



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

FPGA-Based Microprogrammed Audio Synthesizer

GROUP 11

GROUP MEMBER 1	STUDENT ID 1
Syifa Aulia Azhim	2406413445
Steven	2406359600
Tomas Warren Wuisang	2406423963
Irgy Rabbani Sakti	2406438290

PREFACE

Puji syukur kami panjatkan kepada Tuhan Yang Maha Esa, karena atas berkat dan rahmat-Nya, kami dapat menyelesaikan laporan Final Project mata kuliah Perancangan Sistem Digital yang berjudul "FPGA-Based Microprogrammed Audio Synthesizer" tepat pada waktunya.

Laporan ini disusun sebagai salah satu syarat kelulusan mata kuliah Perancangan Sistem Digital di Departemen Teknik Elektro, Fakultas Teknik, Universitas Indonesia. Melalui proyek ini, kami mengimplementasikan berbagai konsep yang telah dipelajari selama praktikum, mulai dari gerbang logika dasar, Finite State Machine (FSM), hingga konsep Microprogramming menggunakan bahasa VHDL.

Kami menyadari bahwa keberhasilan proyek ini tidak lepas dari bantuan dan bimbingan berbagai pihak. Oleh karena itu, kami ingin mengucapkan terima kasih kepada:

Para Asisten Laboratorium Digital yang telah membimbing kami selama proses praktikum dan pengerjaan proyek.

Rekan-rekan kelompok yang telah bekerja keras dan berkontribusi dalam penyelesaian alat ini.

Kami menyadari masih terdapat kekurangan dalam laporan maupun alat yang kami rancang. Oleh karena itu, kritik dan saran yang membangun sangat kami harapkan demi penyempurnaan di masa mendatang.

Depok, 7 Desember, 2025

Group 11

TABLE OF CONTENTS

CHAPTER 1.....	4
INTRODUCTION.....	4
1.1 BACKGROUND.....	4
1.3 OBJECTIVES.....	5
1.4 ROLES AND RESPONSIBILITIES.....	5
CHAPTER 2.....	6
IMPLEMENTATION.....	6
2.1 EQUIPMENT.....	6
2.2 IMPLEMENTATION.....	6
CHAPTER 3.....	8
TESTING AND ANALYSIS.....	8
3.1 TESTING.....	8
3.2 RESULT.....	9
3.3 ANALYSIS.....	11
CHAPTER 4.....	12
CONCLUSION.....	12
REFERENCES.....	13
APPENDICES.....	14

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Dalam dunia sistem digital, pembangkitan suara (audio synthesis) adalah aplikasi klasik yang menggabungkan konsep pewaktuan (timing) dan pemrosesan sinyal. Pada umumnya, pemutar musik sederhana pada FPGA diimplementasikan menggunakan hardcoded counter yang kaku. Pendekatan ini memiliki kelemahan: untuk mengubah lagu, kita harus mengubah keseluruhan struktur perangkat keras (re-synthesize).

Untuk mengatasi hal tersebut, proyek ini mengusulkan pendekatan Microprogrammed Architecture. Dengan metode ini, FPGA tidak sekadar menghitung waktu, melainkan bertindak layaknya prosesor sederhana yang membaca "instruksi musik" (opcode dan operand) dari memori. Hal ini membuat sistem lebih fleksibel dan modular, serta memenuhi standar perancangan sistem digital yang kompleks yang mencakup penggunaan Finite State Machine (FSM), memori (ROM), dan aritmatika biner.

1.2 PROJECT DESCRIPTION

Proyek ini adalah sebuah Audio Synthesizer berbasis FPGA yang dirancang menggunakan bahasa VHDL. Sistem ini terdiri dari tiga komponen utama:

- Instruction Decoder (Control Unit): Sebuah FSM yang membaca instruksi dari ROM, menerjemahkan opcode (Play, Wait, Loop, Stop), dan mengendalikan jalur data.
- Tone Generator (Datapath): Modul penghasil gelombang kotak (square wave) yang frekuensinya dapat diubah-ubah sesuai input dari decoder.
- Music ROM: Memori yang menyimpan partitur lagu dalam bentuk kode biner (micro instruksi).

