Hardware Description Languages, the State of the Practice

Brendan Sechter

2026-07-09

Articles A200 and A201 covered the historical trajectory of hardware description languages and the design space that next-generation languages might occupy. This article addresses the third time frame in the same subject, namely the state of the practice as of mid two thousand twenty-six. The question here is not what happened, which A200 answered, nor what could happen, which A201 explored, but what is actually happening in industrial, open-source, and academic hardware design flows.

The article draws on survey data from the annual Wilson Research Group verification study, on vendor toolchain documentation, on ecosystem-maturity indicators in open-source repositories, and on domain-specific adoption reports. The subject resists precise market-share quantification because proprietary hardware design flows do not typically report their language usage, and because adoption patterns vary substantially by geography, industry segment, and company size. The article identifies the qualitative patterns that the available data supports and notes where the data is soft.

The Industrial Mainstream

The industrial mainstream for digital hardware design remains divided between Verilog and VHDL, with SystemVerilog increasingly absorbing both traditions for new design work. The split between Verilog and VHDL persists along geographic and application lines that were established during the standardisation era. North American commercial designers predominantly use Verilog. European designers, defence contractors, and formal-verification proponents predominantly use VHDL. Within the United States, regional patterns further divide the adoption, with Verilog more prevalent on the West Coast and VHDL more prevalent on the East Coast and in government-adjacent contractor work.

Rough survey estimates from recent industry reports place the combined Verilog and VHDL share of new field-programmable-gate-array design work at approximately eighty-five to ninety percent, with the remaining share distributed across higher-level synthesis targets, embedded-domain-specific-language work, and research prototypes. The percentages depend on the sampling methodology and the definition of new design work, so these figures should be treated as directional rather than precise.

SystemVerilog for verification. The twenty twenty-four Siemens Electronic-Design-Automation and Wilson Research Group functional verification study reports that SystemVerilog and the Universal Verification Methodology dominate testbench work in industrial integrated-circuit and application-specific-integrated-circuit projects. The study notes that first-silicon success rates have declined to approximately fourteen percent, the lowest level in two decades, attributed to the growing complexity of system-on-chip architectures, asynchronous clock domains, and safety-critical requirements. The industry response has been increasing adoption of advanced verification methodologies, principally SystemVerilog assertions, the Universal Verification Methodology, and formal verification techniques.

SystemC for system-level modelling. SystemC retains a substantial adoption in system-on-chip design flows where transaction-level modelling is essential. The language is particularly common in mobile-processor and consumer-electronics design work, where the same description must serve both as a simulation model for early software development and as a synthesisable specification for subsequent hardware implementation. SystemC also occupies a substantial share of high-level synthesis flows, where Xilinx Vivado HLS and Siemens Catapult HLS consume SystemC or C-plus-plus source and generate synthesisable register-transfer-level output.

Bluespec in specialised niches. Bluespec SystemVerilog occupies a substantially smaller share than Verilog or VHDL but maintains adoption in specialised niches where the correctness-by-construction argument justifies the tooling investment. Academic processor design work and some high-frequency trading hardware groups report Bluespec adoption. The language's niche status reflects the trade-off between the tooling ecosystem's smaller size and the technical advantages of guarded-atomic-action semantics.

The Vendor Toolchain Landscape

The proprietary toolchain vendors for field-programmable-gate-array and application-specific-integrated-circuit design flows concentrate in three principal companies, namely AMD (formerly Xilinx), Intel (formerly Altera), and Synopsys. Cadence and Siemens Electronic-Design-Automation provide substantial complementary tooling. The concentration places a substantial dependency on the toolchain vendors for any industrial hardware design flow.

AMD Vivado. Vivado, originally released by Xilinx in two thousand twelve and inherited by AMD after its two thousand twenty-two acquisition of Xilinx, serves as the primary toolchain for AMD's Zynq, Kintex, Virtex, and Versal device families. Vivado supports Verilog, SystemVerilog, VHDL, and SystemC source languages, with integrated synthesis, place-and-route, timing analysis, and device programming. The Vivado ML edition adds machine-learning-based optimisation of timing closure and place-and-route decisions.

Intel Quartus Prime. Quartus Prime, the toolchain for the Agilex, Stratix, Arria, Cyclone, and MAX device families, supports the same source languages as Vivado with comparable feature coverage. Intel acquired Altera in two thousand fifteen for approximately seventeen billion United States dollars and subsequently divested a fifty-one percent majority stake to Silver Lake Partners in two thousand twenty-five for approximately eight point seven five billion United States dollars, with Altera returning to its independent Altera Corporation name and Intel retaining a forty-nine percent minority stake. Altera Corporation continues to develop Quartus Prime alongside the device families.

