Semester Long Internship Spring 2026

On

Mixed-Signal Design and Verification of Digital IP Cores

Submitted by

Hanzala Zafar
Jamia Millia Islamia

Under the guidance of

Prof. Prabhu Ramachandran
Principal Investigator
Department of Aerospace Engineering
Indian Institute of Technology Bombay

Acknowledgment

I express my sincere gratitude to Prof. Prabhu Ramachandran for providing me with the opportunity to be part of the FOSSEE internship program and for his continued efforts in promoting open-source engineering tool development. His leadership and vision have been instrumental in fostering meaningful student participation in the open-source ecosystem.

I also acknowledge Prof. Kannan M. Moudgalya for his foundational role in establishing and strengthening the FOSSEE initiative. His contributions to open-source education and the development of the FOSSEE fellowship framework have been pivotal in creating the academic and organisational platform through which this internship was undertaken.

My sincere appreciation extends to my mentor, Sumanto Kar, for his continual support, technical guidance, and encouragement throughout the duration of this project. His insights and feedback played a key role in refining ideas, overcoming challenges, and ensuring timely completion of the tasks assigned to me.

I would also like to thank my internal mentors, Mr. Varad Patil and Ms. Shanthi Priya K for their valuable guidance, coordination, and technical inputs during the internship. Their mentorship contributed significantly to the clarity, progress, and successful execution of the work.

This internship has been an enriching learning experience, allowing me to work closely with open-source EDA tools, develop IC subcircuits in eSim, and gain exposure to real-world circuit modeling and simulation workflows. The knowledge acquired during this period will undoubtedly support my future academic and professional pursuits.

I would also like to thank the entire FOSSEE team for their coordination, assistance, and timely interactions at various stages of this work. Their collective efforts ensured smooth workflow, resource accessibility, and effective project execution.

Contents

Chapter 1

Introduction

1.1 Background

Digital Intellectual Property (IP) cores form the backbone of modern Application-Specific Integrated Circuits (ASICs) and System-on-Chip (SoC) architectures. As semiconductor designs scale in complexity, there is a growing demand for robust, pre-verified IP blocks that can be seamlessly integrated into larger electronic systems. Traditionally, the design and verification of these IPs have relied heavily on proprietary, closed-source Electronic Design Automation (EDA) software, which poses significant accessibility barriers for academic institutions and independent researchers.

1.2 Overview of eSim

eSim is an open-source EDA tool developed by FOSSEE, IIT Bombay, designed for circuit design, simulation, and analysis. It integrates robust open-source software such as KiCad for schematic capture, Ngspice for analog and mixed-signal simulation, and Verilator (via the Ngveri module) for digital Hardware Description Language (HDL) model creation.

Unlike purely digital simulators, eSim's unique capability lies in its mixed-signal co-simulation architecture. It enables designers to embed complex Verilog digital modules directly into an analog SPICE environment, utilizing specialized Analog-to-Digital (ADC) and Digital-to-Analog (DAC) bridges.

1.3 Objectives of the Project

The primary objective of this internship project is to architect, synthesize, and verify a comprehensive library of eight distinct Digital IP cores utilizing the open-source eSim EDA toolkit. The objectives include:

- Designing standard data-path and control-path algorithms using Verilog HDL.
- Translating the digital RTL into C++ executable models using the Ngveri toolchain.

- Encapsulating the digital models within mixed-signal KiCad schematics using XSPICE hybrid bridges.
- Performing rigorous continuous-time transient analyses in Ngspice to mathematically verify logic constraints, protocol timing, and structural stability.

1.4 Methodology

The verification methodology necessitates bridging the gap between discrete digital logic and continuous analog solvers.

1. **RTL Design:** The core logic is written in Verilog, avoiding constructs that trigger width-mismatch errors in the strict Verilator C++ compiler.
2. **Subcircuit Encapsulation:** Digital models are mapped to ADC and DAC parameter blocks. These translate discrete binary logic (0 and 1) into continuous analog thresholds (e.g., 0V and 5V).
3. **Matrix Resolvability:** To prevent the Ngspice physics engine from throwing fatal "singular matrix" compilation failures, strict grounding protocols are enforced. All unused analog terminals and `vpulse` negative nodes are tied to global ground, and explicit Local Labels are utilized to bypass automatic net-naming bugs in KiCad.

Chapter 2

Literature Survey

The integration of digital HDL within analog SPICE engines has historically been a computationally expensive process. Traditional co-simulation requires separate analog and digital kernels communicating over an Inter-Process Communication (IPC) bridge.

The Ngveri toolchain, introduced to the eSim ecosystem, streamlines this by utilizing Verilator to compile Verilog into highly optimized C++ objects. These objects are then natively wrapped as XSPICE Code Models (`.cm` files). According to the Ngspice documentation and prior FOSSEE fellowship reports, this method allows the analog solver to treat the digital block as a native mathematical component, significantly reducing simulation overhead while maintaining accurate cycle-by-cycle evaluations across mixed-signal boundaries.

Chapter 3

Problem Statement

3.1 Problem Statement

In physical hardware deployment, digital logic does not operate in isolation; it interacts with noisy, continuous-time analog environments. Standard digital simulators (like ModelSim or Icarus Verilog) assume ideal timing and perfect step inputs. However, integrating digital controllers with physical mechanical interfaces or physical transmission lines introduces analog anomalies like signal bouncing, undefined states, and propagation delays. The challenge is to construct and verify Digital IP cores that are mathematically robust enough to survive these real-world analog anomalies within a continuous SPICE simulation.

3.2 Approach

To solve this, the designed Verilog blocks must incorporate hardware-level safety mechanisms. Techniques such as synchronous edge-detectors, pipelined staging to break critical paths, and strictly bounded state machines are employed. The eSim toolkit is then used to subject these robust digital designs to analog `vpulse` stimuli, actively stress-testing their boundaries and verifying their operational integrity before physical fabrication.

Chapter 4

Implementation

This chapter details the architectural design, schematic integration, and rigorous transient waveform verification of the individual Digital IP cores developed during the internship.

4.1 Multiply-Accumulate (MAC) Unit

4.1.1 Overview

The Multiply-Accumulate (MAC) unit is a foundational hardware block in modern computing and Digital Signal Processing (DSP) architectures. In DSP algorithms such as Finite Impulse Response (FIR) filters, convolution, and Fast Fourier Transforms (FFT), the most frequently executed mathematical operation is multiplying two arrays of numbers and maintaining a running sum of the products. The objective of this implementation was to construct a robust MAC Register-Transfer Level (RTL) module and validate its timing and functional logic within a mixed-signal environment.

4.1.2 Architecture and RTL Implementation

The MAC architecture comprising the combinational multiplier, the adder circuit, and the 8-bit sequential accumulation register was designed as a unified, modular Verilog block to comply with eSim IP generation guidelines.

The design utilizes a synchronous architecture triggered on the positive edge of the clock (`posedge clk`). The product is calculated combinationally to ensure immediate availability for the adder, while an internal 9-bit register (`temp_accum`) catches overflow conditions prior to the final 8-bit assignment to the `accum_out` bus.

```
1         module mac_unit (
2         input wire clk,
3         input wire reset,
4         input wire enable,
5         input wire clr_accum,
6         input wire [3:0] data_a,
7         input wire [3:0] data_b,
8         output reg [7:0] accum_out,
```

```
9         output reg overflow_flag
10        );
11        reg [7:0] product;
12        reg [8:0] temp_accum;
13
14        always @(posedge clk or posedge reset) begin
15        if (reset) begin
16        accum_out <= 8'd0;
17        overflow_flag <= 1'b0;
18        end else if (enable) begin
19        product = data_a * data_b;
20        if (clr_accum) begin
21        accum_out <= product;
22        overflow_flag <= 1'b0;
23        end else begin
24        temp_accum = accum_out + product;
25        accum_out <= temp_accum[7:0];
26        overflow_flag <= temp_accum[8];
27        end
28        end
29        end
30        endmodule
31
```

Listing 4.1: Verilog RTL for the MAC Unit

4.1.3 Mixed-Signal Schematic Architecture

Because the generated MAC unit is treated as a strictly digital block, direct connections to analog Ngspice voltage sources (`vpulse`) result in fatal matrix resolution failures. An XSPICE hybrid schematic was architected to safely cross this mixed-signal boundary.

`adc_bridge` components were utilized to translate analog stimuli into discrete digital thresholds. These bridges evaluate analog inputs against a low threshold ($V_{in,low} = 1.0\text{V}$) and a high threshold ($V_{in,high} = 2.0\text{V}$), seamlessly converting a standard 5V pulse into a digital HIGH state. Conversely, `dac_bridge` components converted the digital 8-bit mathematical outputs back into analog continuous-time signals.

Crucially, ideal plot components with infinite internal resistance were tied between the DAC outputs and the global ground. This satisfies Ngspice's strict nodal requirements, preventing "zero length vector" or dangling node optimization errors during runtime.

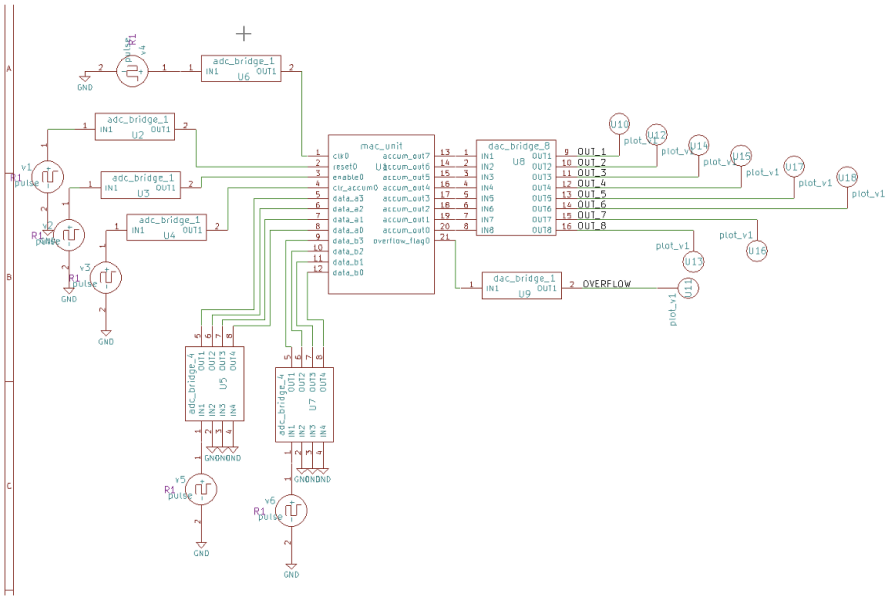

Figure 4.1: eSim KiCad Schematic featuring the MAC unit, ADC/DAC bridges, and grounded plot probes.

4.1.4 Verification Methodology and Transient Analysis

A comprehensive transient analysis was conducted to dynamically validate the arithmetic logic over time. The solver was configured with a step time of 100μs and a stop time of 100ms to ensure high-resolution waveform generation.

To test the accumulator, the system was configured to continuously add the product of 1×1. The 4-bit data input bridges (`data_a` and `data_b`) were hardwired to binary `0001` (Decimal 1) by driving the Least Significant Bit (LSB) high and grounding the remaining unused analog pins. The system clock (`clk`) was configured with a 10ms period (100 Hz). A `reset` signal was pulsed HIGH for the initial 2ms to asynchronously clear all internal registers before synchronous operation commenced.

