat di python_results
3. **Penanganan Input Dinamis:** Sistem terbukti tangguh dalam menangani variabilitas input pengguna, termasuk fitur koreksi kesalahan menggunakan tombol *Backspace*. Logika *smart backspace* dan sinkronisasi indeks soal yang diterapkan pada *testbench* berhasil mencegah terjadinya desinkronisasi data, memastikan bahwa setiap karakter yang diketik dinilai berdasarkan target yang tepat.
4. **Efektivitas Modul Pendukung:** Seluruh modul pendukung seperti *Timer Module* dan *Result Logger* berfungsi sesuai spesifikasi. Timer mampu melacak durasi pengetikan dengan akurat dalam satuan milidetik, sementara Logger berhasil mendokumentasikan seluruh proses pengetikan dan hasil akhir ke dalam format laporan yang terstruktur (output_report.txt).

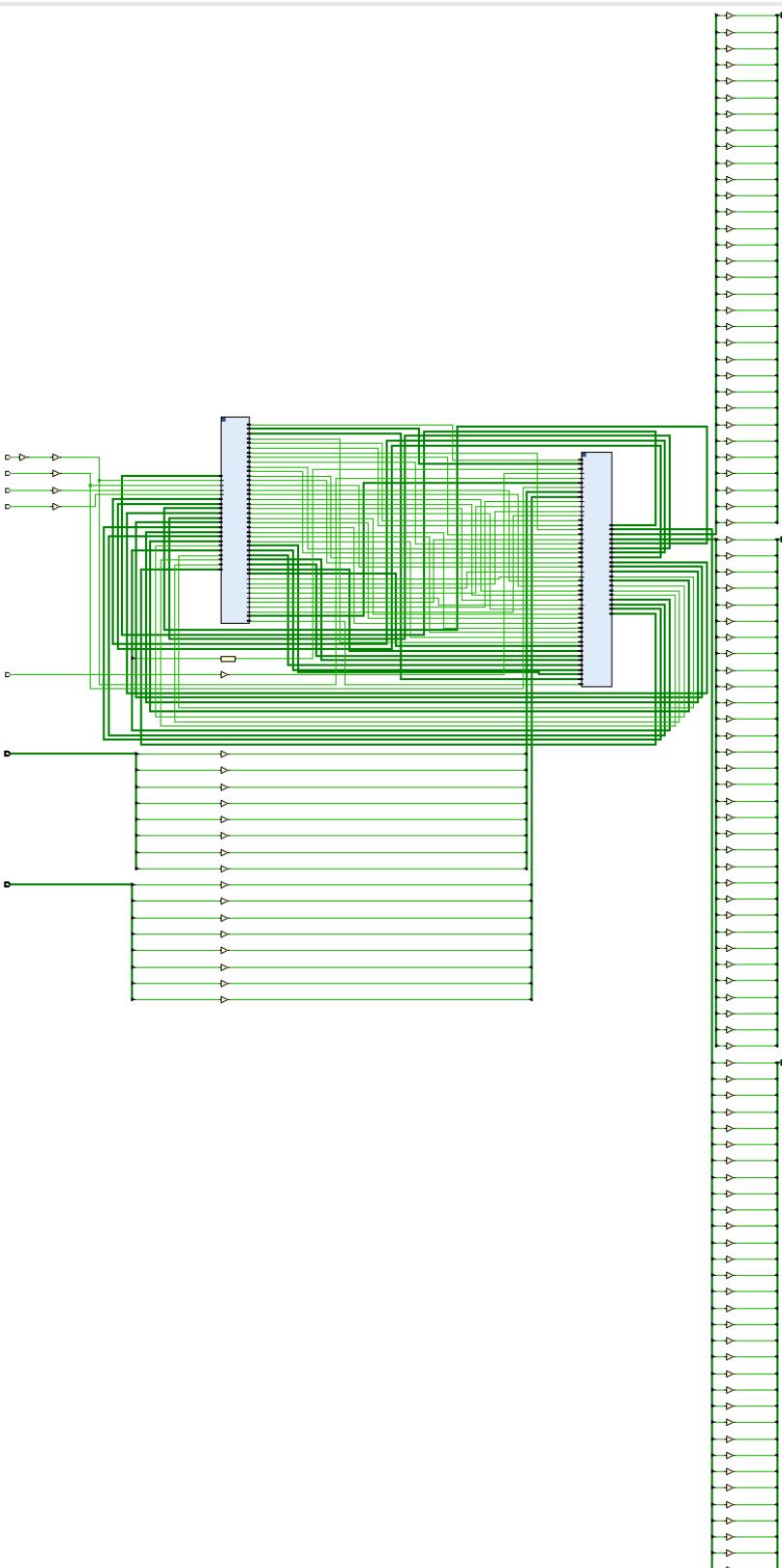
Secara keseluruhan, proyek ini telah memenuhi seluruh tujuan yang ditetapkan, yaitu merancang sistem digital yang mampu mengukur dan menganalisis kecepatan serta akurasi mengetik secara *real-time* dengan tingkat akurasi yang tinggi.

