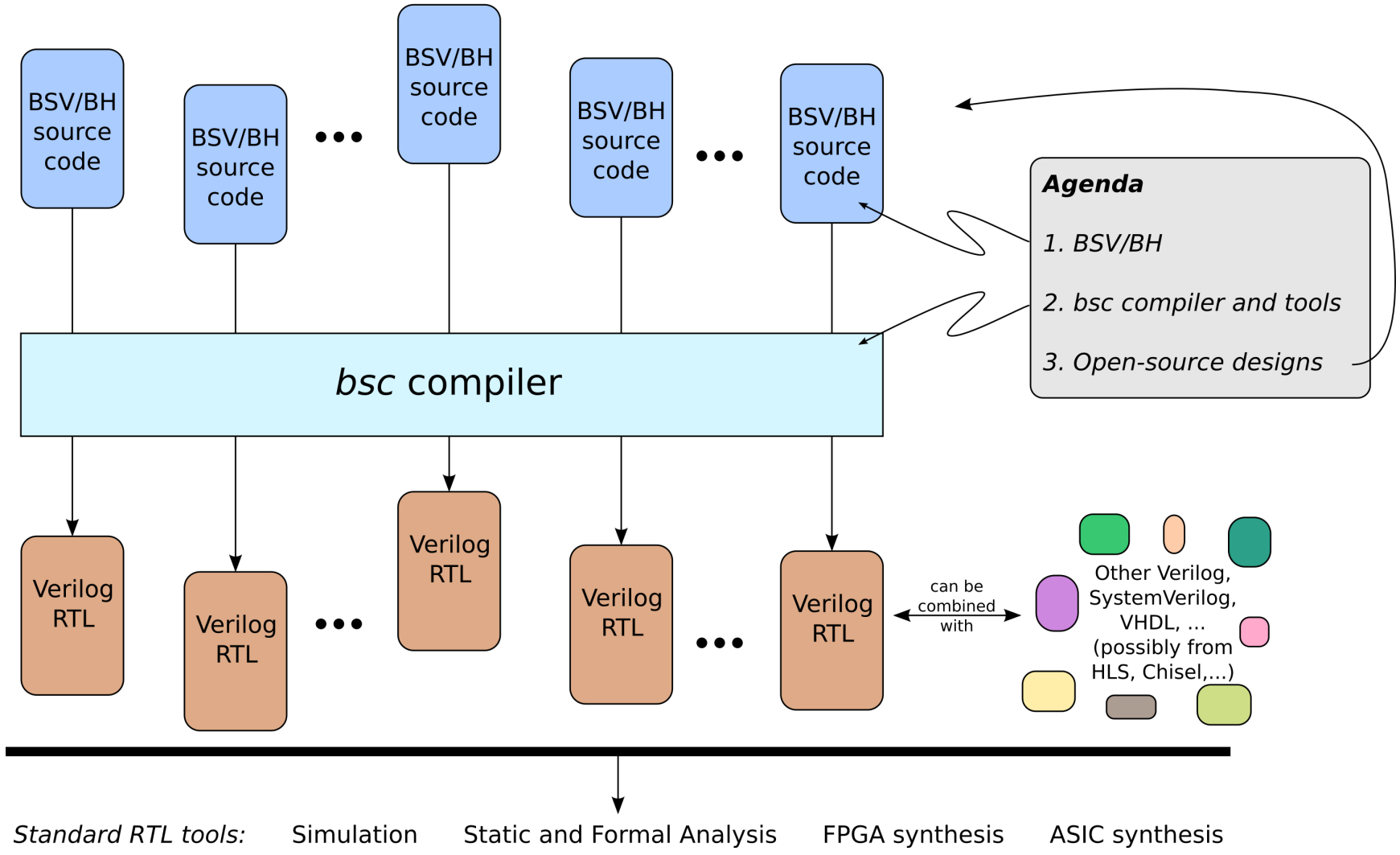
The Open-Source Bluespec bsc Compiler and Reusable Example Designs

Julie Schwartz, Niraj N.Sharma, Darius Rad, Ken Takusagawa,
Joe Stoy & Rishiyur S. Nikhil

