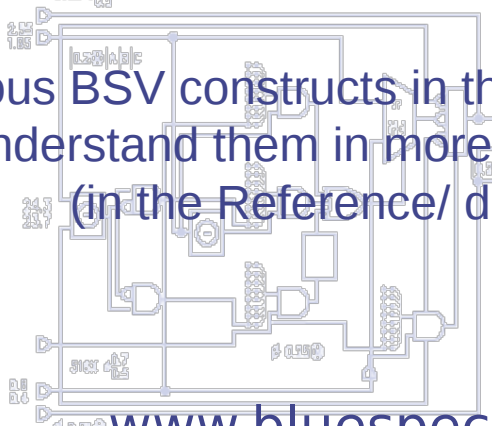
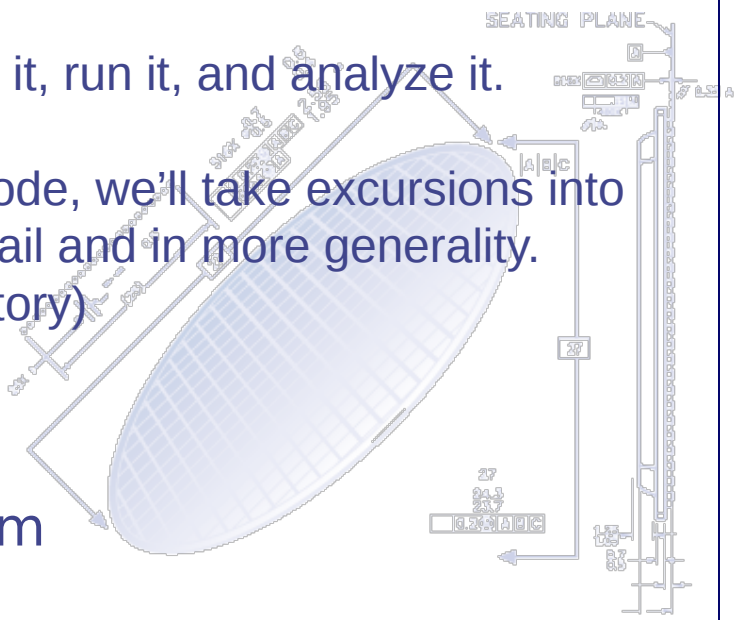
BSV Training

Our training session is structured around a series of complete, runnable examples.
(in the Example_Programs/ directory)

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

As we encounter various BSV 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, lecture slides) are available at:

https://github.com/rsnikhil/Bluespec_BSV_Tutorial.git

Please download a copy for yourself.

Introduction

- BSV is a modern Hardware Design Language (HDL), in use since about 2000 in many major companies and universities worldwide

- BSV 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)

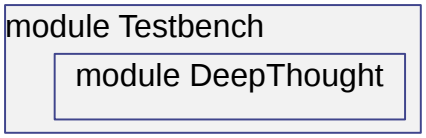
- BSV is not like classical “HLS” (High Level Synthesis), where the source language (C/C++) has a quite different computation model (and algorithmic cost structure) from the target (hardware)
  - BSV is architecturally transparent: you are in full control of architecture and there are no architectural surprises
  - With BSV you think hardware, you think about architectures, you think in parallel

- BSV is “universal” in applicability (like traditional HDLs). BSV 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.

Overview of examples

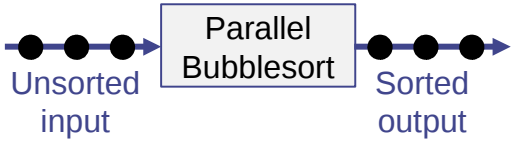
(2a-2c) Basic look-and-feel, tool usage

("Hello World")



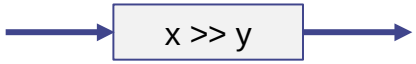
(3a-3e) Basic concurrency and modularity

(Sort 5 integers; generalize to sort 'n' items of type 't')



(4a-4b) Microarchitectures: FSMs and Pipelines

(iterative computation, rigid pipelines, elastic pipelines)

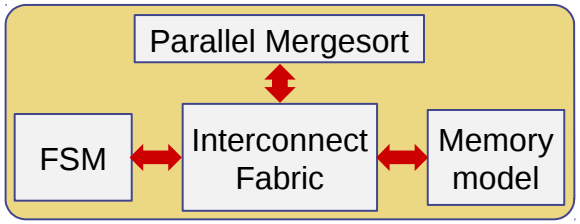


(5a-5b) Greater concurrency with CRegs

(Pipeline and Bypass FIFOs)

(6a-6c) Parallel Mergesort

(Standalone IP; generalize to accelerator in SoC)



(9a) Interfacing with existing HW and protocols

(Non-consecutive example numbers are due to correspondence with forthcoming book chapters)

Note: all BSV code is synthesizable to FPGA/ASIC

bluespec

A little more detail on the examples

Eg02_HelloWorld:
- Eg02a_HelloWorld/ Simple "Hello World" program as a single module
- Eg02b_HelloWorld/ Splits into two separately compiled modules, Testbench and DeepThought
- Eg02c_HelloWorld/ Adds some state-machine functionality to DeepThought