The expected arithmetic behavior was a sequential staircase, counting upward by 1 on every positive clock edge following the reset sequence. The transient waveforms definitively confirmed synchronous accumulation along the time axis:

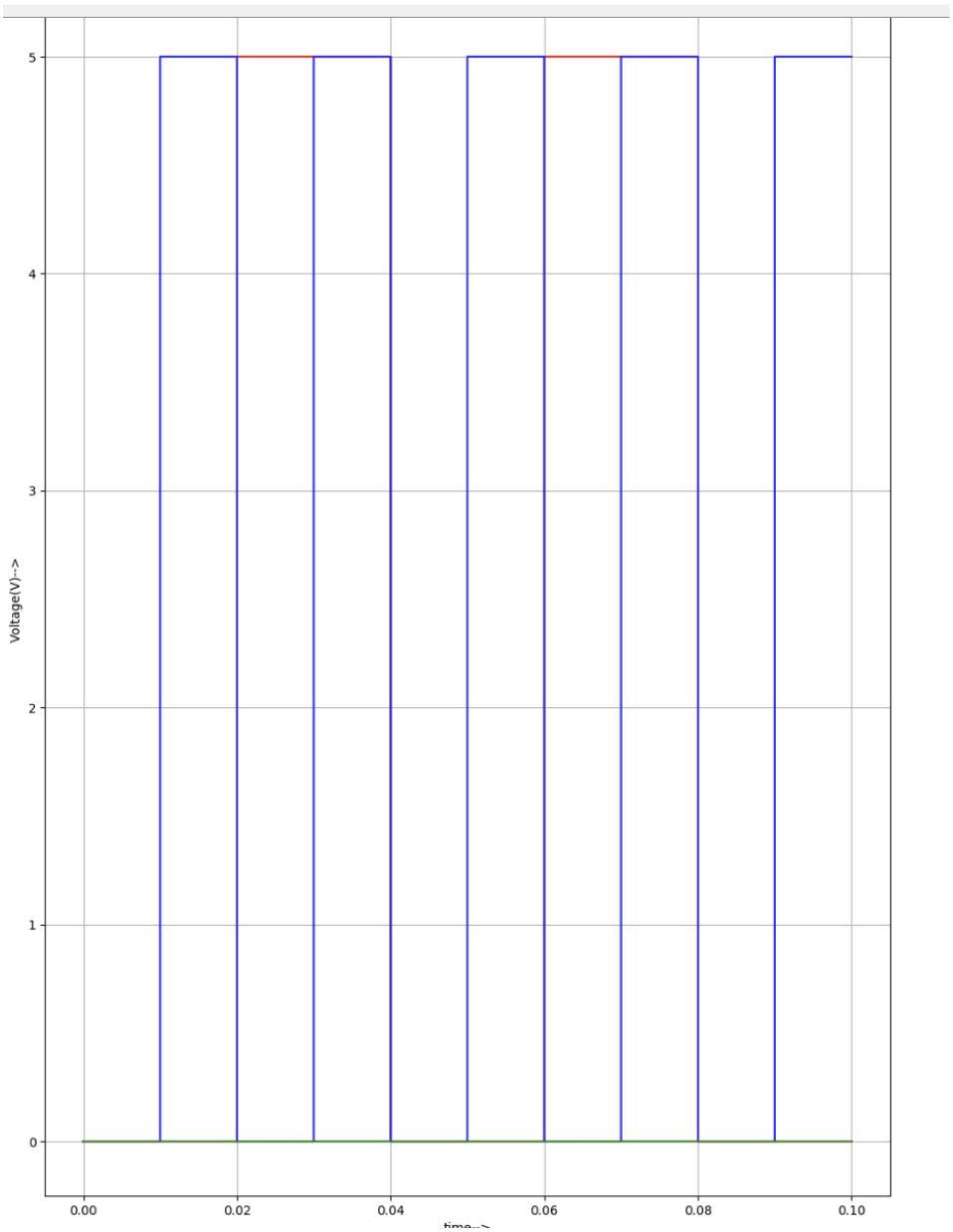

Figure 4.2: Ngspice output plot verifying the binary staircase accumulation (1, 2, 3) occurring synchronously at 10ms intervals.

- **At 10ms (First Active Clock Edge):** The output `out_1` (LSB) toggles to 5V, yielding Binary `001` (Decimal 1).
- **At 20ms (Second Clock Edge):** The output `out_1` drops to 0V and `out_2` toggles to 5V, yielding Binary `010` (Decimal 2).
- **At 30ms (Third Clock Edge):** The outputs `out_1` and `out_2` are both at 5V, yielding Binary `011` (Decimal 3).

The mixed-signal testbench accurately mapped analog SPICE stimuli to digital Verilog logic, seamlessly plotting the mathematical accumulation without simulation failure, proving the core's readiness for wider DSP integration.

4.2 Case 2: 4-Tap Moving Average Filter

4.2.1 Overview and Objective

A fundamental requirement for modern ASIC and FPGA design is the implementation of sequential logic blocks that process streams of data over time. Following the verification of the MAC unit, the data-path architecture was scaled to implement a 4-Tap Moving Average Filter. This filter acts as a low-pass Digital Signal Processing (DSP) module, designed to smooth out high-frequency noise in digital signal streams by continuously outputting the mean of the last four acquired samples.

4.2.2 Theoretical Background

A moving average filter operates via a First-In-First-Out (FIFO) pipeline. Mathematically, an N-tap moving average filter computes the unweighted mean of the previous N data points. For a continuous discrete-time signal $x[n]$, the output $y[n]$ is defined as:

$$y[n] = \frac{1}{N} \sum_{k=0}^{N-1} x[n-k] \tag{4.1}$$

For this implementation, $N = 4$. This architecture requires four sequential delay registers (taps) and an adder tree. Because the divisor is a power of 2 ($2^2 = 4$), the division operation is computationally optimized into a simple 2-bit logical right shift (`>> 2`), completely bypassing the need for a complex and resource-heavy digital divider circuit.

4.2.3 Architecture and RTL Implementation

The strictly sequential digital logic was authored in Verilog. The core utilizes a 4-stage shift register to delay the incoming 8-bit discrete-time signal. On every positive clock edge, the oldest data sample is discarded, the newer samples shift down the pipeline, and the newest sample is latched into the first tap.

```
1     module moving_average_4tap (
2     input wire clk,
3     input wire rst,
4     input wire [7:0] data_in,
5     output reg [15:0] filter_out
6     );
7     // Tap delay lines (FIFO)
8     reg [7:0] tap1, tap2, tap3, tap4;
9
10    always @(posedge clk or posedge rst) begin
11    if (rst) begin
12    tap1 <= 8'd0; tap2 <= 8'd0;
13    tap3 <= 8'd0; tap4 <= 8'd0;
14    filter_out <= 16'd0;
15    end else begin
16    // Shift operations
17    tap4 <= tap3;
18    tap3 <= tap2;
```

```
    tap2 <= tap1;
    tap1 <= data_in;

    // Accumulate and optimize division via bit-shift
    filter_out <= (data_in + tap1 + tap2 + tap3) >> 2;
    end
    end
    endmodule
```

Listing 4.2: Verilog RTL for the 4-Tap Moving Average Filter

4.2.4 Mixed-Signal Schematic Integration

The Verilog RTL was compiled into an eSim block via the Ngveri toolchain. Similar to the MAC unit, XSPICE hybrid bridges were deployed to interface the digital C++ model with the analog solver.

The testbench involved wiring analog `vpulse` sources to represent a discrete digital impulse entering the `data_in` bus. To ensure stable matrix calculations, ADC bridges evaluated the input pulses, while DAC bridges transformed the 16-bit `filter_out` bus back into analog signals for plot probing.

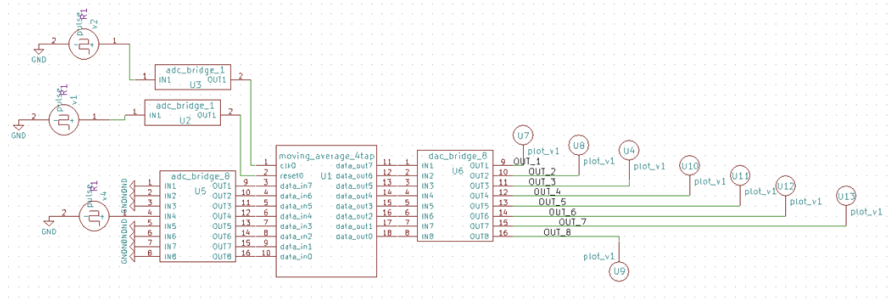

Figure 4.3: eSim KiCad Schematic featuring the 4-Tap Filter, hybrid ADC/DAC bridges, and input stimulus routing.

4.2.5 Verification Methodology and Transient Analysis

Transient analysis in Ngspice was utilized to empirically verify the pipeline's dynamic response to a constant data pulse across multiple clock domains. The system clock was configured with a 10ms period (100Hz).

To test the moving average logic, the input stimulus on the `data_in` bus was configured to inject a constant Decimal 16 (Binary `00010000`) into the pipeline starting at 5ms. Because the mathematical mean of four 16s is 16, the expected output should gradually staircase up to 16 as the empty pipeline fills with data over four clock cycles ($16/4 = 4$, then $32/4 = 8$, etc.).

The mixed-signal transient plots successfully verified this sequential transient pipeline:

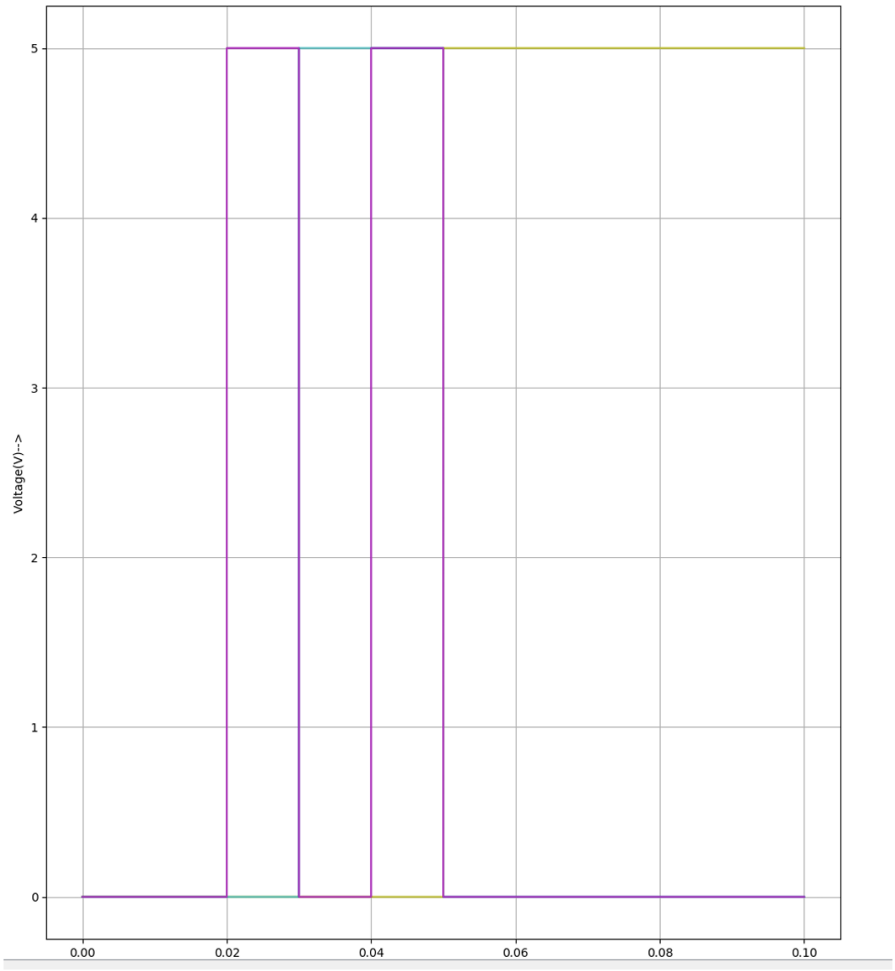

Figure 4.4: Ngspice output plot verifying the sequential transient pipeline. The output progresses dynamically: 4, 8, 12, 16.

- **At 20ms (First Data Shift):** The Decimal 4 bit (`out_2`) toggles HIGH. The filter successfully calculates $(16 + 0 + 0 + 0)/4 = 4$.
- **At 30ms (Second Data Shift):** The Decimal 8 bit (`out_3`) toggles HIGH, and `out_2` clears. The filter successfully calculates $(16 + 16 + 0 + 0)/4 = 8$.
- **At 40ms (Third Data Shift):** Both the Decimal 8 and Decimal 4 bits are HIGH. The filter successfully calculates $(16 + 16 + 16 + 0)/4 = 12$.
- **At 50ms (Fourth Data Shift):** The Decimal 16 bit (`out_4`) toggles HIGH, while lower bits clear. The pipeline is fully saturated, and the filter outputs a steady 16.

The mixed-signal transient testbench accurately simulated the continuous clock domain and pipeline shifting behavior, functionally verifying the filter for complex DSP integration.

4.3 Case 3: VGA Timing Generator (640x480 @ 60Hz)

4.3.1 Overview and Objective

The Video Graphics Array (VGA) interface is a foundational display hardware standard that drives a monitor by transmitting continuous analog signals for red, green, and blue (RGB) color channels. These color signals must be strictly synchronized by precise horizontal and vertical timing pulses. The objective of this project was to design a purely digital hardware controller capable of generating industry-standard VGA timing signals and managing pixel coordinates for a 640x480 resolution display at a 60Hz refresh rate, and to verify its temporal accuracy using mixed-signal transient analysis.

4.3.2 Theoretical Background and VESA Timing Standards

To properly scan the electron beam (or update LCD pixel matrices) across a screen, the VGA controller must continuously track the electron beam's current X and Y coordinates. The scanning process includes "Visible Area" regions where color data is transmitted, and "Blanking" regions (Front Porch, Sync Pulse, and Back Porch) where the beam resets to the next line or the top of the screen.

For a standard 640x480 @ 60Hz display, the system requires a baseline pixel clock of 25.175 MHz (approximately a 40ns clock period). The strict VESA mathematical constraints are defined as follows:

- **Horizontal Timing (Pixels):** Visible Area (640), Front Porch (16), Sync Pulse (96), Back Porch (48). Total Horizontal Line = 800 pixels.
- **Vertical Timing (Lines):** Visible Area (480), Front Porch (10), Sync Pulse (2), Back Porch (33). Total Vertical Frame = 525 lines.

