DLUnit:
A Unit Testing Framework for Simulated Digital Logic Circuits

Zachary Kurmas
kurmasz@gvsu.edu

GRAND VALLEY STATE UNIVERSITY®

DLUnit

- Unit test framework for simulated digital logic circuits
  - Original motivation to help with grading
- But first …

`http://kurmasgvsu.github.io/Software/`

GRAND VALLEY STATE UNIVERSITY

DLUnit

- Unit test framework for simulated digital logic circuits
  - Original motivation to help with grading
- But first …

http://kurmasgvsu.github.io/Software/

GRAND VALLEY STATE UNIVERSITY

Testing Across the Curriculum

Testing Across the Curriculum

- Writing Across the Curriculum

Testing Across the Curriculum

- Writing Across the Curriculum
    - Write in as many classes as reasonable

GRAND VALLEY STATE UNIVERSITY

Testing Across the Curriculum

- Writing Across the Curriculum
  - Write in as many classes as reasonable
    - (sometimes formal instruction; sometimes just “practice”)

Testing Across the Curriculum

- Writing Across the Curriculum
  - Write in as many classes as reasonable
    - (sometimes formal instruction; sometimes just “practice”)
  - GVSU still does this: Supplemental Writing Skills (SWS) courses

Testing Across the Curriculum

- Writing Across the Curriculum
  - Write in as many classes as reasonable
    - (sometimes formal instruction; sometimes just “practice”)
  - GVSU still does this: Supplemental Writing Skills (SWS) courses
  - Why?

Testing Across the Curriculum

- Writing Across the Curriculum
  - Write in as many classes as reasonable
    - (sometimes formal instruction; sometimes just “practice”)
  - GVSU still does this: Supplemental Writing Skills (SWS) courses
  - Why?
    - Writing is a skill (not a set of facts)

GRAND VALLEY STATE UNIVERSITY

Testing Across the Curriculum

- Writing Across the Curriculum
  - Write in as many classes as reasonable
    - (sometimes formal instruction; sometimes just “practice”)
  - GVSU still does this: Supplemental Writing Skills (SWS) courses
  - Why?
    - Writing is a skill (not a set of facts)
    - There is no substitute for practice

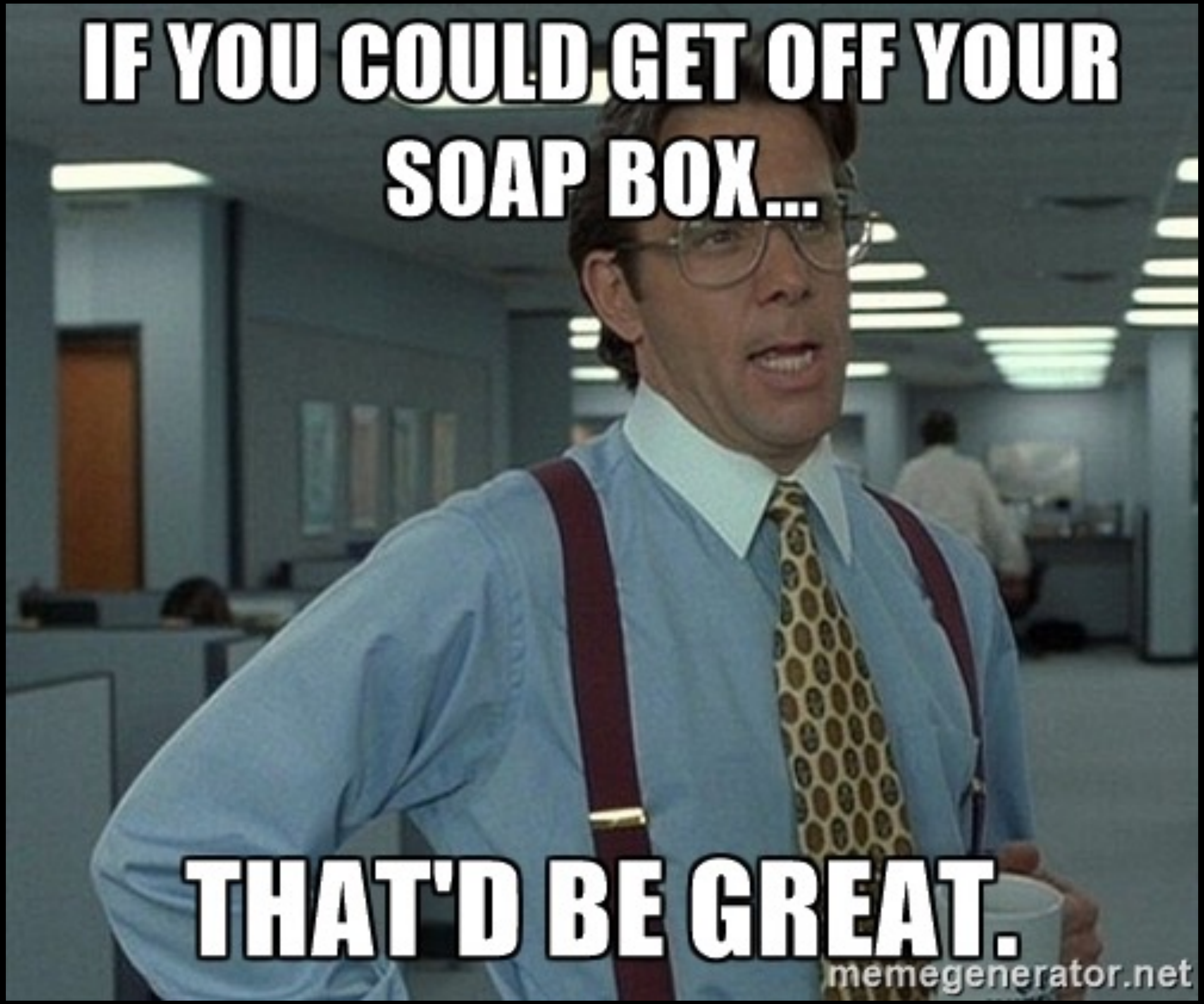
Testing Across the Curriculum

- Writing Across the Curriculum
  - Write in as many classes as reasonable
    - (sometimes formal instruction; sometimes just "practice")
  - GVSU still does this: Supplemental Writing Skills (SWS) courses
  - Why?
    - Testing is a skill (not a set of facts)
    - There is no substitute for practice

