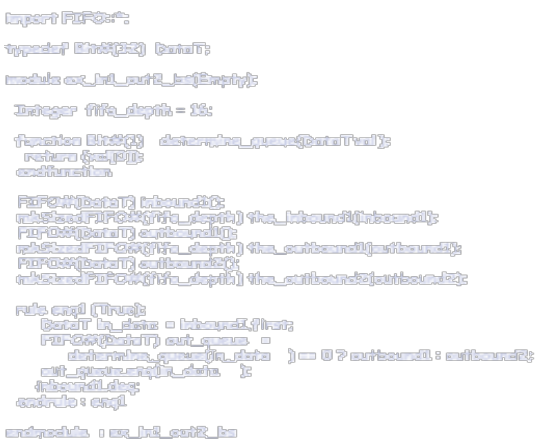
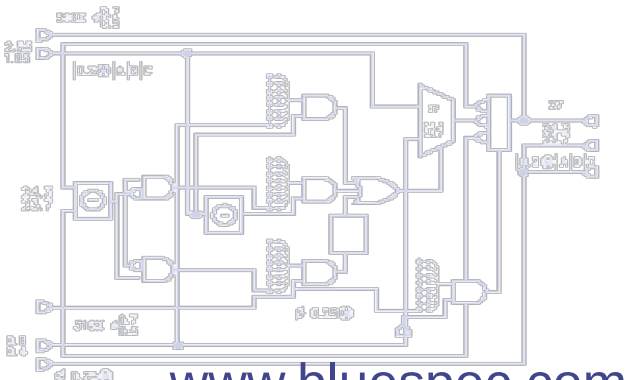
Bluespec Training

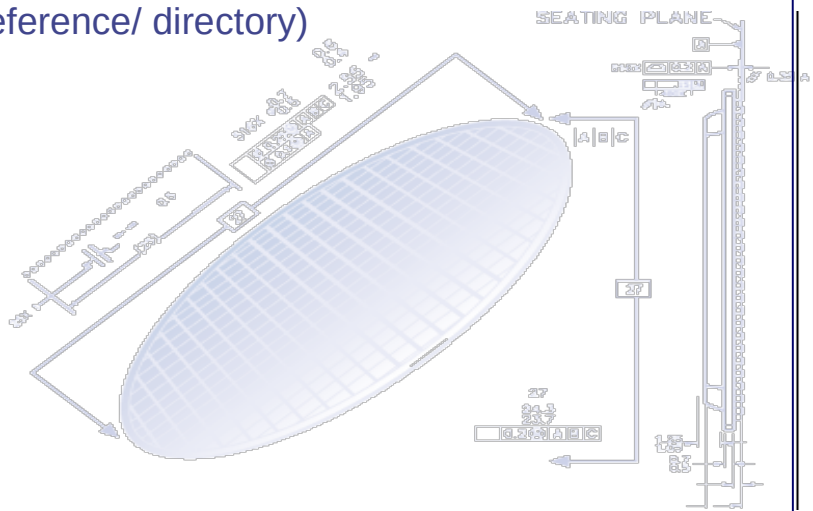
Our training session is structured around a series of complete, runnable examples.

For each example, we study the code, build it, run it, and analyze it.

As we encounter various Bluespec constructs in the code, we'll take excursions into lecture mode to understand them in more detail and in more generality (in the Reference/ directory)







www.bluespec.com

These training materials (examples, slides) are available at:

https://github.com/BSVLang/Main

Please clone a copy for yourself now.