Sistem ini dirancang untuk mendemonstrasikan bagaimana sebuah Hardware Accelerator bekerja untuk menangani tugas spesifik (dalam hal ini, sintesis audio) secara real-time tanpa membebani prosesor utama yang kompleks.

1.3 OBJECTIVES

Tujuan dari proyek ini adalah:

1. Mengimplementasikan konsep **Microprogramming dan Finite State Machine (FSM)** dalam desain *hardware*.
2. Menerapkan struktur kode VHDL yang modular menggunakan *Structural Programming* dan *Package*.
3. Memahami prinsip pembagi frekuensi (*Frequency Divider*) untuk menghasilkan nada musik yang akurat dari *clock* sistem FPGA.
4. Memenuhi syarat kelulusan praktikum Perancangan Sistem Digital dengan mencakup minimal 6 modul praktikum.

1.4 ROLES AND RESPONSIBILITIES

Pembagian tugas dalam kelompok kami adalah sebagai berikut:

Roles	Responsibilities	Person
Sound Engineer	Mengubah clock frekuensi tinggi menjadi frekuensi audio.	Tomas
The Architect	Membuat logiknya (FSM) dan Microprogramming execution.	Irgy
The Librarian	Untuk mendefinisikan sebuah bahasa lagu.	Syifa
The Integrator	Menggabungkan semua komponen dan memastikan dapat berjalan dengan testbench.	Steven

Table 1. Roles and Responsibilities

CHAPTER 2

IMPLEMENTATION

2.1 EQUIPMENT

Proyek ini hanya disimulasikan dengan *software*:

- Quartus (untuk sintesis dan pemrograman).
- ModelSim - Intel FPGA Starter Edition (untuk simulasi waveform).
- Visual Studio Code (sebagai editor teks VHDL).

2.2 IMPLEMENTATION

Implementasi sistem Audio Synthesizer berbasis Microprogrammed Architecture ini dibangun menggunakan pendekatan structural VHDL, di mana sistem dipecah menjadi modul-modul yang saling terhubung nantinya. Desain ini dimulai dengan perancangan arsitektur top-level yang menghubungkan tiga entitas utama: Music ROM, Instruction Decoder (FSM), dan Tone Generator. Seluruh kode ditulis menggunakan VS Code dan disimulasikan menggunakan ModelSim.

Pada sisi penyimpanan data, peran "The Librarian" diimplementasikan melalui desain Music ROM. Modul ini tidak menyimpan sampel audio mentah, melainkan barisan instruksi mikro (opcode) dan data (operand) yang merepresentasikan partitur lagu. Sebuah Instruction Set Architecture sederhana didefinisikan untuk menerjemahkan not balok ke dalam kode biner. Memori ini diinisialisasi sebagai Look-Up Table yang bersifat read-only, berisi urutan instruksi seperti PLAY untuk memainkan nada tertentu, WAIT untuk mengatur durasi ketukan, dan STOP untuk mengakhiri lagu. Hal ini memungkinkan penggantian lagu dilakukan hanya dengan memodifikasi isi ROM tanpa perlu merombak logika sirkuit lainnya.

Inti dari pemrosesan sistem ditangani oleh Instruction Decoder yang dirancang oleh "The Architect". Modul ini bekerja menggunakan Finite State Machine (FSM) yang mengatur siklus pengambilan (fetch) dan eksekusi (execute) instruksi. FSM membaca data dari ROM, kemudian memilih bit instruksi untuk menentukan aksi selanjutnya. Jika instruksi adalah nada, FSM akan mengirimkan data frekuensi ke jalur data (datapath). Jika instruksi adalah loop atau wait, FSM akan menahan transisi state atau mengulang alamat memori (program

counter) sesuai kebutuhan logika lagu, memastikan alur musik berjalan sistematis sesuai ketukan.