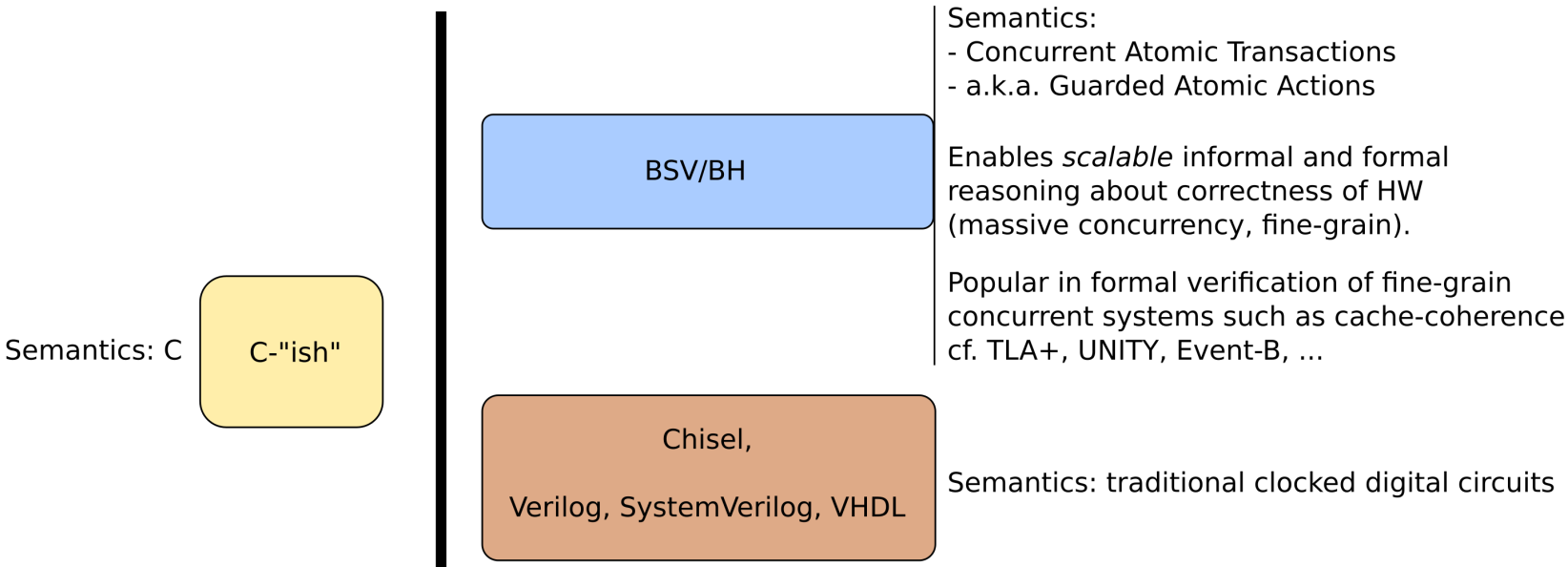
WOSET 2021: Workshop on Open-Source EDA Tools

November 4, 2021

Overview


BSV/BH source code
BSV/BH source code
BSV/BH source code
BSV/BH source code
BSV/BH source code
Agenda
1. BSV/BH
2. bsc compiler and tools
3. Open-source designs
bsc compiler
Verilog RTL
Verilog RTL
Verilog RTL
Verilog RTL
Verilog RTL
can be combined with
Other Verilog, SystemVerilog, VHDL, ... (possibly from HLS, Chisel,...)
Standard RTL tools:
Simulation
Static and Formal Analysis
FPGA synthesis
ASIC synthesis

BSV/BH compared with others: semantics (behavioral model)



HLS/OpenCL

Compiler, not source code, determines microarchitecture (can be guided with hints)

Not universal (limits on automatic parallelization in C compilers).