Synopsys Synplify. Synplify Pro provides device-neutral synthesis that targets multiple field-programmable-gate-array device families from a single design flow. Synplify is common in multi-vendor design work where the same design must target devices from more than one vendor, or where the design house wants to maintain vendor independence in its source-code toolchain.

Cadence and Siemens EDA. Cadence and Siemens Electronic-Design-Automation provide complementary tooling in verification, static timing analysis, formal verification, and integrated-circuit physical design. Cadence's JasperGold formal verification platform and Siemens EDA's Questa simulation platform are particularly prominent in verification-heavy flows. Both companies also provide their own synthesis and place-and-route tools that compete with the device-vendor tools in market segments.

The Open-Source Toolchain Landscape

Open-source end-to-end field-programmable-gate-array design flows have matured across the past decade to production-adjacent capability for some device families. The relevant projects form a coordinated ecosystem rather than a monolithic toolchain.

Yosys provides register-transfer-level synthesis, formal-verification-friendly intermediate representations, and substantial device-independent optimisation. The project was started by Claire Wolf in two thousand twelve and has matured into a substantial open-source synthesis toolchain that several downstream projects depend on.

nextpnr provides device-neutral place-and-route, supporting Lattice iCE40, Lattice ECP5, Lattice Nexus, and several other device families through architecture-specific backends. The project is maintained by YosysHQ, which also maintains Yosys.

F4PGA (formerly SymbiFlow) coordinates the open-source tools into a unified vendor-neutral flow for supported device families, principally Lattice iCE40, Lattice ECP5, and selected Xilinx 7-Series devices. The project is associated with CHIPS Alliance, an industry consortium that promotes open-source semiconductor tools.

Project IceStorm provides the reverse-engineered bitstream documentation for Lattice iCE40 devices that Yosys and nextpnr depend on for device-specific place-and-route. The project was principally the work of Claire Wolf and was the first fully open-source end-to-end field-programmable-gate-array toolchain.

The open-source flow delivers end-to-end synthesis from Verilog or Amaranth source to device bitstream without proprietary dependencies for supported device families. Adoption is substantial in academic, hobbyist, and open-hardware contexts. Industrial adoption remains limited principally by device-family coverage gaps. The open-source flow does not currently support AMD’s UltraScale, Versal, or Zynq device families in a production-ready form, which excludes substantial industrial usage.

Embedded-Domain-Specific-Language Adoption

Article A200 covered the embedded-domain-specific-language revival of the twenty tens. This section records what the individual revival languages have achieved in current adoption.

Chisel. Chisel has achieved substantial adoption in academic and industrial RISC-V design work. The Rocket Chip generator, which produces RISC-V processor implementations, is written in Chisel and is used across several academic and industrial RISC-V projects. SiFive, the commercial RISC-V vendor founded in two thousand fifteen by Krste Asanović, Yunsup Lee, and Andrew Waterman from the University of California Berkeley, the same team that originated Chisel and the RISC-V instruction set architecture, uses Chisel in its core processor design work. Chisel also serves as the source language for the FireSim field-programmable-gate-array-accelerated simulation platform, which several academic groups use for computer architecture research.

Amaranth. Amaranth occupies a substantial share of the hobbyist and open-source field-programmable-gate-array design space. The language’s Python foundation and its integration with the Yosys backend make it the default choice for open-source hardware projects that target Lattice iCE40 and ECP5 devices. Amaranth has produced substantial open-source hardware libraries, including LiteX system-on-chip generators that target open-source soft processors such as VexRiscv and NEORV32.

SpinalHDL. SpinalHDL has achieved adoption in several open-source RISC-V processor projects, notably VexRiscv, which is a commonly used soft processor in open-source field-programmable-gate-array designs. The language’s Scala-based generator paradigm supports substantial parameterisation of processor variants from a single source description.

Clash. Clash occupies a smaller adoption share than the Scala-based or Python-based alternatives, principally because Haskell is less commonly used than Scala or Python in industrial and academic software engineering. The language retains adoption in research groups that work in functional programming and value Haskell's type system for hardware description work.

MyHDL. MyHDL retains a small but persistent adoption in educational and prototype contexts. The language predates Amaranth and introduced several of the design patterns that Amaranth later adopted. Current adoption is substantially smaller than Amaranth's because Amaranth's tighter integration with Yosys and its more active maintenance have displaced MyHDL in much of the Python-based hardware design space.