GRAND VALLEY STATE UNIVERSITY

Testing Across the Curriculum

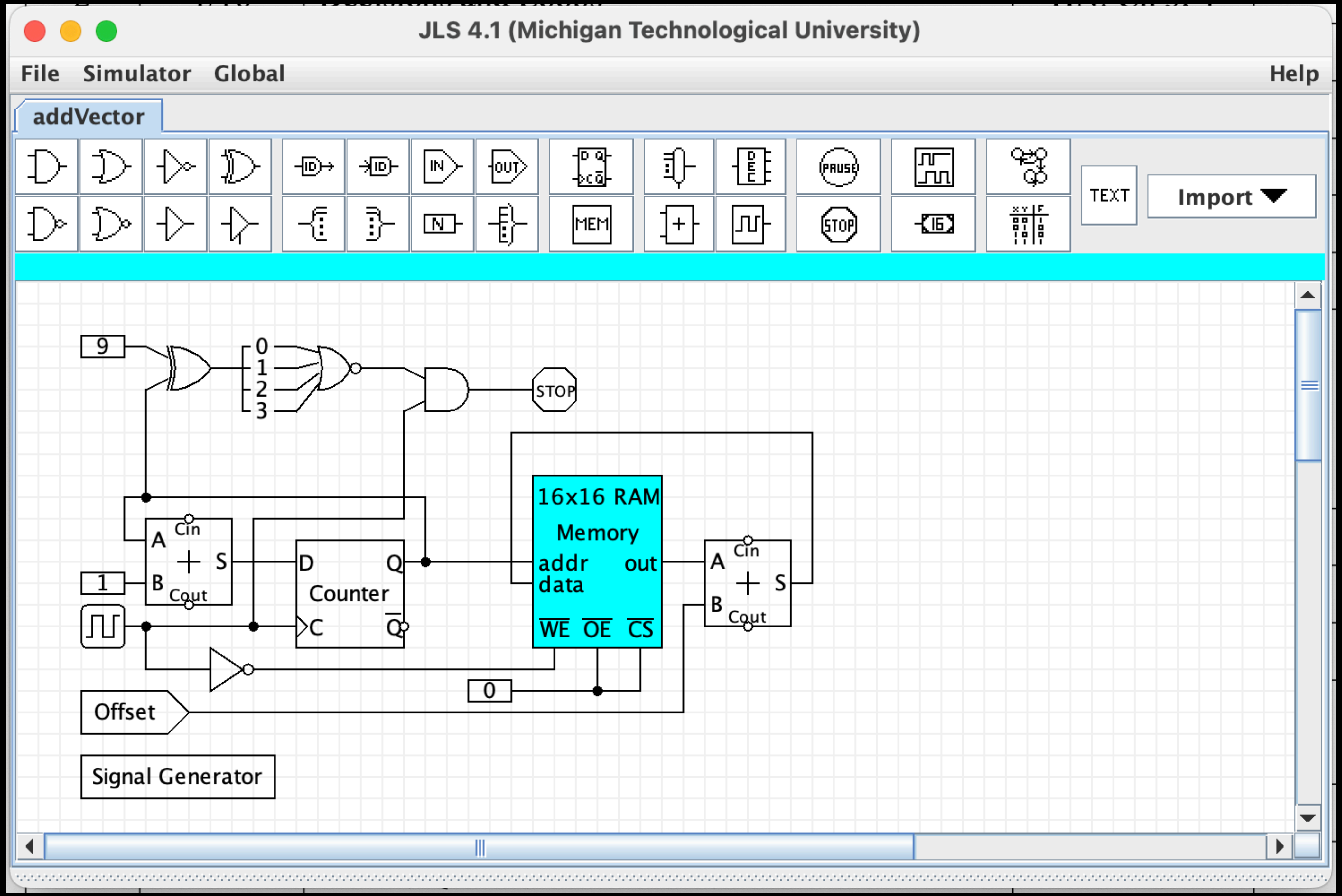
- Writing Across the Curriculum
  - Write in as many classes as reasonable
    - (sometimes formal instruction; sometimes just "practice")
  - GVSU still does this: Supplemental Writing Skills (SWS) courses
  - Why?
    - Testing is a skill (not a set of facts)
    - There is no substitute for practice
      - This means writing tests in more classes than CS 1, CS 2, and Software Engineering

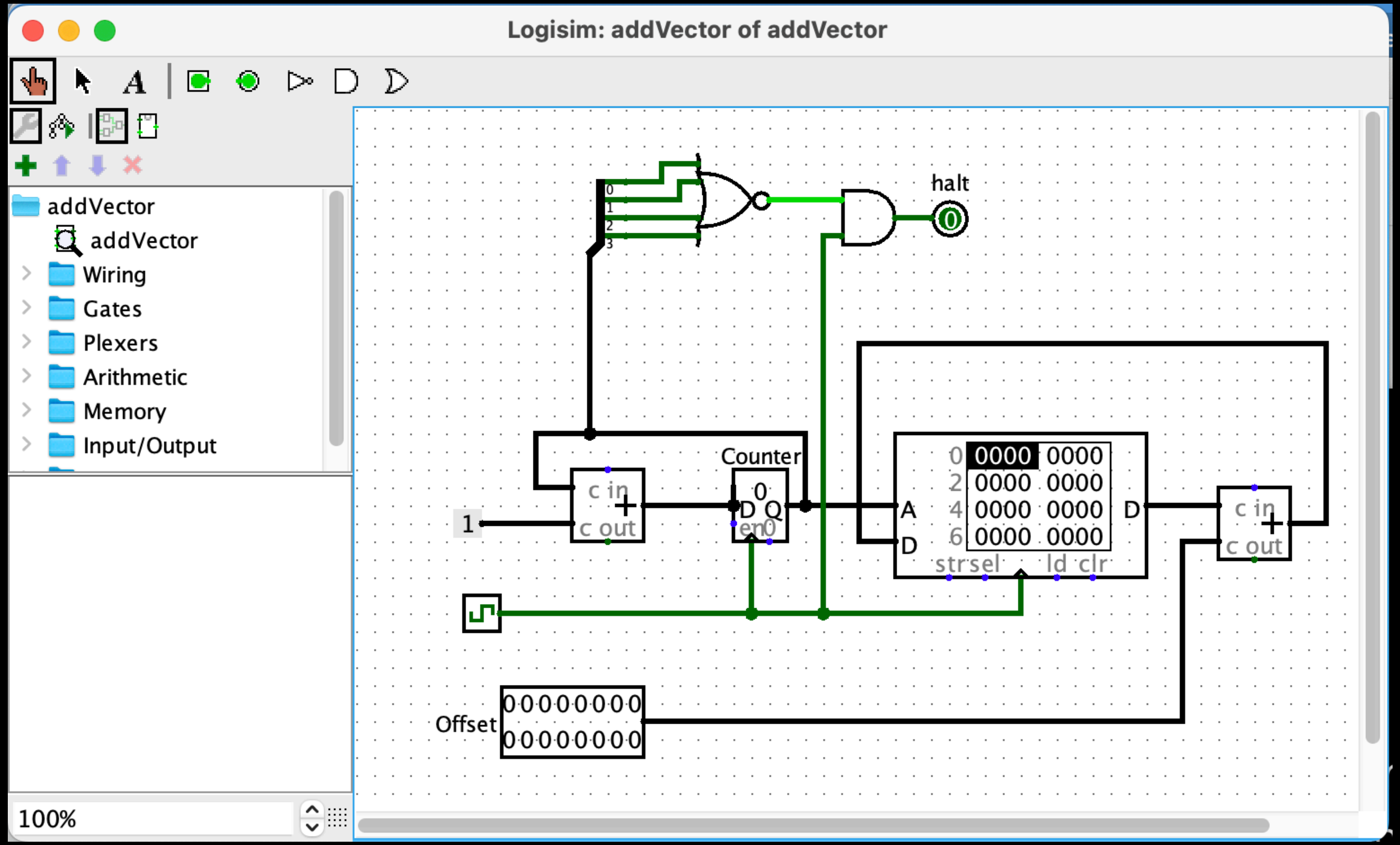


GRAND VALLEY STATE UNIVERSITY

DLUnit

- Just an interface to the JLS and Logisim simulators
- Contains functions to
  - set inputs pins and memory
  - launch the simulation
  - read final state of output pins / memory
- Verify expectations using JUnit assertions

