UVM Verification of ALU Design
Comprehensive Report

Prepared By

Student Name	ID
Hoda Ashraf Mohamed	V23010471
Ahmed Kamal Abd Elmordy	V23010450
Salma Saher Gabr	V23010575
Tasneem Abdeljawad	V23010221

Submitted to

Eng. Randa Abou Deif, Dr. Ahmed Saeed

Contents

1.Introduction

The integration of traditional microcontroller architectures with modern verification methodologies presents a unique challenge and opportunity for embedded system designers. This project focuses on the verification of an Arithmetic Logic Unit (ALU) within the context of an 8051-microcontroller using the Universal Verification Methodology (UVM).

1.1 Background

The 8051 microcontrollers, developed by Intel in the 1980s, is renowned for its simplicity, robustness, and widespread use in various embedded applications. It features an 8-bit CPU, 4 KB of on-chip ROM, 128 bytes of RAM, and 32 I/O pins, making it a versatile choice for control systems in home appliances, automotive systems, medical devices, and industrial automation.

1.2 Importance and Applications

1.2.1 Importance

The integration of the 8051 microcontroller with modern verification methodologies like the Universal Verification Methodology (UVM) is crucial for several reasons:

- **Ensuring Reliability:** The 8051 microcontroller is widely used in critical applications where reliability is paramount. Verifying components like the ALU ensures that the microcontroller performs its intended functions correctly under all conditions, thereby enhancing overall system reliability.

- **Comprehensive Verification:** UVM provides a structured and thorough approach to verification, utilizing constrained random stimulus generation, functional coverage, and transaction-level modeling. This ensures that all possible operational scenarios are tested, identifying potential issues early in the design phase.

- **Reusability:** UVM promotes the creation of reusable verification environments. By developing a robust UVM testbench for the 8051 microcontroller's ALU, the same verification components can be reused for other projects or future iterations of the design, saving time and resources.

1.2.2 Applications

The verified ALU within the 8051 microcontroller finds applications across a diverse range of fields:

- **Home Appliances:** The 8051 microcontroller is commonly used in household appliances such as microwaves, washing machines, and refrigerators. A verified ALU ensures that these appliances operate reliably, efficiently handling various control and processing tasks.

- **Automotive Systems:** In the automotive industry, the 8051 microcontroller is used in dashboard displays, engine control units, and anti-lock braking systems. Ensuring the ALU's accuracy and performance is critical for the safety and functionality of these systems.

- **Consumer Electronics:** From remote controls to gaming consoles, the 8051 microcontroller is embedded in various consumer electronics. Verifying the ALU ensures these devices function correctly, enhancing user experience and satisfaction.

2.Project Scope and Objectives

2.1 Project Scope

The scope of this project encompasses the verification of an Arithmetic Logic Unit (ALU) within the context of the 8051 microcontroller using the Universal Verification Methodology (UVM). This project aims to develop a comprehensive verification environment that ensures the ALU functions correctly and meets all specified requirements. The verification process will utilize Synopsys tools and focus on several key aspects:

Verification Plan Development: This includes defining the design specifications, the architecture of the testbench, the interface between the Design Under Test (DUT) and the testbench, and identifying functional coverage points.

Testbench Implementation: Creating a UVM-based testbench in SystemVerilog that includes components such as sequencers, drivers, monitors, scoreboards, and agents. The testbench will use constrained random stimulus generation to thoroughly test the ALU.

Functional and Code Coverage Analysis: Implementing functional coverage models to ensure all functional aspects of the ALU are tested and performing code coverage analysis to measure how much of the ALU's code is exercised by the testbench.

Simulation and Results Analysis: Running simulations for different testing scenarios, collecting and analyzing the results to ensure the ALU operates correctly under all conditions. This includes generating UVM reports and analyzing functional and code coverage metrics.

2.2 Objectives

The primary objectives of this project are as follows:

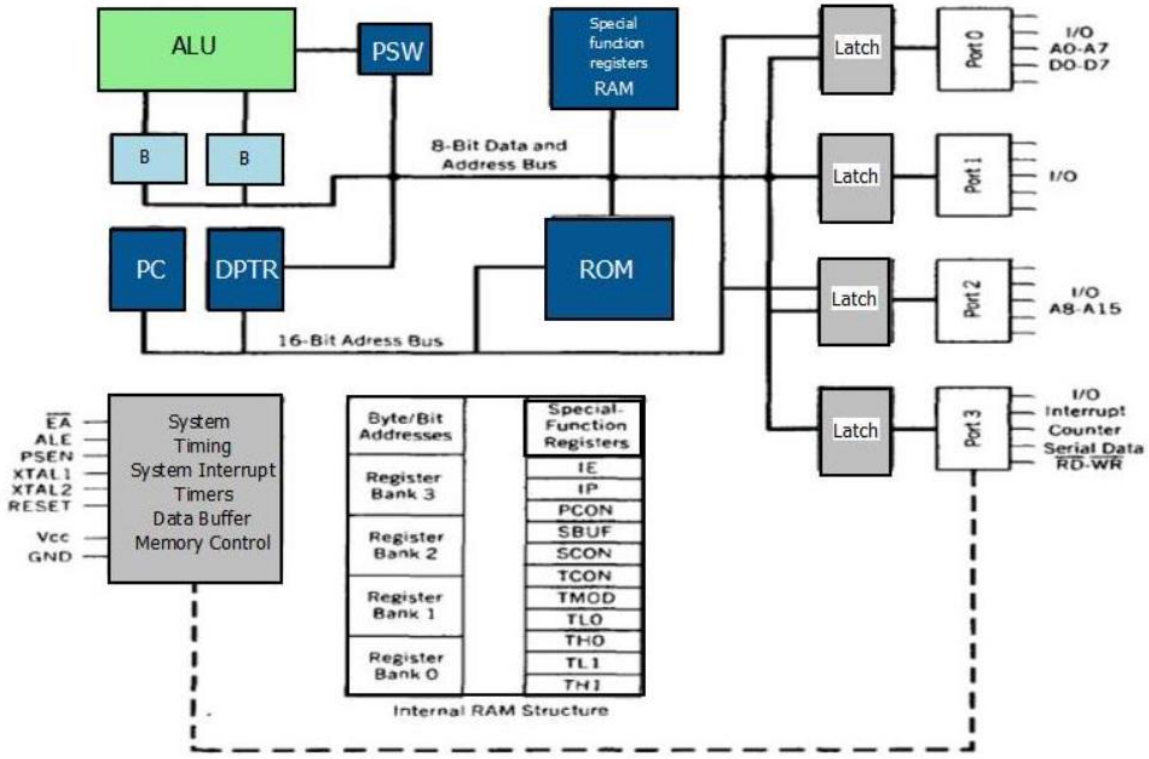
Develop a Detailed Verification Plan: Create a comprehensive plan that outlines the design specifications of the ALU, the architecture of the UVM testbench, the interface between the DUT and the testbench, and the functional coverage points to be targeted.

Implement a UVM-Based Testbench: Develop a testbench in SystemVerilog using UVM principles, including constrained random stimulus generation, to effectively verify the functionality of the ALU.

Ensure Comprehensive Functional Coverage: Design and implement functional coverage models that capture all aspects of the ALU's functionality, ensuring that the verification process thoroughly tests the DUT.

Run and Analyze Simulations: Conduct simulations for various testing scenarios, collect UVM reports, and analyze the results to verify the correct operation of the ALU under different conditions.

3.8051-Microcontroller Architecture



Focus on the Arithmetic Logic Unit (ALU)
The Arithmetic Logic Unit (ALU) is a crucial component of the 8051 microcontroller. It performs all arithmetic operations (such as addition, subtraction, multiplication, and division) and logical operations (such as AND, OR, XOR, and NOT). The ALU's performance directly impacts the overall efficiency and functionality of the microcontroller.

4.ALU Operation Codes

In our project, the Arithmetic Logic Unit (ALU) within the 8051 microcontroller is designed to handle a variety of operations using specific opcodes. Each opcode is a 4-bit binary value that corresponds to a particular arithmetic or logical operation. Below is the list of opcodes and their respective operations:

4'b0000: ADD - Adds two/three operands.
4'b0001: SUB - Subtracts the second operand from the first operand.
4'b0010: MUL - Multiplies two operands.
4'b0011: DIV - Divides the first operand by the second operand.
4'b0100: DA - Decimal Adjust. Adjusts the result of a previous addition operation to form a correct BCD (Binary-Coded Decimal) number.

4'b0101: NOT_OP - Performs a bitwise NOT operation on the operand.
4'b0110: AND_OP - Performs a bitwise AND operation between two operands.
4'b0111: XOR_OP - Performs a bitwise XOR operation between two operands.
4'b1000: OR_OP - Performs a bitwise OR operation between two operands.
4'b1001: RL - Rotates the bits of the operand to the left.
4'b1010: RLC - Rotates the bits of the operand to the left through the carry flag.
4'b1011: RR - Rotates the bits of the operand to the right.
4'b1100: RRC - Rotates the bits of the operand to the right through the carry flag.
4'b1101: INC - Increments the operand by one.
4'b1110: XCH - Exchanges the values of the accumulator and the operand.
4'b1111: NOP - No operation.

