

# Typing Speed Analyzer

Final Project PSD Group 14





# GROUP 14

Naufal Rahman (2406413142)

Dimar Ilham Tamara (2406413804)

Fernanda Raeka Yan Putra (2406415791)

Gogra Friedrik Simatupang (2406487020)



# CONTENTS



1

Introduction

2

Implementation

3

Testing and Analysis

4

Conclusion

5

Appendix

# CHAPTER 1

# INTRODUCTION

Background

Project Description

Objectives

Roles and Responsibilities



# BACKGROUND

Di era digital, efisiensi pengetikan yang menggabungkan kecepatan dan akurasi sangat krusial untuk produktivitas, namun seringkali sulit diukur secara objektif karena variasi kebiasaan pengguna. Mengingat pentingnya keseimbangan antara kecepatan (WPM) dan presisi, kelompok kami mengembangkan Typing Speed Analyzer, sebuah sistem digital berbasis VHDL yang dirancang khusus untuk mengukur dan menganalisis performa pengetikan secara real-time, akurat, dan objektif sebagai bahan evaluasi keterampilan mengetik.



# PROJECT DESCRIPTION

Typing Speed Analyzer adalah **sistem digital berbasis VHDL** yang dirancang untuk **mengukur dan menganalisis performa mengetik secara real-time**. Dengan mengintegrasikan modul timer presisi dan komparator teks, sistem ini memproses input karakter pengguna untuk menghitung parameter, seperti **kecepatan (WPM)**, **jumlah kesalahan**, dan **tingkat akurasi** secara otomatis dan objektif.



# OBJECTIVES

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 (typo).
3. Menerapkan proses perbandingan antara karakter yang diketik dengan karakter target untuk mendeteksi kesalahan pengetikan secara efektif.



# ROLES AND RESPONSIBILITIES

**Timer & System Configuration Engineer → Gogra Friedrik Simatupang**

(typing\_speed\_pkg.vhd dan timer\_module.vhd)

**Performance Analyzer & Output Handler → Dimar Ilham Tamara**

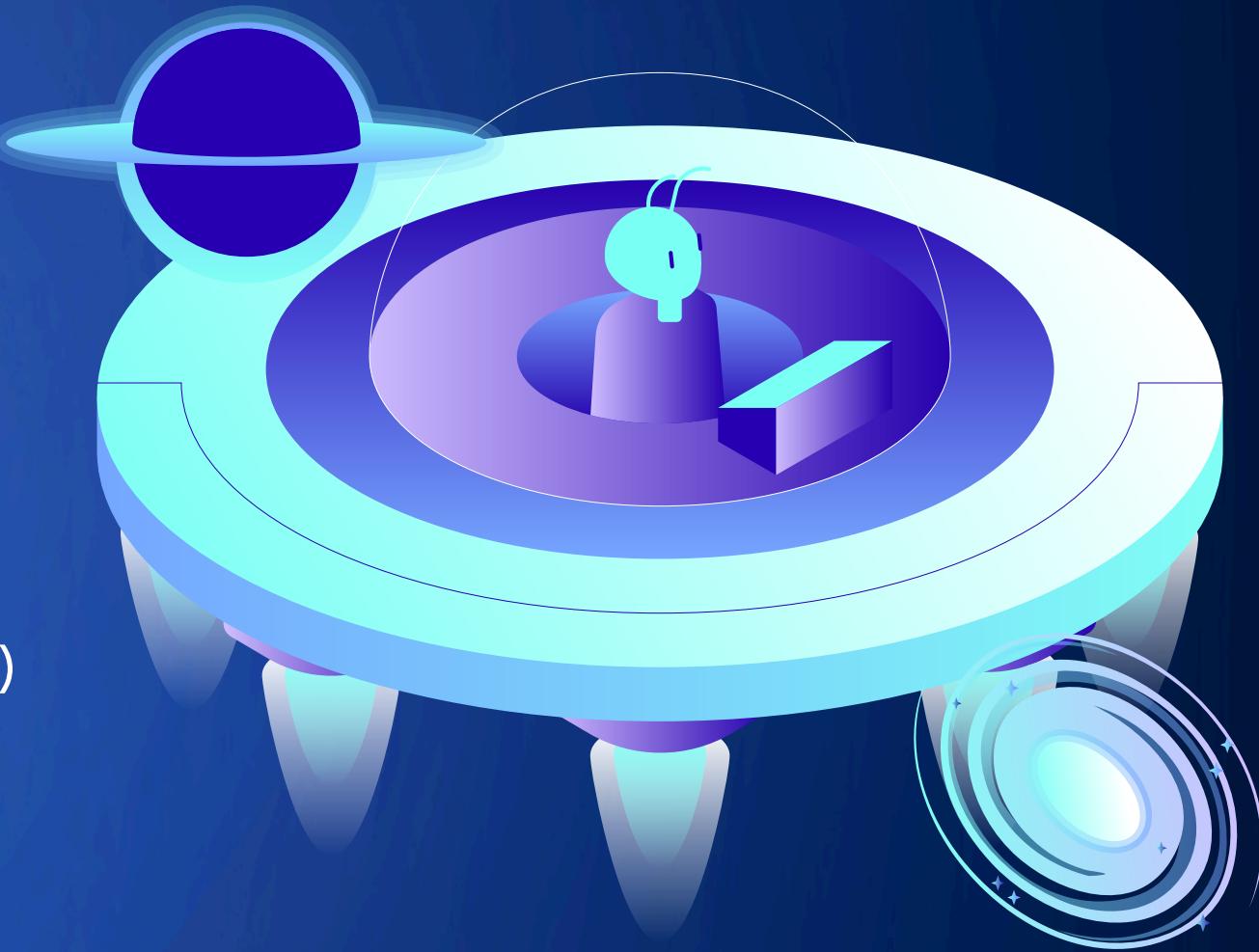
(keystroke\_analyzer.vhd, typing\_speed\_analyzer.vhd, dan result\_txt\_logger.vhd)

**Testbench & Simulation Data Engineer → Fernanda Raeka Yan Putra**

(tb\_typing\_speed.vhd)

**Input Generator & Software Integration Engineer → Naufal Rahman**

(char\_input\_sim.vhd dan typing\_game.py)