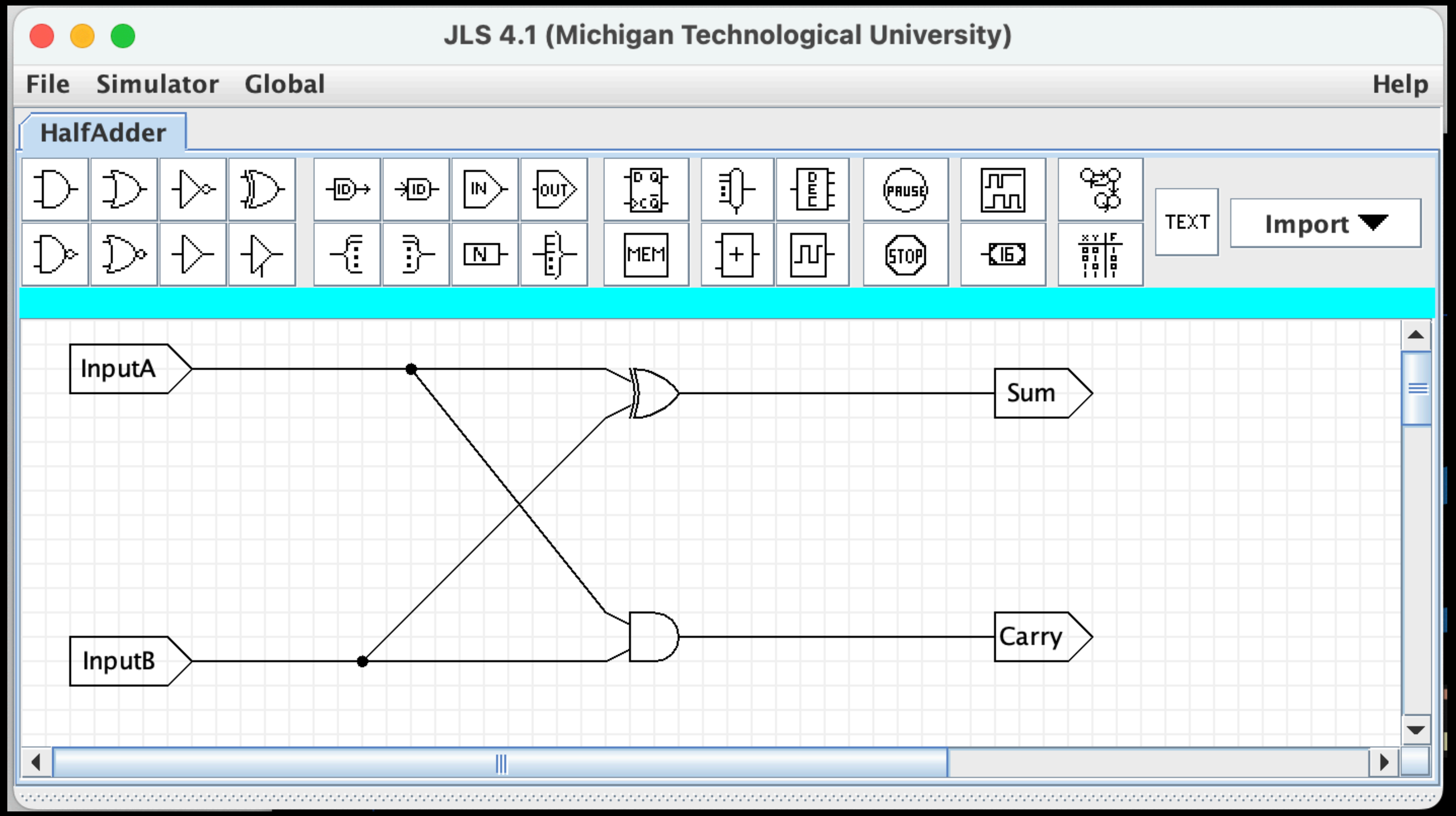




Sample DLUnit Test

```
import static edu.gvsu.dlunit.DLUnit.*;
import org.junit.Assert;
import org.junit.Test;

public class HalfAdderTest {
  @Test
  public void zero_zero() {
    setPin("InputA", false);
    setPin("InputB", false);
    run();
    Assert.assertEquals("Checking Sum", false, readPin("Sum"));
    Assert.assertEquals("Checking Carry", false, readPin("Carry"));
  }
```

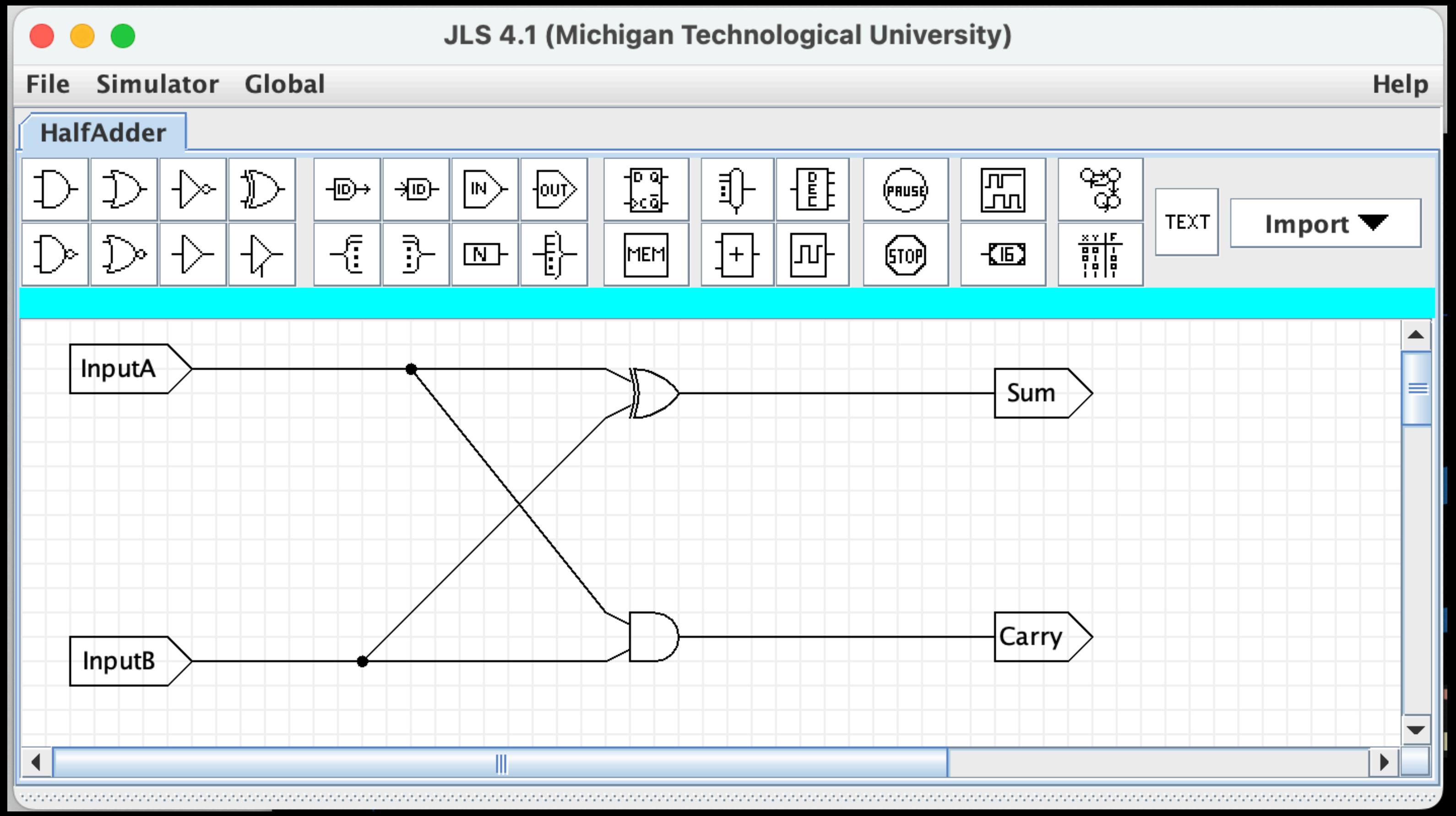


Sample DLUnit Test

```
import static edu.gvsu.dlunit.DLUnit.*;
import org.junit.Assert;
import org.junit.Test;

public class HalfAdderTest {
  @Test
  public void zero_zero() {
    setPin("InputA", false);
    setPin("InputB", false);
    run();
    Assert.assertEquals("Checking Sum", false, readPin("Sum"));
    Assert.assertEquals("Checking Carry", false, readPin("Carry"));
  }
```

- Set both 1-bit pins to logic 0
- `setPin` is a static method to minimize
  1. explicit setup
  2. verbosity of code