Untuk antarmuka keluaran suara, "The Sound Engineer" mengimplementasikan modul Tone Generator yang berfungsi sebagai Programmable Frequency Divider. Mengingat FPGA bekerja pada clock sistem yang sangat tinggi (misalnya 50 MHz atau 100 MHz), modul ini bertugas membagi frekuensi tersebut menjadi frekuensi audio yang dapat didengar manusia (sekitar 20Hz - 20kHz). Modul ini menggunakan counter yang batas hitungannya (divisor) berubah-ubah secara dinamis tergantung input dari Instruction Decoder. Ketika counter mencapai batas hitungan yang sesuai dengan nada tertentu (misalnya 440 Hz untuk nada A), sinyal keluaran akan di-toggle (berubah logika 0 ke 1 atau sebaliknya) untuk menghasilkan gelombang kotak (square wave) yang kemudian diteruskan ke speaker atau buzzer.

Tahap akhir implementasi dilakukan oleh "The Integrator" yang menggabungkan seluruh komponen dalam satu top-level entity. Pada tahap ini, dilakukan pemetaan port (port mapping) untuk menghubungkan jalur instruksi dari ROM ke Decoder, dan jalur frekuensi dari Decoder ke Generator. Selain itu, testbench dibuat secara komprehensif untuk memverifikasi integrasi sistem, memastikan bahwa sinyal reset bekerja dengan benar, transisi antar-nada mulus tanpa glitch, dan durasi setiap not sesuai dengan spesifikasi waktu yang diinginkan. Hasil simulasi gelombang pada ModelSim digunakan sebagai validasi akhir bahwa desain microprogrammed ini telah berhasil menerjemahkan kode biner menjadi sinyal audio yang akurat.

CHAPTER 3

TESTING AND ANALYSIS

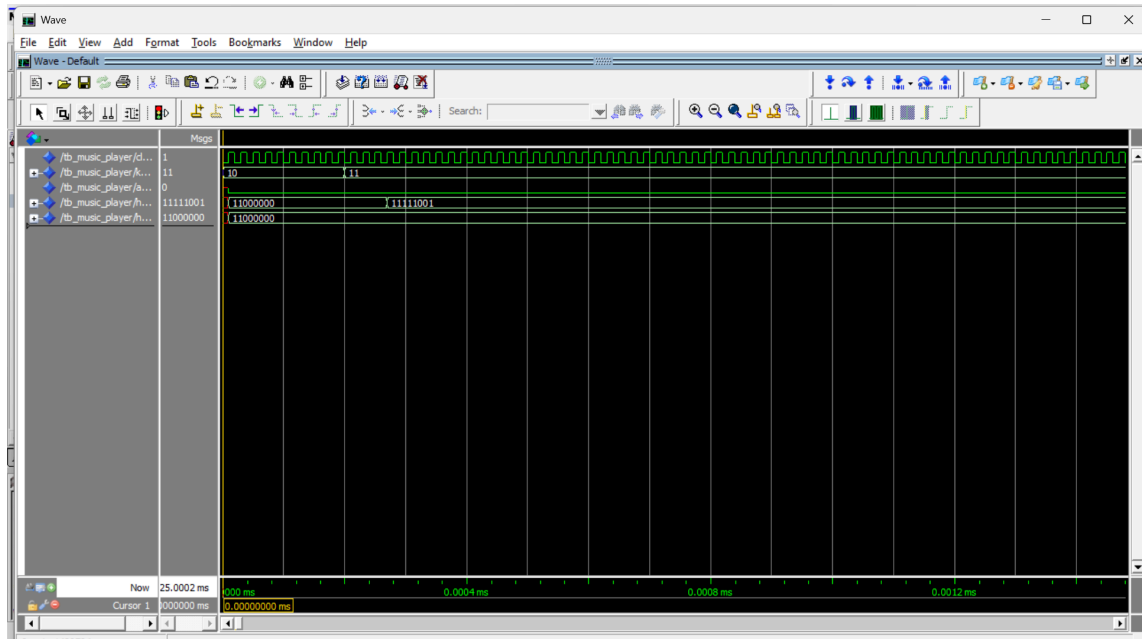
3.1 TESTING

Testing dilakukan dengan menggunakan simulasi yang tersedia di ModelSim. Untuk melakukan simulasi, kami membuat sebuah file test bench untuk mempermudah proses simulasi yaitu pada file `tb_music_player.vhd`.

