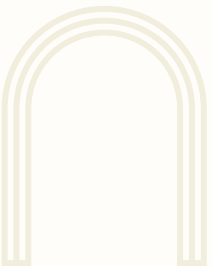


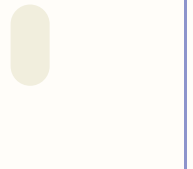


# Full-Stack Web Development

## for Auto-Assessment Platform (AASP)



Chua Chong Yih  
U2022784B



# Table of contents

01

## Introduction

Background, prior works,  
problem statement & objective

02

## Related Works

Related platforms & technologies

03

## Design

Requirements & methodologies

04

## Implementation

System architecture & features

05

## Conclusion

Achievements & future works

06

## Demo

Live demo of main features



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

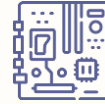
Designed and developed AASP



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

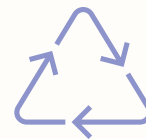
# Objectives



**Build and  
improve AASP**



**Effective  
HDL assessments**



**Scalability**

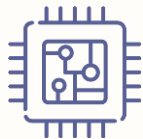




02

# Related Works

# HDLBits



**Circuit Design**



**Reading  
Simulations**



**Writing  
Testbenches**



**Exercise  
Evaluation**



# Circuit Design Exercises

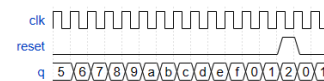
- Combinational and sequential logic designs
- Sequential logic involves timing elements
- Waveform visualization of outputs

## 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

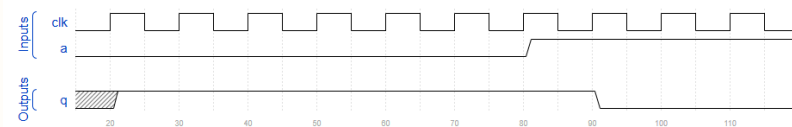
- Presented with simulation waveforms
- Interpret and recreate module design

## Sim/circuit7

[← sim/circuit6](#) Previous

Next [sim/circuit8](#) →

This is a sequential circuit. Read the simulation waveforms to determine what the circuit does, then implement it.



### Module Declaration

```
module top_module (  
    input clk,  
    input a,  
    output q );
```

# 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

## tb/clock — Compile and simulate

Running Icarus Verilog compile: [Show Icarus Verilog compile messages.](#)  
Running Icarus Verilog simulation: [Show Icarus Verilog simulation messages.](#)

### Status: Incorrect

Compile and simulation succeeded, but the circuit's output wasn't entirely correct. The hints below may help.

Hint: Output 'dut.clk' has 1010 mismatches. First mismatch occurred at time 5.  
Hint: Total mismatched samples is 1010 out of 2021 samples

### Timing diagrams for selected test cases

These are timing diagrams from some of the test cases we used. They may help you debug your circuit. The diagrams show inputs to the circuit, outputs from your circuit, and the expected reference outputs. The 'Mismatch' trace shows which cycles your outputs don't match the reference outputs (0 - correct, 1 - incorrect).