Eg03_Bubblesort:
- Eg03a_Bubblesort/ sequential sort of 5 values, each of type Int #(32)
- Eg03b_Bubblesort/ parallel sort, using 'maxBound' special value
- Eg03c_Bubblesort/ generalize '5' to 'n'
- Eg03d_Bubblesort/ generalize 'Int #(32)' to 't'
- Eg03e_Bubblesort/ uses 'Maybe' types instead of 'maxBound' to represent +infinity

Eg04_MicroArchs:
- Eg04a_MicroArchs/
  - Iterative shifter: Bit #(8) << Bit #(3)
  - Rigid pipelined shifter: Bit #(8) << Bit #(3)
  - Elastic pipelined shifter: Bit #(8) << Bit #(3)

- Eg04b_MicroArchs/ generalizes '8' to 'n'
  - Iterative shifter: Bit #(n) << Bit #(TLog#(n))
  - Rigid pipelined shifter: Bit #(n) << Bit #(TLog#(n))
  - Elastic pipelined shifter: Bit #(n) << Bit #(TLog#(n))

Eg05_CRegs_Greater_Concurrency:
- Eg05a_CRegs_Greater_Concurrency/ Example with FIFOs
- Eg05b_CRegs_Greater_Concurrency/ Example with Up/Down counters

Eg06_Mergesort:
- Eg06a_Mergesort/ System: mkConnection (mergesort, mem); 1 merge engine, 1 mem port)
- Eg06b_Mergesort/ System: SoC; same mergesort module (1 merge engine, 1 mem port)
- Eg06c_Mergesort/ System: SoC; n merge engines, n mem ports; reorder buffer

Eg09a_AXI4_Stream/
- Simple example showing interfacing BSV to existing buses

Training plan

These slides ("START_HERE.pdf"). It describes the main thread of the training, which is to go through a series of complete, working examples, which are all provided in the directory "Example_Programs/". All examples are executable and synthesizable.

Please go through Examples serially: for each example (e.g., Example 1),

- Go through the corresponding slide deck: Eg02_HelloWorld.pdf
  It will explain the example, with its possibly multiple versions.

- It will take you through a tour of each version, and show you how to build and run it. In a classroom setting, we study and discuss the source code extensively.
  - We show how to build and run the examples in both Bluesim and Verilog simulation, using command-line Makefiles. Some examples also demonstrate the use of BDW (the Bluespec Development Workstation GUI interface).
  - With both Bluesim and Verilog simulation, we generate VCD waveform files, view the waveforms, and analyze them.

As we go through the examples, we reinforce concepts by looking at the topic-based lectures slides called "Lec_*topic*.pdf" in the "Reference/" directory.

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 BSV 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 BSV code.
- Lec03_Rule_Semantics, Lec04_CRegs
  These two lectures describe BSV's concurrency and parallelism semantics (based on rules and methods). This is the KEY feature distinguishing BSV from other hardware ands of 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 BSV's type system, which is essentially identical to that of the Haskell functional programming language.
- Lec09_BSV_to_Verilog
  Describes how BSV 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 BSV, and how to connect BSV into existing Verilog/VHDL.
- Lec11_Interop_C
  How to import C code into BSV (for simulation only). How to export a BSV subsystem as a SystemC module (for use in a SystemC program).
- Lec12_Multiple_Clock_Domains
  How to create BSV 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.

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
.../Common/	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 ...*

Orientation around each example sub-directory

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

File/directory	*Comments*
.../src_BSV/	Sub-directory containing BSV 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