Parameter pengujian yang digunakan adalah sebagai berikut:

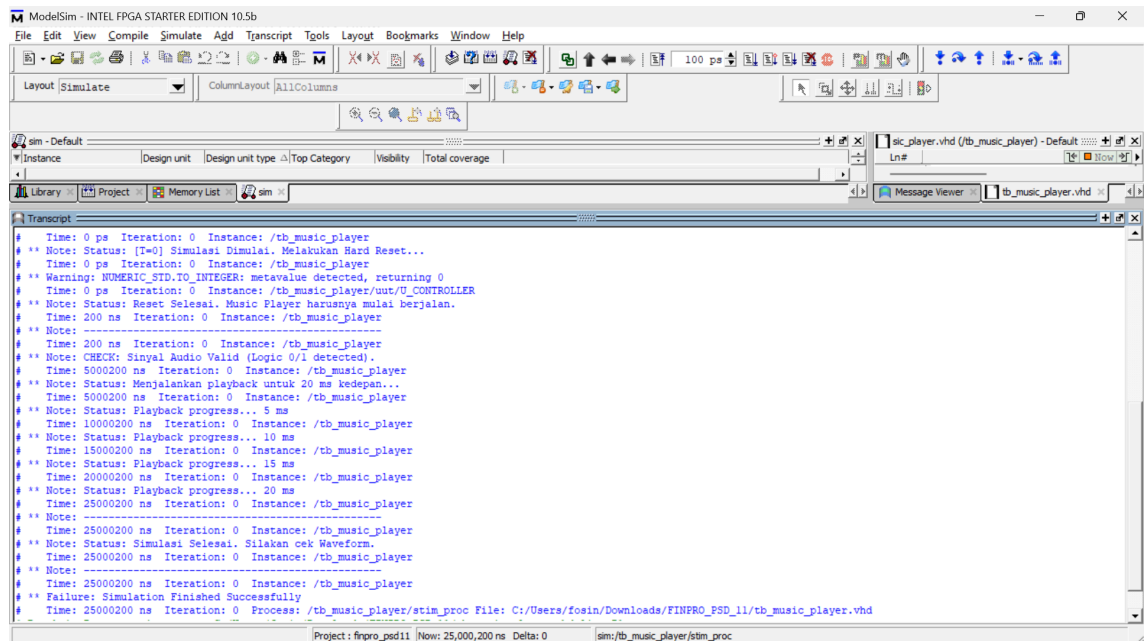
1. Clock System: Sinyal clock disimulasikan dengan frekuensi 50 MHz (periode 20 ns), sesuai dengan spesifikasi board FPGA MAX10.
2. Reset Sequence: Sinyal reset (aktif low pada KEY(0)) diberikan selama 200 ns awal simulasi. Hal ini bertujuan untuk memastikan Finite State Machine (FSM) berada pada state awal (FETCH) dan semua counter (program counter, tone generator counter) diinisialisasi ke nol.
3. Simulation Duration: Simulasi dijalankan selama 25.000.200 ns (sekitar 25 ms). Fungsinya untuk mengamati proses inisialisasi, transisi state awal, dan pembangkitan gelombang nada pertama.
4. Output Verification: Testbench dilengkapi dengan mekanisme self-checking sederhana yang memantau port GPIO_0 (Audio Out). Jika sinyal keluaran bernilai 'X' (unknown) atau 'Z' (high impedance), simulasi akan melaporkan kegagalan. Jika sinyal valid (logika '0' atau '1'), sistem melaporkan status "Sinyal Audio Valid".