The materials for this course are in: Tutorials/Bluespec_Classic_Training

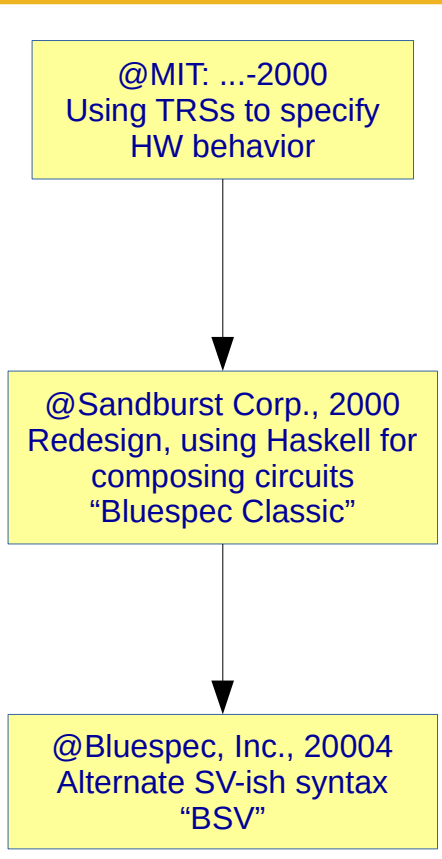
“Lecture” slide material is in: Tutorials/BSV_Training/Reference
(we have not yet made a version for Bluespec Classic)

History and Intellectual Sources

TRSs = “Term Rewriting Systems”.
Ideal for expressing heterogeneous, fine-grain, complex concurrency.
Ideal for proofs of such systems.

See also: Dijkstra's Guarded Commands, Chandy and Mishra's UNITY, Lamport's TLA+, Abrial's Event B, ...

Haskell: Ideal for abstraction, types, composition.



Bluespec Classic and BSV are just two different syntaxes for the same semantics. They are just two different front-end parserson the same compiler. Classic and BSV packages can be mixed-and-matched freely.

bluespec

Introduction

- Bluespec is a modern Hardware Design Language (HDL), in use since about 2000 in many major companies and universities worldwide
- Bluespec embodies a fundamental rethink of what an HDL should be:
  - Circuit generation, not just circuit description, employing the full power of a modern functional programming language (Haskell-like expressivity and types)
  - Atomic Transactional Rules as the fundamental behavioral abstraction, instead of synchronous clocks (scalable, compositional)
- Bluespec is not like classical “HLS” (High Level Synthesis), where the source language (C/C++) has a quite different computation model (and algorithmic cost structure) from the target (hardware)
  - Bluespec is architecturally transparent: you are in full control of architecture and there are no architectural surprises
  - With Bluespec you think hardware, you think about architectures, you think in parallel
- Bluespec is “universal” in applicability (like traditional HDLs). Bluespec has been used for CPUs, caches, coherence engines, DMAs, interconnects, memory controllers, DMA engines, I/O devices, security devices, RF and multimedia signal processing, and all kinds of accelerators.

bluespec

Training Plan

We'll work our way through a series of increasingly complex examples, taking time out to explore some topics in more detail in “lecture mode” as we encounter them in the examples.

10 lines

1500 lines + 16K lines of Piccolo+system

“Hello World”
- Eg02a_HelloWorld/ Simple "Hello World" program as a single module
- Eg02b_HelloWorld/ Split into two separately compiled packages/modules, Top and DeepThought
- Eg02c_HelloWorld/ Add some state-machine functionality to DeepThought

Bubblesort:
- Eg03a_Bubblesort/ Sequential sort of 5 int values
- Eg03b_Bubblesort/ “Chaotic” parallel sort, using 'maxBound' special value
- Eg03c_Bubblesort/ Polymorphic: generalize '5' to 'n’
- Eg03d_Bubblesort/ Polymorphic: generalize 'int' to 't’
- Eg03e_Bubblesort/ Use 'Maybe' types instead of 'maxBound' to represent +infinity

Mergesort accelerator:
- Eg06a_Mergesort/ Pipelined memory-to-memory sorting “accelerator”
- Eg06b_Mergesort/ Organize into an “SoC”-like structure with
  - AXI4 interfaces on blocks and
  - an AXI4 crossbar switch interconnect
- Eg06c_Mergesort/ Replace the test driver with a Piccolo RISC-V CPU
  - Replace test driver with a Bluespec Piccolo RISC-V core
  - Compare Piccolo executing C mergesort program vs using the mergesort accelerator

There is a separate slide deck for each of these 3 groups.