# CHAPTER 2

# IMPLEMENTATION

Equipment

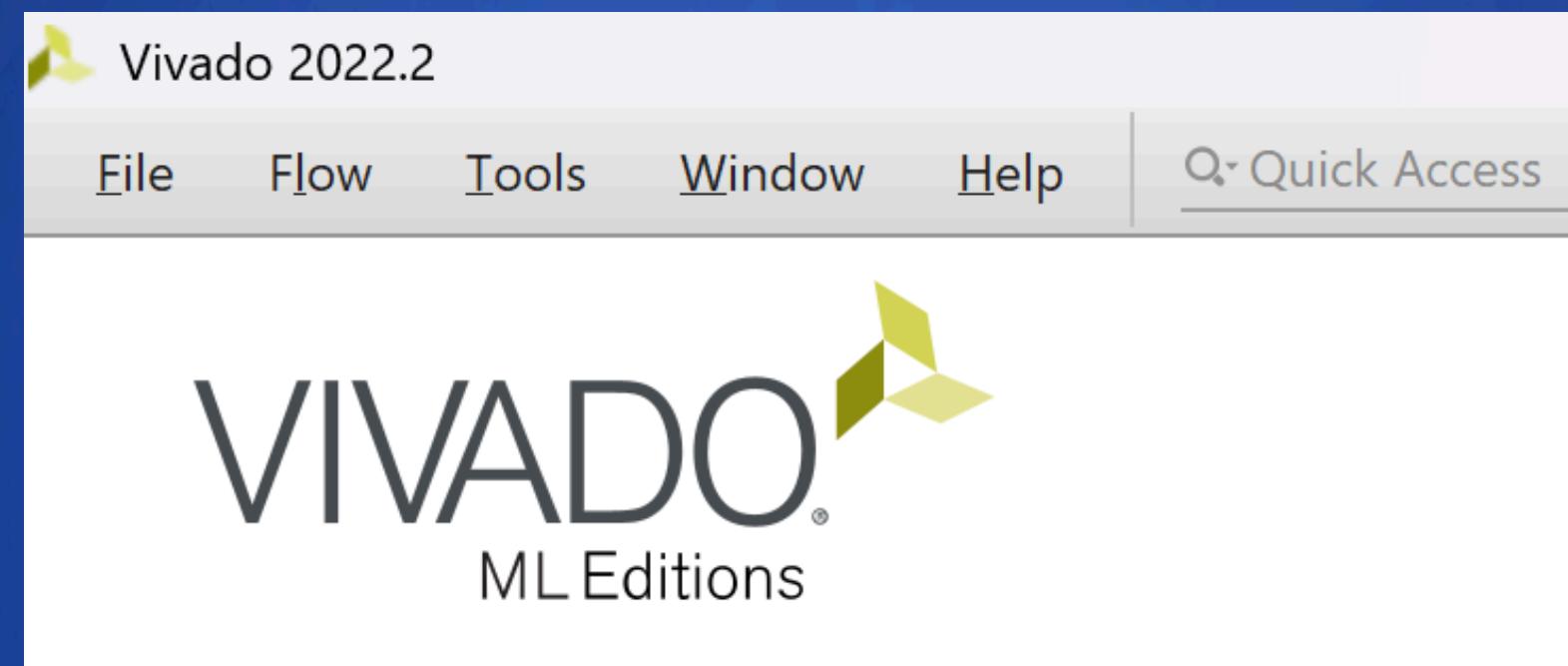
Module Implementation

System Implementation



# EQUIPMENT

Vivado 2022.2



Visual Studio Code



# MODULE IMPLEMENTATION

## Module

2  
(Dataflow Style)

## Implementation

Dataflow digunakan untuk **menetapkan nilai sinyal secara concurrent** (bersamaan) tanpa menunggu clock, seperti menghubungkan sinyal internal ke port output utama.

*Snippet code typing\_speed\_analyzer.vhd:*

```
-- Menghubungkan sinyal internal langsung ke port output
wpm_out      <= wpm_sig;
accuracy_out <= accuracy_sig;
error_out    <= error_sig;
```

# MODULE IMPLEMENTATION

## Module

3  
(Behavioral Style)

## Implementation

Behavioral digunakan di dalam blok **process** untuk **mendeskripsikan algoritma kompleks secara berurutan (sekuensial)**, seperti logika pengecekan huruf dan penghitungan counter.

*Snippet code keystroke\_analyzer.vhd:*

```
process(clk)
    variable total_i      : integer;
    variable correct_i   : integer;
    variable elapsed_i   : integer;
    variable wpm_i       : integer;
    variable acc_i       : integer;
begin
    if rising_edge(clk) then
        if reset = '1' then
            total_keystrokes  <= (others => '0');
            correct_keystrokes <= (others => '0');
            error_count        <= (others => '0');
            wpm_calc            <= (others => '0');
            acc_calc             <= (others => '0');
            char_valid_prev     <= '0';
        else
            -- Logika Pengecekan Huruf
            if char_valid = '1' and char_valid_prev = '0' then
                total_keystrokes <= total_keystrokes + 1;

                if typed_char = target_char then
                    correct_keystrokes <= correct_keystrokes + 1;
                else
                    error_count <= error_count + 1;
                end if;
            end if;
            char_valid_prev <= char_valid;
```

# MODULE IMPLEMENTATION

## Module

4  
(Testbench)

## Implementation

Testbench dirancang sebagai modul penguji tanpa port input/output fisik, yang bertugas membangkitkan sinyal simulasi untuk memverifikasi sistem dan berinteraksi dengan lingkungan eksternal (Python) melalui mekanisme I/O file untuk simulasi Hardware-in-the-Loop (HIL).

*Snippet code tb\_typing\_speed\_file.vhd:*

```
-- Deklarasi variabel file untuk komunikasi dengan Python
file target_file : text;
file input_file  : text;
variable linebuf : line;
variable int_val : integer;
variable status  : file_open_status;

-- Proses sinkronisasi Handshake: Menunggu sinyal START (nilai 1) dari Python
report "STATUS: Waiting for Python START signal..." severity note;

while wait_for_start loop
    file_open(status, input_file, "realtime_input.txt", read_mode);

    if status = open_ok then
        if not endfile(input_file) then
            readline(input_file, linebuf);
            read(linebuf, int_val);

            if int_val = 1 then
                wait_for_start := false;
                report "START SIGNAL DETECTED!" severity note;
            end if;
        end if;
        file_close(input_file);
    end if;

    if wait_for_start then
        for i in 0 to 9999 loop
            wait until rising_edge(clk);
            if timeout_flag = '1' then exit; end if;
        end loop;
    end if;

    if timeout_flag = '1' then
        stop_clock <= '1';
        wait;
    end if;
end loop;
```

# MODULE IMPLEMENTATION

## Module

### 5 (Structural Programming)

## Implementation

Pemrograman struktural digunakan pada modul top-level untuk merakit dan menghubungkan komponen-komponen kecil (Timer dan Analyzer) menjadi satu sistem yang utuh. Sinyal internal seperti elapsed\_ms\_sig digunakan sebagai "kabel" untuk menghubungkan output dari satu modul ke input modul lainnya melalui mekanisme port map.

*Snippet code typing\_speed\_analyzer.vhd:*

```
architecture structural of typing_speed_analyzer is
    -- Deklarasi sinyal penghubung (kabel internal)
    signal elapsed_ms_sig : unsigned(31 downto 0);

begin
    -- Instansiasi Modul Timer
    timer_inst : entity work.timer_module
        port map (
            clk      => clk,
            reset   => reset,
            start    => start,
            stop     => stop,
            elapsed_ms => elapsed_ms_sig -- Dihubungkan ke sinyal internal
        );

    -- Instansiasi Modul Analyzer
    analyzer_inst : entity work.keystroke_analyzer
        port map (
            clk      => clk,
            reset   => reset,
            char_valid => char_valid,
            target_char => target_char,
            typed_char  => typed_char,
            elapsed_ms  => elapsed_ms_sig,-- Menerima data dari Timer
            wpm_out    => wpm_sig,
            accuracy_out => accuracy_sig,
            error_cnt_out => error_sig
        );

```

# MODULE IMPLEMENTATION

## Module

### 6 (Looping Construct)

## Implementation

While loop digunakan secara intensif dalam testbench untuk membaca file input karakter demi karakter secara terus-menerus hingga simulasi selesai.