Formal Verification Adoption

Formal verification for hardware description has grown from a research-only niche to a substantial component of industrial verification flows across the past decade. The growth is principally driven by the same design-complexity forcing function that motivates the embedded-DSL revival, combined with the declining first-silicon success rates that the Wilson Research Group study documents.

Property checking and model checking. Formal verification tools that consume SystemVerilog assertions or Property Specification Language descriptions have substantial adoption in industrial verification flows. Cadence's JasperGold, Synopsys VC Formal, and Siemens EDA Questa Formal are the principal industrial-scale platforms. Adoption has grown across the past decade because the increasing complexity of integrated-circuit designs exceeds the capabilities of simulation-only verification flows.

Coq-based hardware description research. Kami and Koika, both developed at the Massachusetts Institute of Technology under Adam Chlipala's Programming Languages and Verification group, demonstrate formal-verification-integrated hardware description at the source-language level. Kami provides Coq-based modular hardware verification and has been used to verify multicore cache-coherent systems. Koika provides a Bluespec-inspired rule-based hardware description with a formally verified compiler to gates. Both projects remain principally academic at the time of writing, with industrial adoption limited to projects that prioritise correctness proofs over tooling ecosystem size.

Adoption trajectory. The Wilson Research Group study reports that formal verification adoption has grown from approximately thirty percent of industrial verification projects in two thousand fourteen to approximately sixty percent in two thousand twenty-four. The percentages vary across industry segments, with safety-critical automotive and aerospace segments reporting substantially higher adoption than consumer-electronics segments.

Additional and Emerging Languages

Several hardware description languages that articles A200 and A201 did not substantively cover occupy niches in the current landscape.

Silice. Silice is an open-source hardware description language developed by Sylvain Lefebvre at the National Institute for Research in Digital Science and Technology in France. The language provides a C-like syntax that compiles to Verilog and integrates with the open-source Yosys and nextpnr toolchain. Silice is particularly notable for its demonstration corpus, which includes implementations of Doom and Quake level renderers running on low-cost Lattice ECP5 devices. The language occupies a niche in research and hobbyist work where higher-level abstraction than Verilog combined with open-source toolchain compatibility is desired.

DFHDL. DFHDL, also known as DFiant HDL, is a Scala-based hardware description language developed by the DFiantHDL organisation. The language distinguishes itself from Chisel

and SpinalHDL by providing multiple abstraction levels in a single source form, including dataflow, register-transfer, and event-driven abstractions. The dataflow abstraction supports timing-agnostic and device-agnostic hardware description that compiles to platform-specific implementations. DFHDL occupies a research niche at the time of writing but demonstrates a design point that combines multiple abstraction levels in a way that none of Verilog, VHDL, Chisel, or Amaranth currently matches.

LiteX and Migen family. LiteX is a system-on-chip generator framework built on Migen and Amaranth. It provides substantial libraries of pre-designed peripherals, memory controllers, and soft processor cores that Amaranth-based projects can compose into complete system-on-chip designs. LiteX has achieved substantial adoption in academic and open-source hardware projects, particularly for integration with soft processor cores such as VexRiscv and NEORV32.

Migen. Migen, the predecessor to Amaranth, retains some adoption in legacy projects that have not migrated to Amaranth. The Migen ecosystem has substantially shifted to Amaranth over the past several years, so current adoption is principally in maintenance mode rather than in active new development.

PyMTL and MyHDL family. PyMTL, developed at Cornell University, provides a Python-based hardware description and simulation framework that targets academic computer architecture research. The framework has maintained adoption in academic contexts where integration with Python-based analysis tools is valued. Adoption outside the originating academic groups is limited.

Domain-Specific Adoption Patterns

Adoption patterns vary substantially across industry segments. Recording the patterns by domain identifies where each hardware description language occupies its principal niche.

Automotive and aerospace. Safety-critical automotive and aerospace design work predominantly uses VHDL and SystemVerilog with substantial formal verification integration. The verification-methodology adoption in these segments exceeds the industry average, principally because the first-silicon success requirements are substantially more stringent than in consumer segments. Chisel and Amaranth have some adoption in research and prototype work but have not achieved production adoption in these segments.

Consumer electronics and mobile. Consumer electronics and mobile processor design predominantly uses Verilog and SystemVerilog with SystemC-based transaction-level modelling for early software development. The verification-methodology adoption in these segments is substantial but somewhat behind the safety-critical segments, reflecting the different first-silicon success requirements.

RISC-V processor design. RISC-V processor design has substantial adoption of Chisel and SpinalHDL alongside traditional Verilog and SystemVerilog work. The RISC-V ecosystem's academic origins and its close integration with the Berkeley Par Lab where Chisel was developed have established the embedded-DSL revival languages in this segment to a greater extent than in other industrial segments.

