ScalaHDL

by Yao Li

Outline

- Current States
- Future Plan

Test Benches

- FlipFlop
- Bitonic Sort
- Arithmetic
- Comparator
- FIFO

Example: Adder

```scala
package ScalaHDLExample.Arithmetic.Add

import ScalaHDL.Core.ScalaHDL
import ScalaHDL.Core.HDLType
import ScalaHDL.Core.DataType._
import ScalaHDL.Core.DataType.Signals._
import ScalaHDL.Simulation.Simulator

trait Adder extends ScalaHDL {
  defMod.adder('clk, 'rst, 'a, 'b, 'z) {
    sync(1).add {
      when ('rst is 1) {
        'z := 0
      } .otherwise {
        'z := 'a + 'b
      }
    }
  }
}

object Main extends Adder {
  def main(args: Array[String]) {
    val clk = bool(0)
    val rst = bool(0)
    val a = signed(0, 5)
    val b = signed(0, 5)
    val z = signed(0, 6)

    println(convert('adder, clk, rst, a, b, z))
  }
}
```

ScalaHDL code

Example: Adder

to use it
in test bench

means
@ posedge clk

```scala
package ScalaHDLExample.Arithmetic.Add

import ScalaHDL.Core.ScalaHDL
import ScalaHDL.Core.HDLType
import ScalaHDL.Core.DataType._
import ScalaHDL.Core.DataType.Signals._
import ScalaHDL.Simulation.Simulator

trait Adder extends ScalaHDL {
  def adder('clk, 'rst, 'a, 'b, 'z) {
    sync(1) add {
      when ('rst is 1) {
        'z := 0
      } .otherwise {
        'z := 'a + 'b
      }
    }
  }
}

object Main extends Adder {
  def main(args: Array[String]) {
    val clk = bool(0)
    val rst = bool(0)
    val a = signed(0, 5)
    val b = signed(0, 5)
    val z = signed(0, 6)

    println(convert('adder, clk, rst, a, b, z))
  }
}
```

is instead of ==

ScalaHDL code

Example: Adder

```
module adder (
clk,
rst,
a,
b,
z
);

input [4:0] a;
input [4:0] b;
input clk;
input rst;
output [5:0] z;
reg [5:0] z;

always @(posedge clk) begin: _add
if (rst == 1) begin
z <= 0;
end
else begin
z <= a + b;
end
end


endmodule
```

generated Verilog code

Example: Adder

```scala
import org.scalatest.FunSuite

import ScalaHDLExample.Arithmetic.Add.Adder
import ScalaHDL.Core.DataType._
import ScalaHDL.Core.DataType.Signals._
import ScalaHDL.Test.TestHelper

object AdderTestBench extends Adder {
  val A = List(0, 0, 1, 1, 15, 0, 15).iterator
  val B = List(0, 1, 0, 1, 0, 15, 15).iterator
  defMod.Bench('clk, 'rst, 'a, 'b, 'z) {
    delay(1) {
      cycle('clk)
    }

    sync(0) {
      'rst := 0
      'a := A.next()
      'b := B.next()
    }
  }
}

class AdderTest extends FunSuite with TestHelper {
  test("test adder") {
    val clk = bool(0)
    val rst = bool(1)
    val a = unsigned(0, 4)
    val b = unsigned(0, 4)
    val z = unsigned(0, 5)

    val Z = List(0, 1, 1, 2, 15, 15, 30).iterator

    val sim = Simulator(AdderTestBench,
      module('adder, clk, rst, a, b, z),
      module('Bench, clk, rst, a, b, z))
    sim.setTrace("add.vcd")
    sim since 0 until 14 every 2 run {
      assert(clk === 0)
    }
    sim since 1 until 14 every 2 run {
      assert(clk === 1)
      assert(z.value === Z.next)
    }
    sim test
  }
}
```

