Chisel3 Testing Cheat Sheet

Version 0.5 (beta): September 28, 2016

Testing

Chisel provides a evolving family of testers with different capabilities. A tester typically is invoked from scalatest. For example:

```
class CounterSpec extends ChiselPropSpec {
  property("Counter should wrap") {
    assertTesterPasses {
        new WrapTester(42)
      }
    }
  }
```

BasicTester:

`BasicTester`: supports creation of a circuit and provides simple chisel operations and a family of assert statements.

```
class WrapTester(max: Int)
  extends BasicTester {
  val (cnt,wrap) = Counter(Bool(true),max)
  when(wrap) {
    assert(cnt === UInt(max - 1))
    stop()
  }
}
```

Chisel-Testers:

Additional testers can be found in the ucb-arg/chisel-testers repository. Current Testers are:

StandardTester:

`Standard Tester`: is a class with functions for testing `Modules`, connecting and communicating with a simulator:

`reset([n:Int])`	reset the DUT for `n` (default 1) clocks
`step(n:Int)`	steps the DUT for `n` clocks

`poke(data:Bits, x:BigInt)`	writes `x` to wire `data`
`poke(data:Aggregate, x:Array[BigInt])`	writes values from `x` to corresponding wires in `data`
`peek(data:Bits): BigInt`	reads from wire `data`
`peek(data:Aggregate): Array[BigInt]`	reads multiple values from source wires in `data`

`expect(good:Boolean, msg:String): Boolean`	fails unless `good` is `True`, `msg` should describe the test
`expect(data:Bits, target:BigInt): Boolean`	fails unless the value in wire `data` equals `target`

Defining:

Subclass `Tester` with testing code:

```
class MuxTester(c:Mux) extends Tester(c) {
  for (sel <- 0 until 2) {
    poke(c.io.sel, sel)
    poke(c.io.in0, 0); poke(c.io.in1, 1)
    step(1)
    expect(c.io.out, sel)
  }
}
```

Hardware IO Testers:

The Hardware IO Testers run all tests by implementing small FSM's and vectors of input and output values. In general hardware testers are faster than the standard tester. There currrently two forms

Stepped Tester:

`SteppedHWIOTester`: works like the Standard Tester but does not support the peek command

Defining:

Subclass `SteppedHWIOTester` with testing code:

```
class AdderTests extends SteppedHWIOTester {
  val dut = Module( new Adder(10) )
  rnd.setSeed(0L)
  for (i <- 0 until 10) {
    val in0 = rnd.nextInt(1 << dut.w)
    val in1 = rnd.nextInt(1 << dut.w)
    poke(dut.io.in0, in0)
    poke(dut.io.in1, in1)
    expect(dut.io.out, (in0 + in1) & ((1 << du

    step(1)
  }
}
```

Decoupled Tester:

`OrderedDecoupledHWIOTester`: tests Modules with IO implementing `DeqIO`, `EnqIO`, `Valid` or `Decoupled` interfaces, automatically handling the ready/valid protocol.

Defining:

Subclass `OrderedDecoupledHWIOTester` with testing code:

```
class DecoupledRealGCDTests4 extends OrderedDe
  val dut = Module(new RealGCD())
  for {
    i <- 1 to 10
    j <- 1 to 10
  } {
    val gcd_value = computeGcdResults(i, j)

    inputEvent(dut.io.in.bits.a -> i,
               dut.io.in.bits.b -> j)
    outputEvent(dut.io.out.bits -> gcd_value)
  }
}
```