Full-Stack Web Development

for Auto-Assessment Platform (AASP)

Chua Chong Yih
U2022784B

Table of contents

01

Introduction

Background

Programming Assessments

- Evaluate learner's understanding of Computer Science concepts
- Automation help ease educator's burden

Hardware Description Language (HDL)

- Specialized computer language
- Simulate behavior of digital circuits and systems
- Lack of available platforms for evaluation

Hardware Description Language

Module Design

Testbench

Waveform Visualization

Prior Work

Lee Jun Wei

Baseline Model

Designed and developed AASP

Liu Wing Lam

Test Proctoring

Implemented test proctoring feature and enhancements

Chua Chong Yih

Hardware Description Language

Problem Statement

- AASP lacks HDL assessment support
- Limitations in evaluating digital circuit behavior
- Closing this gap is crucial for holistic assessment

Objectives

Build and improve AASP

Effective HDL assessments

Scalability

02

Related Works

HDLBits

Circuit Design

Reading Simluations

Writing Testbenches

Exercise Evaluation

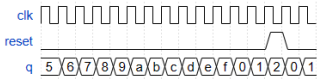
Circuit Design Exercises

- Combinational and sequential logic designs
- Sequential logic involves timing elements
- Waveform visualization of outputs

Count15

dualedge Previous Next count10

Build a 4-bit binary counter that counts from 0 through 15, inclusive, with a period of 16. The reset input is synchronous, and should reset the counter to 0.

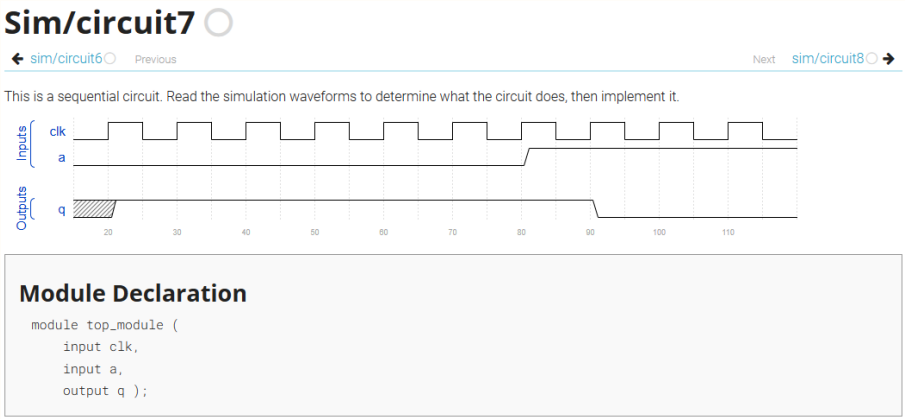


Module Declaration

```
module top_module (
    input clk,
    input reset,      // Synchronous active-high reset
    output [3:0] q);
```

Verification: Reading Simulations

- Presented with simulation waveforms
- Interpret and recreate module design



Verification: Writing Testbenches

- Create testing code
- Generate specific inputs and monitor outputs
- Verify output with expected results

Tb/clock

sim/circuit10 Previous Next tb/tb1

You are provided a module with the following declaration:

```
module dut ( input clk ) ;
```

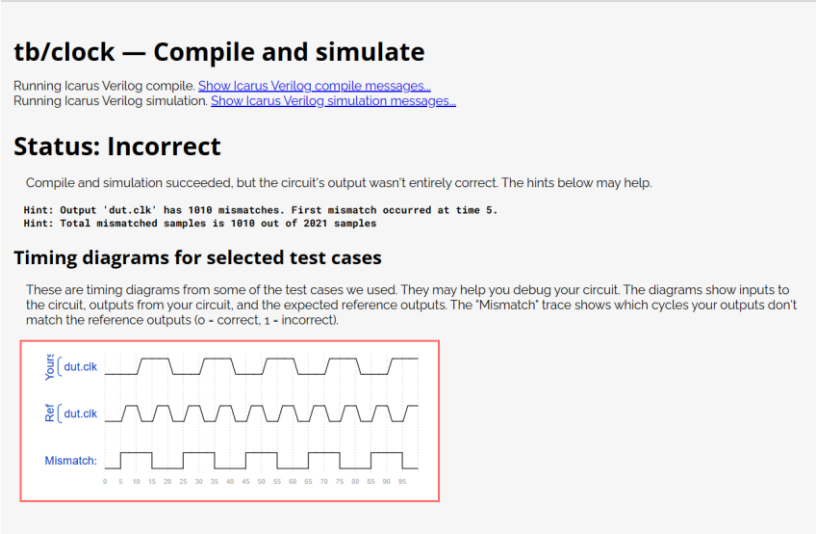
Write a testbench that creates one instance of module `dut` (with any instance name), and create a clock signal to drive the module's `clk` input. The clock has a period of 10 ps. The clock should be initialized to zero with its first transition being 0 to 1.