Test Bench

Example: Adder

```
1  import org.scalatest.FunSuite
2
3  import ScalaHDLExample.Arithmetic.Add.Adder
4  import ScalaHDL.Core.DataType._
5  import ScalaHDL.Core.DataType.Signals._
6  import ScalaHDL.Test.TestHelper
7
8  object AdderTestBench extends Adder {
9    val A = List(0, 0, 1, 1, 15, 0, 15).iterator
10   val B = List(0, 1, 0, 1, 0, 15, 15).iterator
11   defMod.Bench('clk, 'rst, 'a, 'b, 'z) {
12     delay(1) {
13       cycle('clk)
14     }
15
16     sync(0) {
17       'rst := 0
18       'a := A.next()
19       'b := B.next()
20     }
21   }
22 }
23
24 class AdderTest extends FunSuite with TestHelper {
25   test("test adder") {
26     val clk = bool(0)
27     val rst = bool(1)
28     val a = unsigned(0, 4)
29     val b = unsigned(0, 4)
30     val z = unsigned(0, 5)
31
32     val Z = List(0, 1, 1, 2, 15, 15, 30).iterator
33
34     val sim = Simulator(AdderTestBench,
35       module('adder, clk, rst, a, b, z),
36       module('Bench, clk, rst, a, b, z))
37     sim.setTrace("add.vcd")
38     sim since 0 until 14 every 2 run {
39       assert(clk === 0)
40     }
41     sim since 1 until 14 every 2 run {
42       assert(clk === 1)
43       assert(z.value === Z.next)
44     }
45     sim test
46   }
47 }
```

