ScalaHDL

by Yao Li

Outline

- Current State of ScalaHDL.
  - Indentation.
  - ROM / RAM.
- Future Plan.

Indentation

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

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

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


endmodule
```

ROM

```
trait Rom extends ScalaHDL {


  val WIDTH = 64

  val DEPTH = 2048

  val DATA = HDLValueList(2, 3, 6, 7)


  defMod.rom('clk, 'addr, 'dout) {

    sync(0).rom {

      'dout := DATA('addr)

    }

  }

}
```

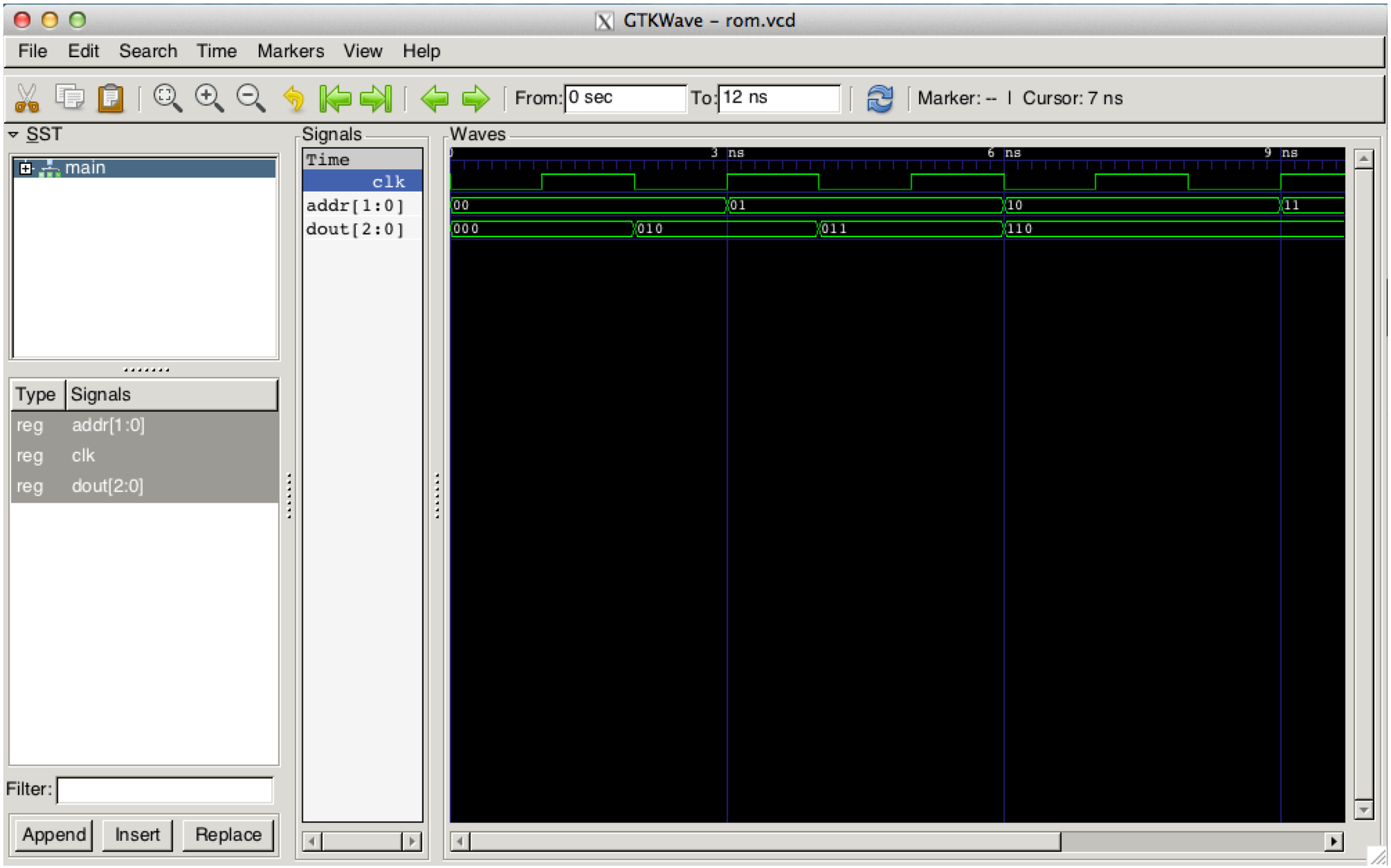
Generated Verilog of ROM

```
module rom (clk, addr, dout);

input [1:0] addr;

input clk;

output [2:0] dout;

reg [2:0] dout;


always @(negedge clk) begin: _rom
  case (addr)
    0: dout <= 2;
    1: dout <= 3;
    2: dout <= 6;
    3: dout <= 7;
  endcase
end
```

ROM Test Bench

```
object RomTestBench extends Rom {

  val WL_ADDR = 2

  val WL_DATA = 3

  val ADDR = List(1, 2, 3).iterator


  defMod.Bench('clk, 'addr, 'dout) {
    delay(1) {
      cycle('clk)
    }
    delay(3) {
      'addr := ADDR.next()
    }
  }
}
```

ROM Simulation Result



RAM