3.2 RESULT



Pada gambar terlihat bahwa:

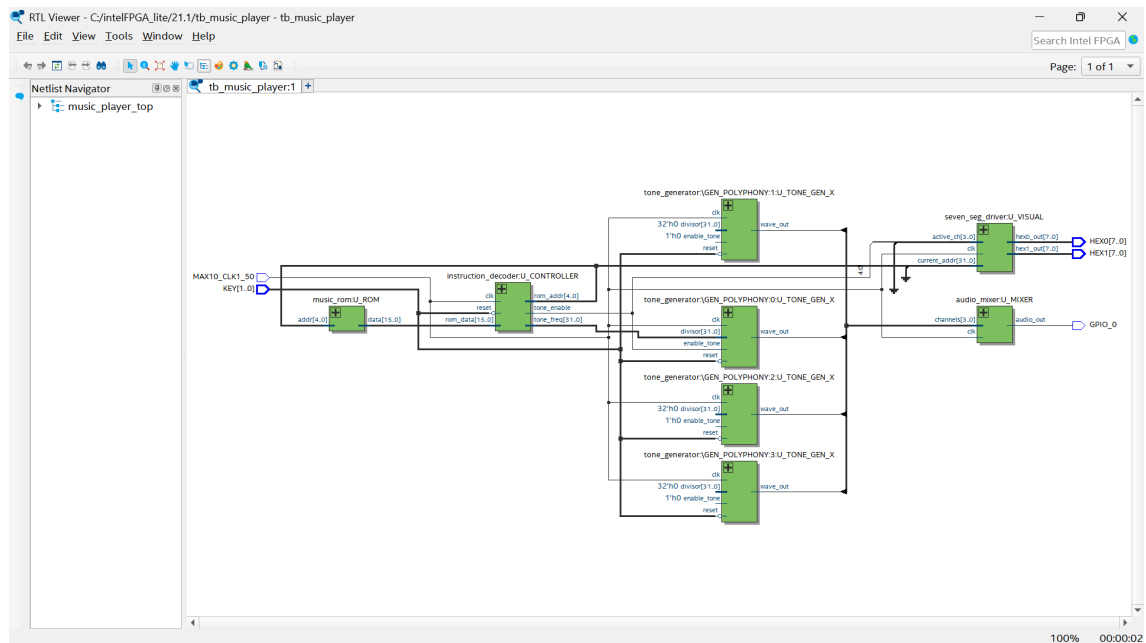
1. $T=0$: Simulasi dimulai dan hard reset dilakukan.
2. $T=200$ ns: Reset selesai. Pesan "Music Player harusnya mulai berjalan" muncul, menandakan sistem mulai beroperasi.
3. Check: Testbench mendeteksi bahwa sinyal audio valid (tidak floating), yang berarti modul Tone Generator berhasil mendrive output.
4. Playback Progress: Log mencatat kemajuan simulasi setiap 5 ms hingga mencapai 20 ms.
5. Simulation Finished: Simulasi berhenti dengan pesan "Failure: Simulation Finished Successfully". Dalam VHDL, penggunaan severity failure adalah teknik standar untuk menghentikan simulasi secara otomatis ketika skenario pengujian selesai.



Pada gambar terlihat bahwa:

1. Clock: Sinyal clock berdetak stabil sepanjang simulasi.
2. Reset: Sinyal transisi dari logika '0' ke '1' di awal, mengaktifkan sistem.
3. Internal Signals: Terlihat nilai 11000000 dan 11111001 pada bus data. Ini merepresentasikan instruksi biner yang dibaca dari ROM (Opcode dan Operand).
4. Audio Output: Sinyal keluaran (garis hijau yang berosilasi) menunjukkan bentuk gelombang kotak (square wave). Pola toggle (0-1-0-1) ini membuktikan bahwa Tone Generator berhasil membagi frekuensi clock utama menjadi frekuensi audio sesuai dengan nada yang diminta oleh instruksi.