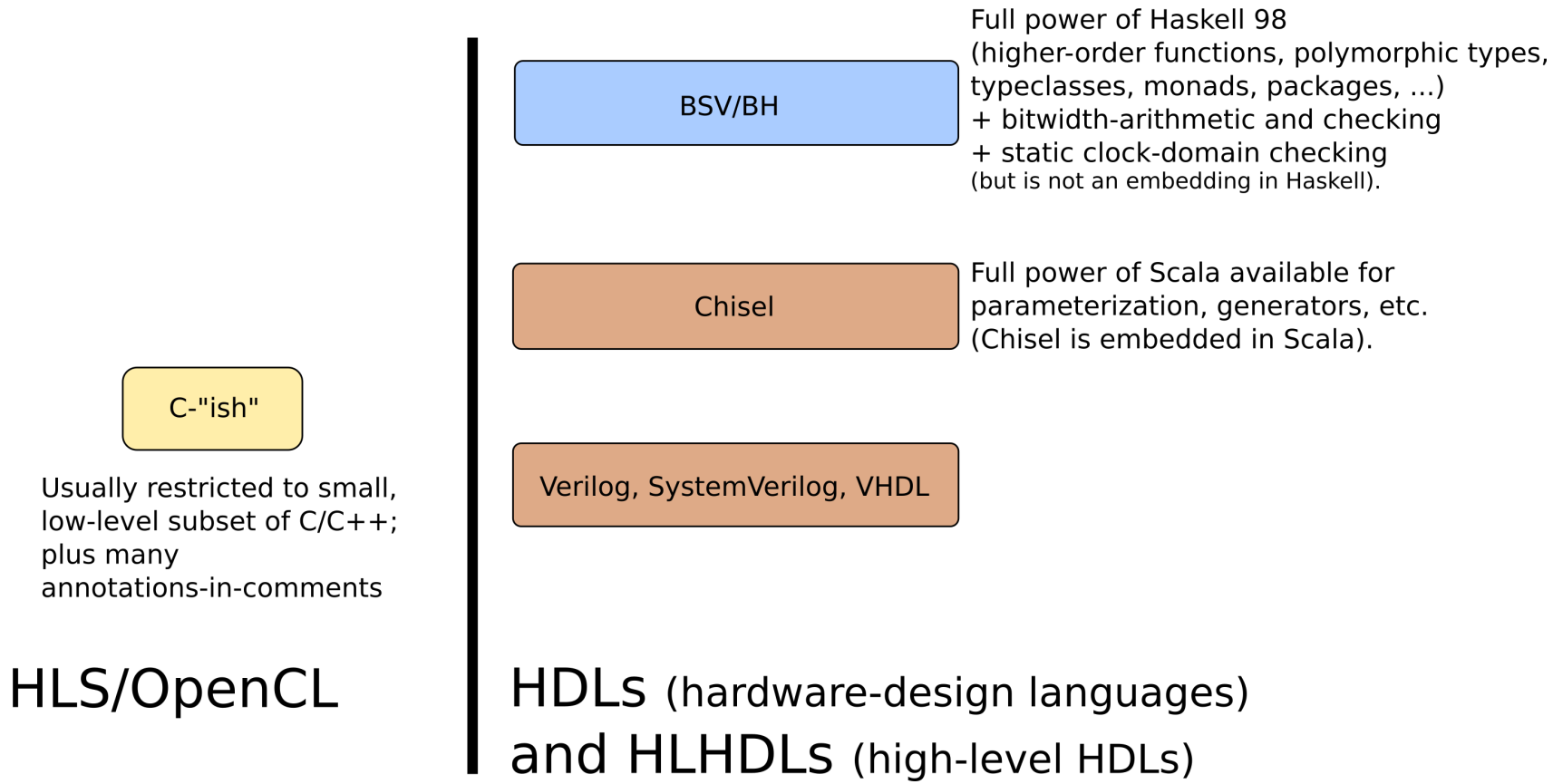
Sweet spot:
dense matrix-and-loop codes, very simple control structures.

HDLs (hardware-design languages) and HLHDLs (high-level HDLs)

Source code, not compiler, determines microarchitecture precisely (designer has direct, full control).

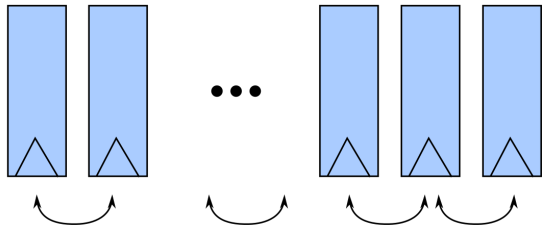
Universal, for any application area (not restricted to any "sweet spot").

