ScalaHDL

Yao Li

Contents

Previous Discussion

- The development of ScalaHDL will be based on Chisel.
- But instead of C++ emulator, ScalaHDL will use a Scala simulator to bring in REPL, etc.
- Some good designs in MyHDL, e.g. syntax, use of annotations, will also be adopted.

Revised Schedule

- 10/01/2013-10/31/2013 Requirements analysis and testbed building.
- 11/01/2013-11/30/2013 Testbench building.
- 12/01/2013-12/31/2013 Software architecture design and key interface definition.
- 01/01/2013-01/31/2013 Software detailed design.
- 02/01/2014-02/28/2014 Software suite development and test.
- 03/01/2014-03/31/2014 Software suite test and debug. Technical report of ScalaHDL.

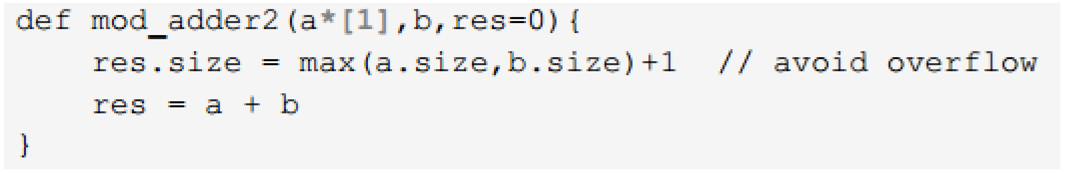
ScalaHDL Syntax Design & Discussion

- DSL implementation: internal vs. external
- Type inference & Auto wiring
- Class, Object, Trait

Internal vs. External

- Internal: Doesn’t need extra parser. But there may be some difficulty in implementing syntax like type inference & auto wiring. May also exists
- External: Need extra parser. May also need extra efforts to support REPL.

Example



```
def mod_adder2(a*[1],b,res=0){
    res.size = max(a.size,b.size)+1  // avoid overflow
    res = a + b
}
```

- This won’t compile!
- Type of every parameter is needed.
- To translate this piece of code into HDL, a structure like AST is needed any way.

Internal vs. External

- Is external DSL a better approach?

PS:

- MyHDL seems to be an internal DSL to me.
- In MyHDL, there’s already a built-in function which can give you an AST of a method.
- I don’t know yet if there’s a Scala alternative for such function.

Type Inference & Auto Wiring

- There are several aspects to be inferred for an identifier: data type, size, input/output, reg/wire.
- How should type be inferred? (See example)

Example

```
@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)
}
```

So far, type of a and b can not be decided.

Example

```
// Test bench
def test_system(){
    reset =  Bool(1)*[1]
    clock =  Bool(0)
    a      = Unsigned(3,3)  // 3 bits, with init value of 3
    b      = Unsigned(2,2)  // 2 bits, with init value of 2
    inst_system1 = system(reset, clock, a, b, notsum)

    cycle_clock()
    reset = 0
    cycle_clock()
    assert notsum == “1010”   // inverted value of 5 plus one bit
}
```

- Type of a and b is decided in test bench.
- But is it a reasonable solution?
- Is it possible that there are several test benches, each with different assignment of type?
- PS: In MyHDL, you need to assign types to variables before their conversion to HDL.

Class, Object, Trait

- So far, we are only talking about functions.
- Supports for some advanced concepts such as Class, Object, Trait will not be implemented in this phase.

Plan for the coming week

A more detailed plan for how our ScalaHDL will work.

Hopefully there would be a simple program for demonstration.

Other Question?