ScalaHDL

by Yao Li

Outline

- Matrix multiplication.
- Things to be done.

Matrix Multiplication

- Convenience.
- Inconvenience.

Matrix Multiplication: Convenience

```
for (i <- 0 until height_a) {
  for (j <- 0 until width_b) {
    val t = (1 until height_b).map(
      (k) => a(i)(k) * b(k)(j)).foldLeft(
      a(i)(0) * b(0)(j))(_ + _)
    c(i)(j) := t
  }
}
```

“for” loop!

“map” and “foldLeft”!

Matrix Multiplication: Inconvenience

```
defMod.mult('clk, 'rst,
  'a00, 'a01, 'a02, 'a03,
  'a10, 'a11, 'a12, 'a13,
  'a20, 'a21, 'a22, 'a23,
  'b00, 'b01, 'b02,
  'b10, 'b11, 'b12,
  'b20, 'b21, 'b22,
  'b30, 'b31, 'b32,
  'c00, 'c01, 'c02,
  'c10, 'c11, 'c12,
  'c20, 'c21, ‘c22)
```

still have to declare
all these elements in matrices

Matrix Multiplication

the bridge.
so we can use "for" loop later

```
val a = Array(
  Array('a00, 'a01, 'a02, 'a03),
  Array('a10, 'a11, 'a12, 'a13),
  Array('a20, 'a21, 'a22, 'a23)
).map(_.map(toHDLType))
```

Matrix Multiplication: Generated Verilog Code

```
c00 <= ((((a00 * b00) + (a01 * b10)) + (a02 * b20)) + (a03 * b30));

c01 <= ((((a00 * b01) + (a01 * b11)) + (a02 * b21)) + (a03 * b31));

c02 <= ((((a00 * b02) + (a01 * b12)) + (a02 * b22)) + (a03 * b32));

c10 <= ((((a10 * b00) + (a11 * b10)) + (a12 * b20)) + (a13 * b30));

c11 <= ((((a10 * b01) + (a11 * b11)) + (a12 * b21)) + (a13 * b31));

c12 <= ((((a10 * b02) + (a11 * b12)) + (a12 * b22)) + (a13 * b32));

c20 <= ((((a20 * b00) + (a21 * b10)) + (a22 * b20)) + (a23 * b30));

c21 <= ((((a20 * b01) + (a21 * b11)) + (a22 * b21)) + (a23 * b31));

c22 <= ((((a20 * b02) + (a21 * b12)) + (a22 * b22)) + (a23 * b32));
```

Things to be Done

- **Better warnings and exceptions.**
    - Especially during conversion.
- More tests.

Any Question?

Thanks!