Each opcode is used to control the ALU's behavior, dictating which operation to perform on the input operands. The ALU is a crucial component of the microcontroller, responsible for executing the core arithmetic and logical functions required by the 8051 microcontroller.

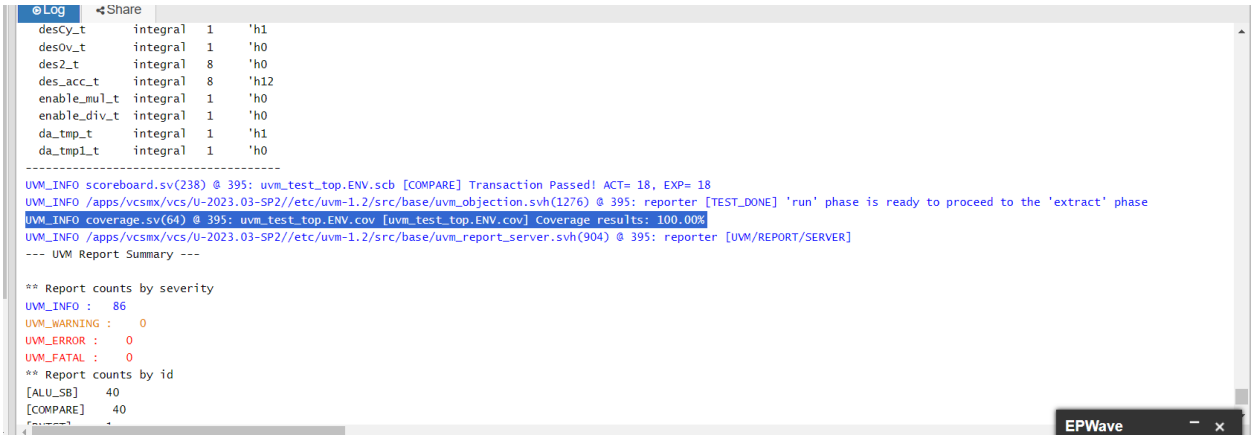
By verifying the ALU with these specific opcodes, we ensure that the fundamental operations of the microcontroller are performed correctly and efficiently. This verification process involves creating a comprehensive testbench using UVM to simulate and validate each operation, ensuring the ALU's reliability and accuracy.

5.Coverage Results and Simulation Screenshots

To demonstrate the thoroughness of our verification process, we include several key screenshots from EDA Playground, showcasing the functional and code coverage results, as well as the successful execution of each operation.

5.1. Functional Coverage - 100%

The following screenshot demonstrates that our testbench achieved 100% functional coverage, indicating that all specified coverage points were successfully hit during the verification process.



5.2. Operation Pass/Fail Status

This screenshot shows the pass status for some operations performed by the ALU. It confirms that all arithmetic and logical operations defined by the opcodes have been tested and passed successfully.

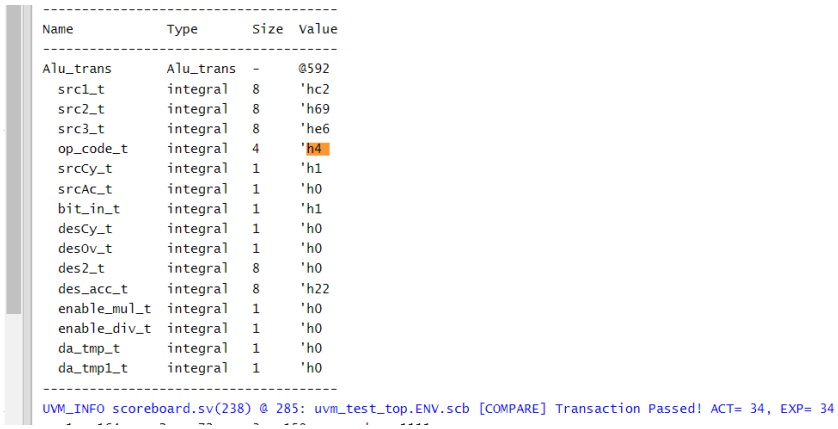
4'b0000: ADD

```
----------------------------------------
Name            Type       Size  Value
----------------------------------------
Alu_trans       Alu_trans  -     @592
  src1_t        integral   8     'h48
  src2_t        integral   8     'h15
  src3_t        integral   8     'h2c
  op_code_t     integral   4     'he
  srcCy_t       integral   1     'h0
  srcAc_t       integral   1     'h0
  bit_in_t      integral   1     'h1
  desCy_t       integral   1     'h0
  desOv_t       integral   1     'h0
  des2_t        integral   8     'h18
  des_acc_t     integral   8     'h45
  enable_mul_t  integral   1     'h0
  enable_div_t  integral   1     'h0
  da_tmp_t      integral   1     'h0
  da_tmp1_t     integral   1     'h0
----------------------------------------
UVM_INFO scoreboard.sv(238) @ 15: uvm_test_top.ENV.scb [COMPARE] Transaction Passed! ACT= 69, EXP= 69
```