REFERENCES

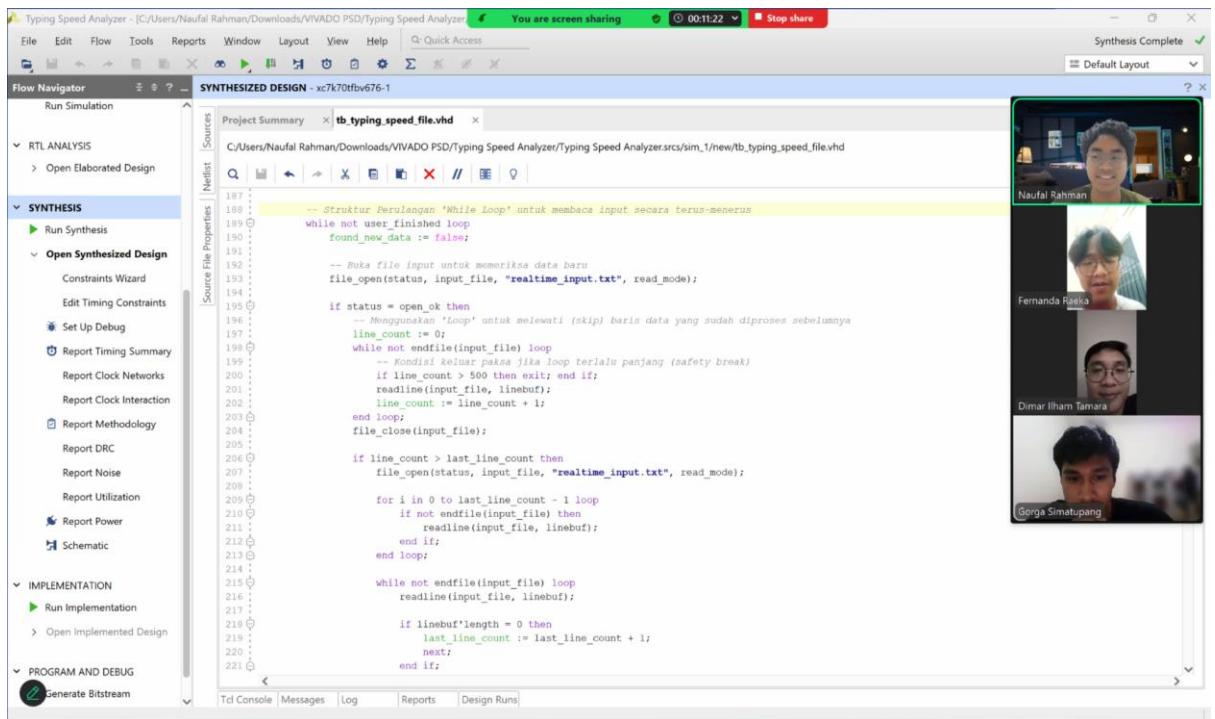
- [1] "What is Hardware-in-the-Loop Testing? | Ansys," *Ansys.com*, 2025.
<https://www.ansys.com/simulation-topics/what-is-hardware-in-the-loop-testing> diakses [Accessed: Dec. 5, 2025].
- [2] "Typing Speed | Typing Pal," [www.typingpal.com](https://www.typingpal.com/en/documentation/school-edition/pedagogical-resources/typing-speed).
<https://www.typingpal.com/en/documentation/school-edition/pedagogical-resources/typing-speed> [Accessed: Dec. 5, 2025].
- [3] Sabanci University, "VHDL Basic I/O," *Sabanci University Electronics Engineering Course Material*. [Online]. Available: https://people.sabanciuniv.edu/erkays/el310/io_10.pdf. [Accessed: Dec. 5, 2025].
- [4] MathWorks, "Basics of Hardware-in-the-Loop Simulation," *MathWorks Documentation*. [Online]. Available: <https://www.mathworks.com/help/simscape/ug/what-is-hardware-in-the-loop-simulation.html>. [Accessed: Dec. 5, 2025].
- [5] University of Toronto, "PS/2 Controller," *University of Toronto ECE241 Course Material*. [Online]. Available: https://www.eecg.utoronto.ca/~jayar/ece241_08F/AudioVideoCores/ps2/ps2.html. [Accessed: Dec. 5, 2025].
- [6] Embedded Thoughts, "FPGA Keyboard Interface," *Embedded Thoughts Blog*, Jul. 5, 2016. [Online]. Available: <https://embeddedthoughts.com/2016/07/05/fpga-keyboard-interface/>. [Accessed: Dec. 5, 2025].

APPENDICES

Appendix A: Project Schematic



Appendix B: Documentation



The screenshot shows the Vivado IDE interface with a video conference overlay. The code editor displays the `tb_typing_speed_file.vhd` file, which contains VHDL code for a 'While Loop' to read data from a file. The video conference shows four participants: Naufal Rahman, Fernanda Riaika, Dimar Ilham Tamara, and Gorga Simatupang.

```
-- Struktur Perulangan 'While Loop' untuk membaca input secara terus-menerus
while not user_finished loop
    found_new_data := false;
    -- Buka file input untuk memeriksa data baru
    file_open(status, input_file, "realtime_input.txt", read_mode);
    if status = open_ok then
        -- Menggunakan 'Loop' untuk melewati (skip) baris data yang sudah diproses sebelumnya
        line_count := 0;
        while not endfile(input_file) loop
            -- Kondisi keluar paks jika loop terlalu panjang (safety break)
            if line_count > 500 then exit; end if;
            readline(input_file, linebuf);
            line_count := line_count + 1;
        end loop;
        file_close(input_file);
        if line_count > last_line_count then
            file_open(status, input_file, "realtime_input.txt", read_mode);
            for i in 0 to last_line_count - 1 loop
                if not endfile(input_file) then
                    readline(input_file, linebuf);
                end if;
            end loop;
            while not endfile(input_file) loop
                readline(input_file, linebuf);
                if linebuf'length = 0 then
                    last_line_count := last_line_count + 1;
                    next;
                end if;
            end loop;
        end if;
    end if;
end loop;
```