1 DFIANT HDL

DFiant [5] is a hardware description language (HDL) and one of the LEGaTO programming models. For a long time, the dominating HDLs have been Verilog, System Verilog, and VHDL, and they all provide the same hardware design abstraction: a register transfer-level (RTL) abstraction. An RTL abstraction burdens designers with explicitly clocked constructs that do not distinguish between design functionality and implementation constraints (e.g., timing, target device). For example, VHDL and Verilog constructs require designers to explicitly place a register, regardless if it is part of the core functionality (e.g., a state-machine state register), an artifact of the timing constraints (e.g, a pipeline register), or an artifact of the target interface (e.g., a synchronous protocol cycle delay). These semantics narrow design correctness to specific timing restrictions, while vendor library component instances couple the design to a given target device. Evidently, formulating complex portable designs is difficult, if not impossible. Finally, these older languages do not support modern programming features that enhance productivity and correctness such as polymorphism and type safety.

High-level synthesis (HLS) tools such as Vivado HLS [7], and high-level HDLs such as Bluespec SystemVerilog [3] and Chisel [1] attempt to bridge the programmability gap. While these tools and languages tend to incorporate modern programming features, they still mix functionality with timing and device constraints, or lack hardware construction and timed synchronization control. For example, designs must be explicitly pipelined in Chisel or Bluespec, while a simple task as toggling a led at a given rate is impossible to describe with C++ constructs in Vivado HLS. Such tools and languages, therefore, fail to deliver a clean separation between functionality and implementation that can yield portable code, while providing general purpose HDL constructs. We explore these gaps further in Section **??**.

In this paper we further extend, a modern HDL whose goal is to improve hardware programmability and designer productivity by enabling designers to express truely portable and composable hardware designs. DFiant decouples functionality from timing constraints (in an effort to end the *"tyranny of the clock"* [6]). DFiant offers a clean model for hardware construction based on its core characteristics: (i) a clock-agnostic dataflow model that enables implicit parallel data and computation scheduling; and (ii) functional register/state constructs accompanied by an automatic pipelining process, which eliminate all explicit register placements along with their direct clock dependency. DFiant borrows and combines constructs and semantics from software, hardware and dataflow languages. Consequently, the DFiant programming model accommodates a middle-ground approach between low-level hardware description and high-level sequential programming.

DFiant is implemented as a Scala library and relies on Scala's strong, extensible, and polymorphic type system to provide its own hardware-focused type system (e.g., bit-accurate dataflow types, input/output port types). The library performs two main tasks: the frontend compilation, which translates dataflow variable interactions into a dependency graph; and the backend compilation, which translates the graph into a pipelined RTL code and a TCL constraints file, followed by a hardware synthesis process using commercial tools. Additionally, the graph can be simulated within the Scala integrated development environment (IDE).

This work focuses on the DFiant language and frontend compiler. DFiant is *not* an RTL language, nor is it a sequential language such as C. The following two sections highlight DFiant's unique semantics by comparing them against modern design language alternatives. For a proof of concept, we implemented a preliminary auto-pipelining backend compiler to compare DFiant and traditional HDL design flows in two test cases: an Advanced Encryption Standard [4] (AES) cipher block and an IEEE-754 [2] floating point multiplier (FPMul). Future work may delve further into the backend compiler and its HLS potential.

The paper is organized as follows. The next two sections describe DFiant's concurrency and state abstractions, followed by Section **??**, which contrasts DFiant with related work. Section **??** details the DFiant type system. Section **??** provides a real-world DFiant code compilation example. Section **??** provides the proof of concept results. Finally, Section **??** concludes the paper.

REFERENCES

[1] Jonathan Bachrach, Huy Vo, Brian Richards, Yunsup Lee, Andrew Waterman, Rimas Avižienis, John Wawrzynek, and Krste Asanović. 2012. Chisel: Constructing Hardware in a Scala Embedded Language. In *ACM/EDAC/IEEE Design Automation Conference (DAC)*.

[2] IEEE. 2008. *754-2008 - IEEE Standard for Floating-Point Arithmetic.*

[3] Rishiyur Nikhil. 2004. Bluespec System Verilog: efficient, correct RTL from high level specifications. In *ACM/IEEE Intl. Conf. on Formal Methods and Models for Co-Design.*

[4] NIST. 2001. Advanced Encryption Standard (AES). *Federal Information Processing Standards Publication* 197, 441 (2001).

[5] Oron Port and Yoav Etsion. 2017. DFiant: A Dataflow Hardware Description Language. In *Intl. Conf. on Field Programmable Logic and Applications.*

[6] I Sutherland. 2012. The tyranny of the clock. *Comm. ACM* 55, 10 (2012), 35–36.

[7] Xilinx. 2015. Vivado High Level Synthesis User Guide. (2015).