Compilers Research

Aditya Srinivasan, Drew Hilton

August 2017

1 Preface

This resource aims to document and formalize aspects of the compilers research project that I am undertaking. It will be updated regularly as the project evolves so as to reflect the most recent and relevant decisions in order to serve as a reference.

2 Overview

This research project is an attempt to develop a programming language for describing hardware and an accompanying compiler. Currently, hardware description languages lack features that would improve ease of expression and modularity, and reduce the potential for errors during execution. Such features include, but are not limited to, strong type systems, parametric polymorphism, and user-defined datatypes. Although there exist languages that strive to provide these primitives, they do not fully consider the semantics of hardware design. The compiler developed through this research project will use the newly developed language as the source and compile to Verilog as the target.

3 Formalizing the Dichotomy

As mentioned in Section 2, there is a need to formalize the dichotomy between the software and hardware components so as to develop a language that respects both entities appropriately. As such, in this section, we will distinguish the types, semantics, and syntax of the two.

3.1 Types

We will first formalize the definitions of the two categories of types: software (SW) and hardware (HW).

```
hw = bit
   | hw[]
   | (hw * hw)
```

(a) HW types

```
sw = int
   | float
   | hw
   | sw list
   | (sw * sw)
   | sw -> sw
```

(b) SW types

Figure 1: Type definitions

As can be seen in Figure 1(a) above, the hardware type is defined as a bit, a fixed-sized array of a hardware type, or a tuple of hardware types. In order to justify this, we argue that all data is represented by bits in hardware, and as such the fundamental data representation of a hardware type must be a bit. We augment the expressive power by introducing fixed-sized arrays, in order to represent bit vectors or bit patterns. Finally, we permit the declaration of tuples in order to group hardware types into one value.

Note that the type declaration for the fixed-size array and tuple are recursive, in that any hardware type can populate these types. This allows for higher-dimensional arrays, arrays of tuples of bits, tuples of arrays of bits, et cetera.