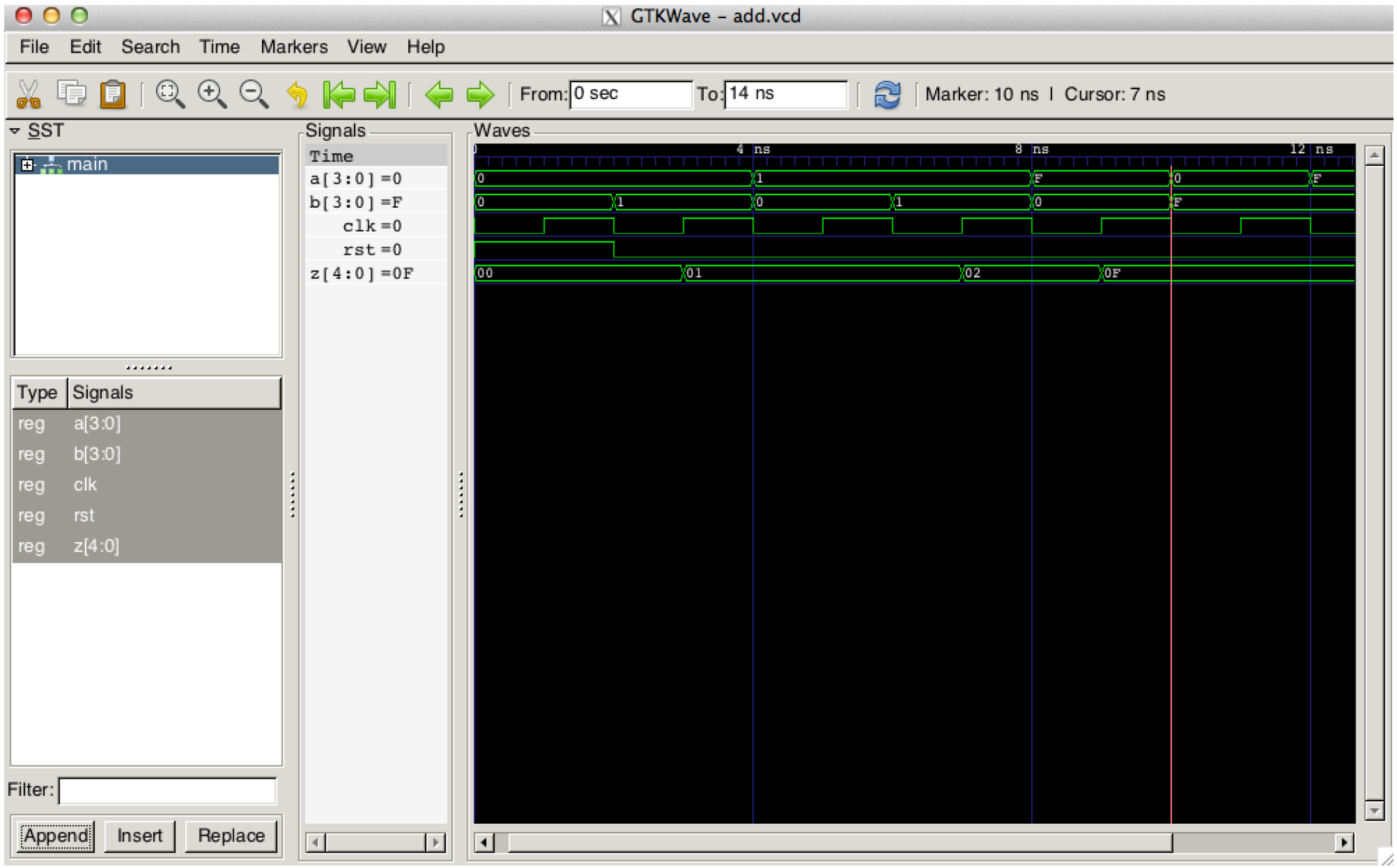
so we can use “next” in line 18

FunSuite is a Scala test suite, TestHelper is a ScalaHDL test helper

some syntax sugar to assert

Test Bench

Example: Adder



generated .vcd file viewed by GTKWave

Examples

```
> test
[info] CompTest:
[info] - test comparator
[info] AdderTest:
[info] - test adder
[info] SubtractorTest:
[info] - test subtractor
[info] FifoTest:
[info] - test fifo
[info] MultiplierTest:
[info] - test multiplier
[info] FlipFlopTest:
[info] - test dff 1
[info] - test dff 2
[info] BitonicSortTest:
[info] - test bitonic sort
[info] Passed: Total 8, Failed 0, Errors 0, Passed 8
[success] Total time: 0 s, completed 2014-2-11 21:43:46
>
```

use **`sbt`** to run test benches

Example: FIFO

```scala
package ScalaHDLExample.FIFO

import ScalaHDL.Core.ScalaHDL
import ScalaHDL.Core.HDLType
import ScalaHDL.Core.DataType._
import ScalaHDL.Core.DataType.Signals._
import ScalaHDL.Simulation.Simulator

trait FIFO extends ScalaHDL {

  val WIDTH = 5
  val DEPTH = 5

  defMod.fifo('clk, 'rst, 'input, 'output) {
    sync(1).fifo {
      val fifo_registers =
        (for (i <- 0 until DEPTH) yield unsigned(0, WIDTH)).map(toHDLType)

      when ('rst is 1) {
        'output := 0
        for (i <- 0 until DEPTH)
          fifo_registers(i) := 0
      } .otherwise {
        fifo_registers(0) := 'input
        'output := fifo_registers(DEPTH - 1)
        if (DEPTH > 1) {
          for (i <- 0 until DEPTH - 1)
            fifo_registers(i + 1) := fifo_registers(i)
        }
      }
    }
  }
}

object Main extends FIFO {
  def main(args: Array[String]) {
    val clk = bool(0)
    val rst = bool(0)
    val input = unsigned(0, WIDTH)
    val output = unsigned(0, WIDTH)

    println(convert('fifo, clk, rst, input, output))
  }
}
```

ScalaHDL code

Example: FIFO

```scala
package ScalaHDLExample.FIFO

import ScalaHDL.Core.ScalaHDL
import ScalaHDL.Core.HDLType
import ScalaHDL.Core.DataType._
import ScalaHDL.Core.DataType.Signals._
import ScalaHDL.Simulation.Simulator

trait FIFO extends ScalaHDL {

  val WIDTH = 5
  val DEPTH = 5

  defMod.fifo('clk, 'rst, 'input, 'output) {
    sync(1).fifo {
      val fifo_registers =
        (for (i <- 0 until DEPTH) yield unsigned(0, WIDTH)).map(toHDLType)

      when ('rst is 1) {
        'output := 0
        for (i <- 0 until DEPTH)
          fifo_registers(i) := 0
      } .otherwise {
        fifo_registers(0) := 'input
        'output := fifo_registers(DEPTH - 1)
        if (DEPTH > 1) {
          for (i <- 0 until DEPTH - 1)
            fifo_registers(i + 1) := fifo_registers(i)
        }
      }
    }
  }
}

object Main extends FIFO {
  def main(args: Array[String]) {
    val clk = bool(0)
    val rst = bool(0)
    val input = unsigned(0, WIDTH)
    val output = unsigned(0, WIDTH)

    println(convert('fifo, clk, rst, input, output))
  }
}
```