Note: all Bluespec code is synthesizable to FPGA / ASIC

Orientation around the training materials

All the training materials are provided to you inside a single top-level directory.

Inside this directory, you will see sub-directories and files. Briefly:

File/directory	*Comments*
START_HERE.pdf	This slide deck
Reference/Lec_*.pdf	Lecture slide decks, by topic
Example_Programs/	Sub-directory
…/Resources/	Common codes used by many/most of the examples
…/Eg02_HelloWorld.pdf	Slides for Eg02
…/Eg02a_HelloWorld/	sub-directory for Eg02a code
…/Eg02b_HelloWorld/	sub-directory for Eg02b code
… similarly, other examples …	*… similarly, other examples …*
.../C_programs_RV32/	C programs and pre-built ELF files to run on Piccolo RISC-V core

Orientation around each example sub-directory

Each example sub-directory (such as Eg03a_Bubblesort/) typically contains:

File/directory	*Comments*
.../Makefile	Makefile for this example
…/src/	Sub-directory containing Bluespec source files
…/dump.vcd	File with waveform data, created during Bluesim and/or Verilog simulation. Can be viewed in any waveform viewer
…/Waves.tiff	Screenshot of waveform viewer displaying waves

Lecture slide decks reading guide

The topic-based lecture slide decks in the "Reference/" directory are intended as a reference, and need not be read sequentially. However, people learning Bluespec on their own for the first time may wish to read them in the following order:

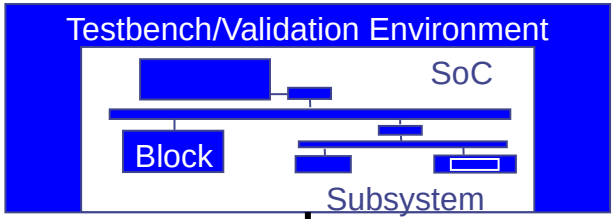
- Lec01_Intro
  General intro to the Bluespec approach, and some comparisons to other Hardware Design Languages and High Level Synthesis.
- Lec02_Basic_Syntax
  Gets you familiar with the "look and feel" of Bluespec code.
- Lec03_Rule_Semantics, Lec04_CRegs
  These two lectures describe Bluespec's concurrency and parallelism semantics (based on rules and methods). This is the KEY feature distinguishing Bluespec from other hardware and software languages.
- Lec05_Interfaces_TLM, Lec06_StmtFSM
  These two lectures describe slightly advanced constructs: more abstract interfaces, and more abstract rule-based processes.
- Lec07_Types, Lec08_Typeclasses
  These two lectures describe Bluespec's type system, which is essentially identical to that of the Haskell functional programming language.
- Lec09_Bluespec_to_Verilog
  Describes how Bluespec is translated into Verilog by the bsc tool. Read this only if you are curious about this, or if you need to interface to other existing RTL modules.
- Lec10_Interop_RTL
  How to import Verilog/VHDL code into Bluespec, and how to connect Bluespec into existing Verilog/VHDL.
- Lec11_Interop_C
  How to import C code into Bluespec (for simulation only).
- Lec12_Multiple_Clock_Domains
  How to create Bluespec designs that use multiple clocks or resets.
- Lec13_RWires
  Some facilities typically used in interfacing to external RTL. These are similar in spirit to CRegs, but lower level.

Cheat-sheet to build and run the example programs

- One-time step:
  - 'cd' into the 'Resources/' dir. At the top of 'Include_Makefile.mk' are definitions for DISTRO, PICCOLO_DIR, VSIM, etc. that should be edited to customize it for your environment.
- 'cd' into the directory for any example: it contains a 'Makefile'
- 'make b_compile', 'make b_link', 'make b_sim': build and run in Bluesim (or, just 'make b_all')
  - The created executable is usually 'mkTop_b_sim' (with a 'mkTop_b_sim.so' shared object)
- 'make v_compile', 'make v_link', 'make v_sim': build and run a Verilog simulation (or, just 'make v_all')
  - The v_compile step generates Verilog files in the 'verilog_RTL/' directory
- 'make clean' and 'make full_clean' to clean up the directories.