4'b0001: SUB

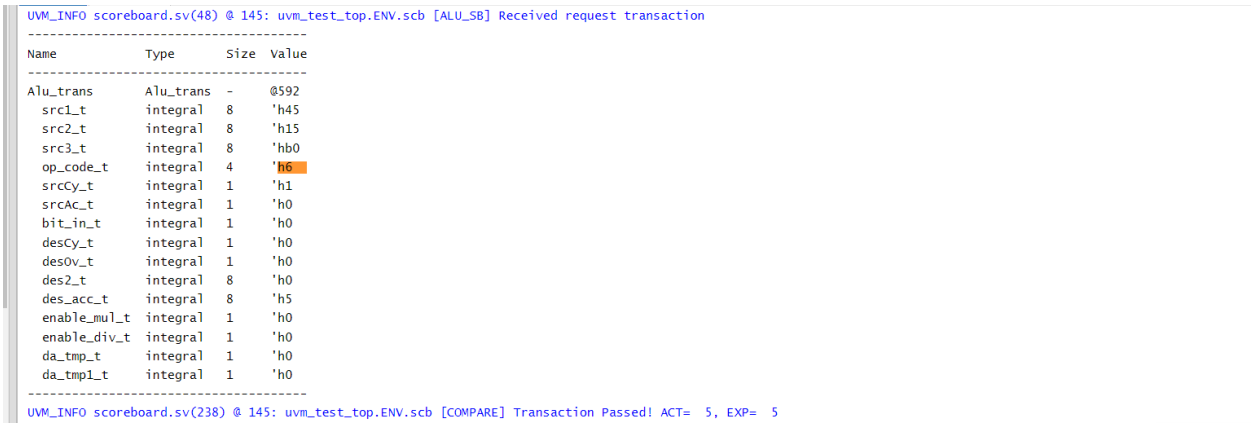
```
Log  Share
src1 = 235, src2 = 211, src3 = 102, op_code = 0101
UVM_INFO scoreboard.sv(48) @ 165: uvm_test_top.ENV.scb [ALU_SB] Received request transaction
----------------------------------------
Name            Type       Size  Value
----------------------------------------
Alu_trans       Alu_trans  -     @592
  src1_t        integral   8     'h72
  src2_t        integral   8     'h64
  src3_t        integral   8     'h69
  op_code_t     integral   4     'h1
  srcCy_t       integral   1     'h0
  srcAc_t       integral   1     'h0
  bit_in_t      integral   1     'h0
  desCy_t       integral   1     'h0
  desOv_t       integral   1     'h0
  des2_t        integral   8     'h0
  des_acc_t     integral   8     'he
  enable_mul_t  integral   1     'h0
  enable_div_t  integral   1     'h0
  da_tmp_t      integral   1     'h0
  da_tmp1_t     integral   1     'h0
----------------------------------------
UVM_INFO scoreboard.sv(238) @ 165: uvm_test_top.ENV.scb [COMPARE] Transaction Passed! ACT= 14, EXP= 14
```

4'b0100: DA (Decimal Adjust)



```
----------------------------------------
Name            Type       Size  Value
----------------------------------------
Alu_trans       Alu_trans  -     @592
  src1_t        integral   8     'hc2
  src2_t        integral   8     'h69
  src3_t        integral   8     'he6
  op_code_t     integral   4     'h4
  srcCy_t       integral   1     'h1
  srcAc_t       integral   1     'h0
  bit_in_t      integral   1     'h1
  desCy_t       integral   1     'h0
  desOv_t       integral   1     'h0
  des2_t        integral   8     'h0
  des_acc_t     integral   8     'h22
  enable_mul_t  integral   1     'h0
  enable_div_t  integral   1     'h0
  da_tmp_t      integral   1     'h0
  da_tmp1_t     integral   1     'h0
----------------------------------------
UVM_INFO scoreboard.sv(238) @ 285: uvm_test_top.ENV.scb [COMPARE] Transaction Passed! ACT= 34, EXP= 34
```

