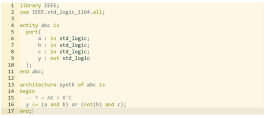
Tufts University
School of Engineering

Analyzing Code Trajectories to Understand VHDL Learning

Brendan Amorin, Shuchin Aeron, Steven Bell, David Hammer, Mark Hempstead, Eric Miller

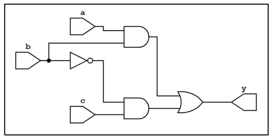
What is VHDL and why is it Difficult?

- VHDL is a hardware description language that can model the behavior of digital systems.



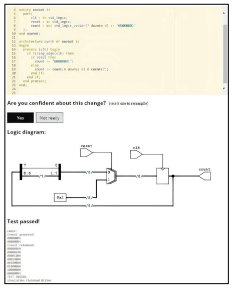
```
library IEEE;
use IEEE.std_logic_1164.all;

entity abc is
  port(
      a : in std_logic;
      b : in std_logic;
      c : in std_logic;
      y : out std_logic
  );
end abc;

architecture synth of abc is
begin
  -- Y = AB + B'C
  y <= (a and b) or (not(b) and c);
end;
```

Simple VHDL Example



Corresponding Hardware

- It is concurrent: statements generally occur simultaneously, as opposed to sequentially.
- Requires a basic understanding of hardware being designed.
- Typically, novice VHDL programmers first learned a sequential programming language.
- Results in a variety of methods, some erroneous, for completing problems.



```
entity onehot is
  port(
      clk : in std_logic;
      reset : in std_logic;
      count : out std_logic_vector(7 downto 0) := "00000001"
  );
end onehot;

architecture synth of onehot is
begin
  process (clk) begin
    if rising_edge(clk) then
      if reset then
        count <= "00000001";
      else
        count <= count(6 downto 0) & count(7);
      end if;
    end if;
  end process;
end;
```

Are you confident about this change? (select one to recompile)

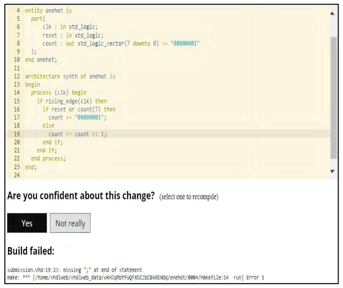
Yes | Not really

Logic diagram:

Test passed!

```
count:
(reset asserted)
00000001
00000001
(reset released)
00000010
00000100
00001000
00010000
00100000
01000000
10000000
00000001
TEST PASSED.
simulation finished @115ns
```

Concatenate-bits Method



```
entity onehot is
  port(
      clk : in std_logic;
      reset : in std_logic;
      count : out std_logic_vector(7 downto 0) := "00000001"
  );
end onehot;

architecture synth of onehot is
begin
  process (clk) begin
    if rising_edge(clk) then
      if reset or count(7) then
        count <= "00000001";
      else
        count <= count << 1;
      end if;
    end if;
  end process;
end;
```

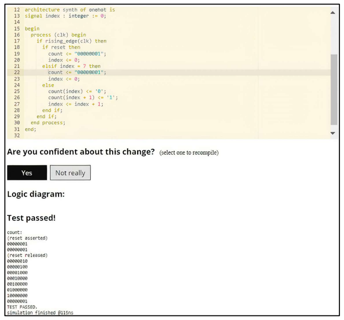
Are you confident about this change? (select one to recompile)

Yes | Not really

Build failed:

```
submission.vhd:19:23: missing ";" at end of statement
make: *** [/home/vhdlweb/vhdlweb_data/wkHlqRbtfuQFXSCZbCBAXENOq/onehot/0004/Makefile:14  run] Error 1
```

C-code Method



```
architecture synth of onehot is
signal index : integer := 0;

begin
  process (clk) begin
    if rising_edge(clk) then
      if reset then
        count <= "00000001";
        index <= 0;
      elsif index = 7 then
        count <= "00000001";
        index <= 0;
      else
        count(index) <= '0';
        count(index + 1) <= '1';
        index <= index + 1;
      end if;
    end if;
  end process;
end;
```

Are you confident about this change? (select one to recompile)

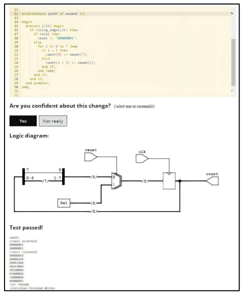
Yes | Not really

Logic diagram:

Test passed!

```
count:
(reset asserted)
00000001
00000001
(reset released)
00000010
00000100
00001000
00010000
00100000
01000000
10000000
00000001
TEST PASSED.
simulation finished @115ns
```