Module Declaration

```
module top_module ( );
```

Exercise Evaluation

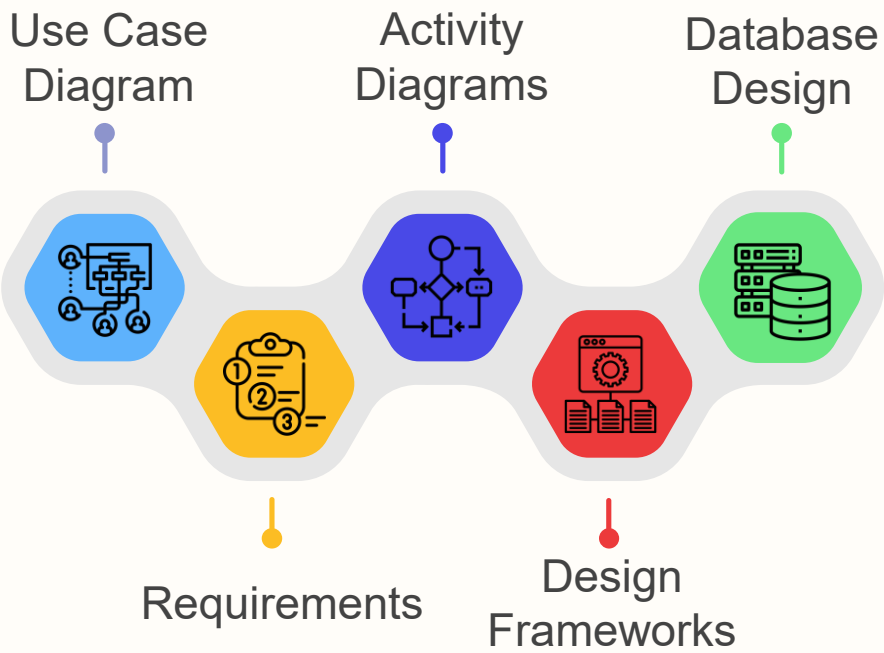
- Evaluate HDL assessment output
- Compares learner solution waveform with expected solution waveform
- Mismatch graph highlight points of divergence



03

Design Methodology

Overview


Use Case Diagram
Activity Diagrams
Database Design
Requirements
Design Frameworks

Summary of Functional Requirements

HDL Assessment Integration

- HDL-based assessments
- Different assessment configurations
- Generate boilerplate code

Component Validation

- Validate correctness of student code

Compilation and Simulation

- Incorporate modified Judge0 framework

Waveform Visualization

- Interactive waveform visualization
- Static waveform visualization
- Mismatch Graph

04

Implementation

Judge0

Online Code Execution Engine

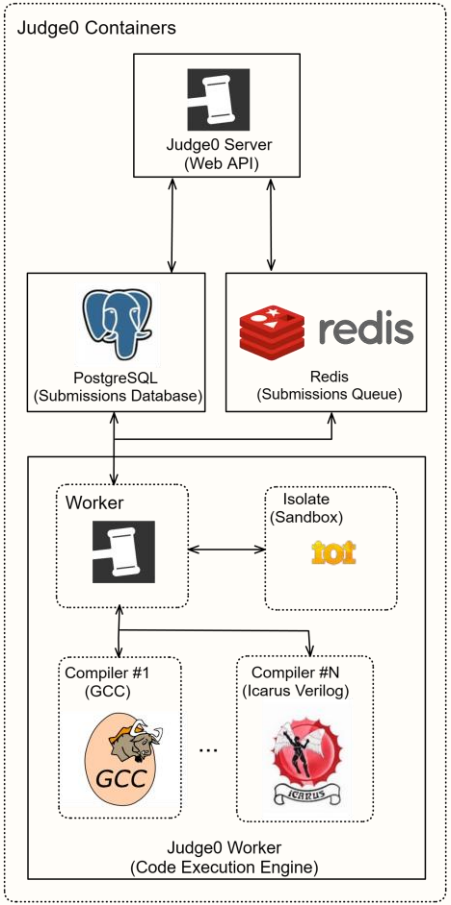
- Open source
- Lack HDL support

Modify compiler Docker image

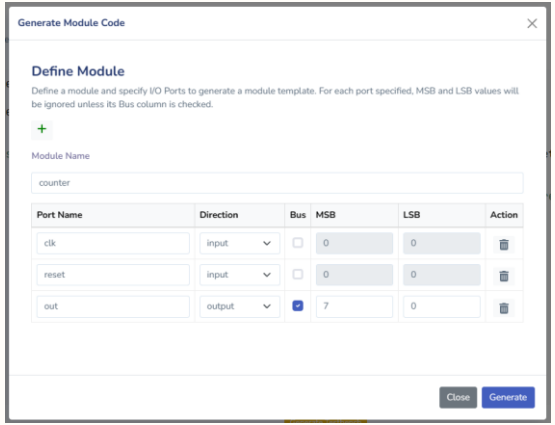
- Add Icarus Verilog compiler
- Curate selected compilers

