CAD COURSE PROJECT

Pattern-Based Digital Lock

A Finite State Machine Implementation in VHDL

— Milad Afkhami, Stu No.: 39950341054243

— Computer-Aided Design

— Prof. Ali Rajabi

— 1st Semester 1404-1405

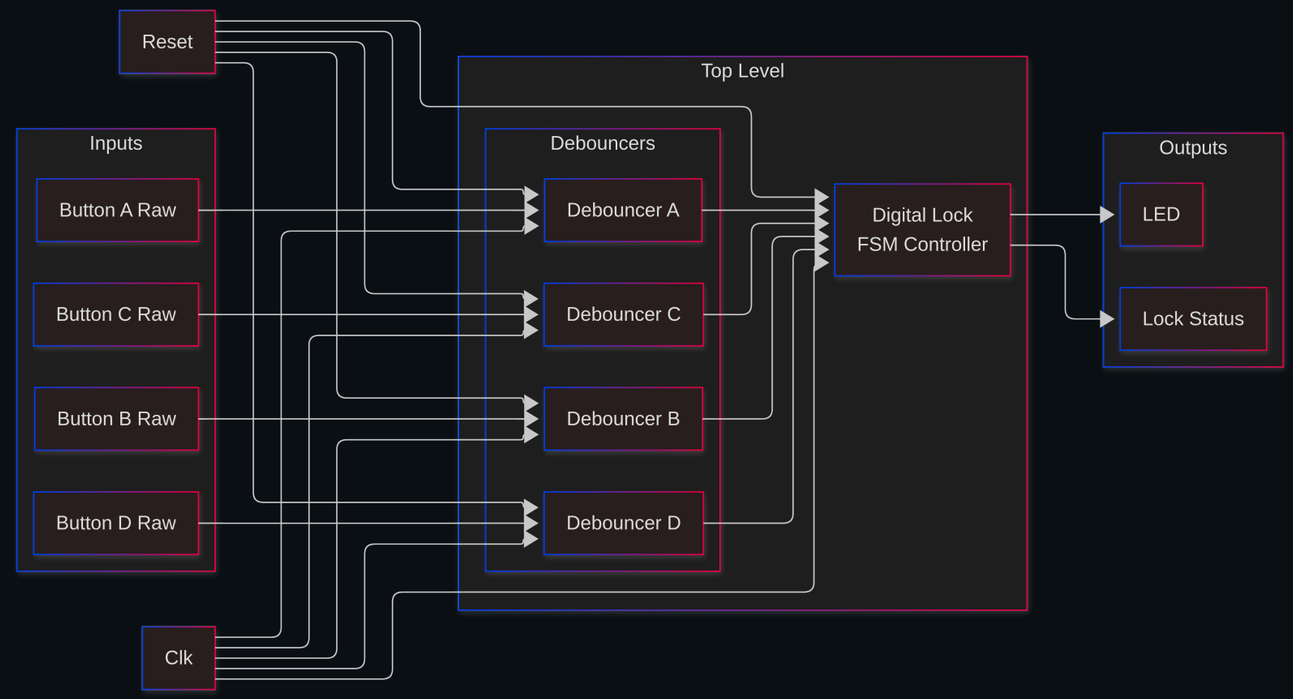
// 01

INTRODUCTION

Project Overview

A hardware-based digital lock system requiring a specific 4-button sequence to unlock, implemented using finite state machine methodology.

- Secure pattern-based authentication (A → B → C → A)
- Button debouncing for reliable physical input
- Automatic re-lock timer functionality
- Wrong input detection and recovery
- Fully synthesizable for FPGA deployment



View Full Diagram

// 02
SYSTEM DESIGN

Hardware Architecture

The system comprises three primary VHDL modules integrated in a hierarchical design pattern.

FSM

digital_lock.vhd

Core FSM implementing 5-state lock controller with auto-relock timer.

156 lines | 4 processes

DEB

button_debouncer.vhd

Counter-based debounce with 2-stage synchronizer and edge detection.