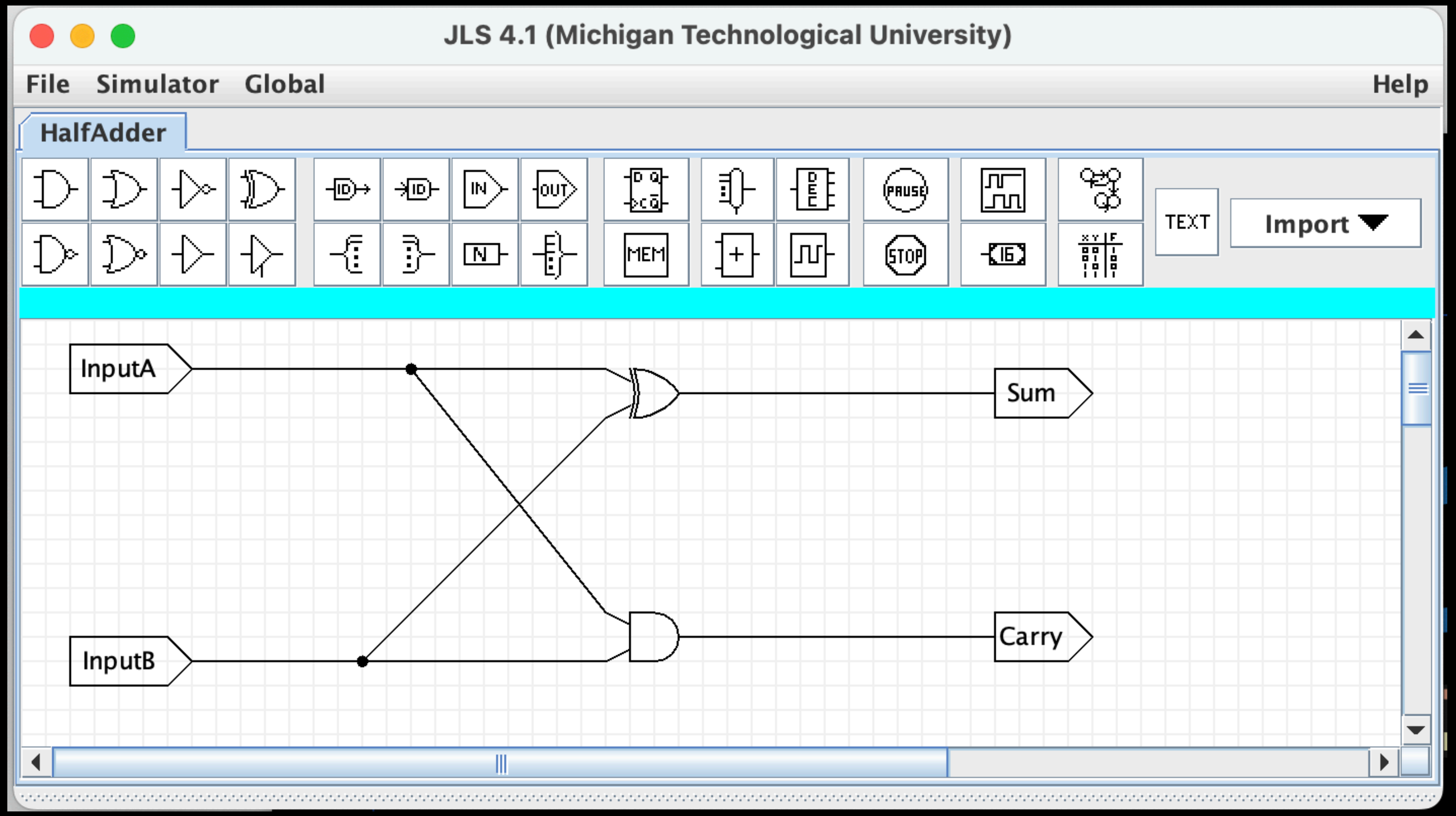
Sample DLUnit Test

```
import static edu.gvsu.dlunit.DLUnit.*;
import org.junit.Assert;
import org.junit.Test;

public class HalfAdderTest {
  @Test
  public void zero_zero() {
    setPin("InputA", false);
    setPin("InputB", false);
    run();
    Assert.assertEquals("Checking Sum", false, readPin("Sum"));
    Assert.assertEquals("Checking Carry", false, readPin("Carry"));
  }
```

- Set both 1-bit pins to logic 0
- `setPin` is a static method to minimize
  1. explicit setup
  2. verbosity of code

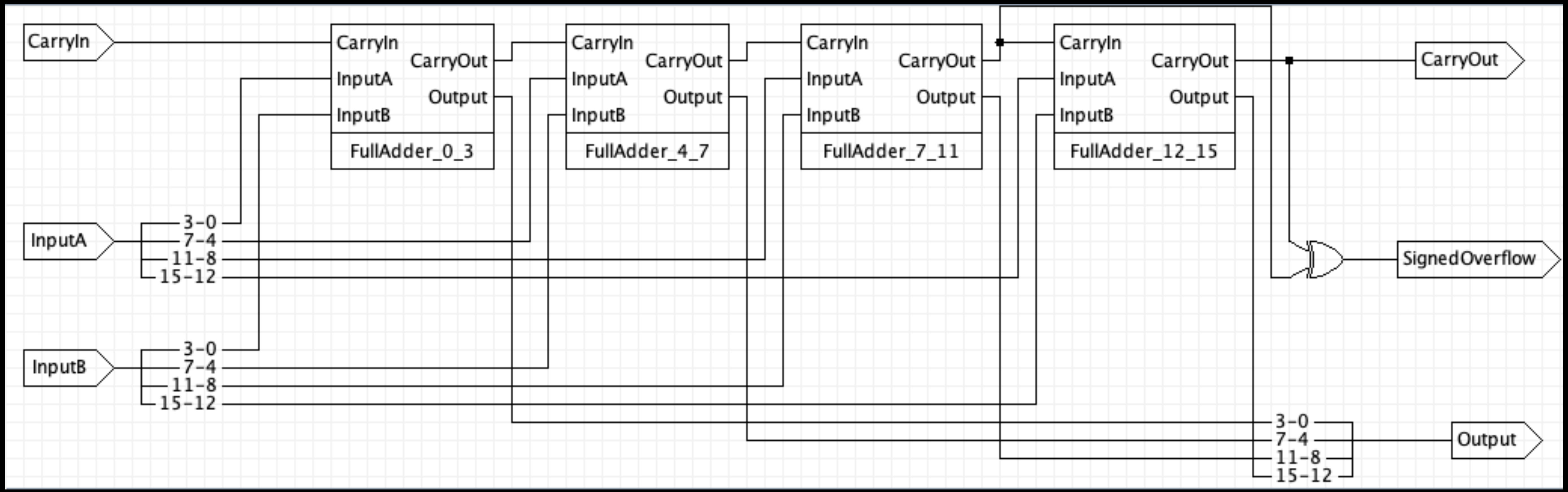
Uses JUnit's built-in assert mechanism


JLS 4.1 (Michigan Technological University)
File Simulator Global Help
HalfAdder
TEXT
Import
InputA
InputB
Sum
Carry

Multi-pin Input

```
import static edu.gvsu.dlunit.DLUnit.*;
import org.junit.Assert;
import org.junit.Test;

public class HalfAdderTest {
  @Test
  public void zero_zero() {
    setPin("InputA", 45);
    setPin("InputB", 37);
    setPin("CarryIn", 0);
    run();
    Assert.assertEquals(82, readPin("Output"));
    Assert.assertEquals(false, readPin("SignedOverflow"));
  }
```

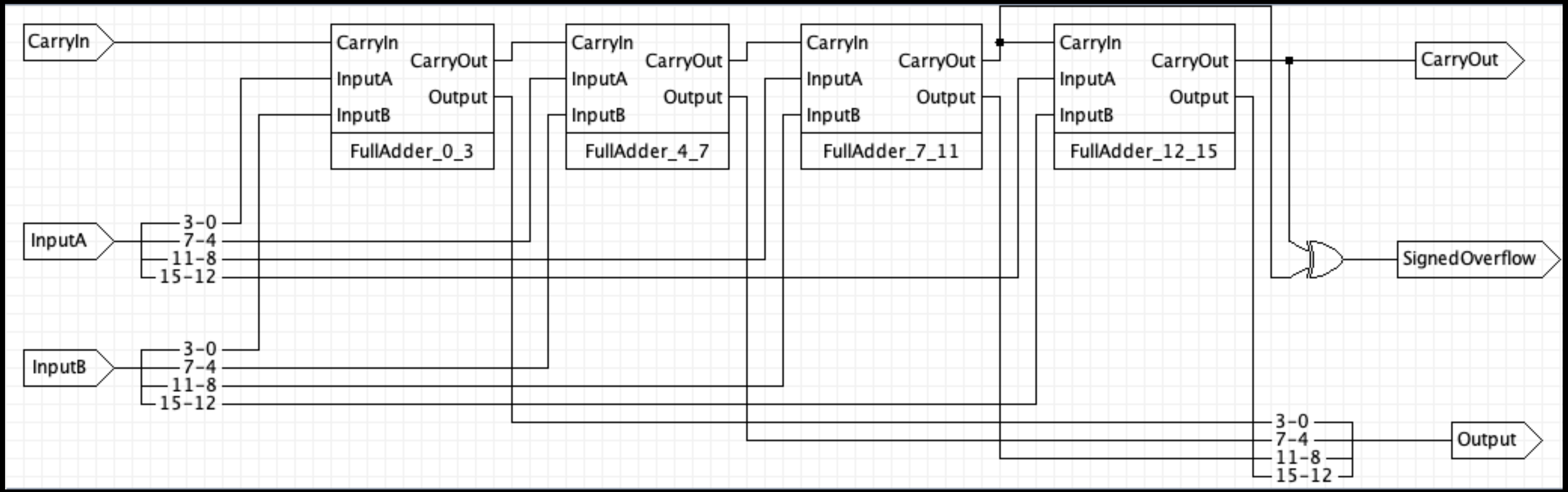


Multi-pin Input

```
import static edu.gvsu.dlunit.DLUnit.*;
import org.junit.Assert;
import org.junit.Test;

public class HalfAdderTest {
  @Test
  public void zero_zero() {
    setPin("InputA", 45);
    setPin("InputB", 37);
    setPin("CarryIn", 0);
    run();
    Assert.assertEquals(82, readPin("Output"));
    Assert.assertEquals(false, readPin("SignedOverflow"));
  }
```