declare internal registers

normal Scala for comprehension

ScalaHDL code

Example: FIFO

```
module fifo (
clk,
rst,
input,
output
);

input [4:0] input;
input clk;
input rst;
output [4:0] output;
reg [4:0] output;
reg [4:0] tmp_0;
reg [4:0] tmp_1;
reg [4:0] tmp_2;
reg [4:0] tmp_3;
reg [4:0] tmp_4;

always @(posedge clk) begin: _fifo
if (rst == 1) begin
output <= 0;
tmp_0 <= 0;
tmp_1 <= 0;
tmp_2 <= 0;
tmp_3 <= 0;
tmp_4 <= 0;
end
else begin
tmp_0 <= input;
output <= tmp_4;
tmp_1 <= tmp_0;
tmp_2 <= tmp_1;
tmp_3 <= tmp_2;
tmp_4 <= tmp_3;
end
end


endmodule
```

generated Verilog code

Example: FIFO

```
module fifo (
clk,
rst,
input,
output
);

input [4:0] input;
input clk;
input rst;
output [4:0] output;
reg [4:0] output;
reg [4:0] tmp_0;
reg [4:0] tmp_1;
reg [4:0] tmp_2;
reg [4:0] tmp_3;
reg [4:0] tmp_4;

always @(posedge clk) begin: _fifo
if (rst == 1) begin
output <= 0;
tmp_0 <= 0;
tmp_1 <= 0;
tmp_2 <= 0;
tmp_3 <= 0;
tmp_4 <= 0;
end
else begin
tmp_0 <= input;
output <= tmp_4;
tmp_1 <= tmp_0;
tmp_2 <= tmp_1;
tmp_3 <= tmp_2;
tmp_4 <= tmp_3;
end
end


endmodule
```

loop unrolled

generated Verilog code

Example: FIFO

```scala
import org.scalatest.FunSuite

import ScalaHDLExample.FIFO.FIFO
import ScalaHDL.Core.DataType._
import ScalaHDL.Core.DataType.Signals._
import ScalaHDL.Test.TestHelper

object FifoTestBench extends FIFO {

  override val WIDTH = 3
  override val DEPTH = 2

  val INPUT = List(0, 7, 0, 3, 0, 1, 0, 0).iterator

  defMod.Bench('clk, 'rst, 'input, 'output) {
    delay(1) {
      cycle('clk)
    }

    sync(0) {
      'rst := 0
      'input := INPUT.next()
    }
  }
}

class FifoTest extends FunSuite with TestHelper {
  test("test fifo") {
    val clk = bool(0)
    val rst = bool(1)
    val input = unsigned(0, FifoTestBench.WIDTH)
    val output = unsigned(0, FifoTestBench.WIDTH)

    val OUTPUT = List(0, 0, 0, 7, 0, 3, 0, 1).iterator

    val sim = Simulator(FifoTestBench,
      module('fifo, clk, rst, input, output),
      module('Bench, clk, rst, input, output))
    sim.setTrace("fifo.vcd")
    sim since 0 until 16 every 2 run {
      assert(clk === 0)
    }
    sim since 1 until 16 every 2 run {
      assert(clk === 1)
      assert(output.value === OUTPUT.next())
    }
    sim test
  }
}
```