BSV/BH compared with others: modern Programming Language features



An example fragment of BSV, for flavor

Concurrent bubblesort of an array of registers



Instantiate vector of registers
- strongly-typed polymorphic size 'n_t',
- strongly-typed register contents 'Maybe #(t)',
  (polymorphic type 't' but
  tagged with Valid/Invalid bits ('Maybe')
- using standard vector higher-order function 'replicateM'
  on standard register module instantiator 'mkReg'
  with reset value 'Invalid'

```
Vector #(n_t,
         Reg #(Maybe #(t))) xs <- replicateM (mkReg(tagged Invalid));

for (Integer i = 0; i < valueOf(n_t)-1; i = i+1)
   rule rl_swap_i (xs [i] > xs [i+1]);
      xs [i]   <= xs [i+1];
      xs [i+1] <= xs [i];
   endrule
```

Implicitly generative loops
(can also be written in Haskell style
maps, lists, recursion, ...
since rules are first-class values).

Though values are polymorphic (type 't'),
comparision '>' is strongly-typed
(using Haskell *typeclasses*).

Adjacent rules (updating common registers) arbitrated automatically due to atomic transaction semantics

Full example (including simpler lead-up examples without parameterization and generators):

`https://github.com/BSVLang/Main.git`
`Tutorials/BSV_Training/Example_Programs/Eg03e_Bubblesort/src_BSV/Bubblesort.bsv`

The true power of *rules* is revealed as we scale atomicity across parameterized module boundaries.
The true power of Haskell features is revealed as we scale to large, parameterized designs.

BSV/BH libraries: a sampling

Basic data types
- Bit vectors
- structs, tagged unions, enums
  (w. automatic $display)
- Vectors (of arbitrary types,
  including modules, interfaces)
  and higher-order functions
  map, fold, scan, zip, ...
- One-hot

Storage Structures
- Default-value Register
- Register File
- BRAM
- Memory Interface

Connectivity
- Connectable typeclass
- GetPut interfaces
- Credit-based GetPut
- ClientServer interfaces
- AXI4, AXI4-Lite interfaces, crossbars

Primitive modules
- Registers
- Concurrent Registers

Math
- Floating Point
- FixedPoint
- Complex
- IFFT
- Random Numbers (LFSRs)

Clock and Reset Domains
- Statically checked Clock and Reset types
- Clock Generators and Transformers
- Clock Division
- Clock Gating and Clock selection
- Synchronizers: Bit, Pulse, Word, FIFO, RAM

FIFOs
- Simple FIFOs
- BRAMFIFO
- LevelFIFO
- SyncFIFO
  (clock-domain crossing)
- GearBox (width-change)
- MIMO (multipe-in/out)
- Reorder Buffer

Counters
- Arbiter
- Up/down counters
- Gray counters
- BufIndex
- WrapNumber

Crypto/signature
- CRC
- AES
- SHA256 (planned)