- Set both 16-bit pins to bit pattern implied by integer



Thorough test using helpers

```java
private static final long testIntegers[] = {0, 1, 2, 13, 127, 128, 129, 0xAAAA, 65534, 65535};

  protected void verify(long a, long b, boolean carryIn) {

    long carryInAsInt = (carryIn ? 1 : 0);
    long expected = a + b + carryInAsInt;
    boolean expectedOverflow = expected > 65535;

    setPinUnsigned("InputA", a);
    setPinUnsigned("InputB", b);
    setPin("CarryIn", carryIn);
    run();
    String message = "of " + a + " + " + b + " with " + (carryIn ? "a " : " no ") + " carry in";

    // Output "wraps around" if there is an overflow
    Assert.assertEquals("Output " + message, (expected % 65536), readPinUnsigned("Output"));
    Assert.assertEquals("CarryOut " + message, expectedOverflow, readPin("CarryOut"));
  }

  @Test
  public void testAll() {
    int count = 0;
    for (long a : testIntegers) {
      for (long b : testIntegers) {
        verify(a, b, false);
        verify(a, b, true);
        count += 2;
      }
    }
    System.out.println("testAll ran " + count + " tests.");
  } // end testAll
```

Thorough test using helpers

```
private static final long testIntegers[] = {0, 1, 2, 13, 127, 128, 129, 0xAAAA, 65534, 65535};

  protected void verify(long a, long b, boolean carryIn) {

    long carryInAsInt = (carryIn ? 1 : 0);
    long expected = a + b + carryInAsInt;
    boolean expectedOverflow = expected > 65535;

    setPinUnsigned("InputA", a);
    setPinUnsigned("InputB", b);
    setPin("CarryIn", carryIn);
    run();
    String message = "of " + a + " + " + b + " with " + (carryIn ? "a " : " no ") + " carry in";

    // Output "wraps around" if there is an overflow
    Assert.assertEquals("Output " + message, (expected % 65536), readPinUnsigned("Output"));
    Assert.assertEquals("CarryOut " + message, expectedOverflow, readPin("CarryOut"));
  }

  @Test
  public void testAll() {
    int count = 0;
    for (long a : testIntegers) {
      for (long b : testIntegers) {
        verify(a, b, false);
        verify(a, b, true);
        count += 2;
      }
    }
    System.out.println("testAll ran " + count + " tests.");
  } // end testAll
```

- Compute the expected output based on the inputs
  - (as opposed to typing them all out by hand.)

Thorough test using helpers

```java
private static final long testIntegers[] = {0, 1, 2, 13, 127, 128, 129, 0xAAAA, 65534, 65535};

  protected void verify(long a, long b, boolean carryIn) {

    long carryInAsInt = (carryIn ? 1 : 0);
    long expected = a + b + carryInAsInt;
    boolean expectedOverflow = expected > 65535;

    setPinUnsigned("InputA", a);
    setPinUnsigned("InputB", b);
    setPin("CarryIn", carryIn);
    run();
    String message = "of " + a + " + " + b + " with " + (carryIn ? "a " : " no ") + " carry in";

    // Output "wraps around" if there is an overflow
    Assert.assertEquals("Output " + message, (expected % 65536), readPinUnsigned("Output"));
    Assert.assertEquals("CarryOut " + message, expectedOverflow, readPin("CarryOut"));
  }

  @Test
  public void testAll() {
    int count = 0;
    for (long a : testIntegers) {
      for (long b : testIntegers) {
        verify(a, b, false);
        verify(a, b, true);
        count += 2;
      }
    }
    System.out.println("testAll ran " + count + " tests.");
  } // end testAll
```

- Compute the expected output based on the inputs
  - (as opposed to typing them all out by hand.)

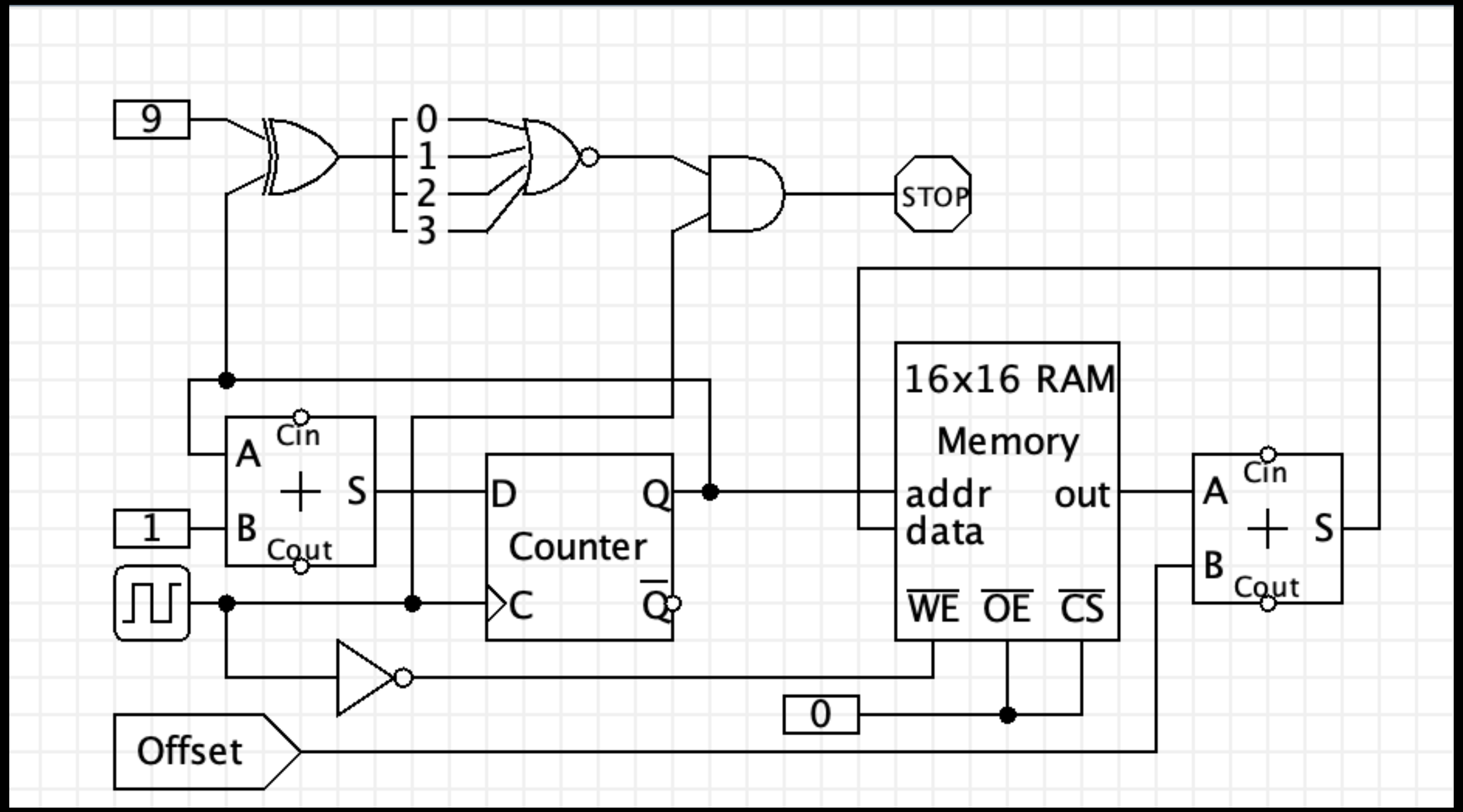
Unconventional practice in software development; but, effective when testing student circuits.