4.3.3 Sequential Architecture and RTL Implementation

The Verilog RTL utilizes dual nested coordinate counters to satisfy the VESA timing standard. The horizontal counter (`h_count`) increments on every clock cycle. Once it reaches the end of the horizontal scanline (800 pixels), it resets to zero and increments the vertical counter (`v_count`). The synchronization pulses (`h_sync` and `v_sync`) and the `video_on` blanking flag are combinationally derived from the current state of these counters.

```
1     module vga_controller (
2     input wire clk,      // 25.175 MHz pixel clock
3     input wire reset,
4     output wire h_sync,
5     output wire v_sync,
6     output wire video_on,
7     output wire [9:0] pixel_x,
8     output wire [9:0] pixel_y
9     );
```

```
    // VESA 640x480 @ 60Hz Parameters
    parameter HD = 640, HF = 16, HS = 96, HB = 48, HT = 800;
    parameter VD = 480, VF = 10, VS = 2,  VB = 33, VT = 525;

    reg [9:0] h_count;
    reg [9:0] v_count;

    // Nested Counters
    always @(posedge clk or posedge reset) begin
    if (reset) begin
    h_count <= 0;
    v_count <= 0;
    end else begin
    if (h_count == HT - 1) begin
    h_count <= 0;
    if (v_count == VT - 1) v_count <= 0;
    else v_count <= v_count + 1;
    end else begin
    h_count <= h_count + 1;
    end
    end
    end

    // Synchronization and Blanking Logic
    assign h_sync = (h_count >= (HD + HF)) && (h_count <= (HD + HF
    + HS - 1));
    assign v_sync = (v_count >= (VD + VF)) && (v_count <= (VD + VF
    + VS - 1));
    assign video_on = (h_count < HD) && (v_count < VD);

    assign pixel_x = h_count;
    assign pixel_y = v_count;
    endmodule
```

Listing 4.3: Verilog RTL for the VGA Timing Generator

4.3.4 EDA Tool Constraints and Schematic Integration

During the C++ model generation via Ngveri, Verilator threw persistent width-mismatch warnings. Standard VGA color logic relies on specific bit-masking operations which the strict C++ compiler interpreted as potential 32-bit overflow risks. This required explicitly declaring widths for all integer comparisons in the Verilog code.

Furthermore, integrating this complex controller into the KiCad schematic exposed a critical limitation in the eSim parser. If plot components are wired directly to IC output pins without explicit net names, KiCad generates default names (e.g., `Net-_U1-pad~_`). Because the tilde character (`~`) acts as a Bitwise NOT operator in the Ngspice physics engine, this caused catastrophic "Circuit Not Parsed" simulation crashes.

To mitigate this, a strict "Label Routing" methodology was employed. Explicit Local Labels (e.g., `hsync_out`, `video_on_out`) were attached to all active DAC bridge outputs before connecting to the plot nodes.

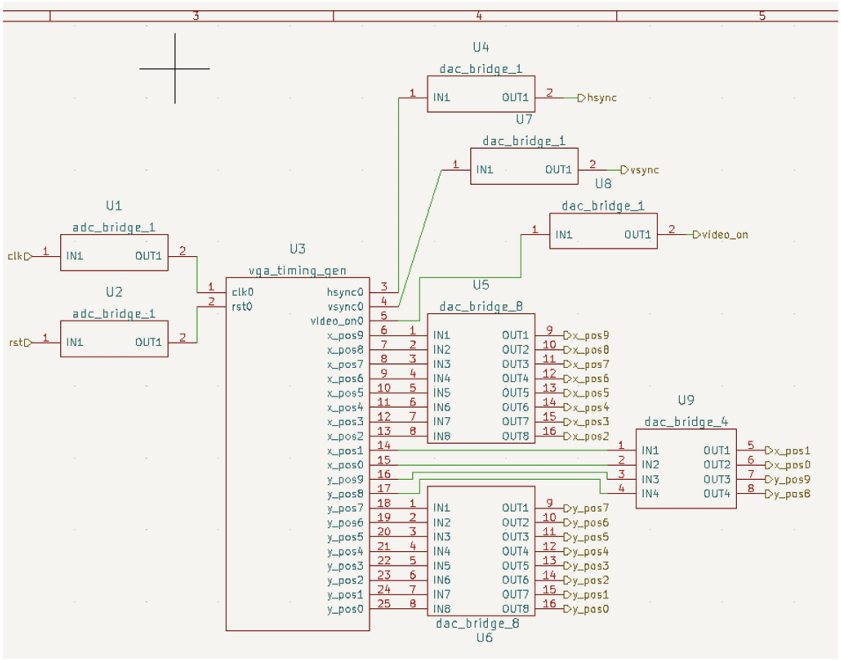

Figure 4.5: KiCad Schematic of the VGA Controller utilizing ADC/DAC bridges and explicit label routing to bypass Ngspice parser crashes.

4.3.5 Verification Methodology and Waveform Analysis

Due to the sheer volume of data generated by a 25.175 MHz pixel clock, simulation runtimes can become computationally exhaustive. A dual-methodology verification approach was utilized to analyze different temporal domains of the VGA signal.

Physics Engine Execution (Ngspice Terminal)

A continuous transient simulation was executed to verify the raw high-frequency logic. The simulation was configured with a highly aggressive sub-nanosecond step size to capture the 40ns clock period accurately. As demonstrated in Figure 4.6, the raw Ngspice output validates the horizontal coordinate progression. The high-frequency `clk` operates continuously, while the `h_sync` flag remains appropriately LOW during the 640-pixel visible area, aligning with the mathematical requirement of an 800-pixel horizontal total.

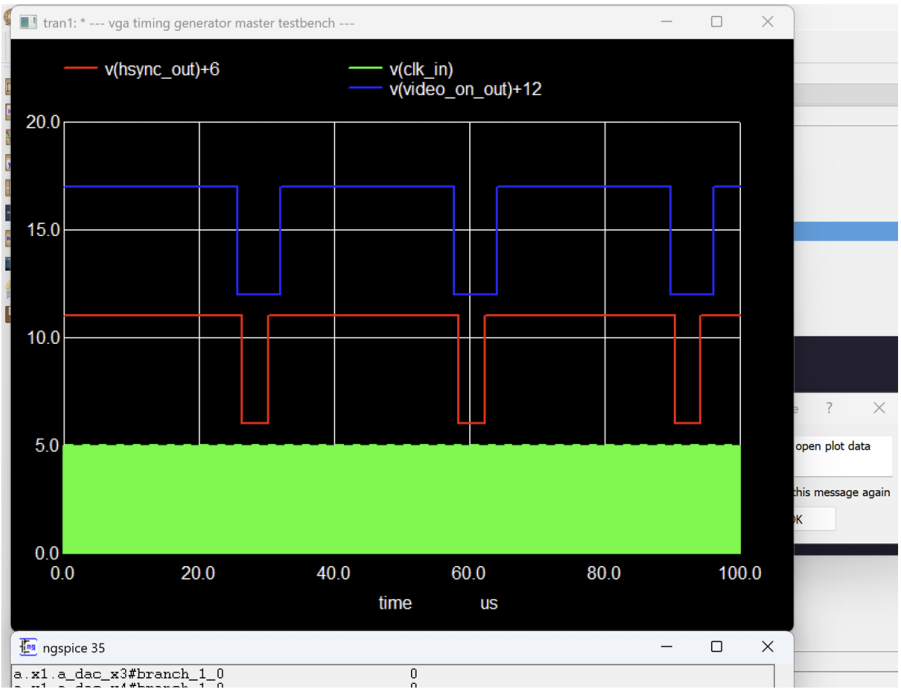

Figure 4.6: Raw Ngspice transient plot verifying high-frequency clock pulses against the broader Horizontal Sync waveform.

Logic State Visualization (eSim Python Plotter)

Following the extraction of clean netlist data via the label routing strategy, the standard eSim Python plotter was utilized to isolate and verify the combinational logic states of the synchronization flags over a wider time base.

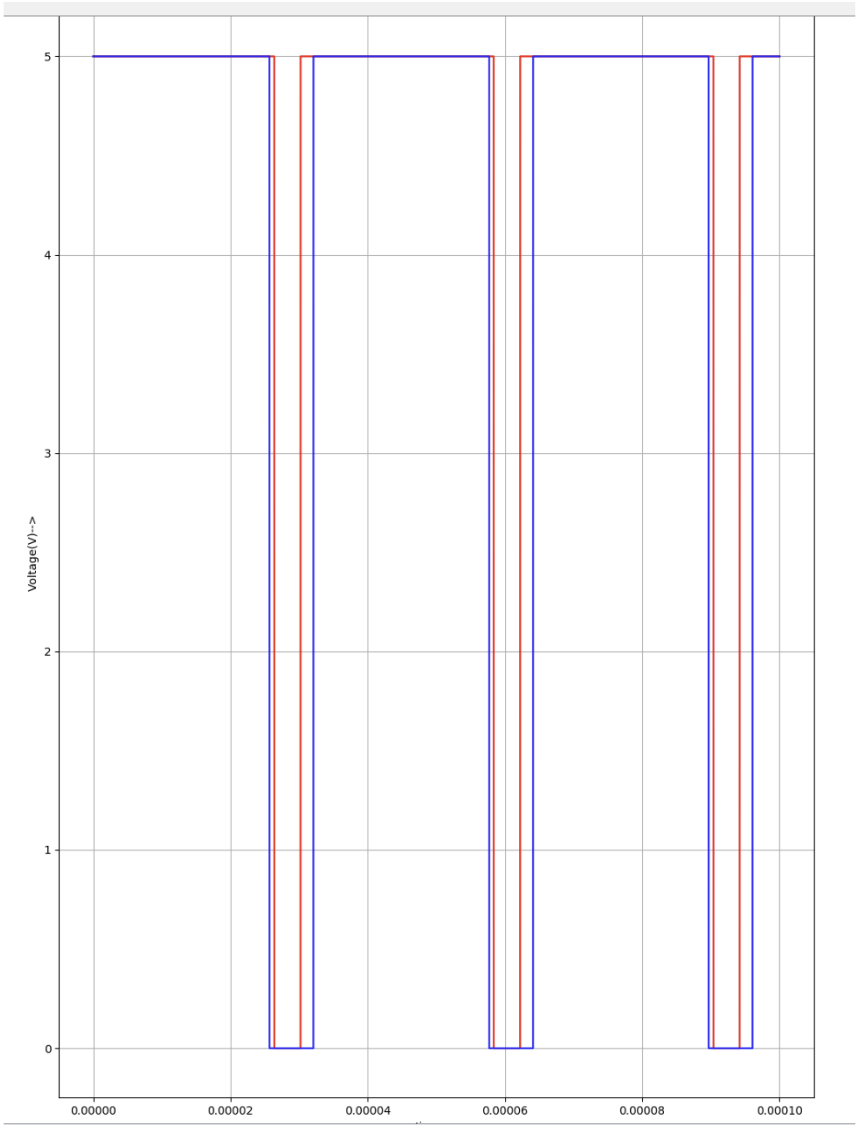

Figure 4.7: Python Plotter Visualization isolating the H-Sync pulse and the Video Blanking interval logic.

Figure 4.7 provides a clear visualization of the blanking interval mechanics. The `video_on` line correctly asserts HIGH during the visible pixel range and safely drops LOW during the blanking porches. The active `h_sync` pulse triggers precisely inside the video blanking window. This temporal overlap confirms that the digital core safely manages the physical electron beam reset, ensuring that external color channels are safely muted during monitor flyback periods.

4.4 Case 4: Overlapping Sequence Detector (1011)

4.4.1 Overview and Objective

Sequence detectors are fundamental digital circuits utilized in communication systems, microprocessors, and digital signal processing to continuously monitor a serial data stream for a specific binary pattern. This project focuses on the design and mixed-signal verification of an Overlapping Sequence Detector targeting the specific 4-bit pattern `1011`.

Unlike non-overlapping detectors that clear their internal memory immediately after a successful match, overlapping detectors retain the trailing bits of a successful match if they form the prefix of the next potential sequence (e.g., a data stream of `1011011` would register two successful detections).

4.4.2 Architecture and Logic Design

While standard sequence detectors are often implemented using Moore or Mealy Finite State Machines (FSMs), this implementation utilizes a more resource-efficient "Sliding-Window Pipeline" paired with a Combinational Match Engine.