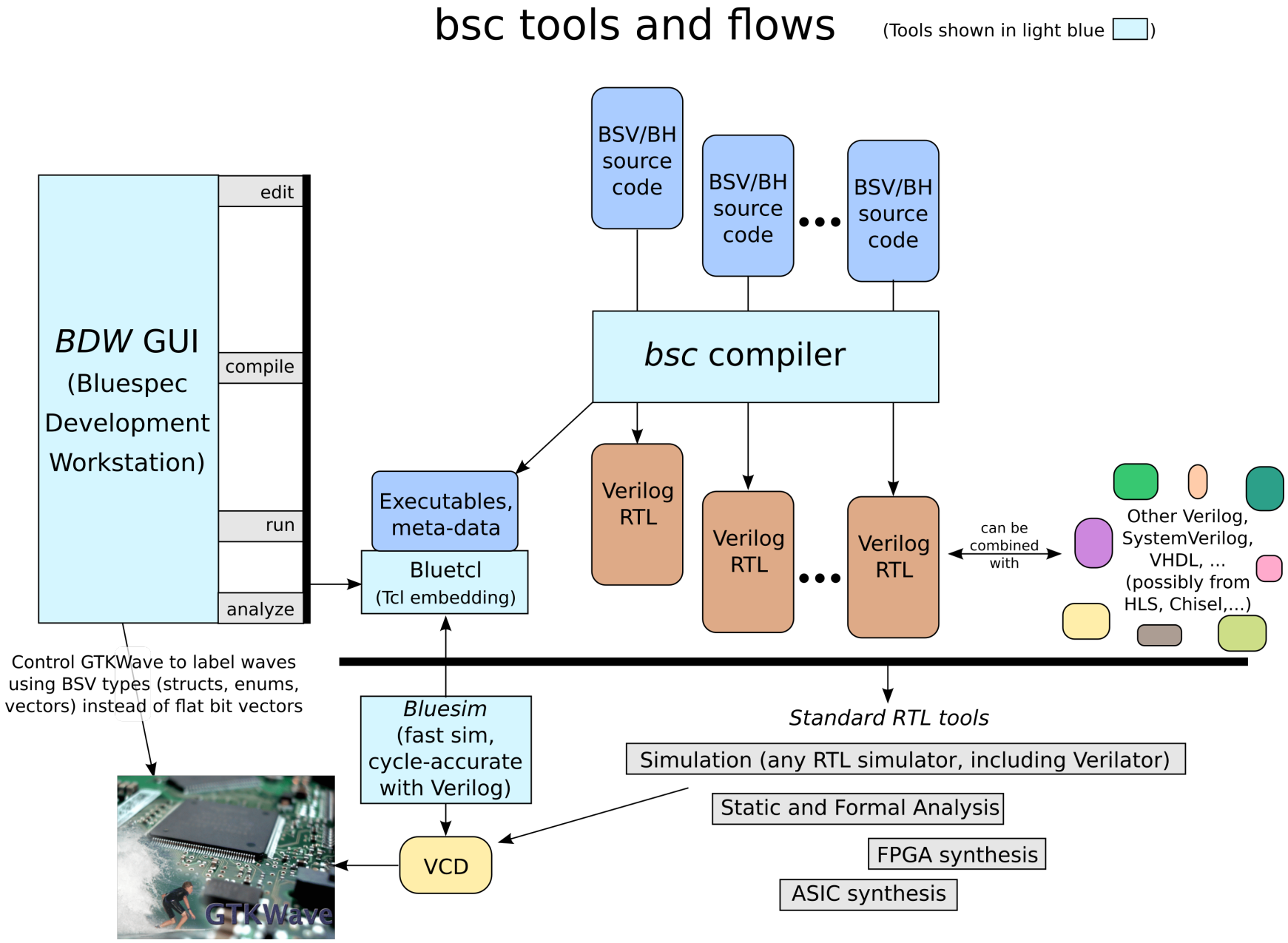
FPGA-specific
- BRAM
- BRAMFIFO
- Xilinx IP wrappers

Composable, Controllable FSMs
- FSM interface
- Smts (abstracted FSM components)
- FSM builders (general, one-shot, stall control)

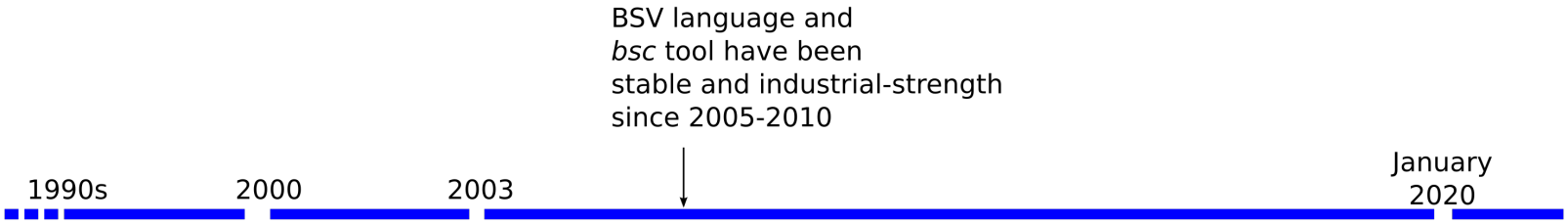
Display
- Format objects
- FShow typeclass