GRAND VALLEY STATE UNIVERSITY

Set/Test Memory

```java
import static edu.gvsu.dlunit.DLUnit.*;
import org.junit.Assert;
import org.junit.Test;

public class AddVectorTest {
  @Test
  public void add5() {
    setPinUnsigned("Offset", 5);
    setRegisterUnsigned("Counter", 3);
    setMemoryUnsigned("Memory", 0, new int[]{0, 10, 20, 30, 40, 50, 60, 70});
    run();
    Assert.assertEquals(35, readMemorySigned("Memory", 3));
    Assert.assertEquals(new long[]{25, 35, 45, 55}, readMemorySigned("Memory", 2, 4));
  }
}
```

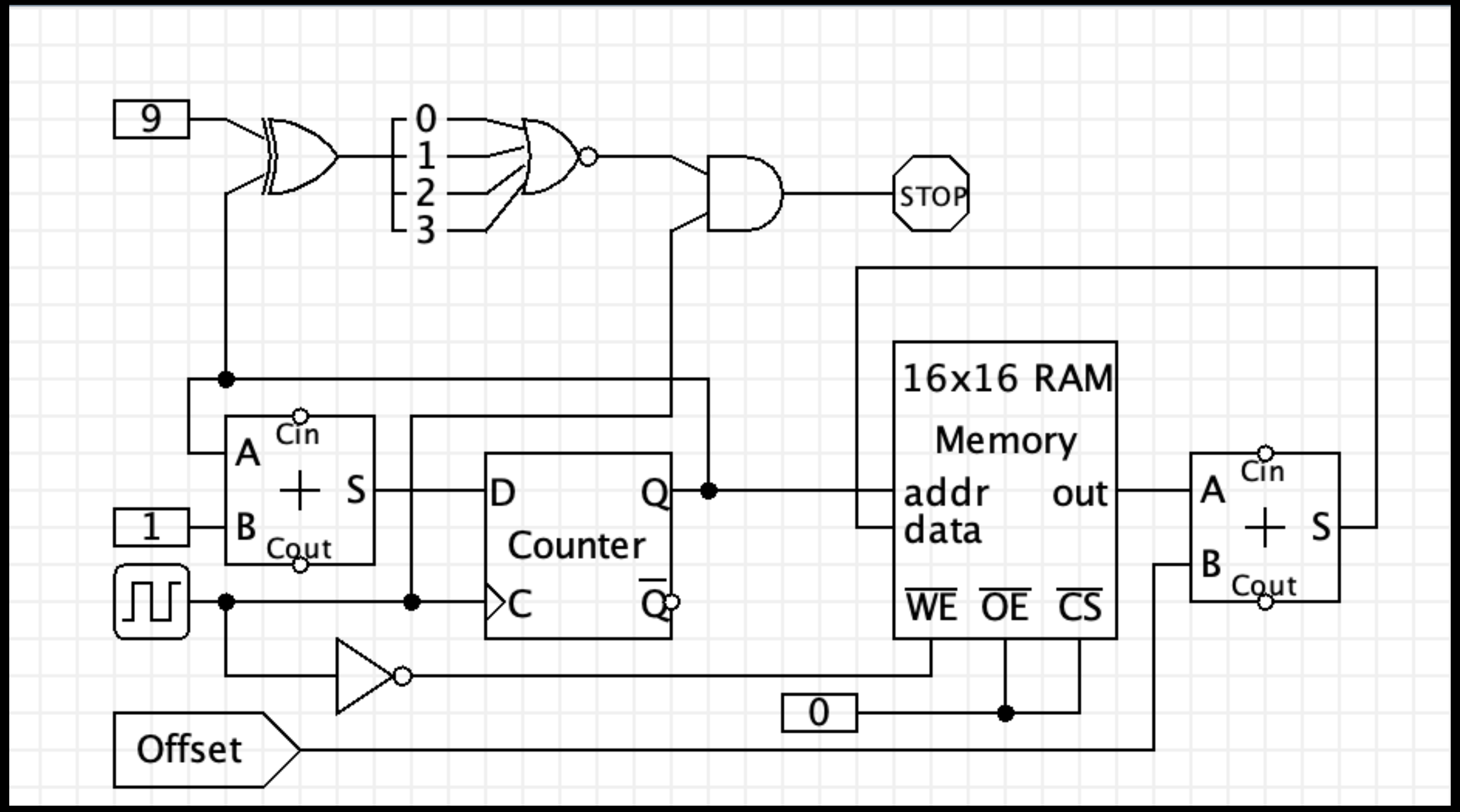


Set/Test Memory

Initialize a register

```
import static edu.gvsu.dlunit.DLUnit.*;
import org.junit.Assert;
import org.junit.Test;

public class AddVectorTest {
  @Test
  public void add5() {
    setPinUnsigned("Offset", 5);
    setRegisterUnsigned("Counter", 3);
    setMemoryUnsigned("Memory", 0, new int[]{0, 10, 20, 30, 40, 50, 60, 70});
    run();
    Assert.assertEquals(35, readMemorySigned("Memory", 3));
    Assert.assertEquals(new long[]{25, 35, 45, 55}, readMemorySigned("Memory", 2, 4));
  }
}
```



