Typesetting Bluespec SystemVerilog with `bluespec.sty`

A whole file

Listing 1: `Gcd.bsv`

```
// A greatest-common-divisor engine, used as the demo listing.
package Gcd;

import FIFOF :: *;
import GetPut :: *;
import ClientServer :: *;

typedef 32 GcdWidth;
typedef Bit#(GcdWidth) GcdOperand;

typedef enum { Idle, Swapping, Subtracting } Phase deriving (Bits, Eq, FShow);

typedef struct {
   GcdOperand a;
   GcdOperand b;
} GcdRequest deriving (Bits, FShow);

interface Gcd;
   interface Put#(GcdRequest) request;
   interface Get#(GcdOperand) response;
   method Phase phase();
endinterface

(* synthesize, always_ready = "phase" *)
module mkGcd (Gcd);

   Reg#(GcdOperand) x    <- mkReg(0);
   Reg#(GcdOperand) y    <- mkReg(0);
   Reg#(Phase)      step <- mkReg(Idle);

   FIFOF#(GcdOperand) out <- mkSizedFIFOF(2);

   (* descending_urgency = "swap, subtract" *)
   rule swap (step == Swapping && x > y && y != 0);
      x <= y;
      y <= x;
      step <= Subtracting;
   endrule

   rule subtract (step == Subtracting && x <= y && x != 0);
      y <= y - x;
      step <= Swapping;
   endrule

   rule finish (step != Idle && (x == 0 || y == 0));
      let result = (x == 0) ? y : x;
      out.enq(result);
      step <= Idle;
`ifdef GCD_TRACE
```

```
      $display("[%0t] gcd = %0d", $time, result);
`endif
   endrule

   interface Put request;
      method Action put(GcdRequest r) if (step == Idle);
         x <= r.a;
         y <= r.b;
         step <= Swapping;
      endmethod
   endinterface

   interface response = toGet(out);

   method Phase phase() = step;

endmodule: mkGcd

/* A testbench that drives a handful of operand pairs through mkGcd
   and stops the simulation once the last answer has come back. */
module mkTb (Empty);
   Gcd dut <- mkGcd;
   Reg#(UInt#(4)) sent <- mkReg(0);

   Vector#(3, GcdRequest) stimulus = newVector;

   rule drive (sent < 3);
      dut.request.put(GcdRequest { a: 100 + extend(pack(sent)), b: 36 });
      sent <= sent + 1;
   endrule

   rule drain;
      let g <- dut.response.get();
      if (sent == 3 && g != 0) $finish(0);
   endrule
endmodule

endpackage: Gcd
```

An inline listing

A rule fires when its guard holds:

```
rule count (cnt < 8'd200 && !full);
   cnt <= cnt + 1;
   $display("cnt = %0h at %0t", cnt, $time);
endrule
```

Short fragments

The literal `8'hFF` is a sized literal, `mkReg` is a module constructor, and `(* synthesize *)` is an attribute.

The language on its own

Ask for `language=BSV` instead of the style and the package paints nothing: the keyword classes, the comments, the strings and the attribute delimiter all take whatever the document already specified. A single `keywordstyle` reaches all five classes, because listings falls back to it for the classes that have no style of their own.

```
(* synthesize *)
module mkCount (Empty);          // a monochrome document keeps its own scheme
   Reg#(Bit#(8)) cnt <- mkReg(0);
   rule count; cnt <= cnt + 1; endrule
endmodule
```