ScalaHDL

First Attempt in Implementing DSL

by Yao Li

What We've Discussed Before

- We agreed that we'll explore both internal and external ways to implement ScalaHDL.
- We talked about the type inference problem.
- We agreed to not support class, object, trait in this phase.
- We expected to see a simple demo, which will convert our DSL into Verilog.

Approaches

- Internal:
  - macro (experimental feature since 2.10.0)
- External:
  - compiler plugin
  - parser combinator

What We Need

- The ability to change Scala's syntax.
- The ability to convert the functions in ScalaHDL into Scala objects, and then HDL code.

Scala Macro

Example:

```
class Queryable[T] {
    def map[U](p: T => U): Queryable[U] =
        macro QImpl.map[T, U]
}

object QImpl {
    def map[T: c.WeakTypeTag, U: c.WeakTypeTag]
        (c: Context)(p: c.Expr[T => U]):
        c.Expr[Queryable[U]] = ...
}
```

However…

- There are many problems in current Scala macros.
- The most critical one for us among them is: Parameters passed to a macro must be a valid Scala expression.
- You can NOT change Scala syntax using macros.
- NOT a feasible approach for ScalaHDL.

Scala Compiler Plugin

- There’re many phases in Scala compiler pipeline:
- parser, namer, packageobjects, typer, superaccessors, pickler, refchecks, liftcode, uncurry, tailcalls, specialize…
- You can insert a phase into this pipeline by writing a plugin.

However…

- You can not insert a phase before the first phase, i.e. parser.
- If there exists syntax error in the code to be compiled, it won't pass the parser phase.
- So you can NOT change Scala syntax, either.
- NOT a feasible approach for ScalaHDL, either.

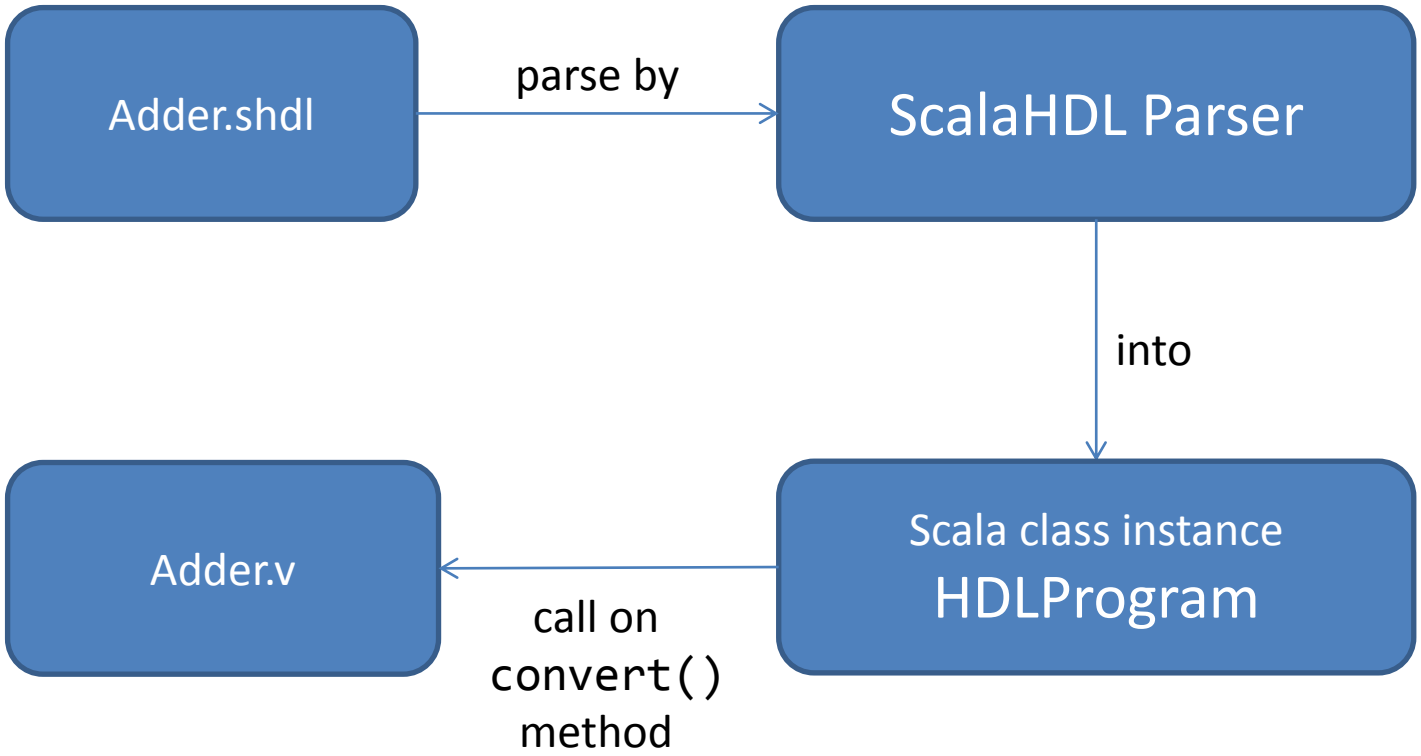
Parser Combinator

- It’s part of Scala standard library.
- The function you write is itself a parser.
- Scala objects can be generated as parse results.
- It’s our current solution.

Example

```
class SHDLParser extends StandardTokenParsers {
    def module: Parser[(String, HDLModule)] =
        "def" ~> moduleName ~ params ~ block ^^ {
            case moduleName ~ params ~ block => {
                val m = HDLModule(params, block)
                (moduleName, m)
            }
        }
    …
}
```

Current Solution



Demo code can be checked in: https://github.com/lastland/ScalaHDL/tree/develop

Bad Parts in Current Solution

- The ScalaHDL source code is written in a distinct file from Scala code.
- You don’t get help from original Scala parser and AST to support Scala syntax in your DSL. Instead, extra parsers and ASTs must be defined.
- Extra efforts is need to support REPL.

Notice

- Current code is written in a very ad-hoc way, with many functionalities not implemented, and some hard code.
- The architecture may change drastically in the future.
- I’m still looking for other alternatives.

Other Methods?

- Modify Scala compiler directly?
  - Users may have to keep different Scala compilers of different versions.
  - May cause conflicts when code is also modified in upstream.

Any Questions
or Suggestions?