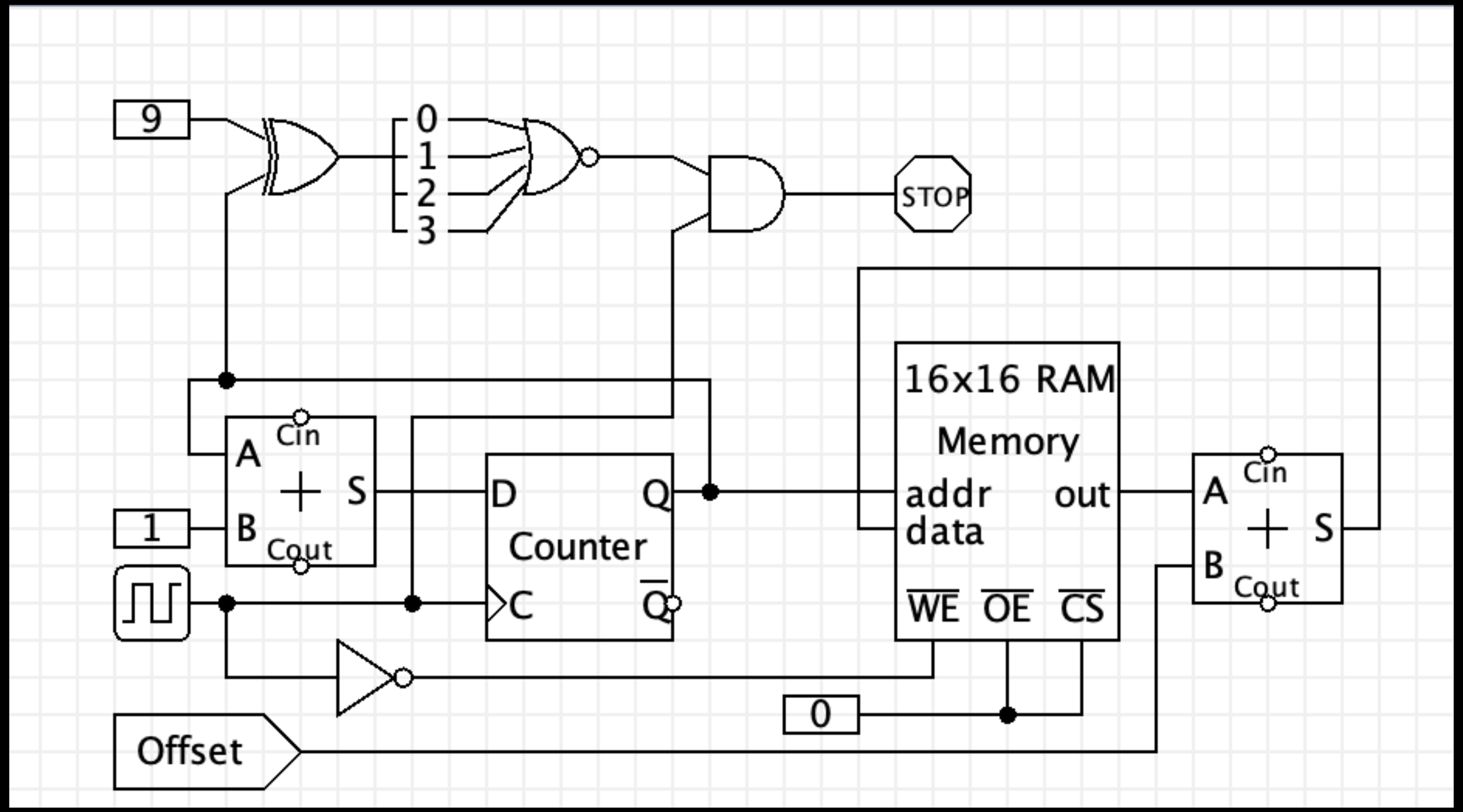
Set/Test Memory

Initialize a register

Initialize memory

```java
import static edu.gvsu.dlunit.DLUnit.*;
import org.junit.Assert;
import org.junit.Test;

public class AddVectorTest {
  @Test
  public void add5() {
    setPinUnsigned("Offset", 5);
    setRegisterUnsigned("Counter", 3);
    setMemoryUnsigned("Memory", 0, new int[]{0, 10, 20, 30, 40, 50, 60, 70});
    run();
    Assert.assertEquals(35, readMemorySigned("Memory", 3));
    Assert.assertEquals(new long[]{25, 35, 45, 55}, readMemorySigned("Memory", 2, 4));
  }
}
```



Set/Test Memory

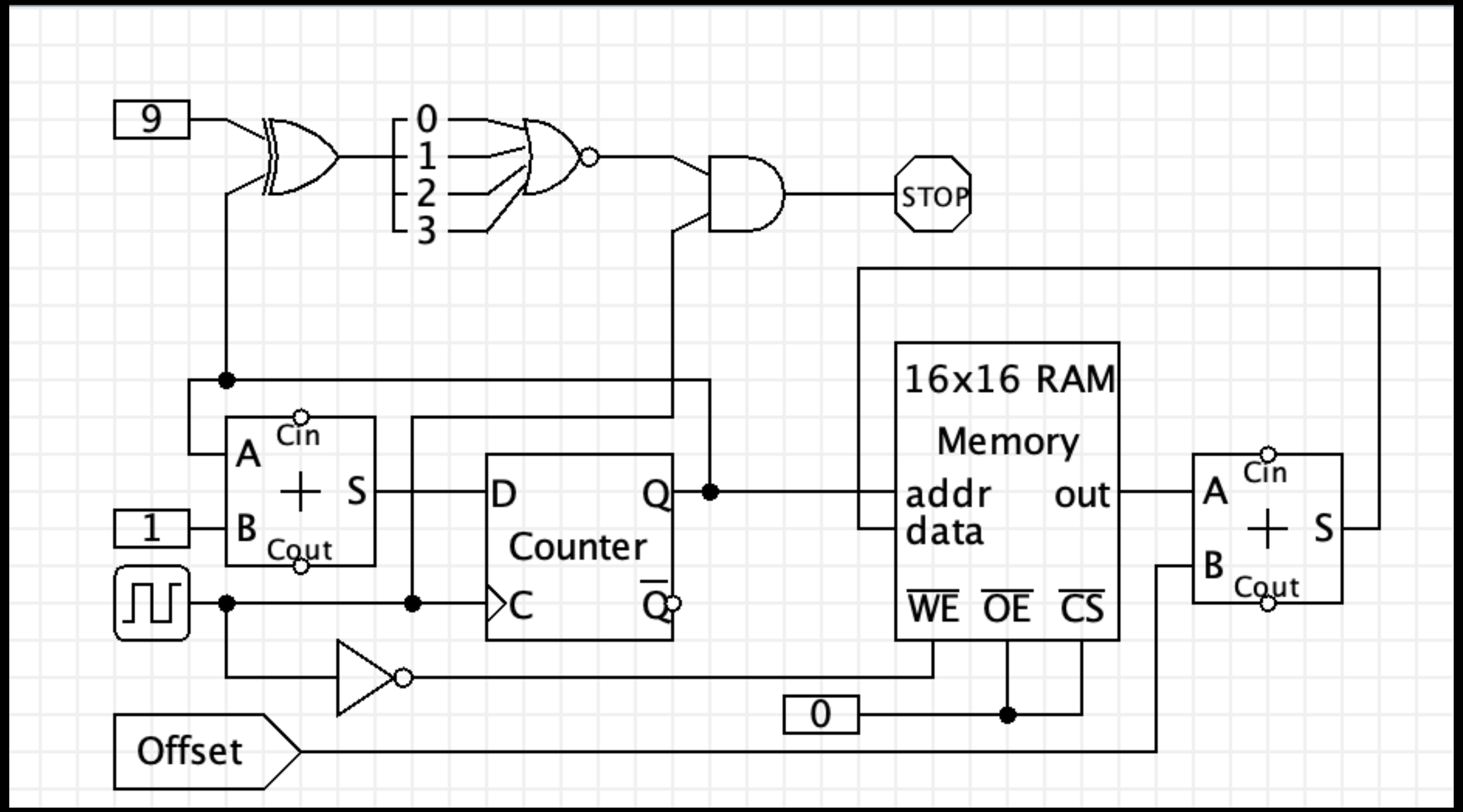
```java
import static edu.gvsu.dlunit.DLUnit.*;
import org.junit.Assert;
import org.junit.Test;

public class AddVectorTest {
  @Test
  public void add5() {
    setPinUnsigned("Offset", 5);
    setRegisterUnsigned("Counter", 3);
    setMemoryUnsigned("Memory", 0, new int[]{0, 10, 20, 30, 40, 50, 60, 70});
    run();
    Assert.assertEquals(35, readMemorySigned("Memory", 3));
    Assert.assertEquals(new long[]{25, 35, 45, 55}, readMemorySigned("Memory", 2, 4));
  }
}
```

Initialize a register

Initialize memory

Can read individual bytes, or entire sequences.



Other JUnit Features

```
 public static int[] initialState = {0x0, 0x10, 0x20, 0x30,
0x40, 0x50, 0x60, 0x70, 0x80, 0x90, 0xa0, 0xb0, 0xc0,
      0xd0, 0xe0, 0xf0};

   @Before
   public void setMemory() {
      setMemoryUnsigned("TheMemory", 0, initialState);
   }
```

Use `@Before` and `@After` to set up and tear down tests

Limitation of DLUnit

- Only works with simulators wirtten in Java
  - Excludes newer tools like CircuitVerse
  - In theory, the tool could launch an external simulator; but, re-launching the simulator for each test would be really slow.

My Approach

- Students use DLUnit to write tests “TDD Style”
  - Begin with example tests from instructor
  - Write more tests
  - If submitted code has bugs
    - Write failing test
    - Fix
    - Resubmit

Future Work

- More tools to support “low overhead” test writing in courses
  - e.g., DLUnit for digital logic simulators
- Study whether “Testing Across the Curriculum” really improves students’ abilities to test software

Summary

- DLUnit has saved me a lot of time grading
  - (not completely automated, though)
- It is a good addition to automated platforms like PrairieLearn, GitHub Classroom, etc.