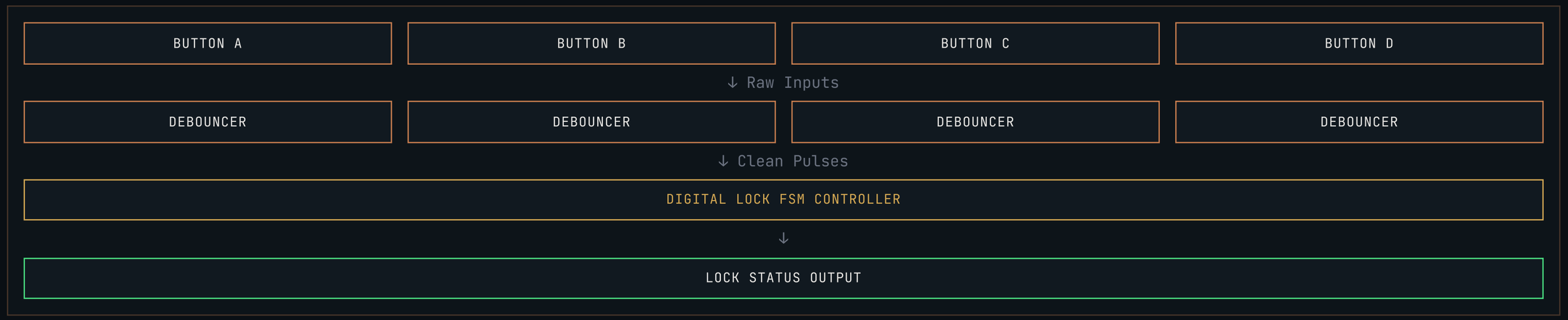
93 lines | 1 process

TOP

top_level.vhd

Integration module: 4 debouncers + 1 FSM with configurable generics.

154 lines | Synthesizable



// 03

FINITE STATE MACHINE

State Machine Design

The lock controller implements a Moore-type FSM with 5 states, following the industry-standard 3-process pattern for reliable synthesis.

A → B → C → A → UNLOCKED

STATE	CORRECT INPUT	OUTPUT
STATE_LOCKED	Button A	lock_status = '0'
STATE_FIRST	Button B	lock_status = '0'
STATE_SECOND	Button C	lock_status = '0'
STATE_THIRD	Button A	lock_status = '0'
STATE_UNLOCKED	—	lock_status = '1'

DEBOUNCER FLOW DIAGRAM

Processing flow from raw input through synchronizer, counter, and edge detector to clean pulse output

View Full Diagram

// 04

INPUT PROCESSING

Button Debouncing

Mechanical buttons exhibit contact bounce—rapid on/off transitions lasting 5-20ms. The debouncer module filters this noise and provides clean, single-cycle pulses to the FSM.

Debouncer Architecture

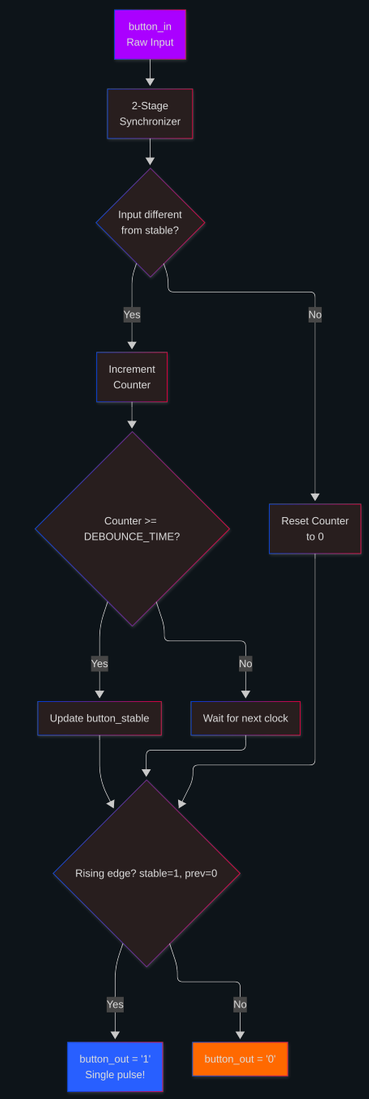
- **2-Stage Synchronizer:** Prevents metastability from asynchronous inputs
- **Counter-Based Filter:** Requires stable input for DEBOUNCE_TIME cycles
- **Edge Detector:** Generates single pulse on rising edge only

VHDL

```
-- Edge detection: rising edge only
if button_stable = '1' and button_prev = '0' then button_out ⩽
'1'; else button_out ⩽ '0'; end if;
```

DEBOUNCER FLOW DIAGRAM

Processing flow from raw input through synchronizer, counter, and edge detector to clean pulse output