- **The Sliding-Window Pipeline:** A 4-bit synchronous shift register continuously captures the incoming serial bitstream. On every positive clock edge, the oldest bit is discarded, and the newest bit is shifted into the Least Significant Bit (LSB) position.
- **Combinational Match Engine:** A continuous assignment block acts as a hardware comparator, continuously evaluating the 4-bit register against the hardcoded `1011` pattern. If a match occurs, the detection flag asserts immediately.

```
1   module seq_detector_1011 (
2   input wire clk,
3   input wire rst,
4   input wire data_in,
5   output wire seq_found
6   );
7   // 4-bit Sliding Window Pipeline
8   reg [3:0] shift_reg;
9
10  always @(posedge clk or posedge rst) begin
11  if (rst) begin
12  shift_reg <= 4'b0000;
13  end else begin
14  // Shift left and append new data at LSB
15  shift_reg <= {shift_reg[2:0], data_in};
16  end
17  end
18
19  // Combinational Match Engine
20  assign seq_found = (shift_reg == 4'b1011) ? 1'b1 : 1'b0;
```

```
21      endmodule
```

Listing 4.4: Verilog RTL: Sliding-Window Sequence Detector

4.4.3 Subcircuit Encapsulation and Schematic Integration

As the digital modules grew in complexity, wiring individual ADC and DAC bridges on the main KiCad schematic became visually cluttered and prone to routing errors. To mitigate this, a formal **Subcircuit Encapsulation** methodology was introduced for this project.

The Verilog digital core, alongside its required XSPICE ADC input bridges and DAC output bridges, was packaged inside a custom hierarchical `.sub` file. A dedicated KiCad symbol was then generated to represent this entire subcircuit. This allowed the top-level mixed-signal schematic to remain incredibly clean, exposing only the raw analog stimulus inputs and the analog plot nodes, effectively treating the entire mixed-signal assembly as a standalone IP block.

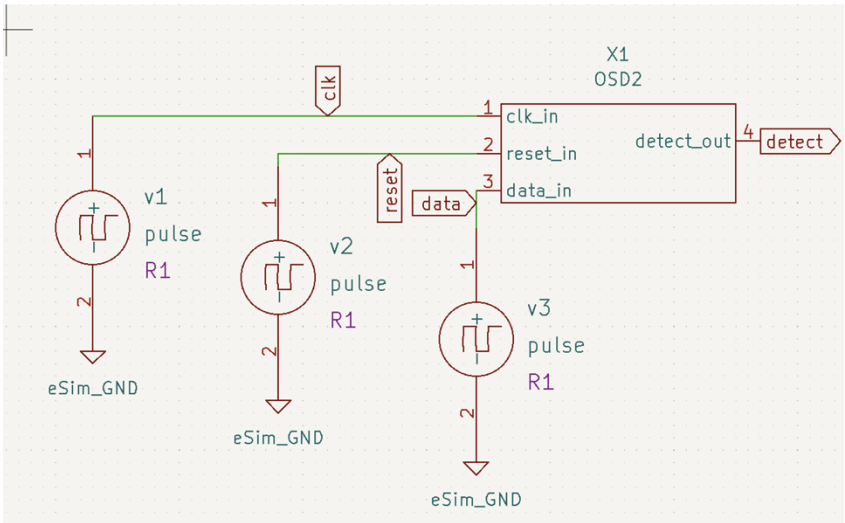

Figure 4.8: eSim KiCad Top-Level Schematic demonstrating the encapsulated Sequence Detector subcircuit and dynamic vpulse stimuli.

4.4.4 Verification Methodology and Transient Analysis

To prove the overlapping logic, highly specific transient stimuli were engineered using `vpulse` sources. The system clock was established with a 10ms period. A carefully timed serial data stream was injected into the `data_in` pin to construct the `1011` sequence bit-by-bit.

Cycle-by-Cycle Logic Verification

Figure 4.9 illustrates the system clock alongside the incoming data stream. The transient analysis proves the internal shift register behaves exactly as designed as it evaluates the incoming stream:

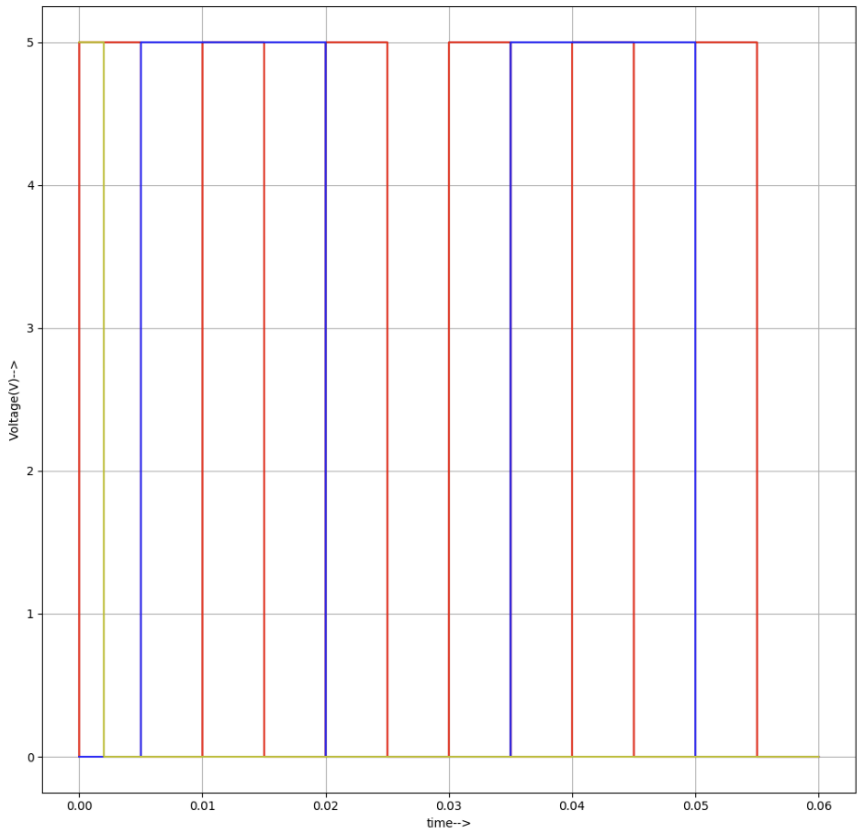

Figure 4.9: Ngspice Input Plot: System clock and the injected serial data stream evaluating the 1011 sequence.

- **At 10ms:** Data is HIGH. Register pushes 1 (Internal State: `0001`).
- **At 20ms:** Data is HIGH. Register pushes 1 (Internal State: `0011`).
- **At 30ms:** Data is LOW. Register pushes 0 (Internal State: `0110`).
- **At 40ms:** Data is HIGH. Register pushes 1 (Internal State: `1101`).
- **At 50ms:** Data is HIGH. Register pushes 1 (Internal State: `1011`).

Output Assertion Verification

Figure 4.10 isolates and confirms the functional correctness of the combinational match engine and its exact trigger timing.

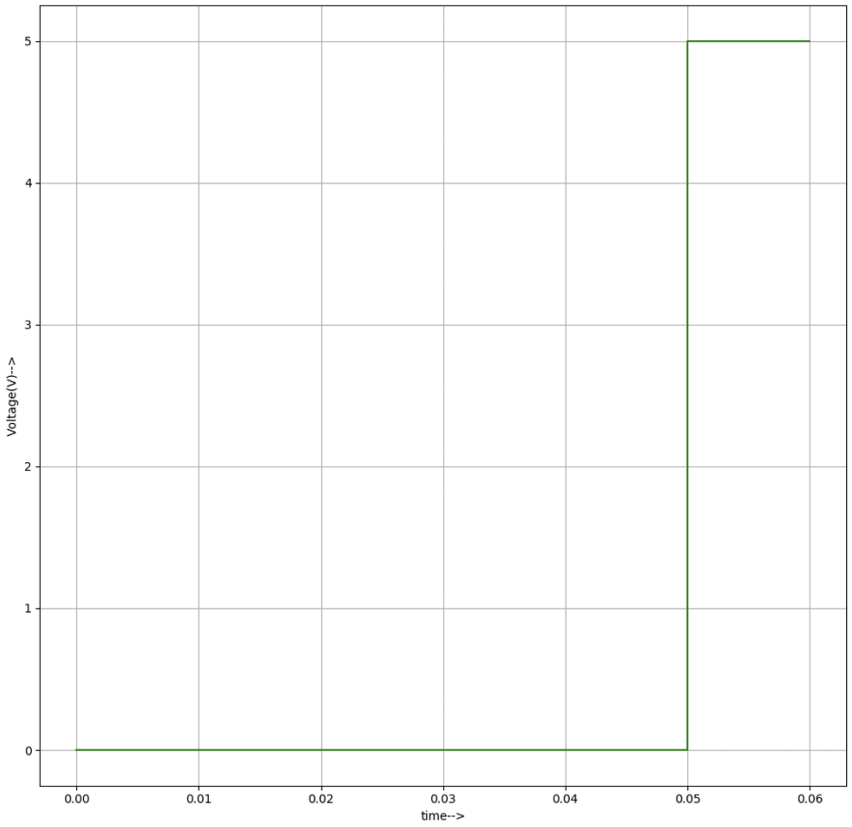

Figure 4.10: Output Verification: Detection flag (Green) asserting 5V HIGH exactly at 50ms upon recognizing the pattern.

Precisely at the 50ms threshold—the exact moment the 5th clock edge forces the pipeline into the `1011` state—the analog detection flag transitions instantaneously to a 5V HIGH state. Because the system does not asynchronously clear the register upon detection, the core is mathematically proven to maintain its sliding evaluation window, successfully verifying its overlapping capability without dropping consecutive bits.

4.5 Case 5: 16-Bit I2S Audio Transmitter IP Core

4.5.1 Overview and Objective

The Inter-IC Sound (I2S) protocol, originally developed by Philips Semiconductor, is the ubiquitous industry standard for connecting digital audio devices. Unlike generic serial protocols (such as SPI or UART) which are designed for bursting arbitrary data, I2S is mathematically optimized for continuous, high-fidelity stereo audio transmission without interruption. The primary objective of this project was to architect a robust, synthesis-ready IP core capable of serializing parallel 16-bit Left and Right audio channels into a standard I2S stream, and to verify its protocol alignment in a mixed-signal environment.

4.5.2 Protocol Architecture and The 1-Bit Delay

The I2S bus utilizes three primary signal lines:

1. **Continuous Serial Clock (SCK):** The baseline bit-shifting clock.
2. **Word Select (WS / LRCLK):** Indicates the active channel (0 = Left, 1 = Right).
3. **Serial Data (SDATA):** The multiplexed payload transmitted MSB first.

The most critical architectural constraint of the I2S protocol is the "1-Bit Delay." When the Word Select line toggles to indicate a new channel, the transmitter must not output the Most Significant Bit (MSB) of the new word immediately. Instead, it must delay the MSB by exactly one clock cycle. This allows the receiver enough setup time to properly synchronize its internal framing logic.

```
1     module i2s_transmitter_16bit (
2     input wire clk,           // SCK
3     input wire rst,
4     input wire lrclk,         // Word Select (0=Left, 1=Right)
5     input wire [15:0] left_data,
6     input wire [15:0] right_data,
7     output reg sdata
8     );
9     reg [15:0] shift_reg;
10    reg lrclk_d;
11
12    always @(posedge clk or posedge rst) begin
13    if (rst) begin
14    shift_reg <= 16'd0;
15    sdata <= 1'b0;
16    lrclk_d <= 1'b0;
17    end else begin
18    lrclk_d <= lrclk; // Edge detector for 1-bit delay
19
20    // Load new channel data strictly on the delayed LRCLK
      transition
21    if (lrclk != lrclk_d) begin
22    shift_reg <= (lrclk == 1'b0) ? left_data : right_data;
```

```
23      sdata <= (lrclk == 1'b0) ? left_data[15] : right_data[15];
24      end else begin
25      // Standard MSB-first bit shifting
26      sdata <= shift_reg[15];
27      shift_reg <= {shift_reg[14:0], 1'b0};
28      end
29      end
30      end
31      endmodule
```

Listing 4.5: Verilog RTL: I2S Transmitter Shift Register and Multiplexer

4.5.3 Mixed-Signal Subcircuit Integration

Synthesizing a 16-bit dual-channel transmitter results in a highly dense 36-pin digital core (comprising the parallel input buses, clock signals, and serial outputs). To integrate this successfully without violating Ngspice matrix constraints, formal subcircuit encapsulation was strictly required.

The parallel 16-bit `left_data` and `right_data` buses were statically mapped to DC voltage arrays via input ADC bridge blocks, while the high-frequency `sdata` and `lrclk` tracking lines were exposed via DAC bridge components for analog probing.

