ScalaHDL

Outline

- If statement
- loop
- recursion
- function parameters
- future plan

If Statement

- `if (a == b)`
  - if a or b is ScalaHDL data type, this will be an HDL if statement.
  - if neither a and b are ScalaHDL data type, this will be executed in Scala.
- Current plan: we will be using macro to overload equals method to implement such feature
- Things could get complicated for:
  - `if (a == (b > c)) …`
  - `a = b > c; if (a) …`
  - `for (s <- sigs if s == a) …`
- What about pattern matching?

Loop

- `for (i <- 1 to n) s(i) := 1`
- Since n is already known, we will simply unroll the loop.
- `for (i <- 1 to n if s(i) == 0) s(i) := 1` will be equivalent to:
  - `for (i <- 1 to n) if (s(i) == 0) s(i) := 1`

Recursion

- Instead of supporting “generate” features in Verilog, unroll the recursion like in MyHDL may be a better choice because:
  - it doesn’t need underlying HDL to support such feature
  - we can learn from MyHDL

Function Parameters

- How to decide if a parameter is input or output? And how to wire ports?

```
defMod.foo(‘a, ‘b, ‘c, ‘d) {
    ‘c := ‘a
    ‘e := ‘b + 1
    ‘bar(‘c, ‘e, ‘d)
}
```

```
defMod.foo(‘a, ‘b, ‘c, ‘d) {
    ‘c := ‘a
    ‘e := ‘b + 1
    ‘bar(‘c, ‘e, ‘d)
}
```

- ‘c will be output because it has been assigned
- ‘e will be a new signal because it doesn’t exist in current symbol table
- whether ‘d is output or not depends on ‘bar
- however, ‘c and ‘e could be input in ‘bar

```
defMod.foo(‘a, ‘b, ‘c, ‘d) {
   ‘c := ‘a
   ‘e := ‘b + 1
   ‘bar(‘c, ‘e, ‘d)
}
```

- What does ‘c := ‘a actually mean here?
  - one time assignment to reg ‘c
  - assignment to wire ‘c

- There’re several alternative solutions:
  - always { ‘c := ‘a } means ‘c is wire
  - init { ‘c := ‘a } means ‘c is reg
  - introduce another operator to distinguish them

Plan

- Implement a bitonic sort in ScalaHDL.

Any Question?