View Full Diagram

INPUT PROCESSING

Debounce in Action

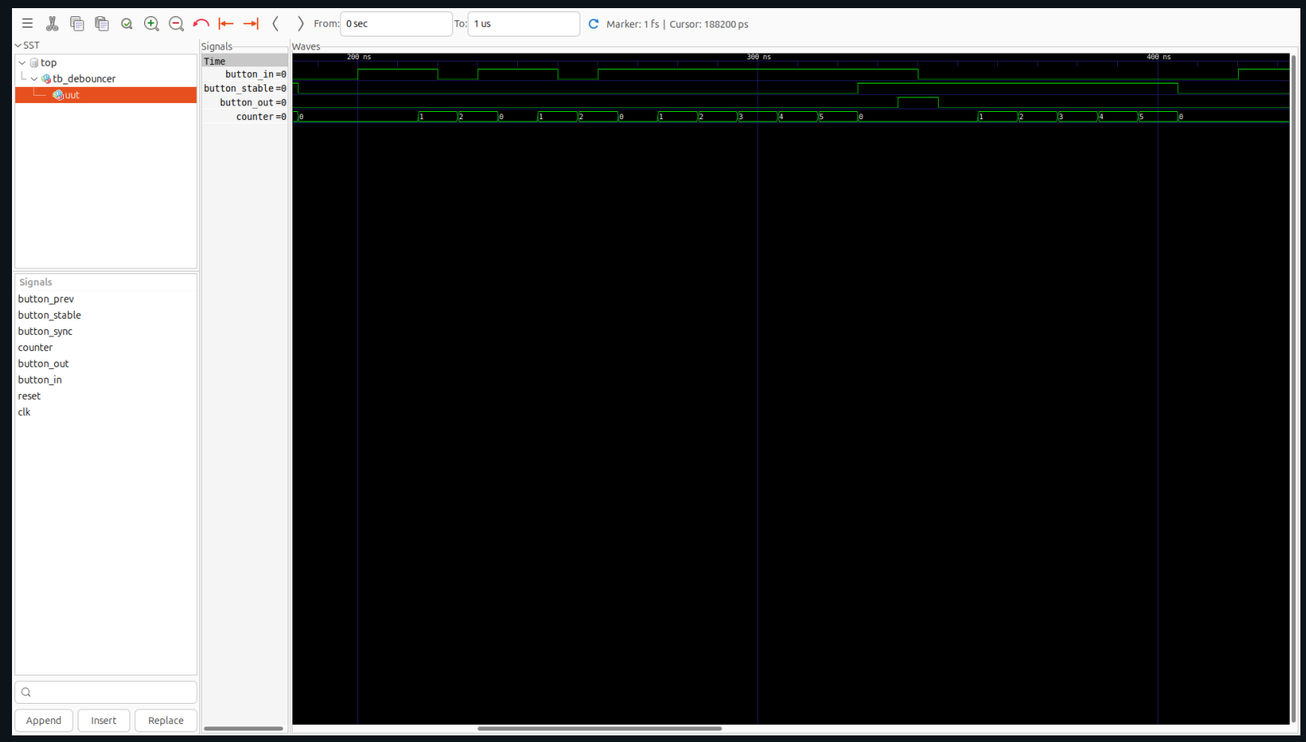
The waveform demonstrates how the debouncer transforms a noisy, bouncing button signal into a clean single-cycle pulse that the FSM can reliably process.

Signal Transformation

SIGNAL	DESCRIPTION
button_in	Raw input with bounce noise
button_stable	Filtered stable value
button_out	Single-cycle output pulse
counter	Stability counter value

BOUNCE VS. DEBOUNCED SIGNAL

Timing diagram comparing raw bouncy input with filtered stable output



VHDL IMPLEMENTATION

Code Architecture

The FSM employs the industry-standard 3-process pattern: state register, next-state logic, and output logic, enhanced with a fourth process for the auto-relock timer.

VHDL

```
-- State type definition
type state_type is ( STATE_LOCKED, STATE_FIRST, STATE_SECOND,
STATE_THIRD, STATE_UNLOCKED ); signal current_state : state_type :=
STATE_LOCKED; signal next_state : state_type; signal unlock_timer :
integer range 0 to UNLOCK_TIME;
```

VHDL

```
-- Process 1: State Register (Sequential)
state_register: process(clk, reset) begin if reset = '1' then
current_state ⩽ STATE_LOCKED; elsif rising_edge(clk) then
current_state ⩽ next_state; end if; end process;
```

VHDL

```
-- Process 2: Next State Logic (Combinational)
next_state_logic: process(current_state, button_A, button_B,
button_C, button_D) begin next_state ⩽ current_state;
-- Default
case current_state is when STATE_LOCKED ⟹ if button_A = '1' then
next_state ⩽ STATE_FIRST; end if;
when STATE_FIRST ⟹ if button_B = '1' then next_state ⩽
STATE_SECOND; elsif button_A = '1' or button_C = '1' or button_D =
'1' then next_state ⩽ STATE_LOCKED; end if;
-- ... additional states
end case;
end process;
```

DESIGN VERIFICATION

Test Methodology

Comprehensive verification was conducted using GHDL simulation with 5 testbenches covering unit testing, integration testing, coverage analysis, and edge case validation.