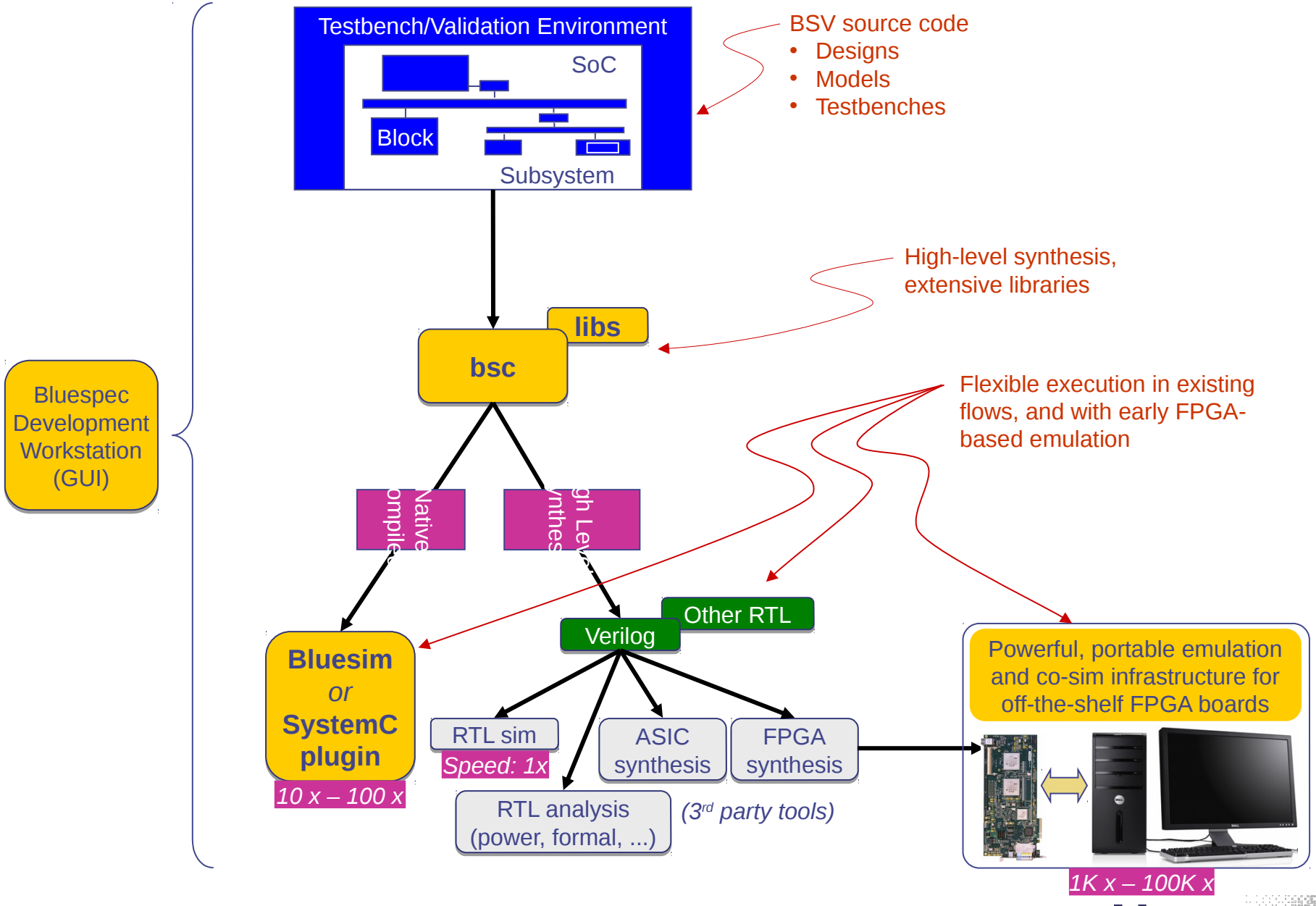
Cheat-sheet to build and run the example programs

To build and run from 'make' commands on the command line:

- 'cd' to the Build/ directory, which contains a Makefile
  - (You may wish to make a copy of the Build directory for each example you build)
  - Edit the top few lines of the Makefile to specify which example you are building, the location of your source files (distribution accompanying this training), and your Verilog simulator
- 'make compile', 'make link', and 'make simulate' to build and run in Bluesim
  - The created executable is usually called 'out' (with an 'out.so' shared object)
- 'make v_compile' to generate Verilog files in the 'verilog_dir/' directory
- If you have a Verilog simulator installed, then:
  - 'make v_link', and 'make v_simulate' to build and run in Verilog simulation
- 'make clean' and 'make full_clean' to clean up directories

Full details on the commands in the Makefiles, and also on using the BDW GUI (Bluespec Development Workstation) are given in the Bluespec User Guide (see "Resources" slide later in this slide deck).

Bluespec HLS tools and flow



What you'll learn about BSV during these exercises

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

- BSV modules, interfaces, module hierarchies, interfaces
- The core semantics of BSV:
  - 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.)
- BSV 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

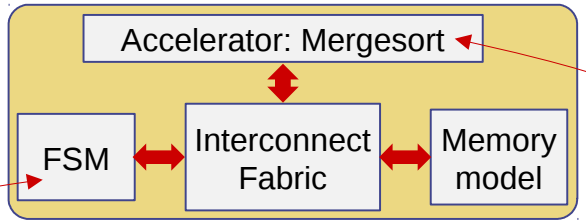
- These examples and lecture slides
  - Also, the $BLUESPEC_HOME/training/BSV directory has more examples, tutorials, and labs
- Language reference guide: $BLUESPEC_HOME/doc/BSV/reference-guide.pdf
  - Complete reference on the BSV language (syntax, semantics, all language constructs, scheduling annotations, 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, the tools, anything:
  - User Forums at bluespec.com (free, after registration)
  - E-mail to 'support@bluespec.com'

Resources: Follow-on Examples

We can provide some follow-on larger examples for later self-study (they are not covered during the training itself). They build on the SoC structure of Example 6:

(6a-6c) Parallel Mergesort

(Standalone IP; generalize to accelerator in SoC)



(20) CPU: replace the FSM by a simple implementation of a complete CPU (Berkeley RISC-V instruction set)

(30) Accelerator: Vector Add

(31) Accelerator: Vector Inner Product

(32) Accelerator: Matrix Multiplication

(33) Accelerator: Blocked Matrix Multiplication

Note: all BSV code is synthesizable to FPGA/ASIC

bluespec

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: "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 BSV 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/

End