*Snippet code tb\_typing\_speed\_file:*

```
-- 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 paksa 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);
```

# MODULE IMPLEMENTATION

## Module

7  
(Procedure,  
Function,  
and Impure  
Function)

## Implementation

Menerapkan **Function (to\_integer)** untuk mengonversi tipe data sinyal menjadi integer agar bisa diproses, **Procedure (write)** untuk menyusun string dan data ke dalam variabel buffer baris (Line), serta **Impure Procedure (writeln)** yang memiliki *side effect* untuk menuliskan baris tersebut secara fisik ke file eksternal.

*Snippet code result\_txt\_logger.vhd:*

```
-- 'to_integer' adalah Function karena ia menerima input (unsigned)
-- dan mengembalikan nilai baru (integer) tanpa mengubah data aslinya.
char_targ := character'val(to_integer(unsigned(target_char)));
char_user := character'val(to_integer(unsigned(typed_char)));

-- 'write' adalah Procedure karena ia melakukan aksi (menulis data ke variabel L)
-- dan tidak mengembalikan nilai (return value), melainkan memodifikasi parameter 'L' secara langsung.
write(L, string'("Target: '"));
write(L, char_targ);

-- 'writeln' disebut Impure karena memiliki side effect
-- yang mengubah kondisi di luar lingkungan VHDL, yaitu menulis data secara fisik ke file 'output_report.txt'.
writeln(outfile, L);
```

# SYSTEM IMPLEMENTATION

File	Deskripsi Singkat
<b>Package Sistem</b> typing_speed_pkg.vhd	Mendefinisikan konstanta sistem, seperti frekuensi clock (100 KHz untuk simulasi).
<b>Modul Utama</b> 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.
<b>Output &amp; Logging</b> 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
<b>Simulasi &amp; Testbench</b> tb_typing_speed_file.vhd	Testbench utama yang mengatur simulasi, sinkronisasi handshake dengan Python, dan pembacaan file input.
<b>User Interface</b> 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.