5	61	100%	0
TESTBENCHES	ASSERTIONS	COVERAGE	FAILURES

TESTBENCH	PURPOSE	TEST CASES	STATUS
tb_digital_lock	FSM unit verification	6 tests	● All Passed
tb_top_level	Full system integration	11 tests	● All Passed
tb_fsm_coverage	State & transition coverage	18 assertions	● All Passed
tb_edge_cases	Boundary conditions	22 assertions	● All Passed
tb_debouncer	Debouncer unit verification	4 tests	● All Passed

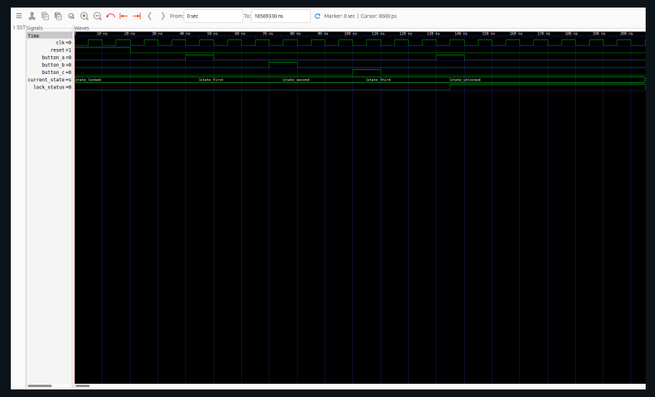
SIMULATION RESULTS

Waveform Analysis

Signal waveforms captured during simulation demonstrate correct FSM behavior across all operational scenarios.

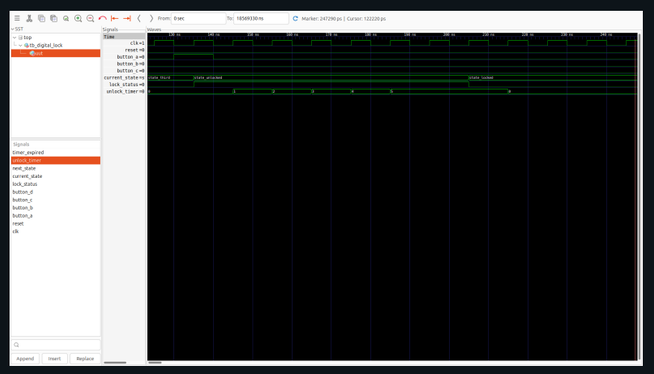
CORRECT UNLOCK SEQUENCE WAVEFORM

Shows state transitions during successful A→B→C→A sequence



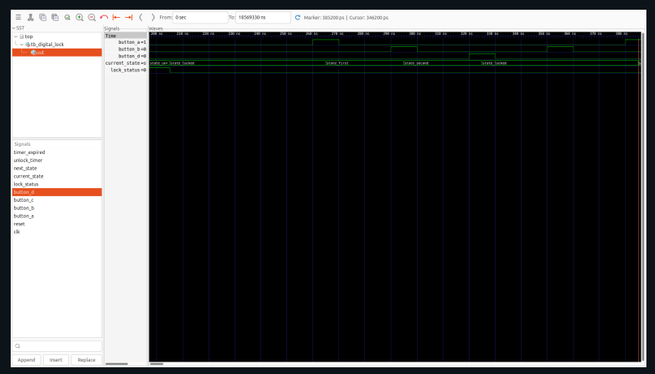
AUTO-RELOCK TIMER WAVEFORM

Demonstrates automatic re-locking after timeout



WRONG SEQUENCE DETECTION WAVEFORM

Shows FSM returning to LOCKED state on incorrect button press



SYNTHESIS RESULTS

FPGA Implementation

The design was verified for synthesizability using GHDL synthesis checking. Resource utilization is minimal, ensuring compatibility with any FPGA device.