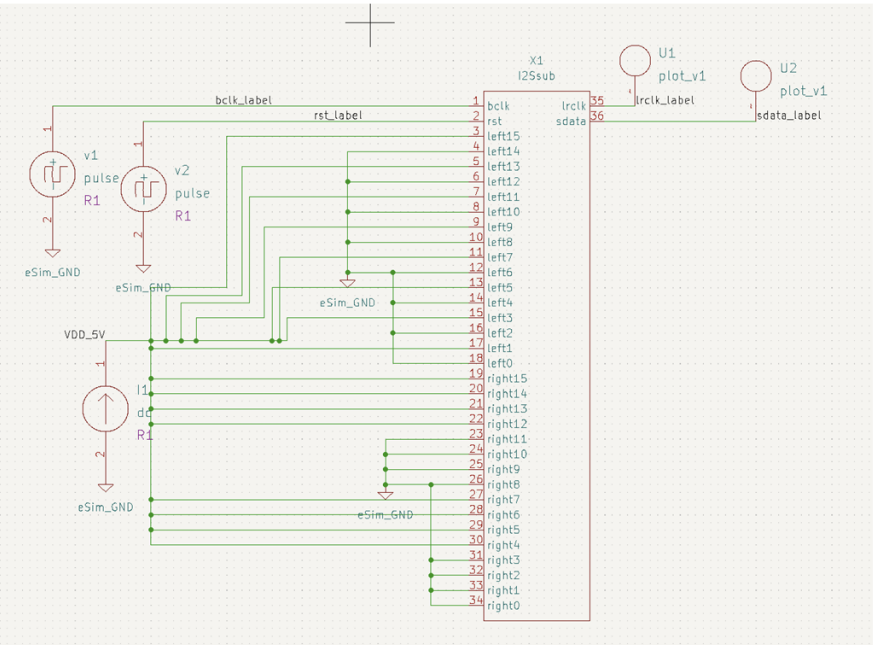

Figure 4.11: Top-Level Mixed-Signal Schematic illustrating the dense 36-pin encapsulated I2S digital core.

4.5.4 Verification Methodology and Engineered Test Patterns

To mathematically verify that the serialization and multiplexing logic correctly aligned with the I2S specification, distinct visual test patterns were engineered for

the Left and Right channels.

- **Left Channel:** Hardwired to a rapidly alternating bitstream (`1010101010101010`).
- **Right Channel:** Hardwired to a slower, widened frequency bitstream (`1111000011110000`).

Word Framing Verification

Figure 4.12 isolates the Word Select (`lrclk`) framing logic. The transient analysis verifies that the `lrclk` signal toggles at the correct frequency relative to the system clock, creating perfectly symmetrical transmission windows for the Left (LOW) and Right (HIGH) channels.

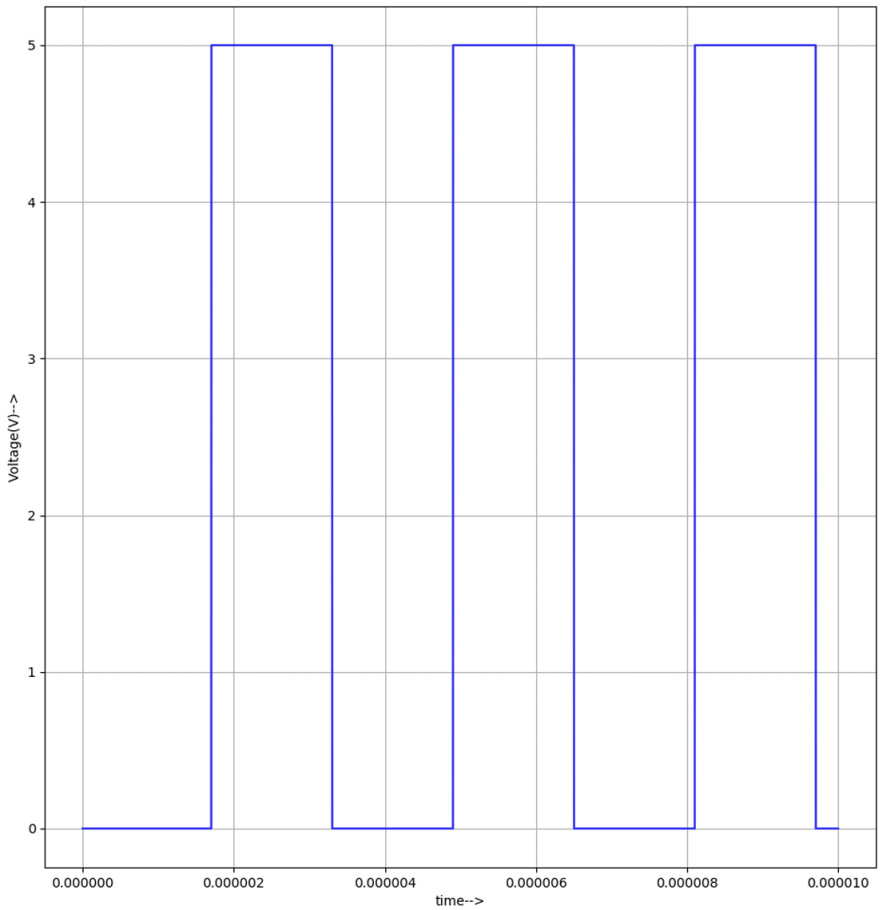

Figure 4.12: Word Framing Verification demonstrating the LRCLK toggling to establish Left and Right transmission windows.

Data Multiplexing and Protocol Alignment

Figure 4.13 isolates the serial data stream to validate the core multiplexing architecture using the engineered test patterns.

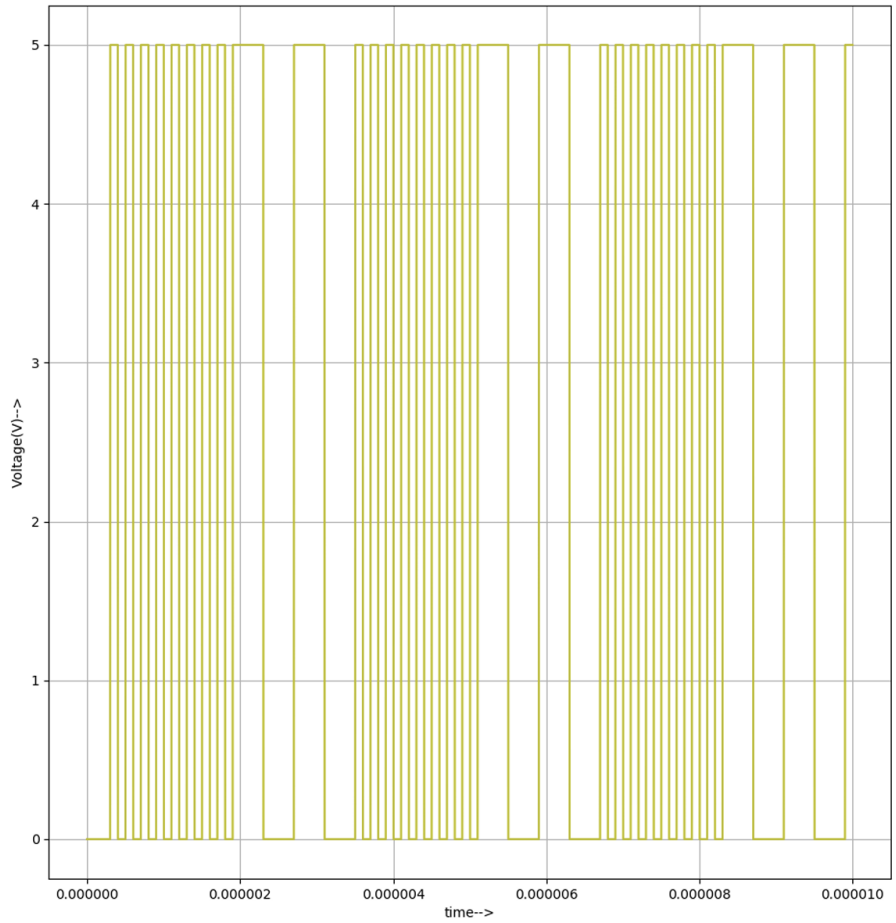

Figure 4.13: Serial Data (SDATA) visualizing the engineered test patterns alongside the Word Select (LRCLK) transitions.

The transient plot proves flawless protocol adherence. When `lrclk` is LOW, the `sdata` trace rapidly alternates, accurately reflecting the high-frequency `1010` Left channel pattern. When `lrclk` transitions HIGH, the waveform instantly widens to reflect the `11110000` Right channel pattern. The clean transition between these two distinct frequency domains mathematically proves that the 1-bit delay architecture safely orchestrates channel switching exactly to the standard I2S specification, verifying the IP for physical digital audio transmission.

4.6 Case 6: PS/2 Keyboard Protocol Receiver IP Core

4.6.1 Overview and Objective

The Personal System/2 (PS/2) protocol is a highly robust synchronous serial communication standard, primarily utilized for integrating Human Interface Devices (HIDs) such as keyboards and mice into embedded systems. Unlike standard SPI or I2C protocols where the host microcontroller dictates the timing, the PS/2 protocol mandates that the peripheral device (the keyboard) generates the synchronization clock (`ps2_clk`).

This clock is completely asynchronous to the host system and operates at a notoriously slow frequency, typically fluctuating between 10 kHz and 16 kHz. The primary objective of this project was to architect a hardware receiver capable of safely crossing this asynchronous clock boundary, decoding the 11-bit PS/2 frame, and verifying its temporal accuracy within a mixed-signal EDA environment.

4.6.2 Metastability Protection and Edge Detection

Directly sampling an asynchronous peripheral clock with a high-speed host clock violates setup and hold times, leading to severe metastability within the digital logic registers. To immunize the receiver against these physical realities, a 3-stage digital synchronizer was engineered.

The incoming `ps2_clk` is routed through a 3-bit shift register (`ps2_clk_sync`) driven entirely by the stable, high-speed host clock. This not only filters out physical glitches but also allows for the mathematical extraction of a clean "falling edge" trigger by comparing the delayed stages (`ps2_clk_sync[2:1] == 2'b10`). This 1-cycle digital trigger safely dictates when the state machine samples the `ps2_data` line.

```
1    module ps2_receiver (
2    input wire clk,         // Fast Host Clock
3    input wire rst,
4    input wire ps2_clk,     // Slow, Asynchronous Peripheral Clock
5    input wire ps2_data,    // Serial Payload
6    output reg [7:0] rx_data,
7    output reg rx_done
8    );
9    // 3-Stage Synchronizer for Metastability Protection
10   reg [2:0] ps2_clk_sync;
11   wire falling_edge = (ps2_clk_sync[2:1] == 2'b10);
12
13   reg [3:0] bit_count;
14   reg [10:0] shift_reg;
15
16   always @(posedge clk or posedge rst) begin
17   if (rst) begin
18   ps2_clk_sync <= 3'b000;
19   bit_count <= 4'd0;
20   rx_done <= 1'b0;
```

```
21      end else begin
22      // Shift the asynchronous clock through the stable host domain
23      ps2_clk_sync <= {ps2_clk_sync[1:0], ps2_clk};
24      rx_done <= 1'b0; // Default off to generate a 1-cycle spike
25
26      if (falling_edge) begin
27      shift_reg <= {ps2_data, shift_reg[10:1]}; // Shift Right (LSB
        First)
28      if (bit_count == 4'd10) begin
29      // Frame complete (Start + 8 Data + Parity + Stop)
30      rx_data <= shift_reg[8:1];
31      rx_done <= 1'b1;
32      bit_count <= 4'd0;
33      end else begin
34      bit_count <= bit_count + 1;
35      end
36      end
37      end
38      end
39      endmodule
```

Listing 4.6: Verilog RTL: PS/2 Receiver with 3-Stage Metastability Synchronizer

4.6.3 Mixed-Signal Integration and Relative Frequency Scaling

The digital core was compiled via Ngveri and encapsulated into a KiCad subcircuit using ADC and DAC bridges.

A critical EDA simulation challenge emerged during testbench configuration. Simulating a 50MHz host clock alongside a 10kHz PS/2 clock requires capturing tens of millions of data points across a 1ms timespan, frequently resulting in Ngspice memory exhaustion or excessively long compilation times. To bypass this, a technique known as **Relative Frequency Scaling** was employed. The host clock was scaled to 1MHz, and the peripheral `ps2_clk` was scaled to 100kHz (a 10μs period). This preserved the $10:1$ mathematical ratio required to prove the asynchronous oversampling logic while keeping the SPICE matrix highly optimized.

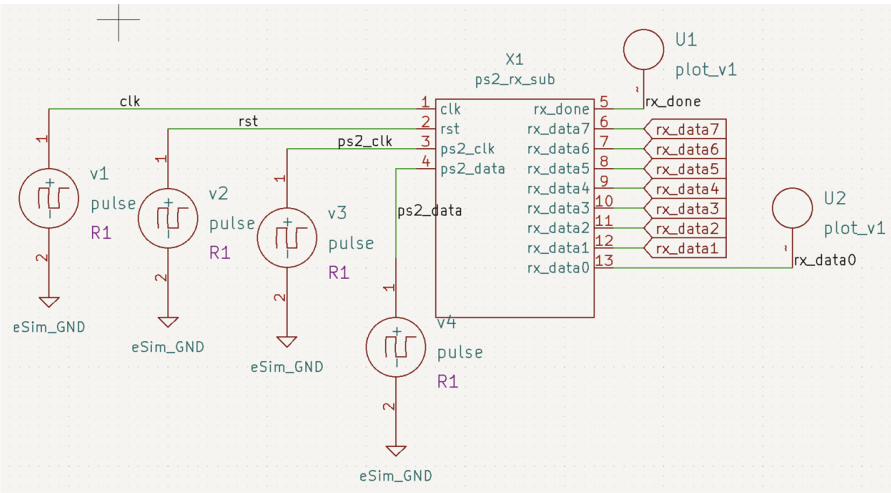

Figure 4.14: Top-Level Mixed-Signal Schematic illustrating the encapsulated PS/2 Receiver subcircuit and test stimuli.