Advanced
- Monadic plumbing
- Configuration bus

Most of this is written in BSV


bsc tools and flows
(Tools shown in light blue)
BSV/BH source code
BSV/BH source code
BSV/BH source code
bsc compiler
BDW GUI (Bluespec Development Workstation)
edit
compile
run
analyze
Executables, meta-data
Bluetcl (Tcl embedding)
Verilog RTL
Verilog RTL
Verilog RTL
can be combined with
Other Verilog, SystemVerilog, VHDL, ... (possibly from HLS, Chisel,...)
Control GTKWave to label waves using BSV types (structs, enums, vectors) instead of flat bit vectors
Bluesim (fast sim, cycle-accurate with Verilog)
Standard RTL tools
Simulation (any RTL simulator, including Verilator)
Static and Formal Analysis
FPGA synthesis
ASIC synthesis
VCD
GTKWave

BSV/BH and *bsc* timeline and maturity



Research at MIT on compiling concurrent atomic transactions to Verilog RTL [Hoe & Arvind 2000]

bsc created and initial use inside Sandburst Corp. (enterprise-class network routing chips; acquired by Broadcom)

bsc licensed as EDA tool to many companies; no-cost licenses for teaching and research.

Some top-ten semis use it for complex IPs in ASIC products.

Some top-ten semis use it for rapid architecture exploration, on FPGAs, of future ASIC products.

Researchers at dozens of universities use it for research/teaching (MIT, CMU, U.T.Austin, U.Cambridge, Ga.Tech, Seoul Natl. U., IIT Madras, IIT Bombay, T.U.Darmstadt, U.Toronto, ...)

bsc released free and open-source

Significant example designs (open source)

The paper briefly describes and provides links for a number of significant, open-source designs (by no means an exhaustive list). All of them run on FPGAs

- Complete Linux-capable RISC-V system running on Xilinx VCU118 boards and Amazon AWS ("AWSteria_RISCV_Virtio"). [**Demo** and next slides.]

- Several RISC-V CPUs from Bluespec, Inc. and MIT (Piccolo, Flute, Toooba/RISCY-OOO). Piccolo and Flute are highly parameterized on the RISC-V ISA options, generating a range of CPUs from small, embedded to powerful, Linux-capable.
- These Bluespec CPU repositories also include several variants of cache systems, Debug Modules, Interrupt Controllers, Timer interruptors, AXI4 and AXI4-Lite interfaces, fabrics and components, etc.
- Security-enhanced CPU designs (MIPS and RISC-V) from U. Cambridge.
- "Shakti" RISC-V CPU family from IIT Madras.
- "CONNECT" NoC (network-on-chip) generator from Carnegie Mellon University.
- "BlueCheck" automatic hardware checker modelled on Haskell QuickCheck from U.Cambridge.

Please see paper for open-source repository links for all the above.

Hardware setup for demo

Intel NUC mini-server
running Debian Linux

("host side")

