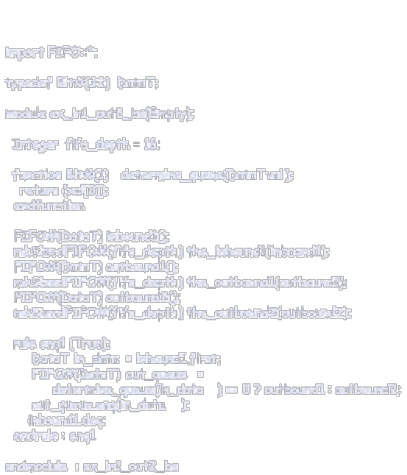
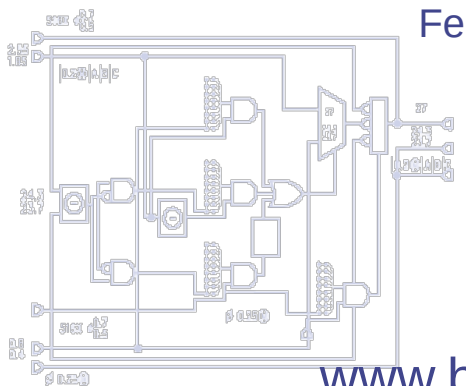
Bluespec for High-Level Hardware Design

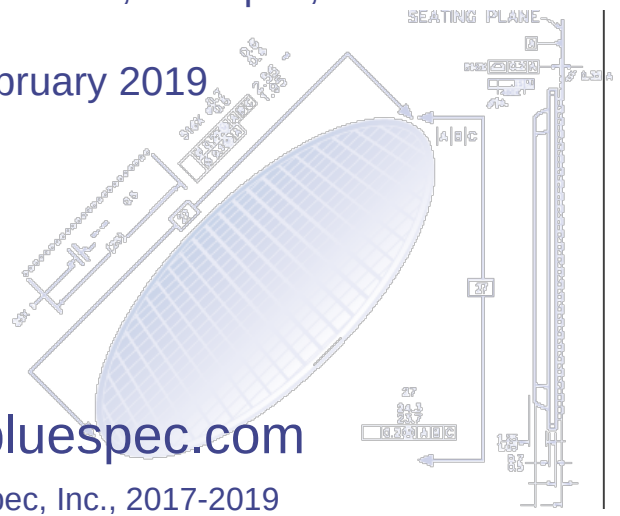
A comparison of HDLs, HLHDLs and HLS

Rishiyur S. Nikhil, PhD
CTO and co-founder, Bluespec, Inc.

February 2019







www.bluespec.com

HDL: Hardware Design Language
HLHDL: High-Level HDL
HLS: High-Level Synthesis

Introduction

Background: Since the mid-1980s, Verilog and VHDL have been the standard Hardware Design Languages (HDLs).

Since 2005, Verilog has evolved into SystemVerilog, with many features of modern programming languages. Many of these modern features (such as object-orientation) are not synthesizable, and are primarily used only in simulation (verification testbenches).

In the following slides, we'll refer to these HDLs collectively as *RTL*.

In the last 10 years, new HLHDLs (High Level Hardware Design Languages) have emerged:

- Bluespec BSV and Classic
- Chisel
- C/C++ with High-Level Synthesis (HLS)

In the following slides we discuss these new HLHDLs, along with RTL.

Two syntaxes (your preference), same semantics

```
module mkAXI4_Fabric #(function Tuple2 #(Bool, Bit #(TLog #(num_slaves)))
                               fn_addr_to_slave_num (Bit #(wd_addr) addr))
                     (AXI4_Fabric_IFC #(num_masters, num_slaves, wd_id, wd_addr, wd_data, wd_user));
   …
   // Transactors facing masters
   Vector #(num_masters, AXI4_Slave_Xactor_IFC  #(wd_id, wd_addr, wd_data, wd_user))
      xactors_from_masters <- replicateM (mkAXI4_Slave_Xactor);
   …
   rule rl_reset (rg_reset);
      $display ("%0d: AXI4_Fabric.rl_reset", cur_cycle);
   endrule
   …
   method Action reset () if (! rg_reset);
      rg_reset <= True;
   endmethod
endmodule: mkAXI4_Fabric
```