```
trait Ram extends ScalaHDL {

  val WIDTH = 64

  val DEPTH = 2048


  defMod.ram('clk, 'we, 'addr, 'din, 'dout) {

    val mem = toHDLList((for (i <- 1 to DEPTH) yield unsigned(0, WIDTH)).toList)

    sync(1).ram {

      when ('we is 1) {

        mem('addr) := 'din

      }

      'dout := mem('addr)

    }

  }

}
```

not so clean for now

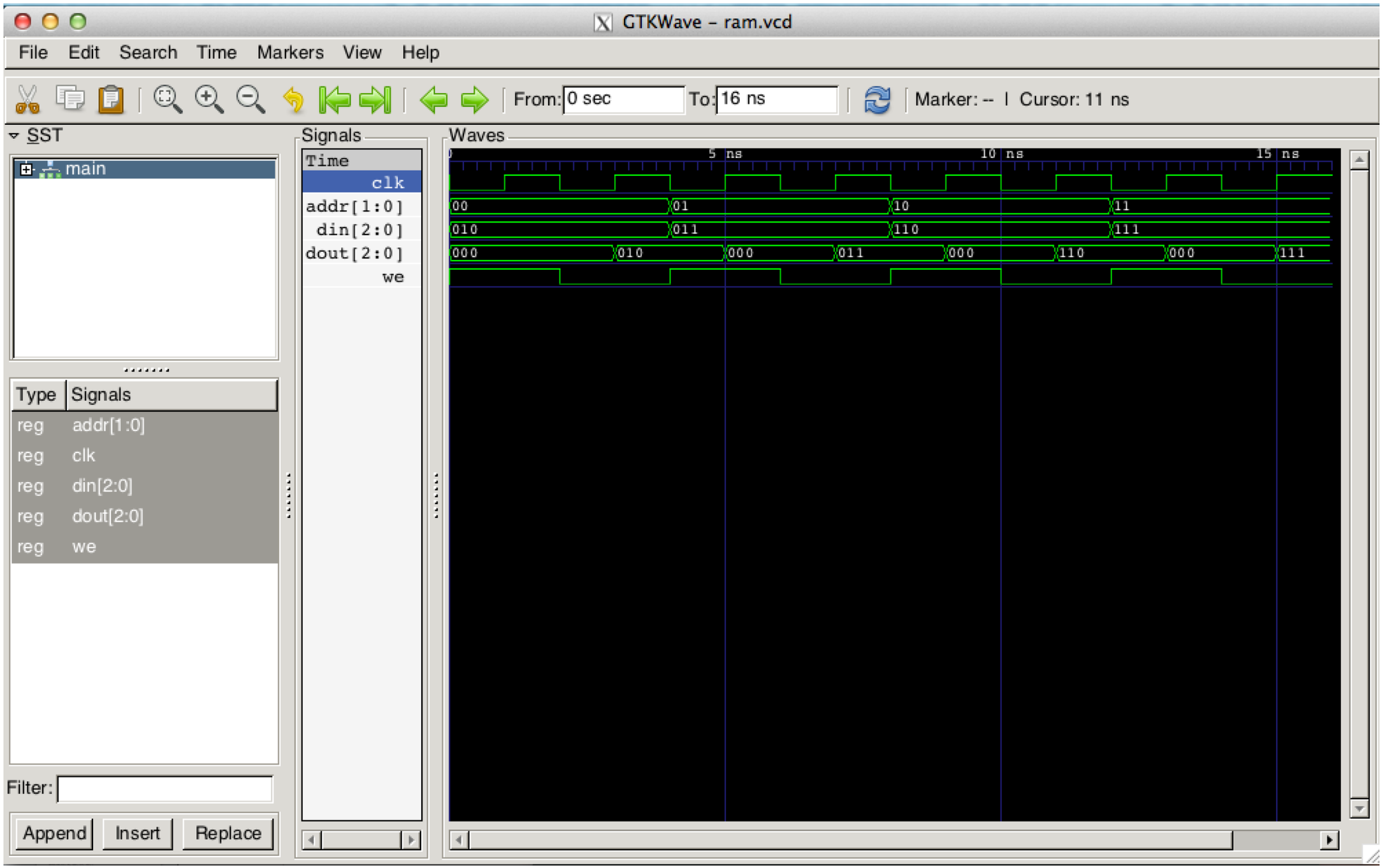
Generated Verilog of RAM

```
module ram (clk, we, addr, din, dout);

input [1:0] addr;

input [2:0] din;

input clk;

input we;

output [2:0] dout;

reg [2:0] dout;

reg [7:0] tmp_0 [0:3];

always @(posedge clk) begin: _ram

  if (we == 1) begin

    tmp_0[addr] <= din;

  end

  dout <= tmp_0[addr];

end

endmodule
```

RAM Test Bench

```
object RamTestBench extends Ram {

  val WL_ADDR = 2
  val WL_DATA = 3

  val ADDR = List(1, 2, 3).iterator
  val DIN = List(3, 6, 7).iterator

  defMod.Bench(
    'clk, 'we, 'addr, 'din, 'dout) {
    delay(1) {
      cycle('clk)
    }
    delay(2) {
      cycle('we)
    }
    delay(4) {
      'addr := ADDR.next()
      'din := DIN.next()
    }
  }
}
```

RAM Simulation Result



Things to be Done

- Better syntax in the RAM example.
- “yield” in test bench.
- Signal wrapping function.
- “if” instead of “when”.
- More tests to make it more robust.
- Better warnings and exceptions.

Any Question?

Thanks!