C integration in SystemVerilog and Scala

Outline

- SystemVerilog DPI
- Java JNI
- JNI in Scala
- Concluding remarks

SystemVerilog Direct Programming Interface

- DPI allows you to call C-code from inside SV (and SV code from inside C)
- Allows you to use C model for verification
- We implemented the scoreboard in C, calling it from SV
  - Incredibly intuitive!

```c
#include "svdpi.h"
#include <stdlib.h>
/**
* @brief Implements the scoreboard checking function in C using the SV Direct Programming Interface
*
* @param din The input data to the ALU
* @param op The opcode used for the ALU
* @param reset Whether reset was asserted (1) or not (0)
* @param fromDUT The result from the DUT
*/
int scoreboard_check(int din, int op, int reset, int fromDUT) {
```

```
import "DPI-C" function int scoreboard_check(int din, int op, int reset, int fromDUT);

class scoreboard_dpi extends scoreboard;
   `uvm_component_utils(scoreboard_dpi);

   extern function void write_1(leros_command t);

   function new(string name = "scoreboard_dpi", uvm_component parent);
      super.new(name, parent);
   endfunction: new

endclass: scoreboard_dpi

function void scoreboard_dpi::write_1(leros_command t);
   int ret;

   ret = scoreboard_check(t.din, t.op, t.reset, t.accu);
```

Java JNI

- Java interface to native code
- Typically libraries in form of .dll or .so files
- Libraries only work on the machine they have been compiled for
  - Not as portable as the DPI!

```scala
@nativeLoader("chisel-uvm0")
class NativeScoreboard {
/**
   * @brief Implements the scoreboard checking function in C using the JNI
   *
   * @param din The input data to the ALU
   * @param op The opcode used for the ALU
   * @param reset Whether reset was asserted (1) or not (0)
   * @return The calculated output value
   */
   @native def calc(din: Int, op: Int, reset: Int): Int;
}
```

```c
#include "NativeScoreboard.h"
#include <ctype.h>

enum leros_op {
   NOP, ADD, SUB, AND, OR, XOR, LD, SHR
};
typedef enum leros_op leros_op_t;

/**
* @brief Implements the scoreboard checking function in C using the JNI
*
* @param din The input data to the ALU
* @param op The opcode used for the ALU
* @param reset Whether reset was asserted (1) or not (0)
* @return The calculated output value
*/
JNIEXPORT jint JNICALL Java_NativeScoreboard_calc
  (JNIEnv* env, jobject obj, jint din, jint op, jint reset) {
```

Using the Scala JNI

- Scala is built on Java, has access to the JNI
- Using plain Scala + JNI is pretty easy
  - Generate C header files, write C-code, compile and run
- Using JNI + Chisel is more difficult when using *sbt* (Scala Build Tool) to manage your build
  - Using external libraries messes with the default JNI integration
  - Requires additional plugins and additional steps to function
- It works, allowed us to re-use the C-tester from previously
  - Not as intuititive as using the DPI

```
sbt "project native; javah; nativeInit cmake chisel-uvm; nativeCompile"
```

Concluding remarks

- The DPI is impressively simple to use
- The JNI is relatively approachable
  - But using Chisel functionality on top makes it more difficult
- Scalability and portability *might* be an issue

- Future work: Developing better/easier integration
  - Forgoing the JNI entirely?