Test Bench

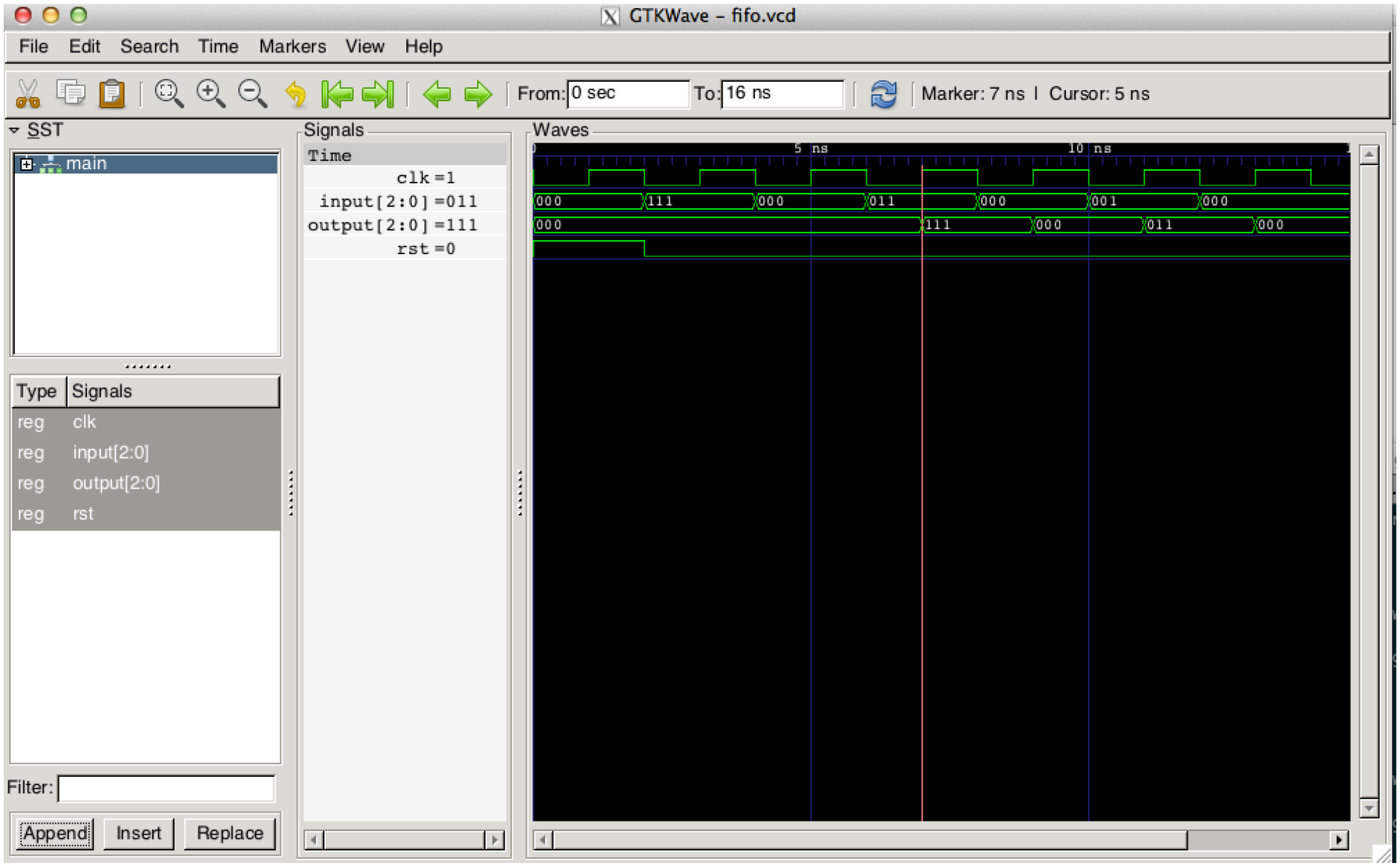
Example: FIFO

normal Scala override

```scala
import org.scalatest.FunSuite

import ScalaHDLExample.FIFO.FIFO
import ScalaHDL.Core.DataType._
import ScalaHDL.Core.DataType.Signals._
import ScalaHDL.Test.TestHelper

object FifoTestBench extends FIFO {

  override val WIDTH = 3
  override val DEPTH = 2

  val INPUT = List(0, 7, 0, 3, 0, 1, 0, 0).iterator

  defMod.Bench('clk, 'rst, 'input, 'output) {
    delay(1) {
      cycle('clk)
    }

    sync(0) {
      'rst := 0
      'input := INPUT.next()
    }
  }
}

class FifoTest extends FunSuite with TestHelper {
  test("test fifo") {
    val clk = bool(0)
    val rst = bool(1)
    val input = unsigned(0, FifoTestBench.WIDTH)
    val output = unsigned(0, FifoTestBench.WIDTH)

    val OUTPUT = List(0, 0, 0, 7, 0, 3, 0, 1).iterator

    val sim = Simulator(FifoTestBench,
      module('fifo, clk, rst, input, output),
      module('Bench, clk, rst, input, output))
    sim.setTrace("fifo.vcd")
    sim since 0 until 16 every 2 run {
      assert(clk === 0)
    }
    sim since 1 until 16 every 2 run {
      assert(clk === 1)
      assert(output.value === OUTPUT.next())
    }
    sim test
  }
}
```

Test Bench

Example: FIFO



generated .vcd file viewed by GTKWave

Unsigned Type Overflow

- Unsigned(value = 0, size = 4) - Unsigned(value = 1, size = 4)
- = Unsigned(value = 15, size = 4)
- 2’s complement

Future Plan

- All the basic circuit examples
- Signed number operations
- “if” statement

Any Question?

Thanks!