`http://kurmasgvsu.github.io/Software/`

GRAND VALLEY STATE UNIVERSITY

DLUnit: A Unit Testing Framework for Simulated Digital Logic Circuits

Zachary Kurmas kurmasz@gvsu.edu http://kurmasgvsu.github.io

http://kurmasgvsu.github.io/Software/

GRAND VALLEY STATE UNIVERSITY®

Testing Across the Curriculum

- Web Programming

Testing Across the Curriculum

- Web Programming

Testing Across the Curriculum

- Web Programming
- Programming Languages

Testing Across the Curriculum

- Web Programming
- Programming Languages
- Compilers
- Artificial Intelligence

Testing Across the Curriculum

- Web Programming
- Programming Languages
- Compilers
- Artificial Intelligence

GRAND VALLEY STATE UNIVERSITY

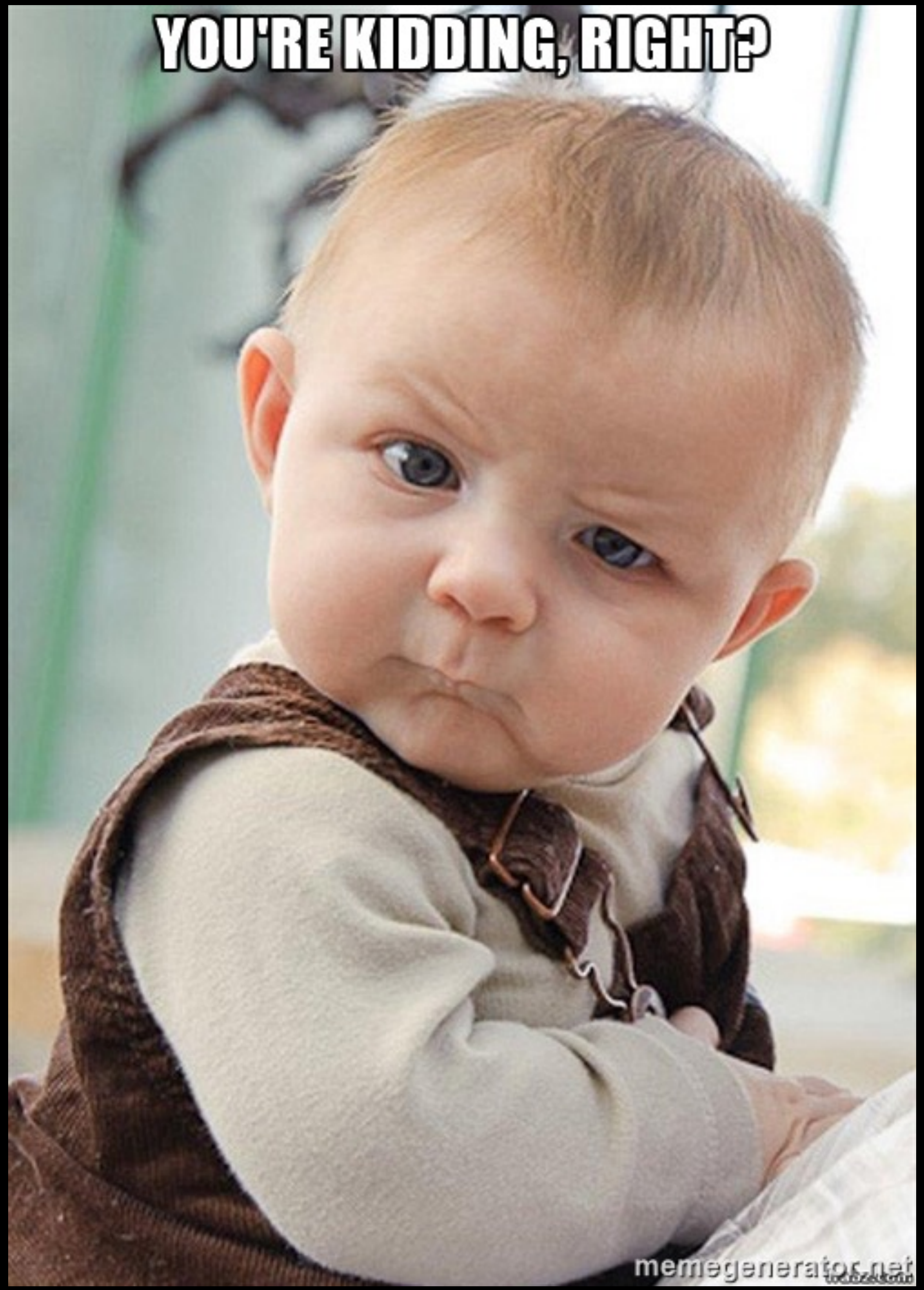
Testing Across the Curriculum

- Web Programming
- Programming Languages
- Compilers
- Artificial Intelligence
- Networking
- Operating Systems
- Database

Grand Valley State University

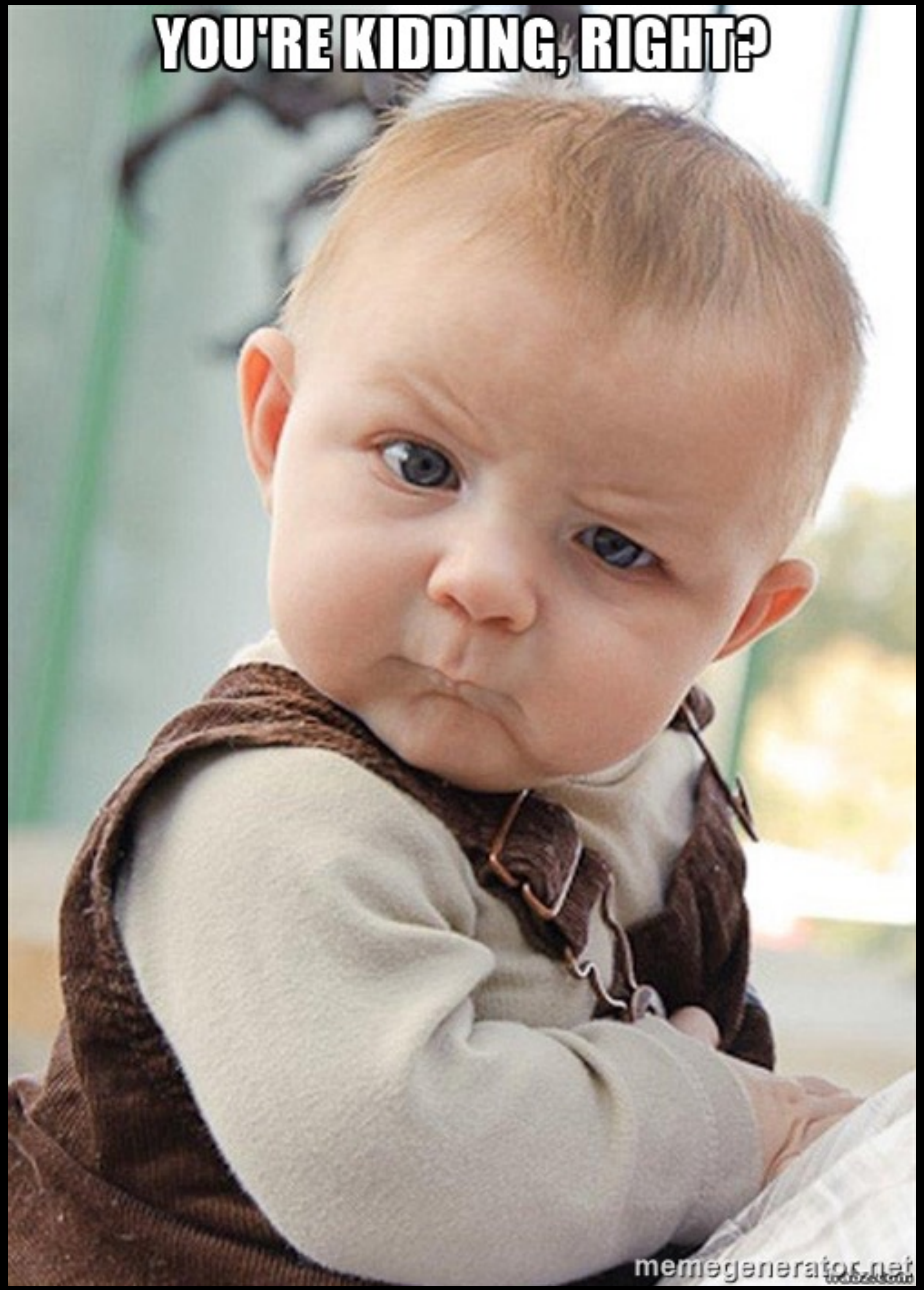
Testing Across the Curriculum

- Web Programming
- Programming Languages
- Compilers
- Artificial Intelligence
- Networking
- Operating Systems
- Database
- Graphics
- Computer Organization/Architecture



Testing Across the Curriculum

- Web Programming
- Programming Languages
- Compilers
- Artificial Intelligence
- Networking
- Operating Systems
- Database
- Graphics
- Computer Organization/Architecture



DLUnit does this

GRAND VALLEY STATE UNIVERSITY