ScalaHDL

The Internal DSL

by Yao Li

What We've Discussed

- The previous solution is not good enough.
- We should investigate more hacks and tricks to implement our expected DSL.
- We prefer an internal approach.

New Solution

- A completely internal DSL implementation.
- All conversion happen at runtime.
- Some experimental feature since Scala 2.9 is used.

Current Solution: Main.scala

```
package ScalaHDL

import ScalaHDL.DataType._

object Main extends ScalaHDL {
  def main(args: Array[String]) {
    module.adder('a, 'b, 'res) {
      'res := 'a + 'b
      'res := 'res + 'a
    }
    val a = Signal(3, 3)
    val b = Signal(2, 3)
    val res = Signal(0, 3)
    println(convert('adder, a, b, res))
  }
}
```

module

```
module.adder('a, 'b, 'res) {
  'res := 'a + 'b
  'res := 'res + 'a
}
```

- module is an object which extends Dynamic class
- module.adder actually calls module.applyDynamic("adder")

module

```
module.adder('a, 'b, 'res) {
  'res := 'a + 'b
  'res := 'res + 'a
}
```

- module.adder('a, 'b, 'res) will return an instance of HDLModule
- HDLModule.apply will take a *pass-by-name* parameter

HDLIdent

```
'res := 'a + 'b
```

- There's an implicit conversion from Symbol to HDLIdent
- := and + are methods of HDLIdent
- This statement will be executed, and return an HDLObject.
- The structure of an HDLObject is like an AST.

Side Effects

- To record the code inside a block, side effects are brought in.
- For each HDLModule, there exists a list of HDLObject representing the ScalaHDL code.
- The list is updated when an assignment operation has happened.

Limitations

- You are asked to use module instead of def.
- There are some annoying single quotes (').

TODO

- More data type and operation.
- Syntax specification.
- Type inference.
- Simulator and Test bench. (top priority?)

Any Question?

Thanks!

module

```
case class HDLModule(name: Symbol, params:
Seq[Symbol]) {
  def apply(f: => HDLObject) = f
}

object module extends Dynamic {
  def applyDynamic(name: String)(params:
Symbol*): HDLModule = {
    HDLModule.createModule(name, params)
  }
}
```