Index-variable Method



```
architecture synth of onehot is

begin
  process (clk) begin
    if rising_edge(clk) then
      if reset then
        count <= "00000001";
      else
        for i in 0 to 7 loop
          if i = 7 then
            count(0) <= count(7);
          else
            count(i + 1) <= count(i);
          end if;
        end loop;
      end if;
    end if;
  end process;
end;
```

Are you confident about this change? (select one to recompile)

Yes | Not really

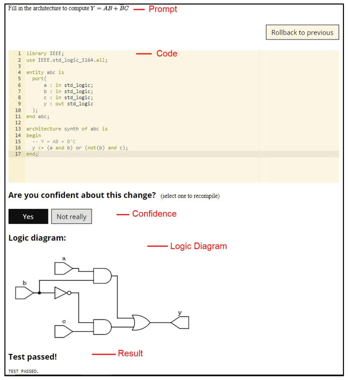
Logic diagram:

Test passed!

```
count:
(reset asserted)
00000001
00000001
(reset released)
00000010
00000100
00001000
00010000
00100000
01000000
10000000
00000001
TEST PASSED.
simulation finished @115ns
```

For-loop Method

What Data We Have



Fill in the architecture to compute $Y = AB + \overline{B}C$ — Prompt

Rollback to previous

— Code

```
library IEEE;
use IEEE.std_logic_1164.all;

entity abc is
  port(
      a : in std_logic;
      b : in std_logic;
      c : in std_logic;
      y : out std_logic
  );
end abc;

architecture synth of abc is
begin
  -- Y = AB + B'C
  y <= (a and b) or (not(b) and c);
end;
```

Are you confident about this change? (select one to recompile)

Yes | Not really — Confidence

Logic diagram: — Logic Diagram

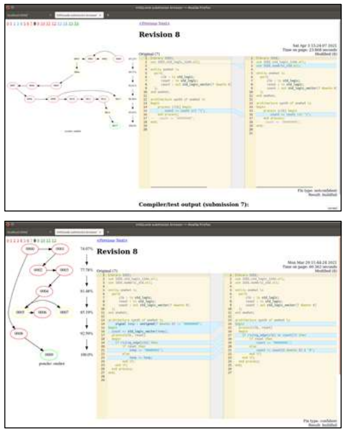
Test passed! — Result

```
TEST PASSED.
```

Also, have date and time when code was submitted.

Drawing Solution Trajectories

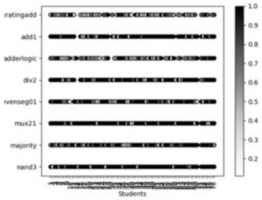
- Goal of connecting a student's submissions based on their different approaches to solving a problem.
- Use text-based comparison of code.
- Find first passing submission and compare all previous submissions to it to indicate progress.
- Submissions equally like the final submission have the same rank, and submissions that are identical share the same node.
- Edge between current submission and most similar previous submission.



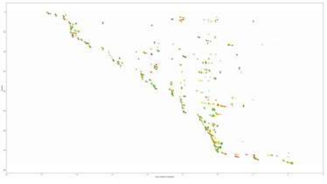
Sample Student Solution Trajectories along with Diffviewer Tool

Solution Trajectory Metrics

- Use Dijkstra's algorithm to determine the shortest path from initial submission to final submission, which suggests the percentage of submissions that contributed to the student's final solution.
- Timestamps can reveal trends about when students typically submit their assignments.



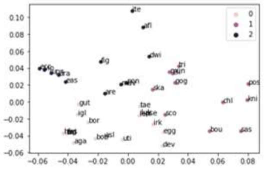
Percentages for a Selection of Problems



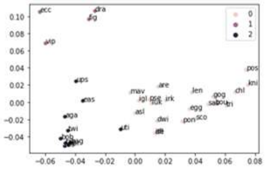
Timeline of when Students Submitted Their Code Relative to the Deadline

Grouping Students by Code:

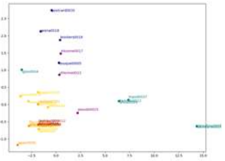
- Goal of grouping students based on similarities in coding process and solution.
- Affinity matrix calculated with two methods: using Wasserstein metric from optimal transport between different students' code submissions and computing similarity between students' solutions to a problem.
- Spectral Embedding performed on affinity matrix and then k-means clustering used.
- Alternate approach: counting occurrences of keywords in solutions and use these totals as features in k-means clustering.



Optimal-Transport Method



Similarity Method



Bag-of-Keywords Method