Academic computer architecture. Academic computer architecture research uses a diverse mix of hardware description languages, with Chisel, Bluespec SystemVerilog, PyMTL, and Kami and Koika all occupying substantial shares of the academic project space. The academic segment has adopted the embedded-DSL revival languages substantially earlier than industrial segments, principally because academic projects prioritise design elegance and generator flexibility over tooling ecosystem maturity.

Hobbyist and open-source hardware. Hobbyist and open-source hardware projects predominantly use Amaranth, Verilog, and Silice with the open-source Yosys and nextpnr

toolchain. This segment has the highest adoption of open-source-first design flows because proprietary toolchains represent a substantial cost and dependency that hobbyist projects cannot easily justify.

Adoption Trajectory

The overall trajectory across the past decade shows several persistent patterns. The traditional Verilog and VHDL mainstream has proven remarkably durable, with SystemVerilog gradually absorbing both traditions for new design work. The embedded-DSL revival languages have achieved substantial adoption in academic and open-source contexts, but have not displaced Verilog and VHDL in industrial mainstream flows. Formal verification has grown substantially across the same period, principally driven by the declining first-silicon success rates and the increasing complexity of system-on-chip designs. Open-source toolchains have matured into production-adjacent capability for some device families, particularly Lattice iCE40 and ECP5, but have not achieved device-family coverage sufficient to displace proprietary toolchains for most industrial work.

The most substantial recent adoption shift is the integration of formal verification methodologies into industrial verification flows. The Wilson Research Group study's reported growth from approximately thirty percent to approximately sixty percent formal-verification adoption over a decade represents a substantial industry response to the design-complexity forcing function that articles A200 and A201 discussed. The trajectory suggests that formal verification integration will continue to grow across the coming decade, with increasing pressure on hardware description languages that do not support source-language-level formal verification.

The most substantial open question for the coming decade is whether any of the embedded-DSL revival languages or the formal-verification-integrated research languages will achieve industrial mainstream adoption, or whether the mainstream will remain divided between Verilog and VHDL with SystemVerilog extensions for several more decades. The historical record covered in article A200 suggests that mainstream displacement requires substantial industrial commitment and takes several decades to complete, so significant mainstream shifts in the coming decade appear unlikely absent substantial external forcing functions.

Conclusion

The state of the practice in hardware description languages as of mid two thousand twenty-six shows a persistent industrial mainstream of Verilog and VHDL with SystemVerilog absorbing new design work. Formal verification adoption has grown substantially across the past decade, principally driven by declining first-silicon success rates. The embedded-domain-specific-language revival languages, including Chisel, Amaranth, SpinalHDL, and Clash, have achieved substantial adoption in academic, open-source, and some research-adjacent industrial contexts, but have not displaced the traditional mainstream. Open-source toolchains, principally Yosys, nextpnr, and F4PGA, have matured into production-adjacent capability for some device families but have not achieved device-family coverage sufficient to displace proprietary toolchains for most industrial work.

The adoption trajectory across the coming decade appears principally shaped by formal verification integration and by open-source toolchain device-family expansion. Mainstream displacement of Verilog and VHDL by any of the embedded-DSL revival languages appears unlikely absent substantial external forcing functions. Articles A200 and A201 covered the historical and design-space dimensions of this subject. This article completes the three-timeframe survey by recording the state of the practice that the historical trajectory has produced and the design space must engage with.

References

Reference

- AMD Vivado design suite
- Amaranth hardware description library
- Cadence Design Systems
- Catapult HLS high-level synthesis
- CHIPS Alliance
- DFHDL Scala-based dataflow hardware description language
- FireSim field-programmable-gate-array-accelerated simulation
- Intel Quartus Prime design software
- LiteX system-on-chip generator framework
- Rocket Chip RISC-V processor generator
- Siemens Electronic-Design-Automation
- Silice hardware description language
- Synopsys Synplify Pro
- Vivado HLS high-level synthesis

Related Post

- A History of Hardware Description Languages, article A200 in this blog
- The Design Space for Next-Generation Hardware Description Languages, article A201 in this blog

Research

- Choi, Vijayaraghavan, Sherman, Chlipala, and Arvind, Kami, a Platform for High-Level Parametric Hardware Specification and its Modular Verification, ICFP 2017
- Bourgeat, Pit-Claudel, and Chlipala, The Essence of Bluespec, a Core Language for Rule-Based Hardware Design, PLDI 2020 (Koika)
- Twenty Twenty-Four Wilson Research Group Integrated-Circuit and Application-Specific-Integrated-Circuit Functional Verification Trend Report