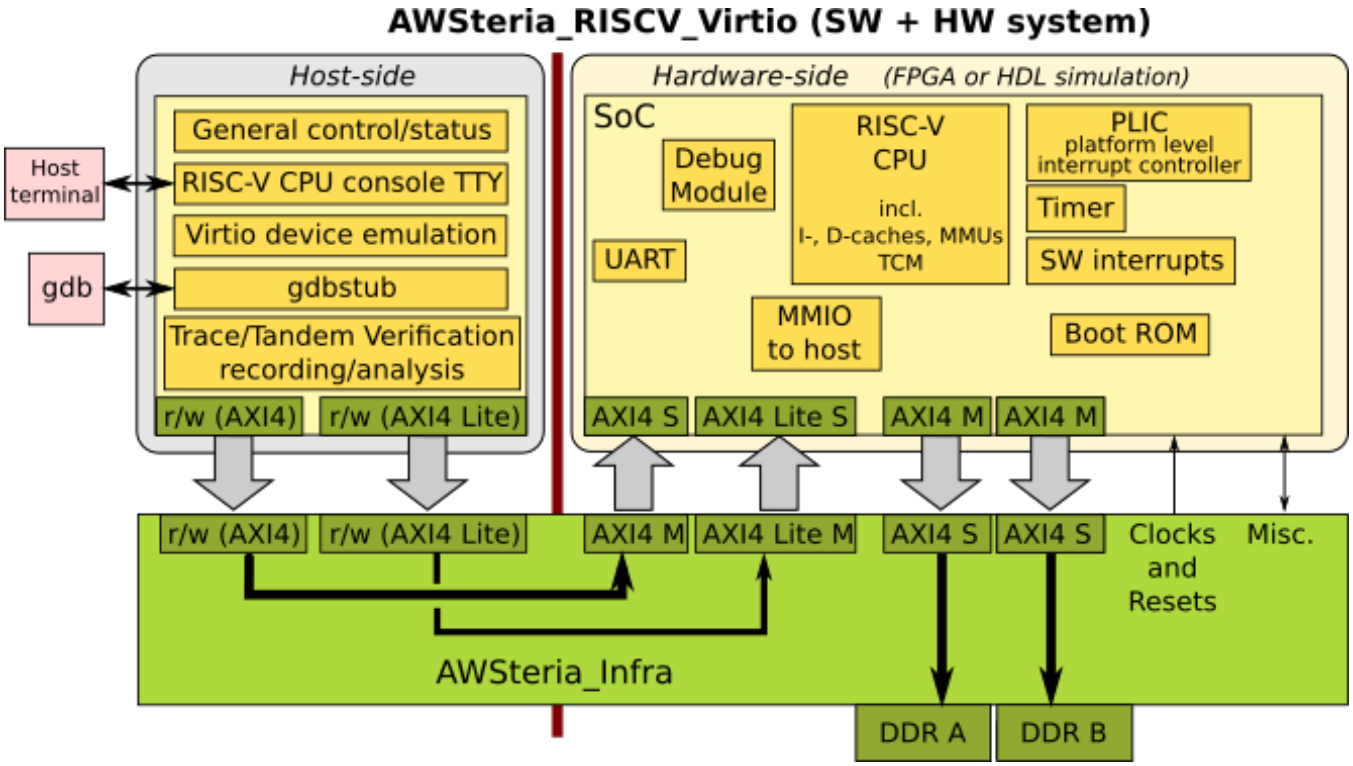
Xilinx VCU118
inside a Node Pro PCIe cage
with PCIe connected via
Thunderbolt cable to NUC

("hardware/FPGA side")

Thunderbolt cable
carrying PCIe

A significant open-source example design (live demo)

- Runs in simulation and on Xilinx VCU118 and Amazon AWS; boots FreeBSD (also Linux capable). Virtio support provides networking and block-storage to FreeBSD/Linux on RISC-V.
- RISC-V CPU on FPGA-side is controllable from GDB on host-side.



Entire hardware-side, and much of AWSteria_Infra is written in BSV. Host-side code is in C.
Flute CPU, RISC-V components (Debug Module, PLIC, Timer, SW intr), AXI: `https://github.com/bluespec/Flute`
AWSteria_RISCV_Virtio: `https://github.com/GaloisInc/BESSPIN-CloudGFE/tree/rsn3/AWSteria_RISCV_Virtio` (rsn3 branch; will move to main branch)
AWSteria_Infra: `https://github.com/bluespec/AWSteria_Infra`

Conclusion

We invite the community to join us in using BSV/BH, and in continued development of the *bsc* compiler and tools.

Key links:

- *bsc* compiler source code (written in Haskell), including libraries: `https://github.com/B-Lang-org/bsc`
- Additional contributed libraries: `https://github.com/B-Lang-org/bsc-contrib`
- Bluespec Development Workstation (GUI): `https://github.com/B-Lang-org/bdw`
- Tutorial materials. Comprehensive textbook(s) are in progress. Meanwhile, the following links are useful:
  - Bluespec, Inc.'s training materials `https://github.com/BSVLang/Main`: working examples, slides, and a PDF copy of the "BSV by Example" book.
  - ICFP 2020 tutorial: `https://github.com/rsnikhil/ICFP2020_Bluespec_Tutorial`
  - Video recording of ICFP 2020 tutorial: `https://www.youtube.com/watch?v=JCxE3JQAXY0`

Please see paper for open-source repository links for the rest of the mentioned artefacts.

Thank you!