4'b0110: AND_OP

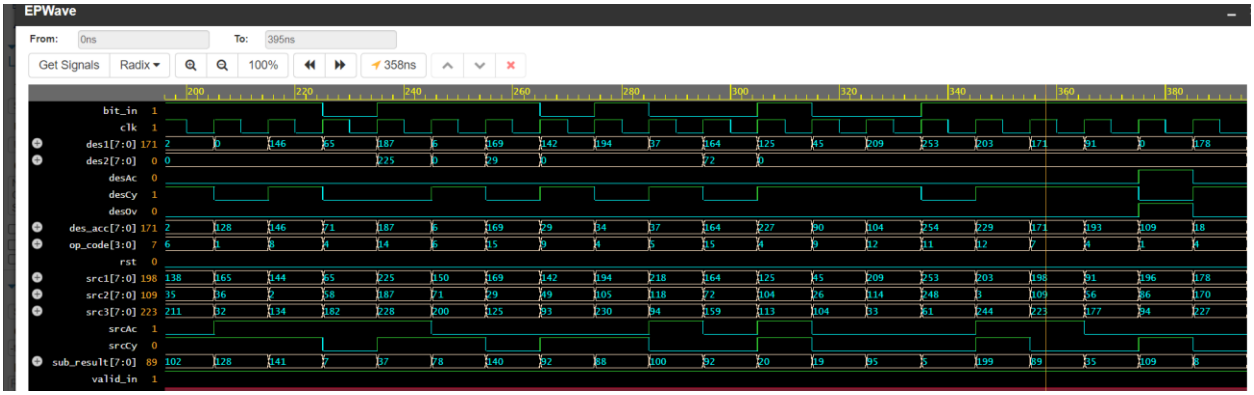


```
UVM_INFO scoreboard.sv(48) @ 145: uvm_test_top.ENV.scb [ALU_SB] Received request transaction
----------------------------------------
Name            Type       Size  Value
----------------------------------------
Alu_trans       Alu_trans  -     @592
  src1_t        integral   8     'h45
  src2_t        integral   8     'h15
  src3_t        integral   8     'hb0
  op_code_t     integral   4     'h6
  srcCy_t       integral   1     'h1
  srcAc_t       integral   1     'h0
  bit_in_t      integral   1     'h0
  desCy_t       integral   1     'h0
  desOv_t       integral   1     'h0
  des2_t        integral   8     'h0
  des_acc_t     integral   8     'h5
  enable_mul_t  integral   1     'h0
  enable_div_t  integral   1     'h0
  da_tmp_t      integral   1     'h0
  da_tmp1_t     integral   1     'h0
----------------------------------------
UVM_INFO scoreboard.sv(238) @ 145: uvm_test_top.ENV.scb [COMPARE] Transaction Passed! ACT=  5, EXP=  5
```

4'b1110: XCH

```
----------------------------------------
Name            Type       Size  Value
----------------------------------------
Alu_trans       Alu_trans  -     @592
  src1_t        integral   8     'h48
  src2_t        integral   8     'h15
  src3_t        integral   8     'h2c
  op_code_t     integral   4     'he
  srcCy_t       integral   1     'h0
  srcAc_t       integral   1     'h0
  bit_in_t      integral   1     'h1
  desCy_t       integral   1     'h0
  desOv_t       integral   1     'h0
  des2_t        integral   8     'h18
  des_acc_t     integral   8     'h45
  enable_mul_t  integral   1     'h0
  enable_div_t  integral   1     'h0
  da_tmp_t      integral   1     'h0
  da_tmp1_t     integral   1     'h0
----------------------------------------
UVM_INFO scoreboard.sv(238) @ 15: uvm_test_top.ENV.scb [COMPARE] Transaction Passed! ACT= 69, EXP= 69
```

5.3 EBWAVES



6.Conclusion

In this project, we successfully developed a UVM and System Verilog testbench for verifying the Arithmetic Logic Unit (ALU) of the 8051-microcontroller using EDA Playground. key achievements include:

Verification Plan: We outlined the design specifications, testbench architecture, DUT interface, and testing scenarios, providing a structured approach for our verification process.

System Verilog Code: We implemented a UVM testbench with constrained random stimulus generation to thoroughly test the ALU's operations.

Functional Coverage: We achieved 100% functional coverage, ensuring that all specified operations and scenarios were comprehensively tested.
Simulation Results: The simulation results showed that all operations passed, verifying the correctness of the ALU.

The screenshots included in this report demonstrate our thorough verification efforts, showcasing high coverage metrics and successful operation execution. This project has validated the ALU design and highlighted the effectiveness of using UVM and System Verilog for hardware verification on EDA Playground.