"BSV": SystemVerilog-ish
Packages are files with ".bsv" extension

```
mkAXI4_Fabric :: (Bit wd_addr → (Bool, Bit (TLog num_slaves)) → Module AXI4_Fabric_IFC num_masters num_slaves wd_id wd_addr wd_data wd_user
mkAXI4_Fabric    fn_addr_to_slave_num =
  module
   …
    -- Transactors facing masters
    xactors_from_masters :: Vector  num_masters  (AXI4_Slave_Xactor_IFC  wd_id wd_addr wd_data wd_user)
        <- replicateM mkAXI4_Slave_Xactor
    …
    "rl_reset": when rg_reset ==>
        $display  "%0d: AXI4_Fabric.rl_reset"  cur_cycle
    …
    reset ()
        rg_reset := True;
     when (not rg_reset)
```

"Classic": Haskell-ish
Packages are files with ".bs" extension

- These are merely different front-end parsers for *bsc.*
- Types and semantics are identical.
- You can mix-and-match .bsv and .bs packages freely.

In the following, "Bluespec" will refer to either BSV or Classic

bluespec

Note

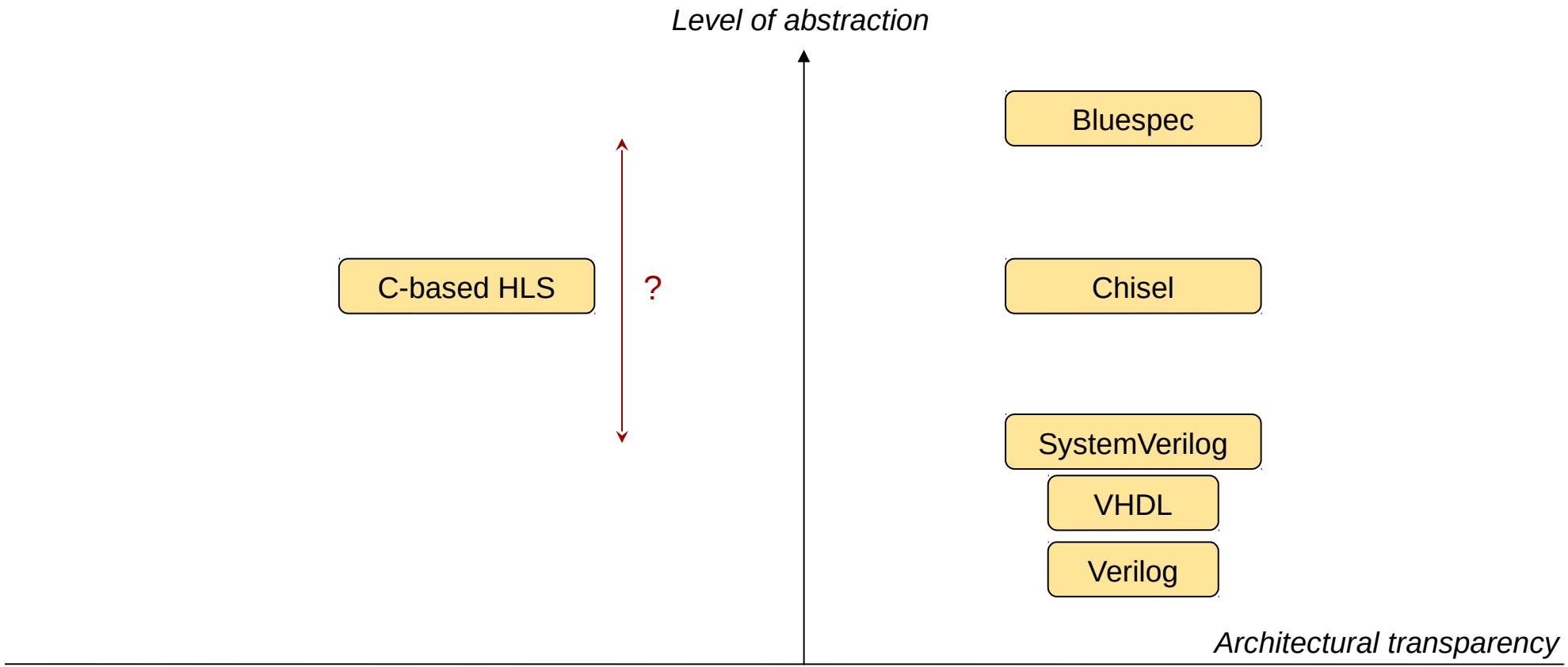
Assessing the "level of abstraction" or "expressive power" of any formal language is a subjective exercise.