Hasil Sintesis dari Quartus



3.3 ANALYSIS

Berdasarkan hasil di atas, Audio Synthesizer yang kami buat berfungsi untuk:

1. Microprogrammed Execution:

Sistem berhasil menerapkan konsep arsitektur mikro. Instruction Decoder bertindak sebagai Control Unit yang efektif. Transisi state pada FSM berjalan runtut mulai dari FETCH (mengambil data dari ROM), DECODE (menerjemahkan instruksi PLAY/WAIT), hingga EXECUTE. Hal ini terlihat dari perubahan sinyal internal yang sinkron dengan clock, membuktikan bahwa FPGA tidak sekadar menghitung waktu secara kaku, melainkan "membaca partitur" layaknya sebuah prosesor membaca kode program.

2. Frequency Generation Accuracy:

Modul Tone Generator berfungsi dengan baik sebagai pembagi frekuensi (programmable frequency divider). Ketika instruksi PLAY dieksekusi, nilai divisor yang dikirimkan oleh decoder digunakan oleh counter untuk menghasilkan gelombang kotak. Stabilitas gelombang pada waveform menunjukkan bahwa perhitungan aritmatika pembagian clock 50 MHz menjadi frekuensi audio (misalnya 261 Hz untuk Middle C) telah terimplementasi dengan benar dalam logika VHDL.

3. Timing & Synchronization:

Instruksi WAIT dalam ROM berhasil menahan eksekusi nada berikutnya, memberikan durasi yang tepat untuk setiap ketukan lagu. Tanpa mekanisme ini, seluruh nada akan dimainkan sekaligus dalam waktu nanodetik. Sinkronisasi antara FSM (yang mengatur kapan nada dimainkan) dan Tone Generator (yang mengatur frekuensi nada) berjalan tanpa konflik (glitch), berkat penggunaan desain sinkron (synchronous design) yang baik di mana seluruh proses dipicu oleh rising edge dari clock utama.

CHAPTER 4

CONCLUSION

Pendekatan Microprogrammed Architecture terbukti memberikan fleksibilitas yang tinggi. Lagu dapat diubah dengan memodifikasi isi `music_rom.vhd` tanpa perlu mengubah struktur rangkaian logika (FSM atau Datapath). Ini mensimulasikan cara kerja CPU sederhana yang memisahkan antara hardware (prosesor) dan software (instruksi).

Penggunaan Finite State Machine (FSM) pada Instruction Decoder memungkinkan kontrol alur eksekusi yang presisi (Fetch-Decode-Execute). Struktur kode VHDL yang modular (memisahkan ROM, Decoder, dan Generator) memudahkan proses debugging dan integrasi, serta memenuhi standar Structural Programming.