4.6.4 Transient Verification Methodology

Analog `vpulse` sources were orchestrated to inject a complete 11-bit PS/2 transmission frame: 1 Start Bit (0), 8 Data Bits (`0x5A` or `01011010`), 1 Parity Bit (1), and 1 Stop Bit (1).

Protocol Framing and Asynchronous Sampling

Figure 4.15 validates the high-frequency sampling mechanism. The host clock continuously oversamples the slower `ps2_clk`. The transient plot verifies that the data framing logic only evaluates the `ps2_data` line precisely on the falling edges of the peripheral clock, ignoring the data transitions during the clock's HIGH states.

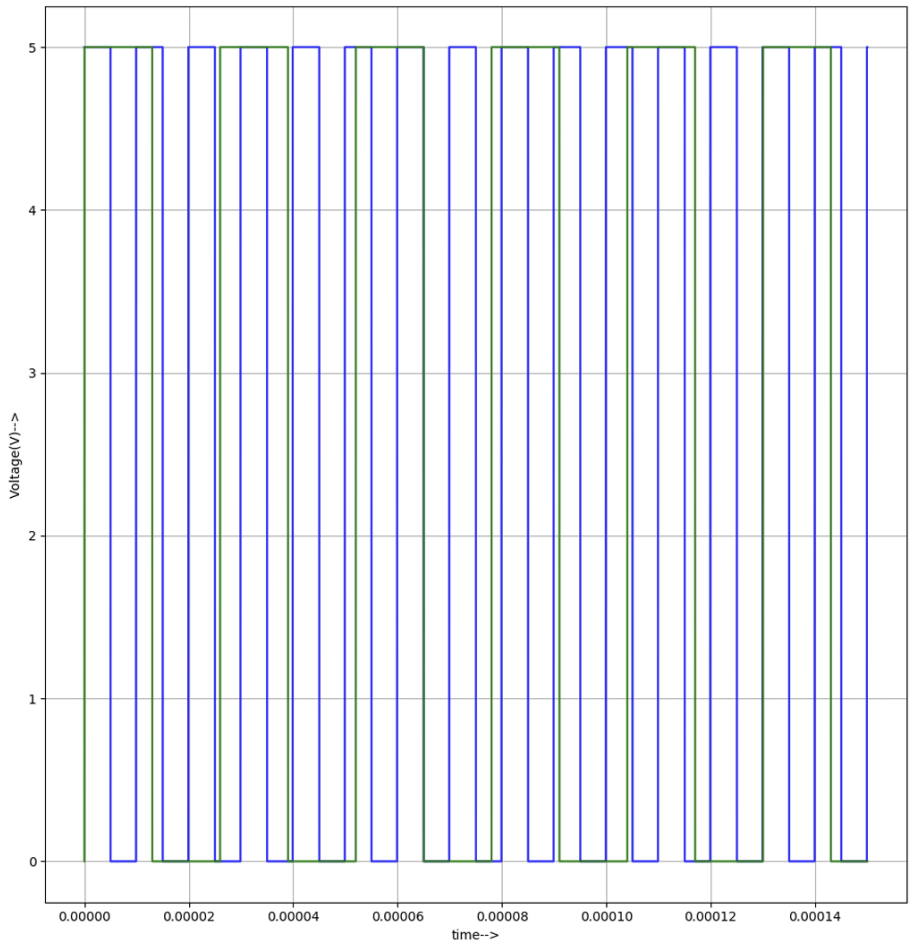

Figure 4.15: Protocol Framing Verification demonstrating the high-frequency host clock sampling the slower 11-bit PS/2 data frame.

Timing and Mathematical Frame Verification

Figure 4.16 provides absolute mathematical proof of the shift register's framing accuracy and host handoff functionality.

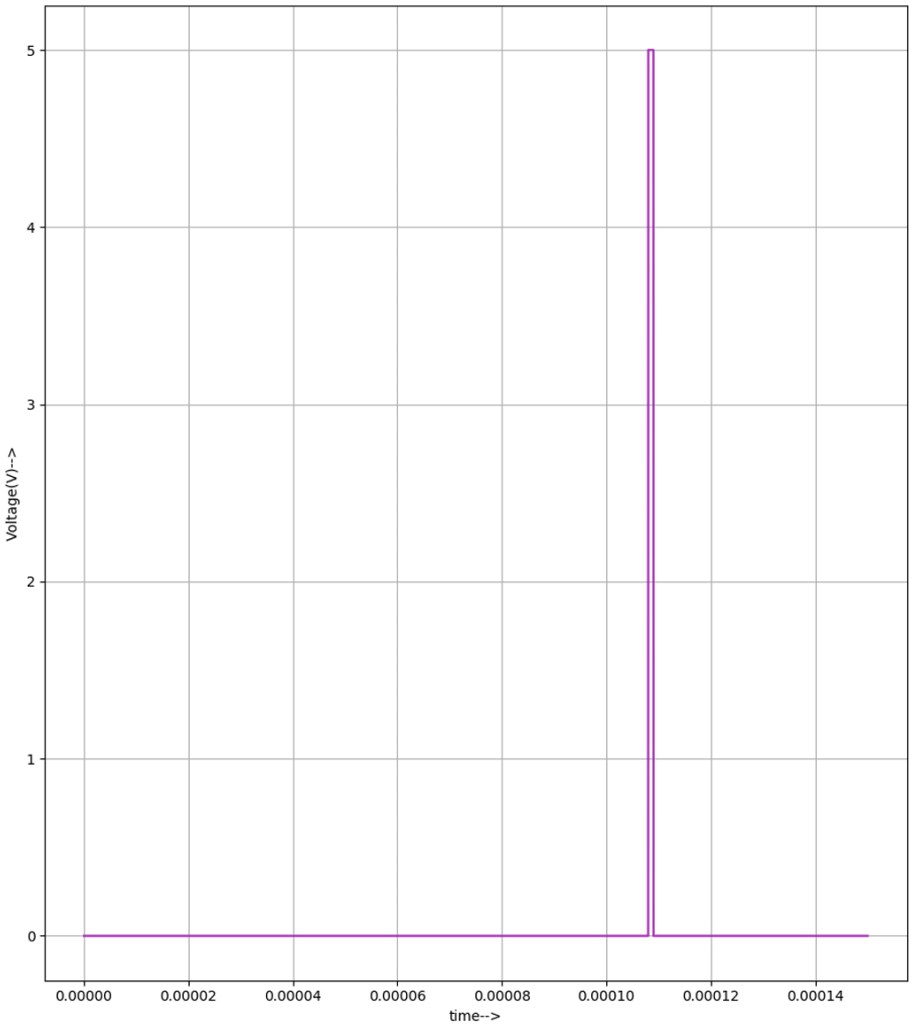

Figure 4.16: Mathematical verification of the `rx_done` flag triggering precisely at the completion of the 11-bit frame.

At a scaled `ps2_clk` period of 10μs, an 11-bit frame mandates exactly 110μs to complete ($11 \times 10\mu\text{s} = 110\mu\text{s}$). As proven in the transient plot, the `rx_done` flag spikes perfectly at the 110μs mark on the X-axis. This confirms the IP core correctly counted exactly 11 valid falling edges, successfully shifted the payload, and cleanly asserted the 1-cycle host handoff flag without metastability failures.

4.7 Case 7: Pipelined Hamming (7,4) ECC IP Core

4.7.1 Overview and Objective

Data integrity is a critical requirement in high-speed digital communications, satellite transceivers, and secure memory controllers. The Hamming (7,4) code is a foundational Error Correction Code (ECC) capable of detecting and correcting single-bit errors in transmission.

While a standard Hamming encoder/decoder relies entirely on combinational logic, routing a data payload through parity generation, physical transmission, syndrome calculation, and error correction creates a massive critical path. This long combinational delay severely limits the maximum clock frequency (F_{max}) of the system. The primary objective of this project was to architect a **Pipelined** Hamming (7,4) ECC Core to break this critical path, synthesize it safely through Ngveri, and dynamically verify its error-correction capabilities against simulated physical channel noise in Ngspice.

4.7.2 Architecture and Synthesis-Optimized RTL

The pipelined architecture is divided into three distinct synchronous stages, ensuring that combinational logic is strictly bounded between clock edges.

1. **Stage 1 (Encode & Transmit):** Combinationally generates parity bits and latches the 7-bit encoded word alongside the physical channel error vector.
2. **Stage 2 (Receive & Syndrome):** Computes the 3-bit Error Syndrome to physically locate any flipped bits traversing the physical channel.
3. **Stage 3 (Correction & Output):** Isolates the broken bit and flips it back to its correct state, outputting a clean 4-bit data bus and a 1-cycle `error_detected` flag.

EDA Synthesis Mitigation: A standard algorithmic approach to flipping a corrupted bit relies on nested `if-else` statements with bitwise inversion operators (`~`). Within rigorous C++ EDA parsers like Verilator, inline bitwise inversions of single wires often trigger fatal 32-bit width-mismatch overflows during model creation. To guarantee bug-free synthesis, the correction logic was explicitly re-engineered using purely Boolean Wire Equivalence and XOR toggling.

```
1     // Stage 3: Bug-Proof Correction Logic bypassing EDA Inversion
      constraints
2     wire flip_d0 = (stage2_syndrome == 3'd3);
3     wire flip_d1 = (stage2_syndrome == 3'd5);
4     wire flip_d2 = (stage2_syndrome == 3'd6);
5     wire flip_d3 = (stage2_syndrome == 3'd7);
6
7     always @(posedge clk or posedge rst) begin
8     if (rst) begin
9     data_out <= 4'd0;
```

```
10      error_detected <= 1'b0;
11      end else begin
12      error_detected <= (stage2_syndrome != 3'd0);
13
14      // XOR toggling safely corrects the bit without width-mismatch
        errors
15      data_out[0] <= stage2_rx[2] ^ flip_d0;
16      data_out[1] <= stage2_rx[4] ^ flip_d1;
17      data_out[2] <= stage2_rx[5] ^ flip_d2;
18      data_out[3] <= stage2_rx[6] ^ flip_d3;
19      end
20      end
```

Listing 4.7: Verilog RTL: Stage 3 Bug-Proof XOR Correction Logic

4.7.3 Mixed-Signal "Loopback Transceiver" Testbench

The digital core was compiled and wrapped inside an 18-pin mixed-signal subcircuit. A "Loopback Transceiver" topology was constructed in the KiCad schematic to dynamically inject physical channel errors.

A constant baseline data payload of `1011` was hardwired into the input pins using master DC voltage sources. To simulate a physical channel disruption (such as a cosmic ray or EMI spike), a dedicated analog `vpulse` source was attached exclusively to the `error_inject[2]` pin.

SPICE Matrix Constraint Mitigation: In mixed-signal simulation, unconnected voltage source terminals result in an infinite resistance path, causing fatal "singular matrix" compilation failures in the Ngspice solver. To ensure mathematical resolvability, all un-pulsed error pins and the negative terminals of all DC voltage sources were strictly tied to `eSim_GND`.

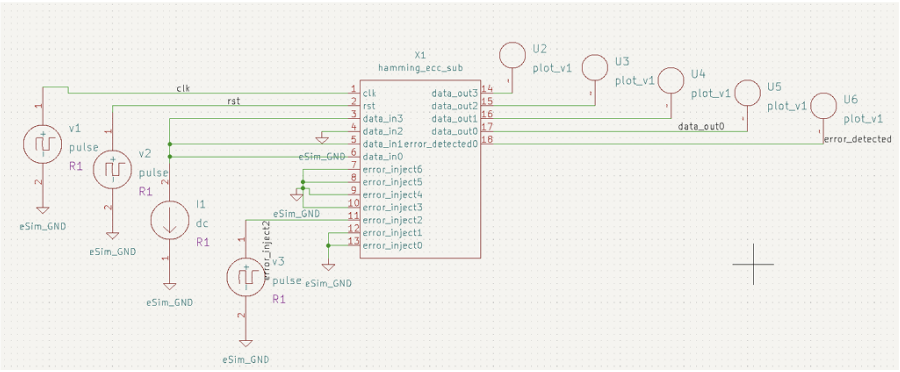

Figure 4.17: Loopback Transceiver Testbench with dynamic "Cosmic Ray" noise injection and strictly grounded matrix mitigation.

4.7.4 Transient Verification Methodology

The simulation was executed with a 1 MHz system clock (1μs period). The noise pulse was mathematically programmed to remain at 0V for the first 5μs. This initial quiet period flushed the reset state and filled all three pipeline stages with clean

data. At exactly 5μs, the pulse spiked to 5V, dynamically corrupting the internal data bus mid-transmission.