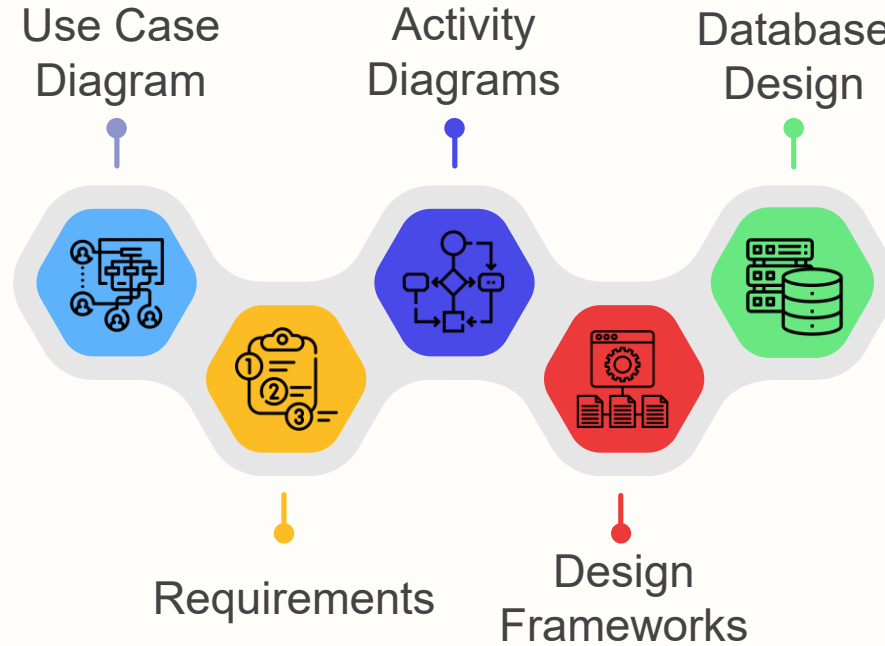


03

# Design Methodology



# Overview





# 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

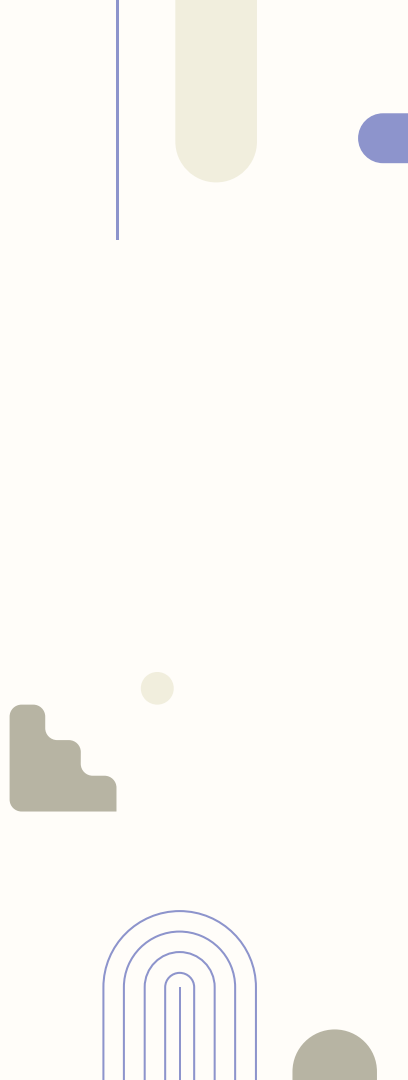
- Incorporate modified Judge0 framework

## Waveform Visualization

- Interactive waveform visualization
- Static waveform visualization
- Mismatch Graph

04

# Implementation



# Judge0



## Online Code Execution Engine

- Open source
- Lack HDL support



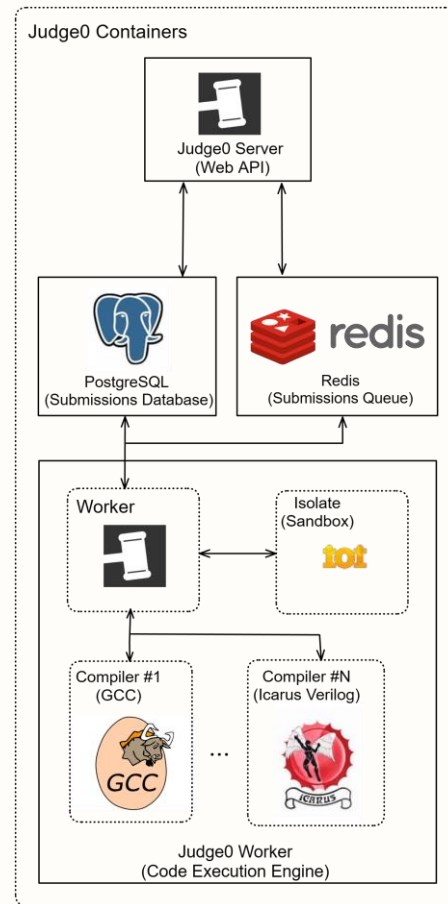
## Modify compiler Docker image

- Add Icarus Verilog compiler
- Curate selected compilers



## Modify core Judge0 Docker image

- Add additional parameters



# Boilerplate Code Generation

**Generate Module Code**

**Define Module**

Define a module and specify I/O Ports to generate a module template. For each port specified, MSB and LSB values will be ignored unless its Bus column is checked.