PARAMETER	SIMULATION	HARDWARE
DEBOUNCE_TIME	10 cycles	2,000,000 cycles
UNLOCK_TIME	5 cycles	500,000,000 cycles
Clock Frequency	N/A	100 MHz
Debounce Period	~100 ns	20 ms
Unlock Duration	~50 ns	5 seconds

SYNTHESIS CHECK OUTPUT

Terminal output showing successful synthesis verification

```
milad@milad-VivoBook-ASUSLaptop-X515EP:~/Desktop/projects/Uni/cad/pattern-based-digital-lock$ ./scripts/synth.sh

==========================================
  Digital Lock - Synthesis Check
==========================================

Analyzing source files...

Running synthesis check...

Checking digital_lock...

[Code Omitted]

✓ digital_lock is synthesizable

Checking button_debouncer...

[Code Omitted]

✓ button_debouncer is synthesizable

Checking top_level...

[Code Omitted]

✓ top_level is synthesizable

==========================================
  Synthesis Check Passed!
==========================================

The design is synthesizable and ready for FPGA deployment.
```

SYNTHESIS RESULTS

Resource Utilization

Estimated FPGA Resources

- LUTs: ~50-100
- Flip-Flops: ~30-50
- Maximum Frequency: >200 MHz
- Power: Minimal (simple logic)

Design Characteristics

- Fully synchronous design
- Single clock domain
- No latches or combinational loops
- Clean reset behavior

5	4	3	100%
FSM STATES	DEBOUNCERS	VHDL MODULES	SYNTHESIZABLE

SUMMARY

Conclusions

This project successfully demonstrates the application of finite state machine methodology to implement a practical digital lock system in VHDL.

Key Achievements

- Implemented 5-state FSM using industry-standard 3-process pattern
- Developed robust button debouncing with metastability protection
- Achieved 100% state and transition coverage in verification
- Verified synthesizability for FPGA deployment
- Created comprehensive documentation and test infrastructure

Learning Outcomes

- FSM design methodology and state encoding
- Synchronous digital design principles
- Hardware verification techniques
- VHDL coding best practices

RESOURCES

Project Repository

COMPLETE TEST SUITE OUTPUT

Terminal showing all tests passing

```
milad@milad-VivoBook-ASUSLaptop-X515EP:~/Desktop/projects/Uni/cad/pattern-based-digital-lock$ ./scripts/test.sh --no-wave
==========================================
  Digital Lock - Test Suite
==========================================

Building source files...

--------------------------------------
Running: tb_digital_lock (FSM Unit Tests)
--------------------------------------
testbench/tb_digital_lock.vhd:91:9:@0ms:(report note): === Starting Digital Lock Testbench ===
testbench/tb_digital_lock.vhd:96:9:@0ms:(report note): TC1: Testing reset functionality
testbench/tb_digital_lock.vhd:105:9:@40ns:(report note): TC1 PASSED: Reset works correctly
testbench/tb_digital_lock.vhd:111:9:@40ns:(report note): TC2: Testing correct sequence A->B->C->A
testbench/tb_digital_lock.vhd:140:9:@160ns:(report note): TC2 PASSED: Correct sequence unlocks the system
testbench/tb_digital_lock.vhd:145:9:@160ns:(report note): TC5: Testing auto-relock timer
testbench/tb_digital_lock.vhd:153:9:@260ns:(report note): TC5 PASSED: Auto-relock works correctly
testbench/tb_digital_lock.vhd:158:9:@260ns:(report note): TC3: Testing wrong sequence A->B->D
testbench/tb_digital_lock.vhd:181:9:@350ns:(report note): TC3 PASSED: Wrong sequence keeps system locked
testbench/tb_digital_lock.vhd:186:9:@350ns:(report note): TC4: Testing wrong first button
testbench/tb_digital_lock.vhd:218:9:@500ns:(report note): TC4 PASSED: System recovers from wrong first button
testbench/tb_digital_lock.vhd:229:9:@540ns:(report note): TC6: Testing reset during sequence
testbench/tb_digital_lock.vhd:251:9:@640ns:(report note): TC6 PASSED: Reset during sequence works correctly
testbench/tb_digital_lock.vhd:256:9:@640ns:(report note): === All Test Cases Completed ===
testbench/tb_digital_lock.vhd:257:9:@640ns:(report note): === Digital Lock Testbench PASSED ===
```

Project Structure

pattern-based-digital-lock/

- src/ — VHDL source files
- testbench/ — Verification testbenches
- docs/ — Comprehensive documentation
- scripts/ — Automation tools
- simulation/ — Waveform outputs

403

LINES OF SOURCE

1198

LINES OF TESTS