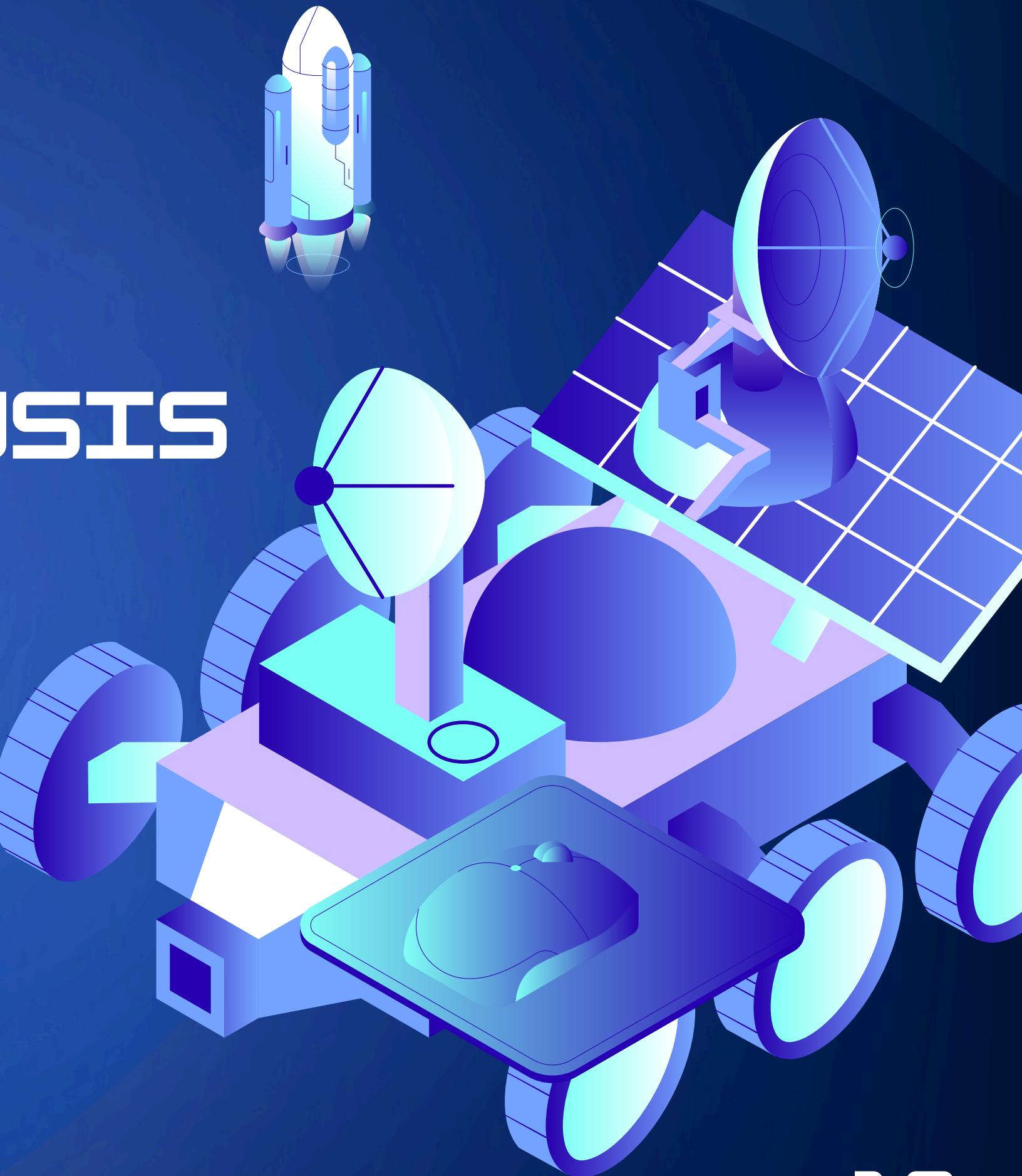


# CHAPTER 3

## TESTING AND ANALYSIS

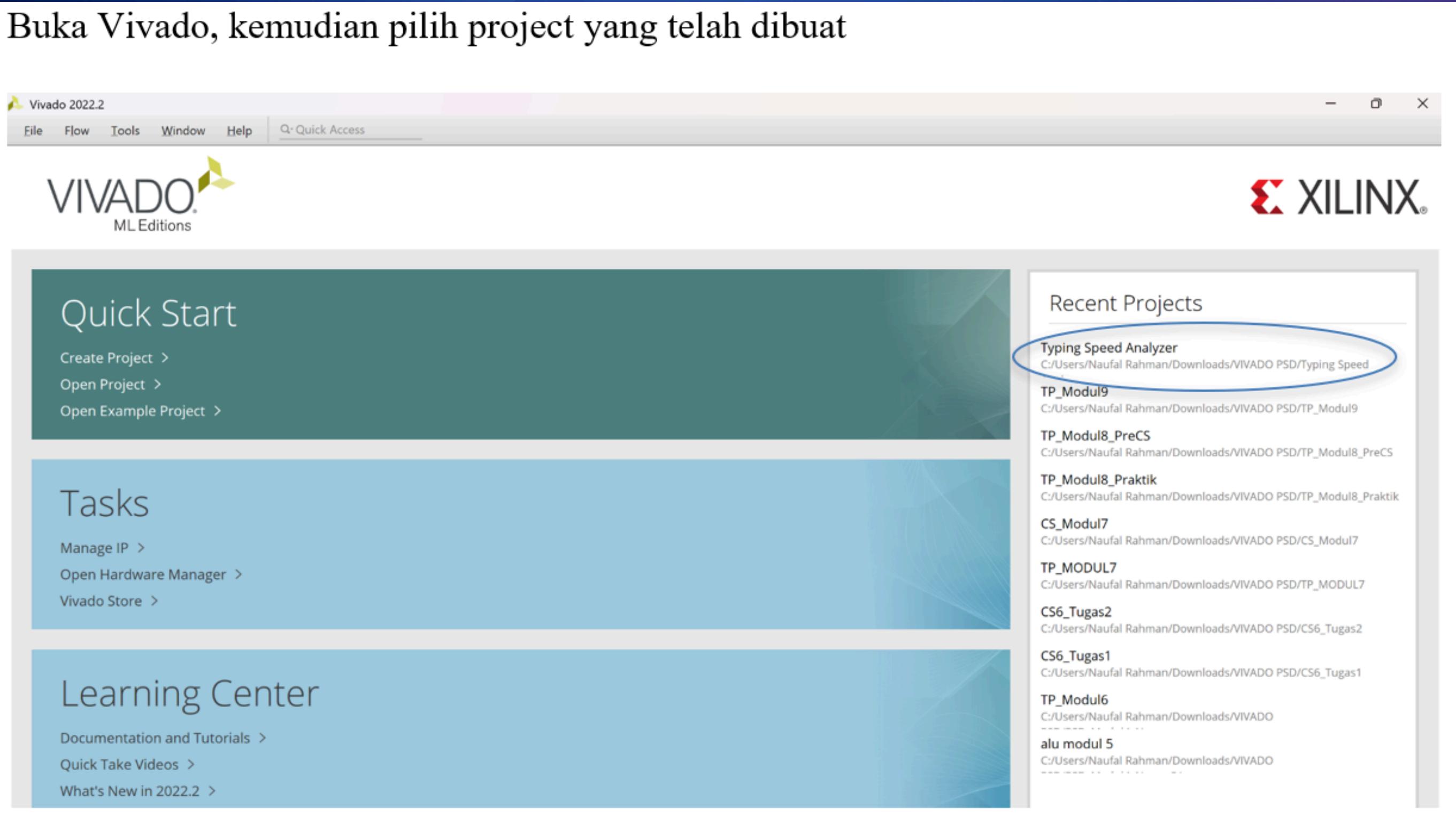
Testing

Analysis



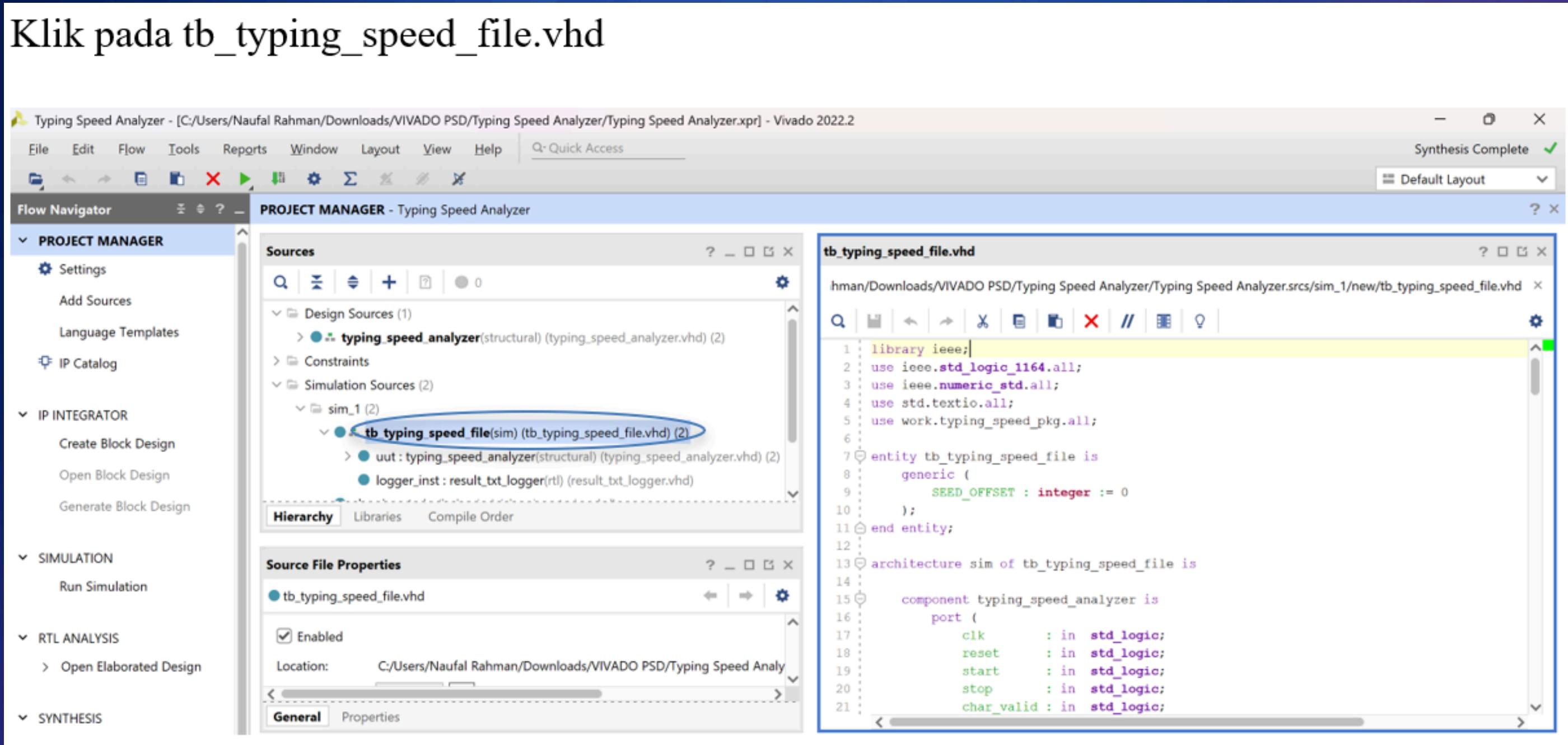
# TESTING

Buka Vivado, kemudian pilih project yang telah dibuat



# TESTING

Klik pada tb\_typing\_speed\_file.vhd



The screenshot shows the Vivado 2022.2 Project Manager interface. The 'PROJECT MANAGER' section is open, displaying the 'Sources' tree. Under 'Design Sources', there is a structural source named 'typing\_speed\_analyzer'. Under 'Simulation Sources', there is a folder 'sim\_1' containing two items: 'tb\_typing\_speed\_file' (selected) and 'uut'. The 'tb\_typing\_speed\_file' item is highlighted with a blue oval. In the bottom right corner, the code editor window for 'tb\_typing\_speed\_file.vhd' is visible, showing the VHDL code for the testbench.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use std.textio.all;
use work.typing_speed_pkg.all;

entity tb_typing_speed_file is
    generic (
        SEED_OFFSET : integer := 0
    );
end entity;