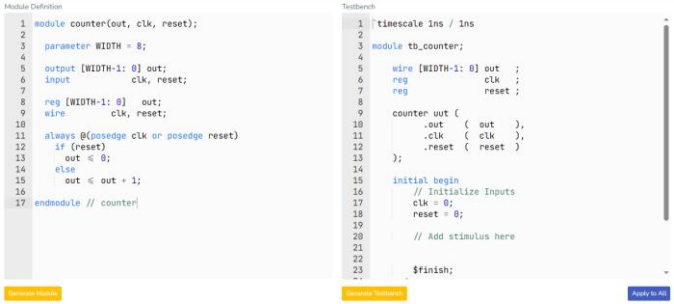
Modify core Judge0 Docker image

- Add additional parameters



Boilerplate Code Generation





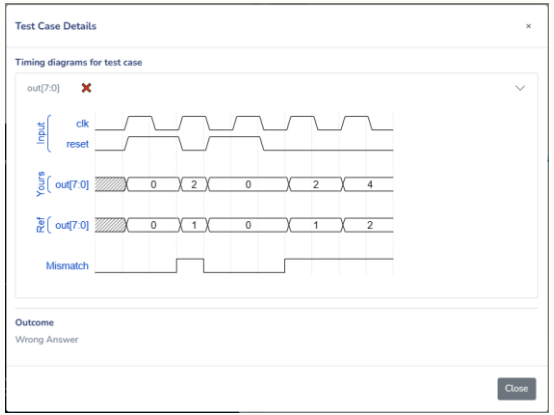
Module Code

Generate module design code from form

Testbench

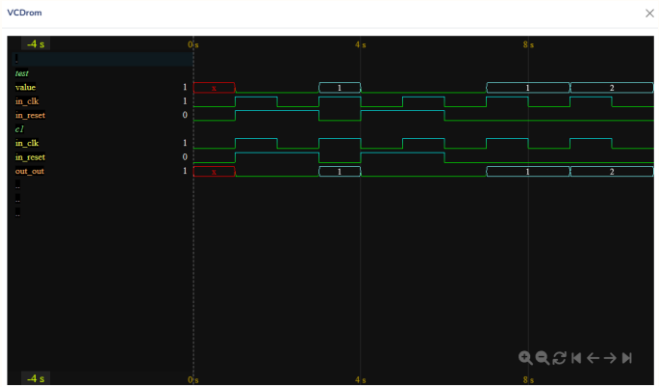
Generate testbench code from module design code

Waveform Visualization



WaveDrom

Static compiled output visualization and mismatch graph comparison



VCDrom

Navigate through compiled output with an interactive UI

05

Conclusion

Conclusion

Achievements

- Developed and enhanced AASP to include HDL assessments
- Implemented HDL related technologies to enhance assessment experience

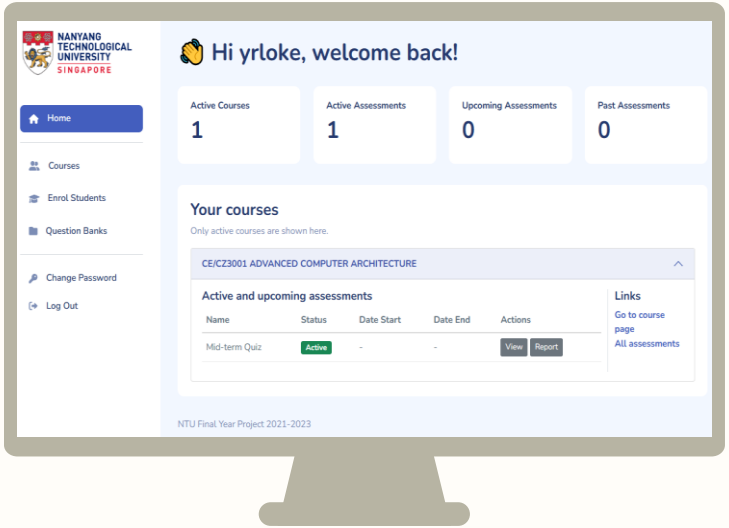
Future Works

- Additional HDL
- More Question Types
- Unit Test

06

Live Demo

http://172.21.148.181



Thanks!

Does anyone have any questions?