The assessments on the following slides are the author's personal opinions, based on deep technical knowledge and experience spanning nearly four decades (going back to Fortran IV and Algol 60):

- Continuously studying and using a wide variety of advanced programming languages (both production languages and research languages)
- Designing a handful of advanced programming languages for parallel processing and hardware description, and implementing their compilers and runtime systems
- Designing and implementing dozens of DSLs (domain specific languages)
- Studying leading research on languages, compilers and language runtime systems
- Studying and using formal language semantics (denotational, axiomatic, operational semantics) for sequential and concurrent languages.

The author would be happy to have a deeper technical discussion on any of the topics that follow.

Designer's Control over Architecture



Opaque (tool-determined)

Architecture determined internally by tool
(with some user-level directives for guidance; can make quality-convergence difficult)

Transparent (deterministic/ predictable)

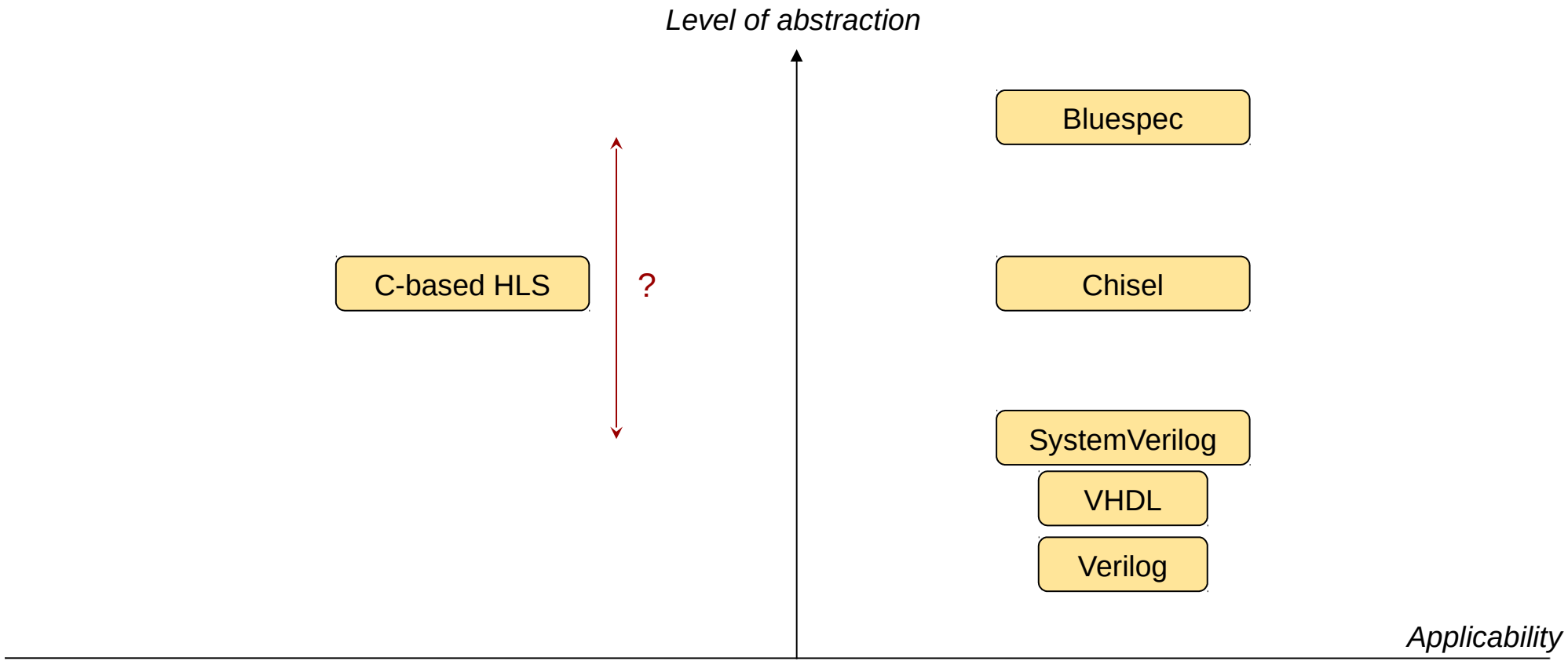
Architecture specified by designer

bluespec

Designer's Control over Architecture: Notes

- In hardware design the first-order determinant of quality is *architecture / microarchitecture.*
  - Just as, in software, the first-order determinant is *algorithm.*
- With RTL, Chisel and Bluespec (right-hand side of the previous slide), the designer describes architecture / microarchitecture precisely and directly.
- With HLS (left-hand side), the tool customizes a proprietary internal datapath+control architecture for the application, providing some knobs (such as loop unrolling). The generated architecture can be inscrutable and surprising.
- For all HDLs, raising the level of abstraction provides greater architectural flexibility-- easier to iterate changes and do tuning, robustly.