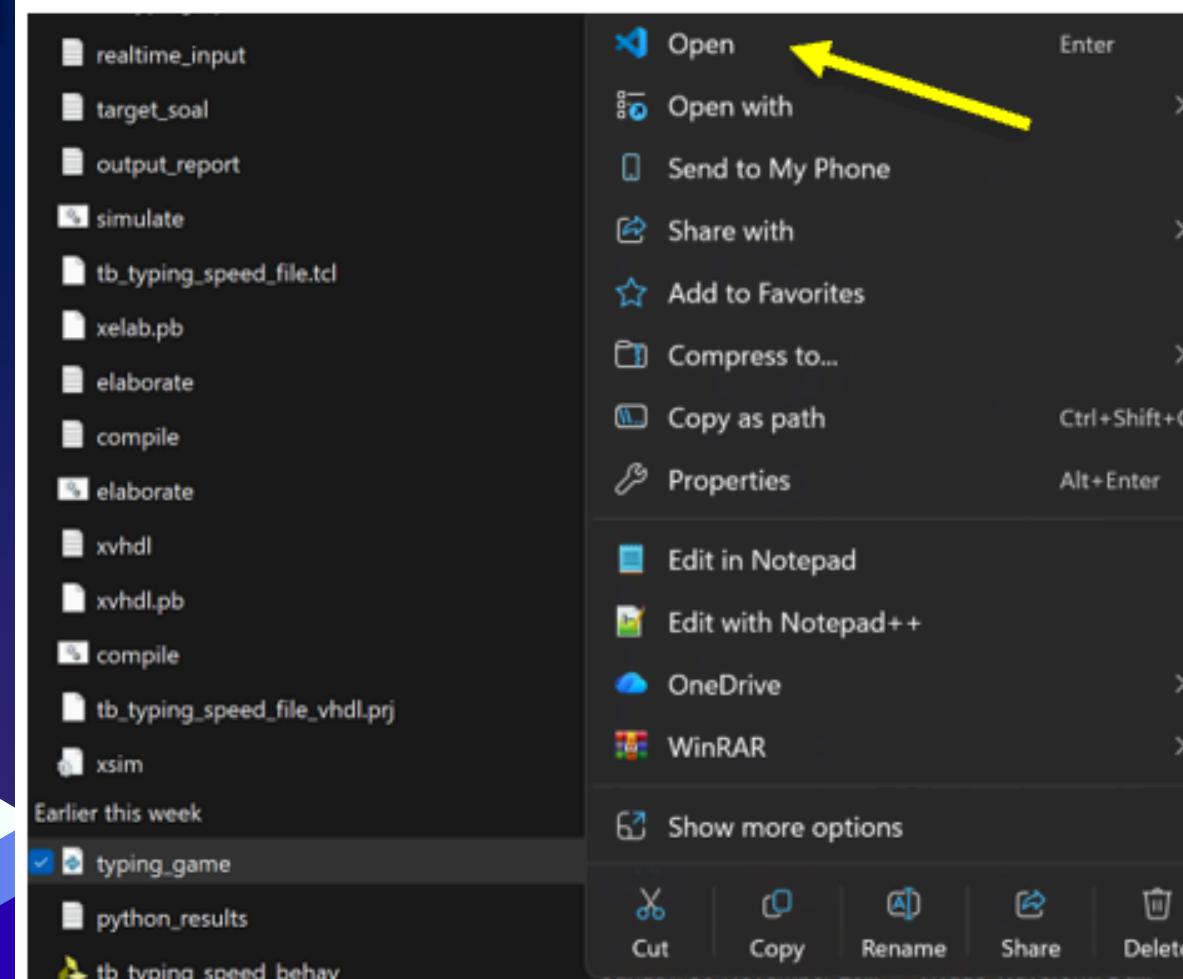
architecture sim of tb_typing_speed_file is
begin
    component typing_speed_analyzer is
        port (
            clk      : in  std_logic;
            reset   : in  std_logic;
            start   : in  std_logic;
            stop    : in  std_logic;
            char_valid : in  std_logic
        );
    end component;
    uut : typing_speed_analyzer;
    logger_inst : result_txt_logger;
    -- ...
end architecture;
```

# TESTING

Buka file typing\_game.py di direktori berikut ini (harus ditaruh disini filenya):

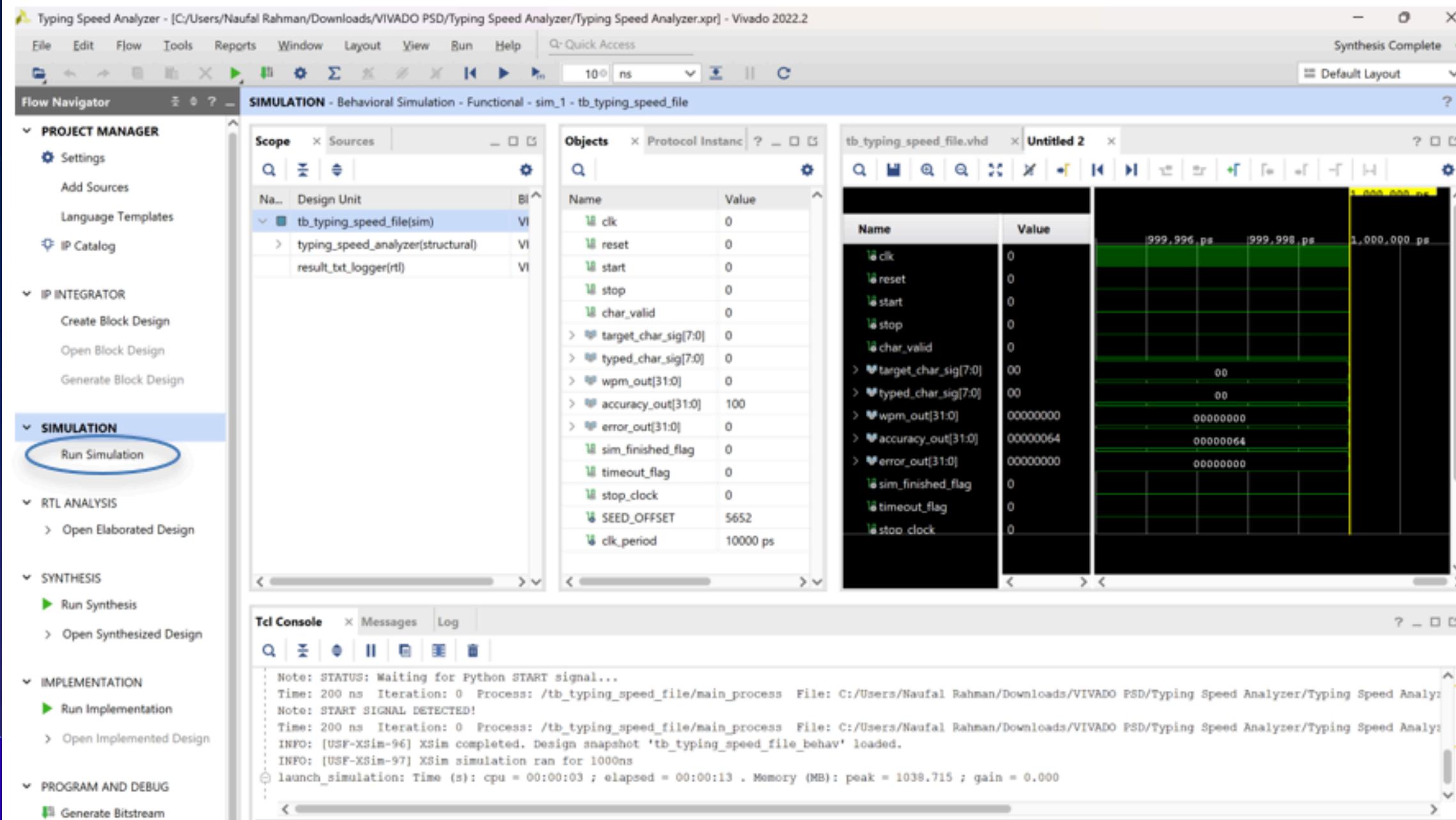
VIVADO PSD > Typing Speed Analyzer > Typing Speed Analyzer.sim > sim\_1 > behav > xsim >

Klik kanan dan pilih Open



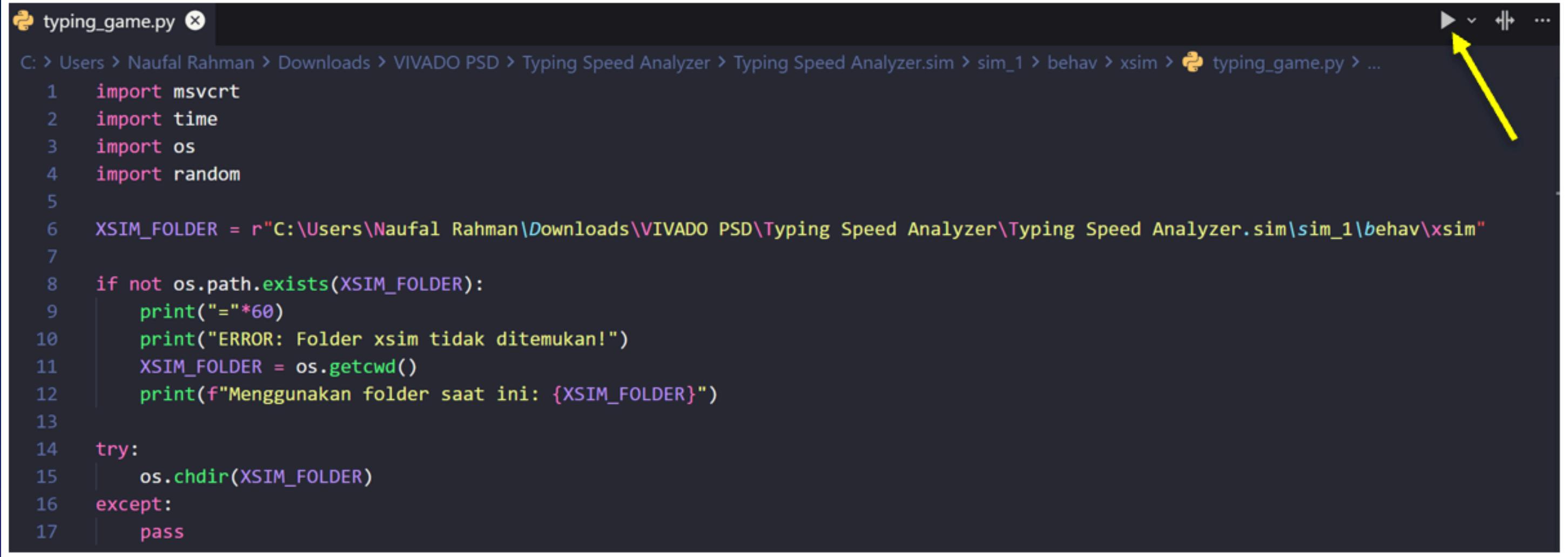
# TESTING

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



# TESTING

Buka script python di VS Code, kemudian klik Run Code



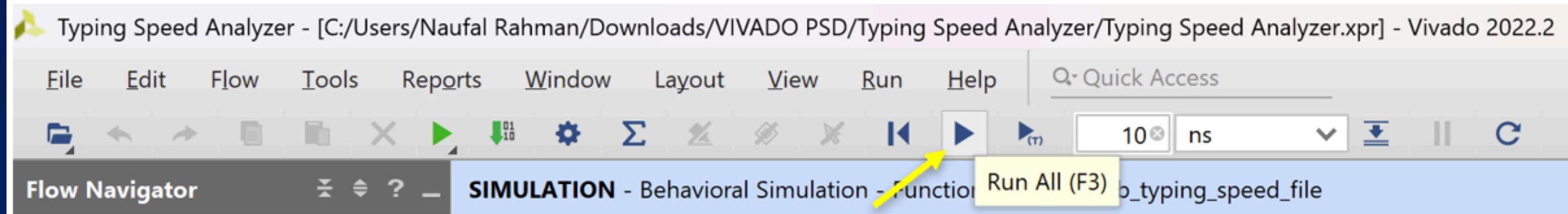
```
typing_game.py

