ScalaHDL

Yao Li

Objectives

- Allow the description of hardware using a minimalist and intuitive syntax.
- Convert the description into either VHDL or Verilog.
- Provide a simulation methodology that can drive and generate output results based on the description provided.
- Generate traces of the described system's signals.

Related Work

- MyHDL (Python to HDL)
- Chisel (Scala to HDL)
  - by UC Berkeley
  - open sourced on GitHub with modified BSD License

An example of MyHDL

```
from myhdl import *

def dff(q, d, clk):

    @always(clk.posedge)
    def logic():
        q.next = d

    return logic
```

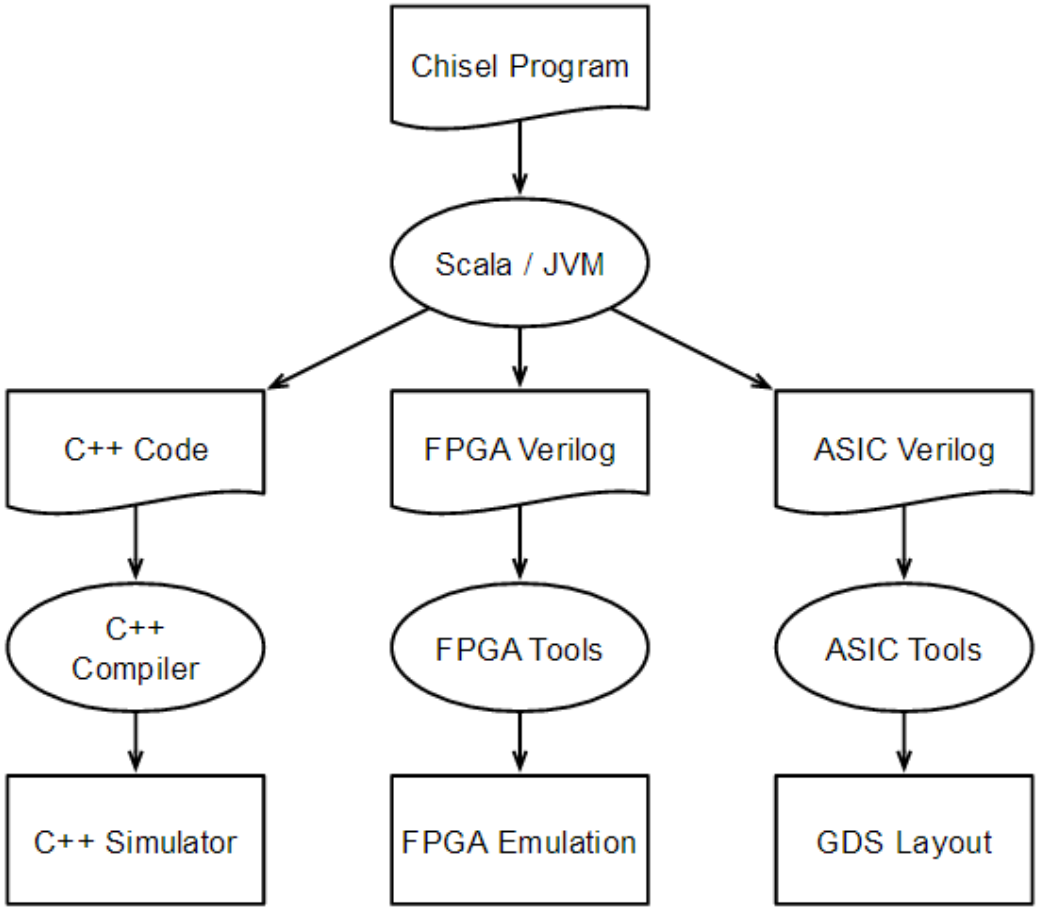
An example of MyHDL

```
from random import randrange

def test_dff():

    q, d, clk = [Signal(bool(0)) for i in range(3)]

    dff_inst = dff(q, d, clk)

    @always(delay(10))
    def clkgen():
        clk.next = not clk

    @always(clk.negedge)
    def stimulus():
        d.next = randrange(2)

    return dff_inst, clkgen, stimulus


def simulate(timesteps):
    tb = traceSignals(test_dff)
    sim = Simulation(tb)
    sim.run(timesteps)

simulate(2000)
```