Applicability across range of designs



Loop-and-array computations

Depends on automatic parallelization, whose "sweet spot" is loop-and-array computations (e.g., signal processing), not complex control systems (e.g., CPUs, interconnects, …)

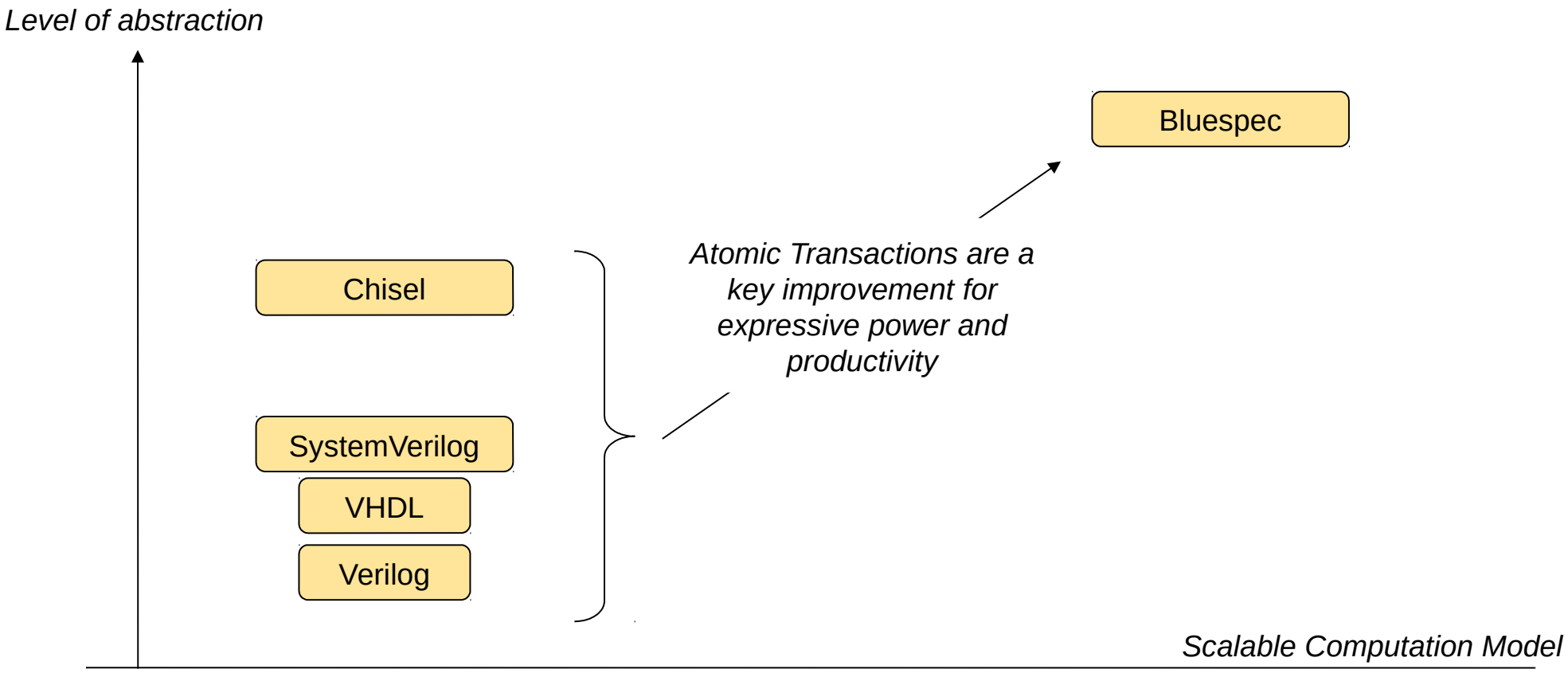
Universal

Architecture specified by designer

Applicability across range of designs: Notes

- RTL, Chisel and Bluespec (right-hand side of the previous slide) can be used for *any* kind of hardware design, since the designer expresses architecture directly.
  - This includes heavily control-oriented applications (CPUs, MMUs, cache-controllers, complex protocol engines, …)
- HLS (left-hand side) is best used where the automatic parallelization and tool-generated architectures are a good match: loop-and-array computation kernels.