C: > Users > Naufal Rahman > Downloads > VIVADO PSD > Typing Speed Analyzer > Typing Speed Analyzer.sim > sim_1 > behav > xsim > typing_game.py > ...

1 import msvcrt
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
```

# TESTING

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



Mulai mengetik di VS Code sesuai dengan kalimat yang muncul

A screenshot of the VS Code terminal window. The tabs at the top are PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined in pink), PORTS, and GITLENS. The terminal output is as follows:

```
=====
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
```

# TESTING

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

Lakukan pengecekan hasil pada output\_report.txt dan python\_results.txt:

Name	Date modified	Type	Size
Today			
output_report	Friday, 05 December 2025...	Text Document	3 KB
tb_typing_speed_file_behav	Friday, 05 December 2025...	Vivado Waveform Da...	73,366 KB
python_results	Friday, 05 December 2025...	Text Document	1 KB

# TESTING

Hasil testing pada output\_report.txt dan python\_results.txt:

output\_report.txt

```
[target: 'k' | User: 'k' -> [OK]
Target: 'o' | User: 'o' -> [OK]
Target: 'm' | User: 'm' -> [OK]
Target: 'p' | User: 'p' -> [OK]
Target: 'u' | User: 'u' -> [OK]
Target: 't' | User: 't' -> [OK]
Target: 'e' | User: 'e' -> [OK]
Target: 'r' | User: 'r' -> [OK]
Target: 'n' | User: 'n' -> [OK]
Target: 'g' | User: 'g' -> [OK]
Target: 'l' | User: 'e' -> [OK]
Target: 'i' | User: 'n' -> [OK]
Target: 'e' | User: 'e' -> [OK]
Target: 'r' | User: 'r' -> [OK]
Target: 'a' | User: 'a' -> [OK]
Target: 's' | User: 's' -> [OK]
Target: 'l' | User: 'l' -> [OK]
Target: 'd' | User: 'd' -> [OK]
Target: 't' | User: 't' -> [OK]
Target: 'e' | User: 'e' -> [OK]
Target: 'r' | User: 'r' -> [OK]
Target: 'b' | User: 'b' -> [OK]
Target: 'a' | User: 'a' -> [OK]
Target: 'r' | User: 'r' -> [OK]
Target: 'u' | User: 'u' -> [OK]
Target: 'e' | User: 'e' -> [OK]
Target: 'n' | User: 'n' -> [OK]
Target: 'g' | User: 'g' -> [OK]
Target: 'g' | User: 'g' -> [OK]
Target: 'u' | User: 'u' -> [OK]
Target: 'n' | User: 'n' -> [OK]
Target: 'a' | User: 'k' -> [TYPO]
Target: 'e' | User: 'a' -> [TYPO]
Target: 'r' | User: 'r' -> [OK]
Target: 'n' | User: 'n' -> [OK]
Target: 't' | User: 'n' -> [OK]
Target: 't' | User: 't' -> [OK]
Target: 'e' | User: 'e' -> [OK]
Target: 'k' | User: 'r' -> [OK]
Target: 'n' | User: 'k' -> [OK]
Target: 'o' | User: 'o' -> [OK]
Target: 'l' | User: 'l' -> [OK]
Target: 'g' | User: 'g' -> [OK]
Target: 'l' | User: 's' -> [OK]
Target: 't' | User: 't' -> [OK]
Target: 'q' | User: 'q' -> [OK]
Target: 'u' | User: 'u' -> [OK]
Target: 'a' | User: 'a' -> [OK]
Target: 'n' | User: 'n' -> [OK]
Target: 't' | User: 't' -> [OK]
Target: 'u' | User: 'u' -> [OK]
Target: 'm' | User: 'm' -> [OK]
Target: 'p' | User: 'p' -> [OK]
Target: 'u' | User: 'u' -> [OK]
Target: 'e' | User: 'e' -> [OK]
Target: 'r' | User: 'r' -> [OK]
Target: 'n' | User: 'n' -> [OK]
Target: 'g' | User: 'g' -> [OK]
Target: 'r' | User: 'r' -> [OK]
Target: 'e' | User: 'e' -> [OK]
Target: 'r' | User: 'r' -> [OK]
Target: 's' | User: 's' -> [TYPO]
Target: 'l' | User: 'o' -> [OK]
Target: 'n' | User: 'n' -> [OK]
Target: 't' | User: 'e' -> [OK]
Target: 'n' | User: 'v' -> [OK]
Target: 'o' | User: 'o' -> [OK]
Target: 'm' | User: 'm' -> [OK]
Target: 't' | User: 'l' -> [OK]
Target: 'u' | User: 'u' -> [OK]
Target: 'e' | User: 's' -> [OK]
Target: 'r' | User: 'r' -> [OK]
Target: 'e' | User: 'e' -> [OK]
Target: 'r' | User: 'r' -> [OK]
=====
VHDL CHARACTER-BY-CHARACTER LOG
=====
Log completed. Check 'python_results.txt' for final WPM and accuracy results.
```

python\_results.txt

```
=====
LAPORAN HASIL TYPING SPEED GAME
=====

Tanggal : 2025-12-05 20:53:02

LAPORAN PERFORMA:

-----
Waktu : 21.69 detik
Kecepatan : 45.9 WPM | 230 CPM
Akurasi : 93.5 %
Jumlah Typo : 6
=====
```

# ANALYSIS

Berdasarkan hasil pengujian integrasi sistem Hardware-in-the-Loop (HIL), modul VHDL Typing Speed Analyzer **berhasil melakukan sinkronisasi data secara real-time** dengan interface input Python.

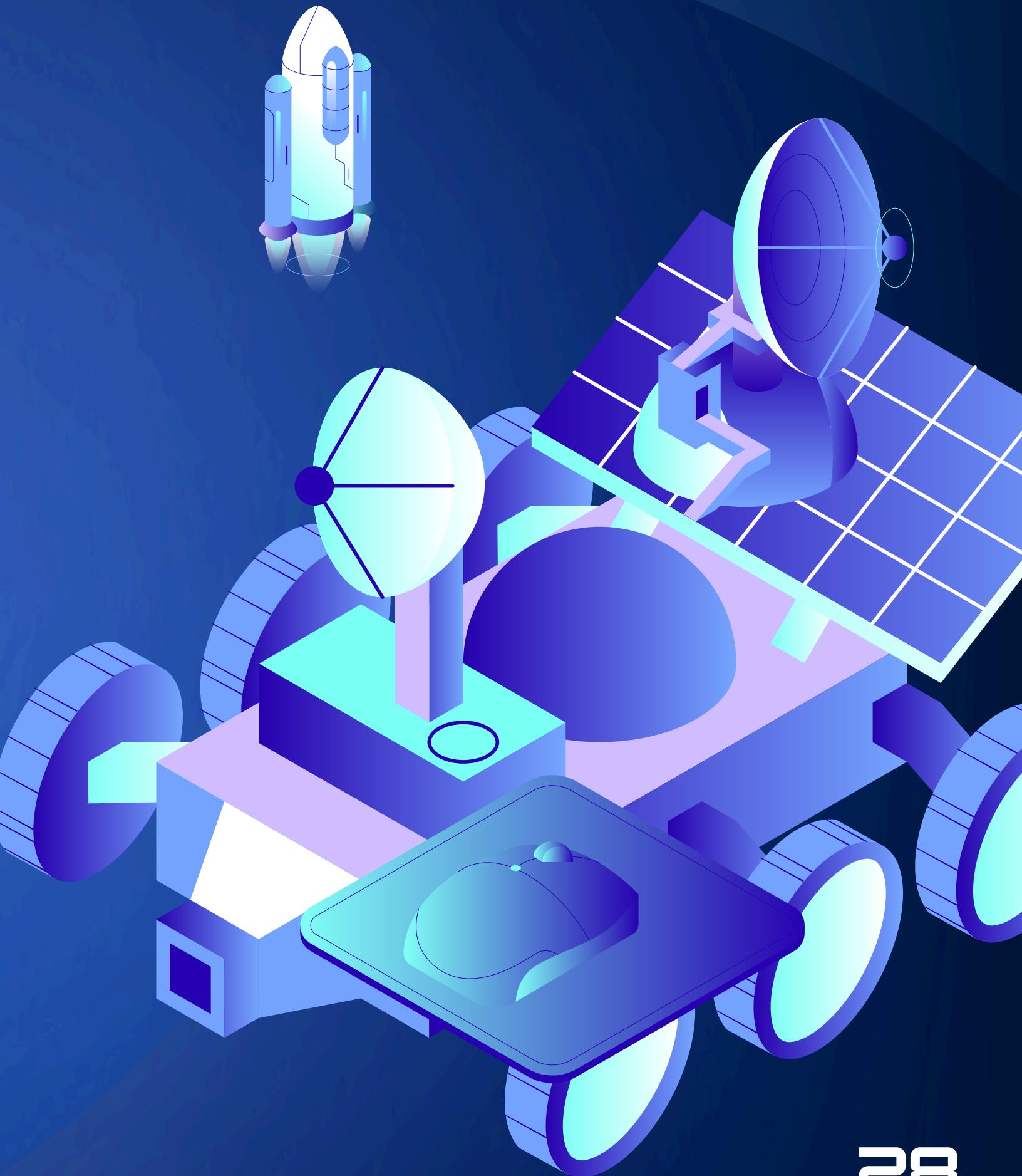
Sistem mampu mendeteksi karakter input dengan presisi tinggi, termasuk penanganan koreksi input (seperti tombol Backspace) tanpa menyebabkan desinkronisasi pada logika pengecekan karakter. Setiap karakter yang diketik pengguna diproses secara akurat oleh VHDL, menghasilkan perhitungan performa (CPM, WPM, dan Akurasi) yang konsisten dan valid.



# CHAPTER 4

## CONCLUSION

Conclusion



# CONCLUSION

- **Integrasi Hardware-Software Sukses:** Berhasil menerapkan konsep Hardware-in-the-Loop (HIL) menghubungkan logika VHDL dan interface Python secara real-time melalui pertukaran file teks.
- **Pengukuran Presisi:** Modul Keystroke Analyzer mampu menghitung metrik WPM/CPM, akurasi (%), dan jumlah typo dengan hasil yang valid dan konsisten.
- **Penanganan Input Dinamis:** Sistem tangguh menangani variabilitas input, termasuk koreksi backspace tanpa desinkronisasi data berkat logika smart backspace.
- **Keandalan Modul Pendukung:** Timer Module akurat melacak durasi dalam milidetik, dan Result Logger berhasil mendokumentasikan laporan akhir secara terstruktur.
- Secara keseluruhan, sistem **berhasil** memenuhi tujuan perancangan alat ukur kecepatan mengetik digital yang akurat dan real-time.



# CHAPTER 5

## APPENDIX

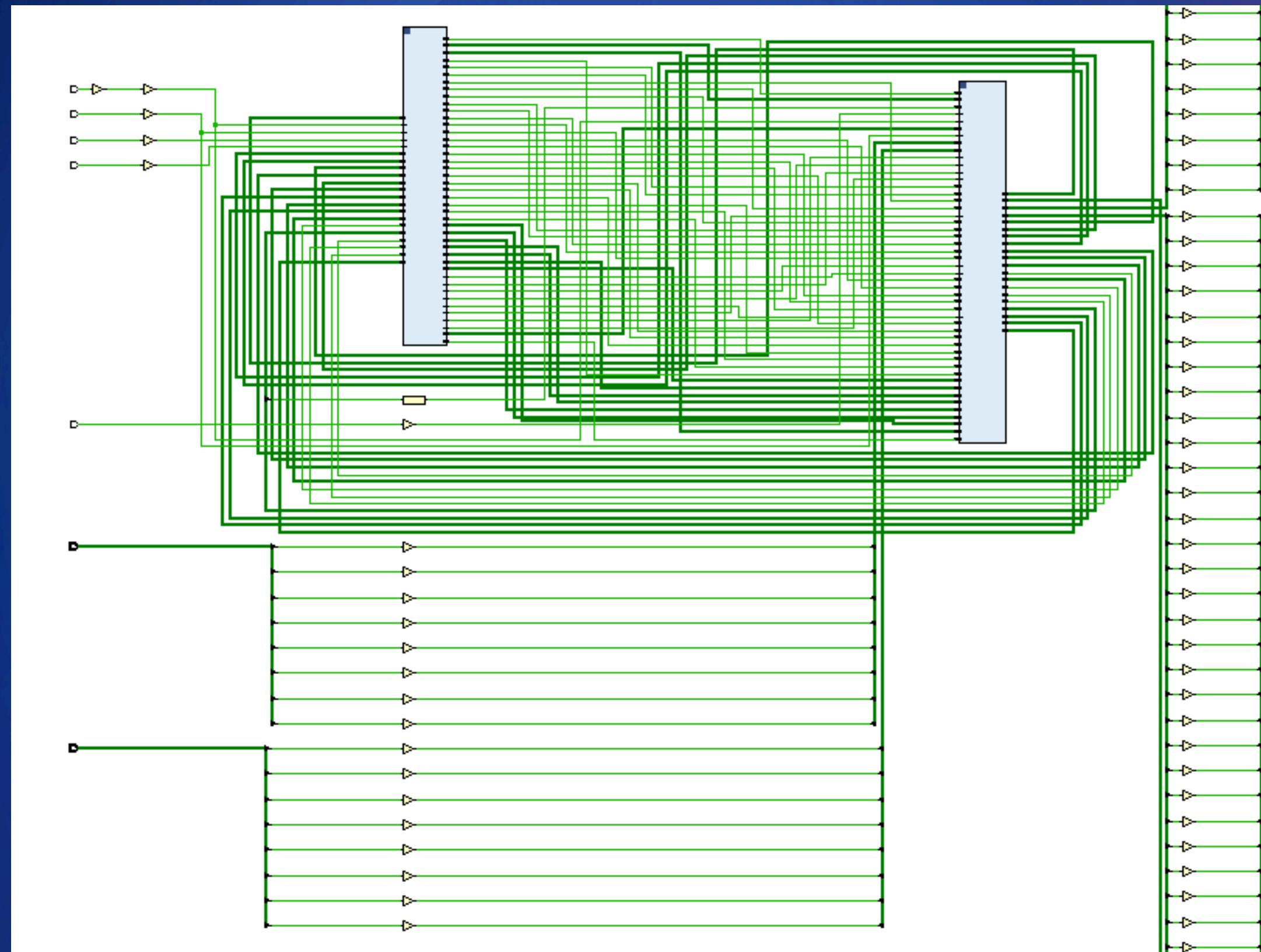
Project Schematic

Documentation

Sample Output Video



# PROJECT SCHEMATIC



# DOCUMENTATION

The screenshot shows the Vivado IDE interface with the following details:

- Title Bar:** Typing Speed Analyzer - [C:/Users/Naufal Rahman/Downloads/VIVADO PSD/Typing Speed Analyzer]
- Toolbar:** File, Edit, Flow, Tools, Reports, Window, Layout, View, Help, Quick Access.
- Flow Navigator:** Run Simulation, RTL ANALYSIS, SYNTHESIS (selected), Open Synthesized Design, Implementation, Program and Debug.
- Sources Tab:** tb\_typing\_speed\_file.vhd
- Code Editor:** tb\_typing\_speed\_file.vhd (Synthesis Complete)

```
-- 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 paksa 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);

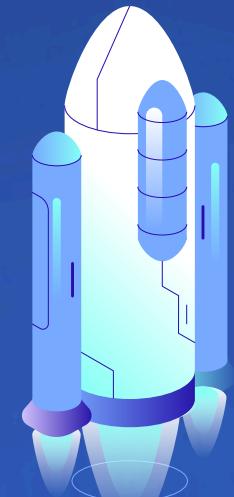
            for i in 0 to last_line_count - 1 loop
                if not endfile(input_file) then
                    readline(input_file, linebuf);
                end if;
            end loop;

            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;
```
- Video Call:** You are screen sharing, 00:11:22, Stop share. Participants: Naufal Rahman, Fernanda Raeka, Dimar Ilham Tamara, Gorga Simatupang.
- Bottom Navigation:** Tcl Console, Messages, Log, Reports, Design Runs.

# SAMPLE OUTPUT VIDEO

<https://youtu.be/pO2amU3Jnng>



# THANK YOU