Full details on the commands in the Makefiles are given in the Bluespec User Guide (see “Resources” slide later in this slide deck).

Bluespec HLS tools and flow



BSV / Classic source code
- Designs
- Models
- Testbenches

Legend: Bluespec tools

High-level synthesis from atomic rules to synchronous RTL, extensive libraries

libs

bsc

Flexible execution in existing flows, and with early FPGA-based emulation

Native Compiled

Compilation to Verilog

Other RTL

Verilog

Bluesim

10 x – 100 x

RTL sim

Speed: 1x

ASIC synthesis

FPGA synthesis

(3rd party tools)

RTL analysis (power, formal, ...)

Powerful, portable emulation and co-sim infrastructure for off-the-shelf FPGA boards

1K x – 100K x

What you'll learn during these exercises

By going through these examples, and with excursions into lectures for deeper discussion, you'll learn about most of the Bluespec language and its capabilities:

- Bluespec modules, interfaces, module hierarchies, interfaces
- The core semantics of Bluespec:
  - Rules, methods, and rule concurrency, scheduling
    - Tighter concurrency using CRegs and RWires
  - Parallel ("instantaneous") Actions within rules/methods
- Types: structured types, polymorphism, strong typing
- User-extensible overloading: typeclasses, instances, provisos
- Numeric types and constraints using typeclasses to establish relationships between sizes of components
- Importing C (for Bluesim and Verilog simulation)
- Interfacing with existing hardware (Verilog, VHDL, etc.)
- Bluespec packages and separate compilation

Things we are not planning to cover during this training (please ask for separate sessions, if you wish):

- Higher-order parameterization: parameterizing functions and modules with other functions and modules
- Multiple clock domains
- Facilities for quick deployment and debugging on FPGAs, controlled by host software

Resources

- These examples and lecture slides
  - Also, the $BLUESPEC_HOME/training/BSV directory has more examples, tutorials, and labs
- Language reference guide: $BLUESPEC_HOME/doc/BSV/reference-guide.pdf
  - Complete reference on the BSV language (syntax, semantics, all language constructs, scheduling annotations, importing C and Verilog, extensive libraries
- Tool usage guide: $BLUESPEC_HOME/doc/BSV/user-guide.pdf
  - How to use BDW (Bluespec Development Workstation), how to compile and link using *bsc*, how to simulate using Bluesim and Verilog sim, how to generate and view waveforms, etc.
- BSV-by-Example book (authors: Nikhil and Czeck):
  - Around 60 examples, each focusing on one topic, with ready-to-run source code
  - Hardcopy version: purchase at Amazon.com
  - Free PDF of book: $BLUESPEC_HOME/doc/BSV/bsv_by_example.pdf
  - All the example code: $BLUESPEC_HOME/doc/BSV/bsv_by_example_appendix.tar.gz
- General questions about BSV or Classic, the tools, anything:
  - User Forums at bluespec.com (free, after registration)
  - E-mail to 'support@bluespec.com'

3rd-party Resources

Note: Bluespec has no official relationship with the organizations mentioned below (other than providing Bluespec tools through its standard University program), and is not responsible for the content posted by them. These links are provided here for information only, for your convenience. Please contact these organizations directly with any questions about this content.

- MIT Courseware: "6.004" (First course in Digital Systems)
  - Fall 2018 edition, taught by Prof. Arvind; taught entirely using Bluespec.
- MIT Courseware: "Complex Digital Systems"
  - FPGA project-oriented digital design course
  - Courseware (lectures, labs, …): http://csg.csail.mit.edu/6.375/6_375_2013_www/index.html
- MIT Courseware: "Computer Architecture: A Constructive Approach"
  - Teaching processor architectures with executable Bluespec code
  - Courseware (lectures, labs, …): http://csg.csail.mit.edu/6.S195/index.html
- Univ. of Cambridge (UK) BSV examples (Prof. Simon Moore)
  - http://www.cl.cam.ac.uk/~swm11/examples/bluespec/

bluespec

End