From classical Clocked Logic to Atomic Transactions



Classical clocked logic

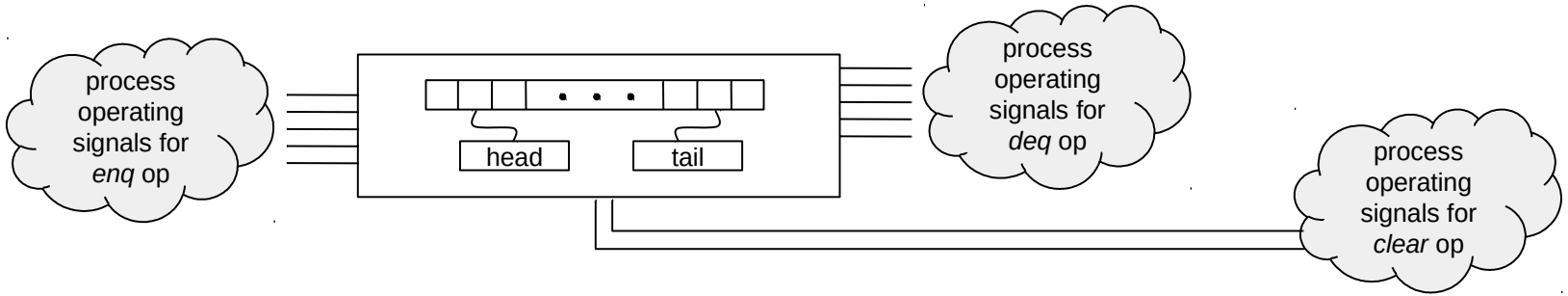
Traditional "always-block" processes on registers and wires; Module interfaces are wires.

Rules with globally atomic semantics

Natural "condition-action" spec of behavior; module interfaces are "methods" with rule semantics.

From classical Clocked Logic to Atomic Transactions (contd.)

Classical clocked logic can be tricky even for individual small modules. Consider a shared queue:



- What if *enq(X)* and *clear* are operated concurrently? Is the queue empty, or does it contain *X*?
- Can *enq(X)* and *deq* operate concurrently when the queue is empty (passing *X* through from *enq* to *deq*)?
- Can *enq(X)* and *deq* operate concurrently when the queue is full (placing *X* in the slot vacated by *deq*)?

With classical Clocked Logic, a designer must anticipate all these combinations of concurrent interactions (which can be quite subtle) and produce appropriate control logic. (We have seen commercial IP that fails in this regard!)

With Bluespec's Atomic Transactions, the compiler produces control logic that behaves predictably and correctly under all concurrent activity.

Keeping track of interactions due to concurrency gets even more complicated as we try to reason about systems with multiple modules (i.e., bad scaling).

BSV rules are *globally atomic*, making it easy to reason about a process' actions even if it spans module boundaries.

Maturity of Bluespec BSV / Classic and its tools

- Bluespec BSV and Classic have been in production use in industry, academia and research since 2005
  - Has been used for designing complex IP blocks, CPUs and SoCs, synthesizable verification IP, emulation infrastructure, and more
- The languages are very stable; extensive IP libraries are available; extensive training materials are available.

- The key tool is *bsc*, Bluespec's compiler from Bluespec BSV and Classic into synthesizable Verilog (which is then processed by standard Verilog tools).
  - Has been used both for ASIC and FPGA targets.
  - Final hardware quality comparable to designs directly coded in RTL.
- *bsc* is well-engineered, solid, stable, and well-supported.
- *bsc* is fast. Supporting separate compilation, a module's compilation time into Verilog is typically measured in seconds or minutes.

Summary

Bluespec BSV and Classic significantly improve the productivity of hardware designers by significantly raising the level of abstraction, with:

- Atomic transactions as the fundamental computation model
  - Atomic *rules* in modules
  - Atomic *methods* for module interfaces
- Very expressive type system (= Haskell's):
  - Algebraic types (enums, structs, unions), vector types, polymorphic types
  - Strong numeric types (for size compatibility)
  - Higher-order functions
  - Typeclasses
    - Similar to C++ template types, except with strong, modular type-checking
    - Encapsulation and changeability of bit representations
  - Monadic types
    - To encapsulate "plumbing" through module hierarchy (e.g, signal probes)
- Very expressive functional programming for powerful static elaboration
  - Higher-order functions and recursion
  - Full parameterizability (function/module parameters can be functions/modules/…)

without loss of generated hardware quality.

Many of these capabilities are unique to Bluespec.

bluespec

End