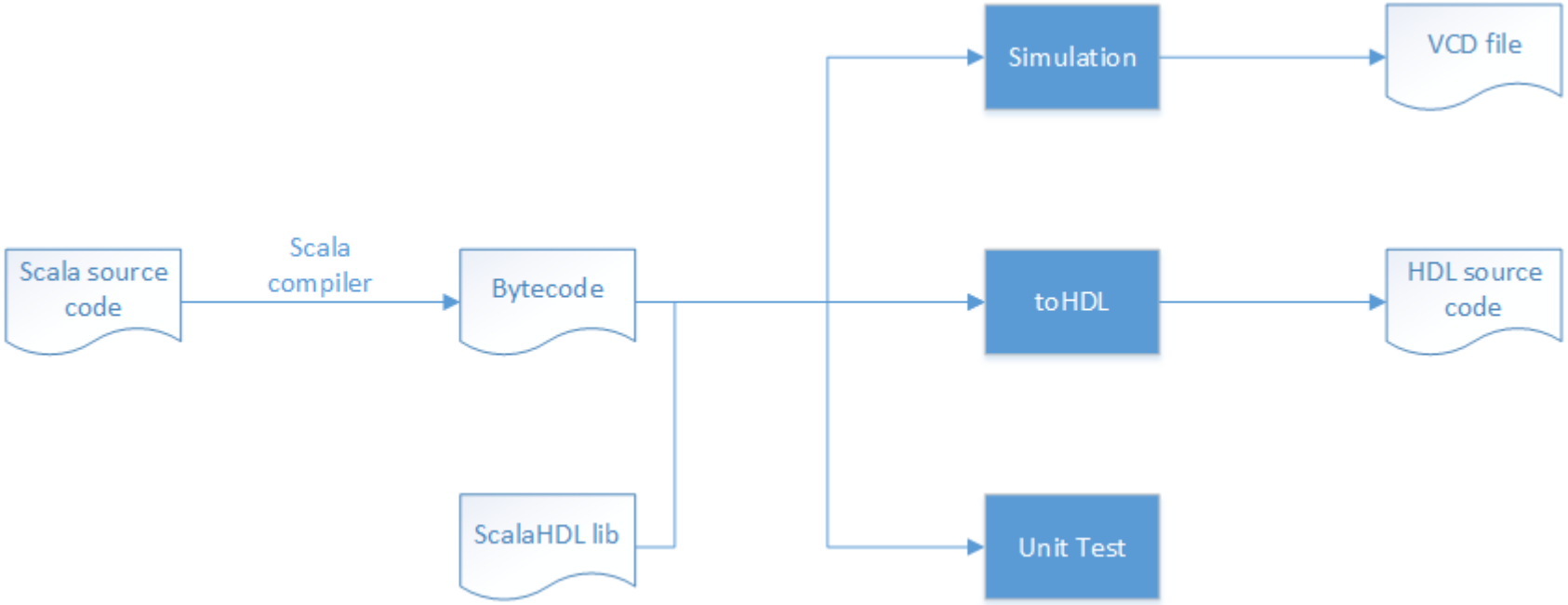
An example of Chisel

```
import Chisel._

class HelloModule extends Module {

    val io = new Bundle {}

    printf("Hello World!\n")
}

class HelloModuleTests(c: HelloModule) extends Tester(c, Array(c.io)) {
    defTests {
        true
    }
}

object hello {
    def main(args: Array[String]): Unit = {
        chiselMainTest(Array[String]("--backend", "c", "--genHarness"),
            () => Module(new HelloModule())){c => new HelloModuleTests(c)}
    }
}
```

Chisel workflow



Expected workflow



Our Plan

- The development will be based on Chisel.
- Some good designs in MyHDL will also be adopted in our ScalaHDL.
  - e.g. syntax, simulation, use of annotations, etc.

Syntax

```
@init(reset=1) // output res will be set to 0 when reset is 1
@sync(clock=1)
def mod_adder2(a*[1],b,res=0){
    res.size = max(a.size,b.size)+1  // avoid overflow
    res = a + b
}

@async
def mod_not(in,out){ out = not in }

// System definition with wires automatically inferred
def system(reset, clock, a, b, notsum){
    mod_adder(a,b, sumres*[2])
    mod_not(sumres,notsum)
}
```

Simulation

- Chisel uses a C++ emulator. (And it's C++ 11 code!)
- A native Scala emulator may be expected for better convenience.

Schedule

- 10/01/2013-10/31/2013 Requirements analysis and testbed building.
- 11/01/2013-11/30/2013 Software architecture design and key interface definition.
- 12/01/2013-12/31/2013 Software detailed design.
- 01/01/2014-01/31/2014 Software suite development and test.
- 02/01/2014-02/28/2014 Software suite test and debug. Technical report of ScalaHDL.

Plan for the coming week

- A more detailed design for our ScalaHDL to be proposed, and discussed, especially designs for syntax, simulation, and test.

Q&A

Thank you!