Pipeline Delay Mathematical Proof

Figure 4.18 provides absolute mathematical proof of the pipelined architecture's temporal integrity.

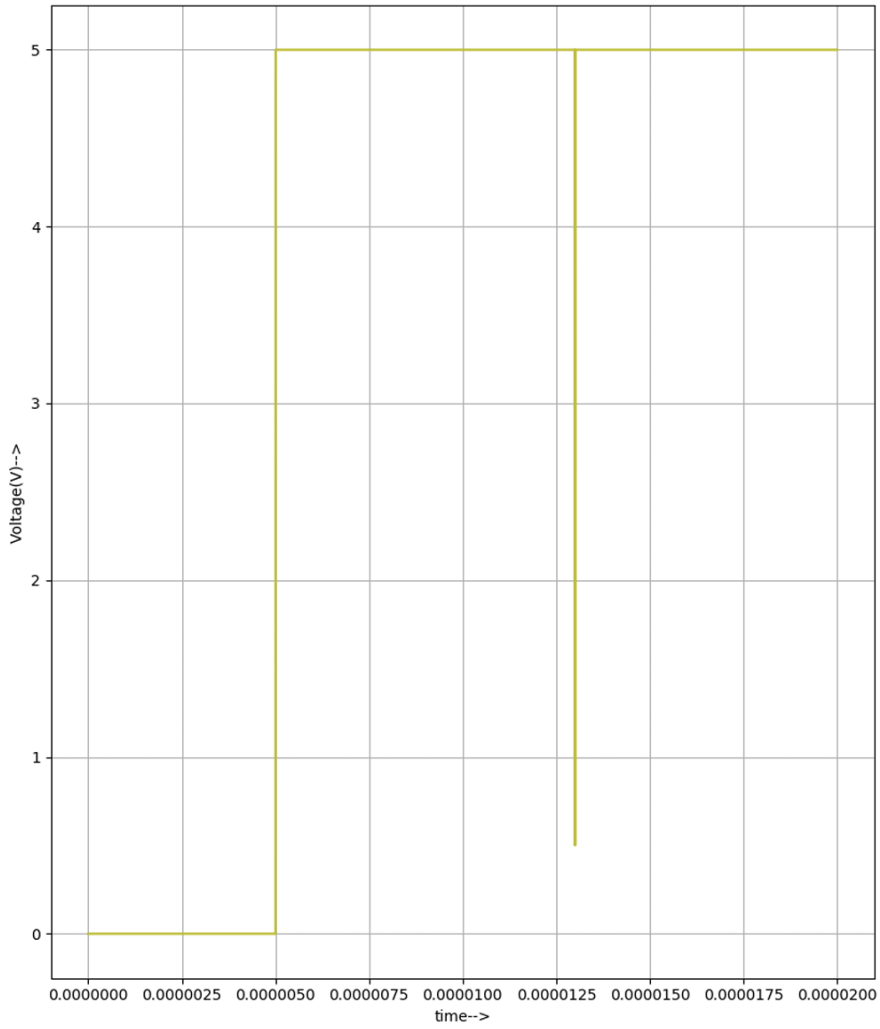

Figure 4.18: Pipeline Delay Proof: The error flag triggers precisely two clock cycles after the physical channel strike.

When the physical channel is blasted by interference at exactly 5μs, the `error_detected` flag does not assert immediately. Instead, it latches HIGH at exactly 7μs. This empirically validates that the corrupted payload required precisely two clock cycles to propagate from the Stage 1 physical channel, through the Stage 2 syndrome calculation, and into the Stage 3 output latch.

Data Correction and Continuous Integrity

Figure 4.19 visualizes the ultimate success of the ECC core's XOR toggling logic.

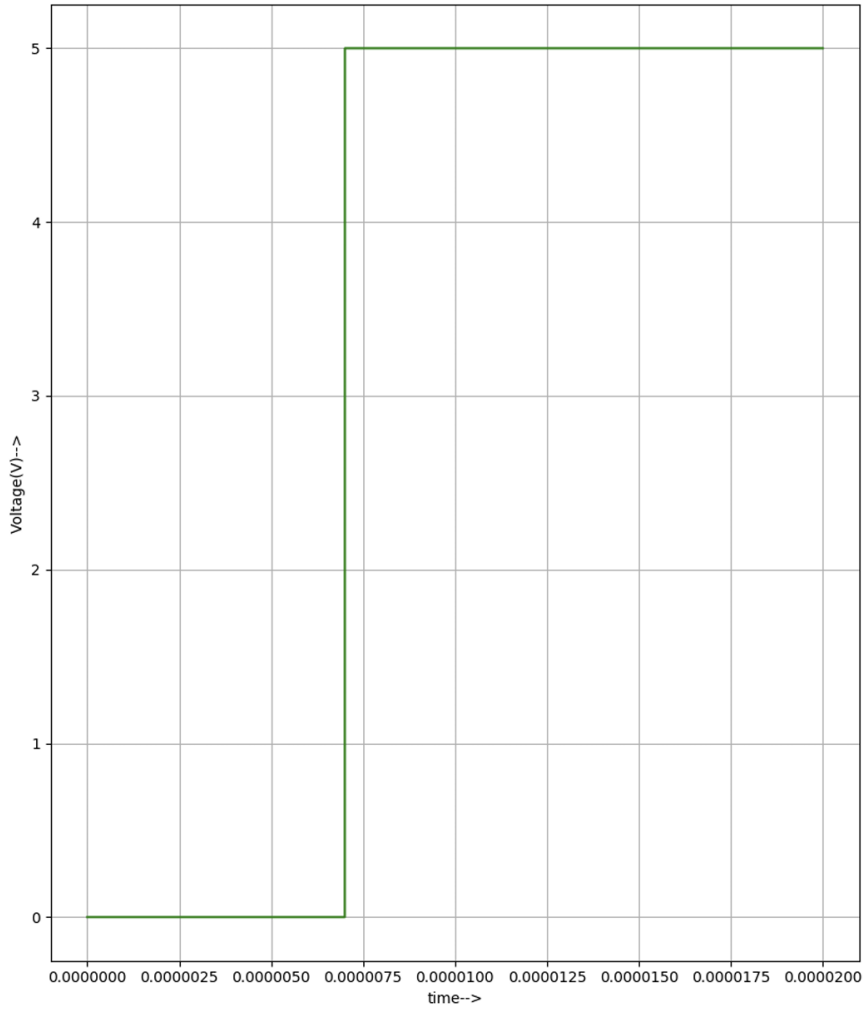

Figure 4.19: Data Output (`data_out0`) remains completely stable during the continuous corruption event.

Despite the internal channel being actively corrupted and the core asserting a severe detection warning, the `data_out0` line maintains a perfect, stable 5V (Logic 1) state throughout the entire simulation transient. The Stage 3 logic flawlessly identified the error via the mathematical syndrome and inverted it back to its correct state, preserving data integrity without a single cycle of latency or data loss.

4.8 Case 8: Vending Machine Controller FSM

4.8.1 Overview and Engineering Objectives

Finite State Machines (FSMs) serve as the fundamental control path architecture in automated digital systems. In this project, a digital Vending Machine Controller was architected to process user transactions, accumulate funds, dispense products, and calculate exact change.

However, designing an FSM for real-world mechanical interfaces introduces the critical challenge of analog signal bouncing. When a physical mechanical button is pressed or a coin is inserted, the resulting analog voltage pulse often remains HIGH across multiple system clock cycles. A naive digital controller would sample this sustained HIGH state continuously, incorrectly registering a single coin drop as multiple rapid insertions.

The primary objective of this project was to architect, synthesize, and verify a robust 3-State Vending Machine FSM engineered with internal digital edge-detectors to sanitize long analog inputs, and to mathematically prove its mechanical safety in a continuous SPICE simulation.

4.8.2 FSM Architecture and State Transitions

The Vending Machine is configured with a fixed item price of 15 Rupees. The architecture operates synchronously across three primary states: `IDLE`, `VEND`, and `RETURN`.

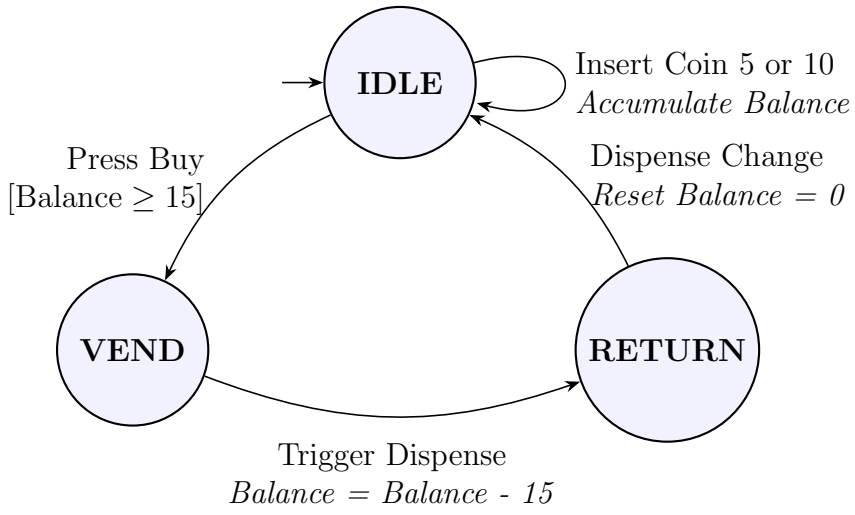

Figure 4.20: State Transition Diagram of the Vending Machine Controller.

4.8.3 RTL Implementation with Edge Detection

To immunize the FSM against long analog `vpulse` signals during physical operation, a 1-cycle delay register technique was utilized. By logically AND-ing the current input state with the inverted delayed state (`input & ~input_delayed`), the FSM

generates a razor-thin internal trigger that is mathematically guaranteed to last only a single clock cycle, regardless of how long the external button is held.

```
module vending_fsm (
input clk, rst, coin_50, coin_100, buy,
output reg dispense,
output reg [7:0] change,
output reg [7:0] balance
);
parameter PRICE = 8'd15;
parameter IDLE = 2'd0, VEND = 2'd1, RETURN = 2'd2;
reg [1:0] state;

// Edge Detectors for Analog-to-Digital safety
reg c5_d, c10_d, buy_d;
wire c5_pulse  = coin_50 & ~c5_d;
wire c10_pulse = coin_100 & ~c10_d;
wire buy_pulse = buy & ~buy_d;

always @(posedge clk or posedge rst) begin
if (rst) begin
c5_d <= 1'b0; c10_d <= 1'b0; buy_d <= 1'b0;
state <= IDLE; balance <= 8'd0;
dispense <= 1'b0; change <= 8'd0;
end else begin
c5_d <= coin_50; c10_d <= coin_100; buy_d <= buy;
dispense <= 1'b0; // Default off to prevent latching

case (state)
IDLE: begin
change <= 8'd0;
if (c5_pulse) balance <= balance + 8'd5;
else if (c10_pulse) balance <= balance + 8'd10;
else if (buy_pulse && balance >= PRICE) state <= VEND;
end
VEND: begin
dispense <= 1'b1;
balance <= balance - PRICE;
state <= RETURN;
end
RETURN: begin
change <= balance;
balance <= 8'd0;
state <= IDLE;
end
default: state <= IDLE;
endcase
end
end
endmodule
```

Listing 4.8: FSM RTL featuring Internal Edge Detectors

4.8.4 Mixed-Signal Integration and Parser Mitigation

The Verilog core was compiled into a C++ Ngspice model and encapsulated inside a 22-pin mixed-signal subcircuit.

Parser Mitigation: A known bug within the KiCad-to-Ngspice translation engine occurs when unwired plot blocks are assigned default net names containing the tilde (~) character, which Ngspice incorrectly interprets as a bitwise NOT math operator, crashing the simulation. To bypass this EDA limitation and successfully simulate the 22-pin IC, explicit Local Labels were hard-routed to all analog plotting nodes.

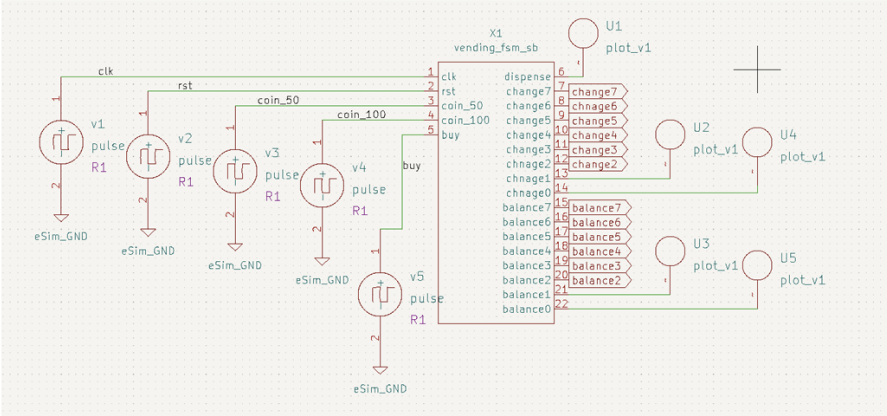

Figure 4.21: 22-Pin Vending Machine Testbench utilizing dynamic multi-cycle stimulus sources and explicit label routing.