Module Name  
counter

Port Name	Direction	Bus	MSB	LSB	Action
clk	input	<input type="checkbox"/>	0	0	
reset	input	<input type="checkbox"/>	0	0	
out	output	<input checked="" type="checkbox"/>	7	0	

Close Generate

## Module Code

Generate module design code  
from form

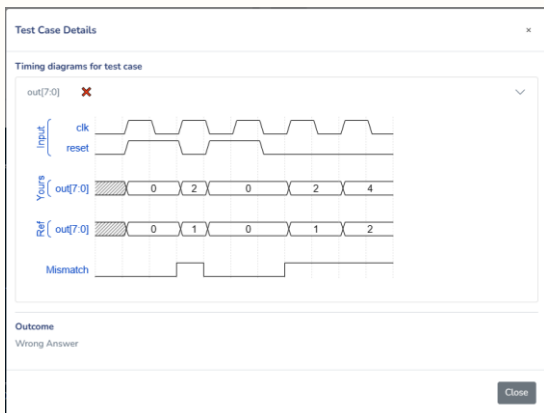
```
Module Definition
1 module counter(out, clk, reset);
2
3   parameter WIDTH = 8;
4   output [WIDTH-1: 0] out;
5   input  clk, reset;
6   reg [WIDTH-1: 0] out;
7   wire clk, reset;
8
9   always @(posedge clk or posedge reset)
10      if (reset)
11         out <= 0;
12      else
13         out <= out + 1;
14   endmodule // counter

Testbench
1 timescale 1ns / 1ns
2 module tb_counter;
3   wire [WIDTH-1: 0] out;
4   reg clk;
5   reg reset;
6
7   counter uut (
8       .out (out),
9       .clk (clk),
10      .reset (reset)
11  );
12
13   initial begin
14       // Initialize Inputs
15       clk = 0;
16       reset = 0;
17       // Add stimulus here
18
19   $finish;
20 endmodule
```

## Testbench

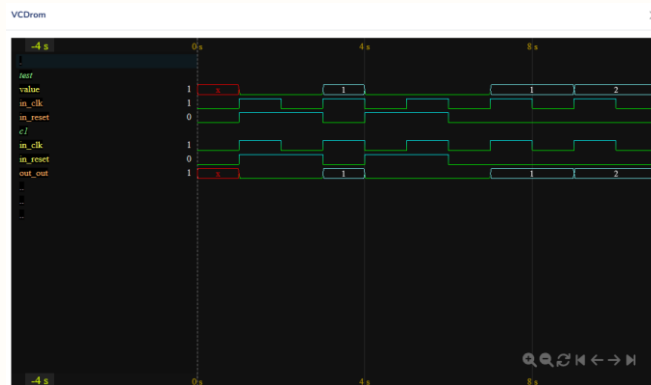
Generate testbench code from  
module design code

# 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



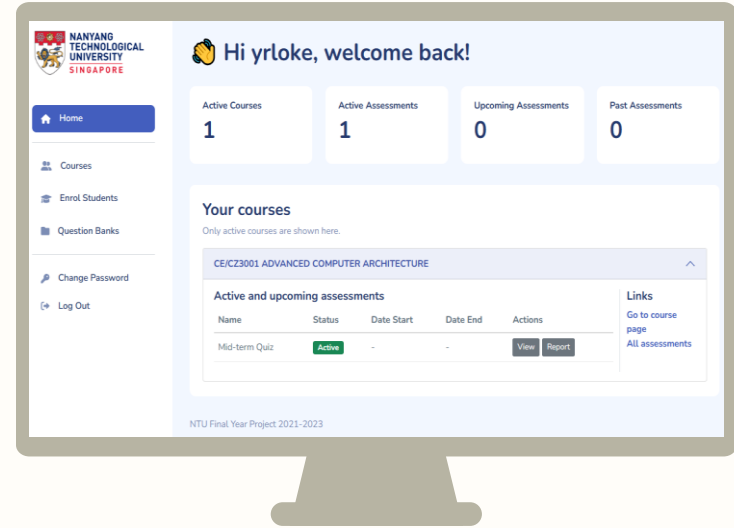
## Future Works

- Additional HDL
- More Question Types
- Unit Test

# 06

# Live Demo

<http://172.21.148.181>







# Thanks!

**Does anyone have any questions?**

**CREDITS:** This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**

