ScalaHDL

by Yao Li

Contents

- Current Simulator
- Implementation
- Follow-up Works

Current Simulator

```
sync('clk is 1)

defMod.logic('d, 'q, 'clk) {

  'q := 'd

}



delay(10)

defMod.clkGen('clk) {

  cycle('clk)

}



sync('clk is 0)

defMod.stimulus('d, 'clk) {

  'd := Random.nextInt(2)

}
```

```
> sbt run
>
time = 0
time = 10
ScalaHDL.Core.DataType.Signal@327b9698 is now 1
ScalaHDL.Core.DataType.Signal@4fdbef0c is now 1
time = 20
ScalaHDL.Core.DataType.Signal@327b9698 is now 0
ScalaHDL.Core.DataType.Signal@702b25c9 is now 1
time = 30
ScalaHDL.Core.DataType.Signal@327b9698 is now 1
ScalaHDL.Core.DataType.Signal@4fdbef0c is now 1
time = 40
ScalaHDL.Core.DataType.Signal@327b9698 is now 0
ScalaHDL.Core.DataType.Signal@702b25c9 is now 0
                              …
```

Implementation: Simulator

```
while (true) {
  for (sig <- hdl.siglist)
    waiters = sig.update() ::: waiters
  hdl.siglist = List()
  for (waiter <- waiters) {
    val wl = waiter.next()
    wl.foreach(w => w match {
      case x: DelayWaiter =>
        schedule(runtime + x.time, x)
      case _ => ()
    })
}
waiters = List()
… // future events
```

Implementation: Waiter

```
class Waiter(stmts: Seq[HDLObject], sigMap: Map[Symbol, Signal]) {

  val iter =

    iterator[List[Waiter]] {

      var senslist: List[Waiter] = List()

      while (true) {

        for (stmt <- stmts) stmt.exec(sigMap)

        yld(senslist)

      }

    }


  def next() = iter.next

}
```

And…

- Code in SHDL.scala, DataType.scala has also been changed drastically.
- Some parts are still (sort of) hard-code.
- There’re lots of stuff NOT implemented yet.

Follow-up Works

- Tests for Simulator.
- Finish some not implemented and hard-code parts.
- More advanced features. (suggestions?)

Any Questions?

Thank You!