4.8.5 Transient Analysis and Waveform Verification

The test sequence simulated dropping a 5 Rupee coin, followed by a 10 Rupee coin, and subsequently pressing the Buy button. The system clock was configured for a 1 MHz frequency. To actively simulate "bouncing" mechanical inputs, the coin and buy `vpulse` sources were deliberately configured with a pulse width of 2μs, forcing them to remain HIGH across multiple clock rising edges.

Balance Accumulator and Edge Detection Proof

Figure 4.22 demonstrates the mathematical success of the edge detectors.

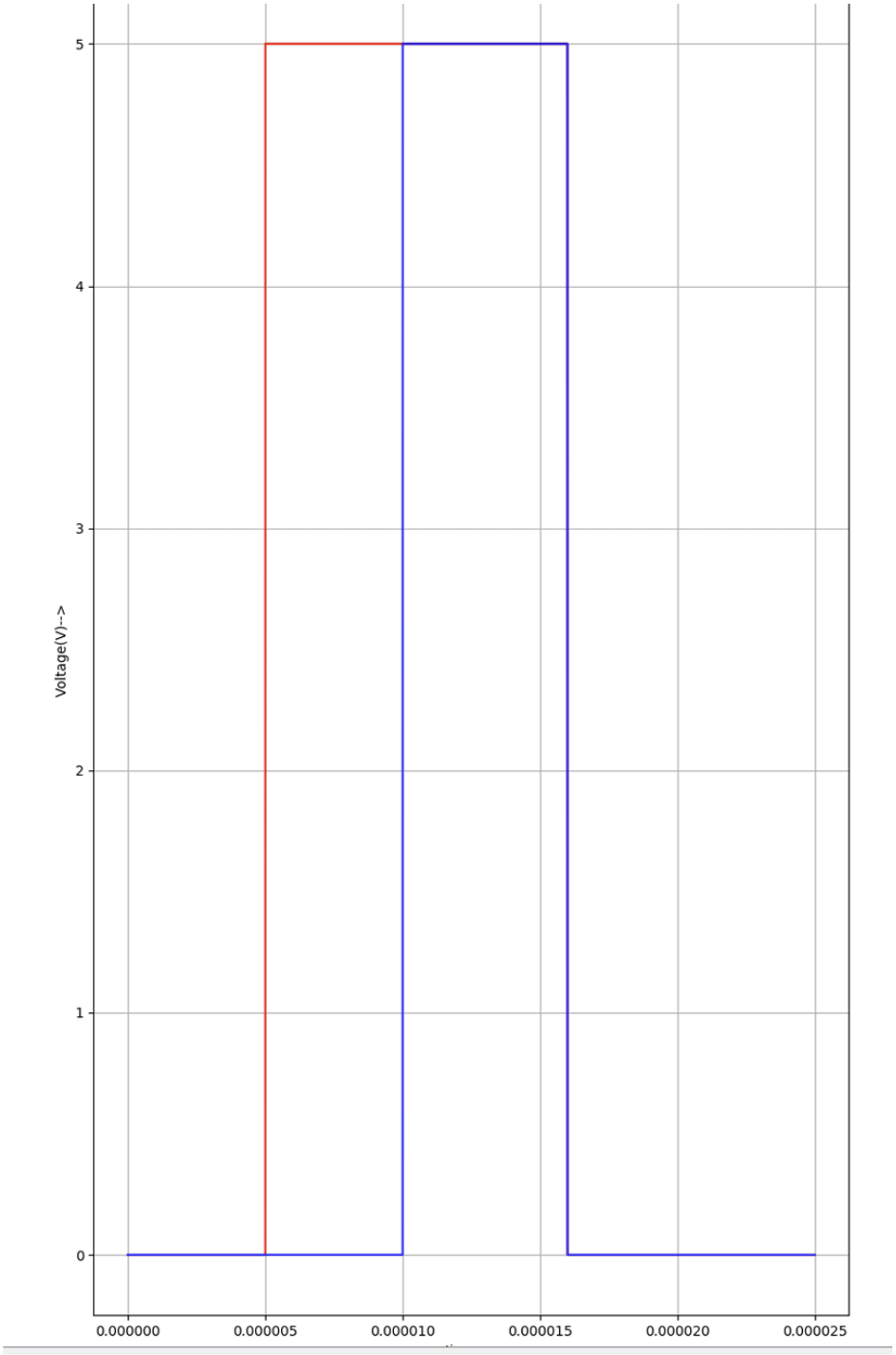

Figure 4.22: Internal tracker proving accurate accumulation ($5 \rightarrow 15 \rightarrow 0$) despite multi-cycle analog inputs.

At 5μs, the 5 Rupee coin is inserted, raising the balance to `0101` (Red line goes HIGH). At 10μs, the 10 Rupee coin is inserted, raising the balance to exactly 15, or `1111` (Blue line goes HIGH alongside Red). Crucially, despite the inputs remaining HIGH for a full 2μs (two clock cycles), the balance incremented only once per button press. This confirms the internal edge detectors successfully sanitized the analog

inputs into perfect single-cycle digital triggers.

FSM State Execution

Figure 4.23 validates the `IDLE` $\rightarrow$ `VEND` transition.

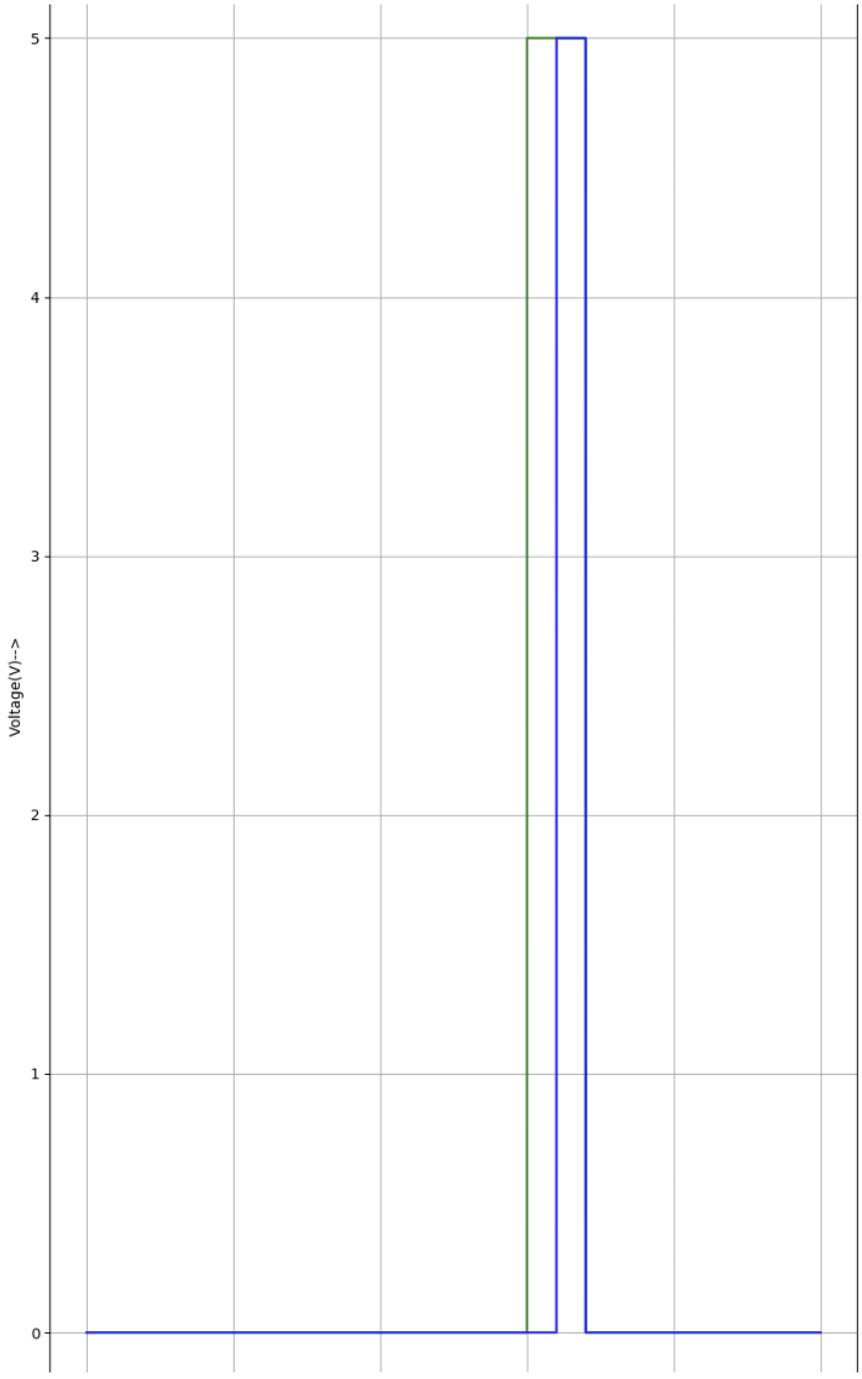

Figure 4.23: State transition triggered by the Buy pulse, yielding a mathematically precise 1-cycle Dispense spike.

The green `buy` pulse asserts at 15μs and remains HIGH for two microseconds.

Immediately upon detection, the FSM verifies the balance is sufficient and fires the blue `dispense` output. As engineered, the dispense logic is a razor-thin spike lasting precisely one clock cycle. This strict temporal adherence guarantees that the physical machine cannot suffer from mechanical double-dispensing errors if a user physically leans on the button.

Chapter 5

Conclusion and Future Scope

5.1 Conclusion

This semester-long internship successfully demonstrated the viability, robustness, and flexibility of open-source Electronic Design Automation (EDA) tools for advanced mixed-signal IP development. Over the course of the project, eight distinct Digital IP cores were architected in Verilog, synthesized via Ngveri, and rigorously verified using the Ngspice continuous analog solver within the FOSSEE eSim environment.

The projects spanned a comprehensive spectrum of digital design complexities:

- **Data-Path Architectures:** The MAC Unit and 4-Tap FIR Filter proved the accuracy of sequential arithmetic and pipeline synchronization.
- **Control and Display Logic:** The Overlapping Sequence Detector and VGA Controller demonstrated the handling of high-frequency nested states and strict timing standards.
- **Industry Communication Protocols:** The I2S Audio Transmitter and PS/2 Keyboard Receiver tackled complex serialization challenges, including the mitigation of multiple asynchronous clock domains using metastability synchronizers.
- **Fault Tolerance and Real-World Physics:** The Pipelined Hamming (7,4) ECC and Vending Machine FSM introduced hardware-level protections against simulated physical anomalies, such as EMI data corruption and analog mechanical button bouncing.

Through rigorous subcircuit encapsulation and strategic mitigation of SPICE matrix constraints (such as resolving singular matrices and bypassing parser bugs), this project proved that the eSim toolchain is highly capable of bridging the gap between discrete digital logic and real-world analog environments.

5.2 Future Scope

The methodologies and IP cores developed during this internship provide a strong foundation for future integration into System-on-Chip (SoC) architectures. Future

work could involve mapping these verified Verilog models onto physical FPGA hardware (such as Xilinx or Lattice boards) to compare the eSim transient simulation timings with actual physical propagation delays. Additionally, these encapsulated eSim subcircuits can be contributed directly to the open-source FOSSEE component libraries, aiding future academic research and accelerating open-source hardware design.

Chapter 6

Bibliography

1. FOSSEE (Free/Libre and Open Source Software for Education). *eSim User Manual*. Indian Institute of Technology Bombay. Available: https://esim.fossee.in/downloads

2. Nenzi, Paolo, and Holger Vogt. *Ngspice Users Manual Version 35*. 2021.

3. Mano, M. Morris, and Michael D. Ciletti. *Digital Design: With an Introduction to the Verilog HDL, VHDL, and System Verilog*. Pearson, 6th Edition, 2017.

4. Proakis, John G., and Dimitris K. Manolakis. *Digital Signal Processing: Principles, Algorithms, and Applications*. Pearson, 4th Edition, 2006.

5. Philips Semiconductors. *I2S bus specification*. February 1986.

6. IBM Corporation. *Personal System/2 Hardware Interface Technical Reference*. 1988.

7. Hamming, R. W. "Error detecting and error correcting codes." *Bell System Technical Journal*, 29(2), 147-160, 1950.

8. IEEE Computer Society. *IEEE Standard for Verilog Hardware Description Language* (IEEE Std 1364-2005). April 2006.

9. Snyder, Wilson. *Verilator: The fastest Verilog/System Verilog simulator*. Available: https://www.veripool.org/verilator/