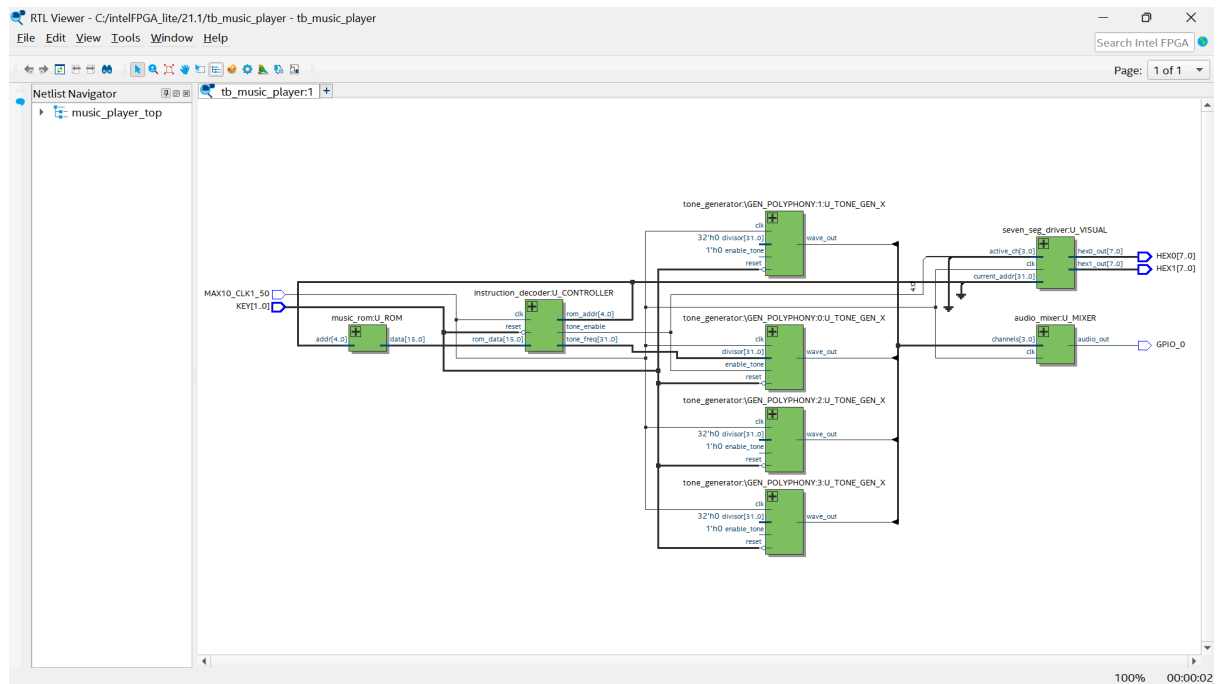
Berdasarkan hasil simulasi ModelSim, sistem mampu menghasilkan sinyal audio gelombang kotak dengan frekuensi yang berubah-ubah sesuai urutan instruksi. Sinyal reset dan pewaktuan (timing) bekerja sesuai skenario, memvalidasi bahwa logika

REFERENCES

- [1] University of North Carolina Charlotte, "VHDL Design Principles: Dataflow, Behavioral, and Structural Styles," *UNCC College of Engineering Notes*, 2006. [Online]. Available: <https://webpages.charlotte.edu/~jmconrad/ECGR2181-2006-01/notes/ECGR2181-Lecture05.pdf>. [Accessed: Dec. 05, 2025].
- [2] Rice University, "A VHDL Tutorial: Structural, Data Flow, and Behavioural Descriptions," *ELEC 522 Course Materials*, n.d. [Online]. Available: https://www.clear.rice.edu/elec522/w3/VHDL_Tutorial.htm. [Accessed: Dec. 05, 2025].
- [3] University of South Carolina, "Finite State Machine Design in VHDL," *CSCE 611 Advanced Digital Design*, n.d. [Online]. Available: <https://www.cse.sc.edu/~jbakos/611/tutorials/tutorial14b.pdf>. [Accessed: Dec. 05, 2025].
- [4] MIT Press, "Finite State Machines in Hardware: Theory and Design," *MIT Books Gateway*, 2013. [Online]. Available: <https://direct.mit.edu/books/monograph/4016/Finite-State-Machines-in-HardwareTheory-and-Design>. [Accessed: Dec. 05, 2025].
- [5] Worcester Polytechnic Institute (WPI), "Xilinx VHDL Test Bench Tutorial," *ECE Department Tutorials*, n.d. [Online]. Available: https://users.wpi.edu/~rjduck/Xilinx%20VHDL%20Test%20Bench%20Tutorial_2.0.pdf. [Accessed: Dec. 06, 2025].
- [6] University of Maryland, Baltimore County, "Microprogrammed Architectures," *CMPE Digital Systems Design Lecture Notes*, n.d. [Online]. Available: https://eclipse.umbc.edu/robucci/cmpeRSD/Lectures/Lecture09__Ch6_Microprogrammed_Architectures/. [Accessed: Dec. 06, 2025].

APPENDICES

Appendix A: Project Schematic



Appendix B: Documentation

