Legend	
✗	Not available or is heavily handicapped
✓-	Mostly available but also with noticeable limitations
✓	Mostly available with almost no limitations
✓+	Exemplary capabilities

Property Group	Properties	HLS	DF-HDL	HL-RTL	RTL	Comments
	Tools & Languages	VivadoHLS, LegUp, Catapult, Synphony, HercuLeS, OpenCL, MaxJ	DFiant	Chisel, SpinalHDL, PyRTL, Migen, MyHDL, Bluespec, Cx, JHDL	VHDL, Verilog, SystemVerilog	
Hardware Design Fundamentals	Concurrent Semantics	✓-	✓	✓	✓	RTL semantics are based on clock-scheduled concurrency. HL-RTL semantics describe hardware construction and are therefore concurrent. DF-HDL semantics are dataflow-based which means that independent paths are concurrent and dependent paths are sequenced together. Most HLS languages are inherently sequential, and special pragmas and tooling allow concurrent and pipelined synthesis.
	Bit Accurate Type Safety	✓-	✓+	✓	✓	Some HLS languages do not support bit accurate types. DFiant is embedded in Scala which is a type-safe language. DFiant has implemented special type-level macros reflect bit-accurate type errors to the Scala presentation compiler, which in turn provides error positions and messages to the editor. This provides an immediate visible indication of most errors, and thus increases productivity.
	Clear Synthesizability	✗	✓	✓	✗	RTL and HLS languages were developed for simulation and software development, respectively, and only later were adopted for hardware synthesis. Consequently, these languages include significant unsynthesizable syntax that confuses hardware developers. Contrarily, DFiant and many HL-RTL languages focus on hardware construction syntax that is inherently synthesizable.
	Hierarchy, Composition, IO Ports, Connectivity, RTL Blackbox Interface,Top Design	✓-	✓+	✓+	✓	HLS languages offer only limited connectivity and are not designed for top-level hardware description. DFiant and Some HL-RTL languages surpass the capabilities of RTLs by allowing direct connection between sibling components and even between ports at different hierarchy levels.
	Combinational/Sequential Expressions	✓-	✓	✓	✓	HLS is sometimes too high-level to allow direct developer control over combinational and sequential paths. However, such a control is rarely required in HLS application domains.
	Clock and Reset Management	✓-	✓-	✓	✓	HLS offers very limited clock and reset management capabilities due to its abstraction. We have yet to implement multiple-clock domains capabilities in DFiant, but we see no limitation to have equivalent RTL-control with the proper constraints.
Design Abstraction & Automation	State Abstraction	✓	✓+	✗	✗	RTLs and HL-RTLs express a state as a register. HLS languages abstract over registers and provide a construct to express a global/static variable which is considered a state. DFiant also abstracts over registers, but also abstracts over explicit state declaration. Thanks to its dataflow ordered semantics, DFiant generates a state only if the developer accesses a stream's token history.
	Time Abstraction	✗	✓-	✗	✗	HLS rarely provides time abstraction (e.g., no UART is written in HLS). DF-HDLs have Timer constructs that provide the necessary abstraction, but we have yet to implement them in DFiant.
	Auto-Balanced Manual Pipelining	✓-	✓	✗	✗	DFiant supports manual user pipelining constraint tag to be applied on selective junctions in the design. The DFiant compiler will make sure that all joined/forked paths are properly balanced.
	Automatic Pipelining	✓+	✓	✗	✗	HLS and DFiant support automatic pipelining and automatic flow control. However, DFiant is an HDL and does support control loops. Instead, control loops are explicitly described as state machines and therefore manually scheduled for better performance.
	Automatic Flow-Control	✓+	✓	✓-	✗	
	Control Loop Pipelining	✓+	✗	✗	✗	
Meta-Programming	Conditional & Looped Gen.	✓	✓	✓	✓	
	Recursive Construction	✓-	✓+	✓	✗	
	Inheritance/Typeclasses	✓-	✓	✓	✗	
	Polymorphic Interfaces	✓-	✓	✓	✓-	
Post-Design Capabilities	Legible RTL Output		✓		✓	
	Simulation	✓	✓-			
	Verification	✓				
	Debugging		✓			
Other Properties	Open Source	✗	✓	✓	✓	
	Maturity	✓	✗	✓-	✓	
	Documentation	✓	✗	✓-	✓	
	Familiar Ergonomics	✓-	✓-	✓-	✓	