Introduction to Bluespec: A new methodology for designing Hardware

Arvind
Computer Science & Artificial Intelligence Lab.
Massachusetts Institute of Technology

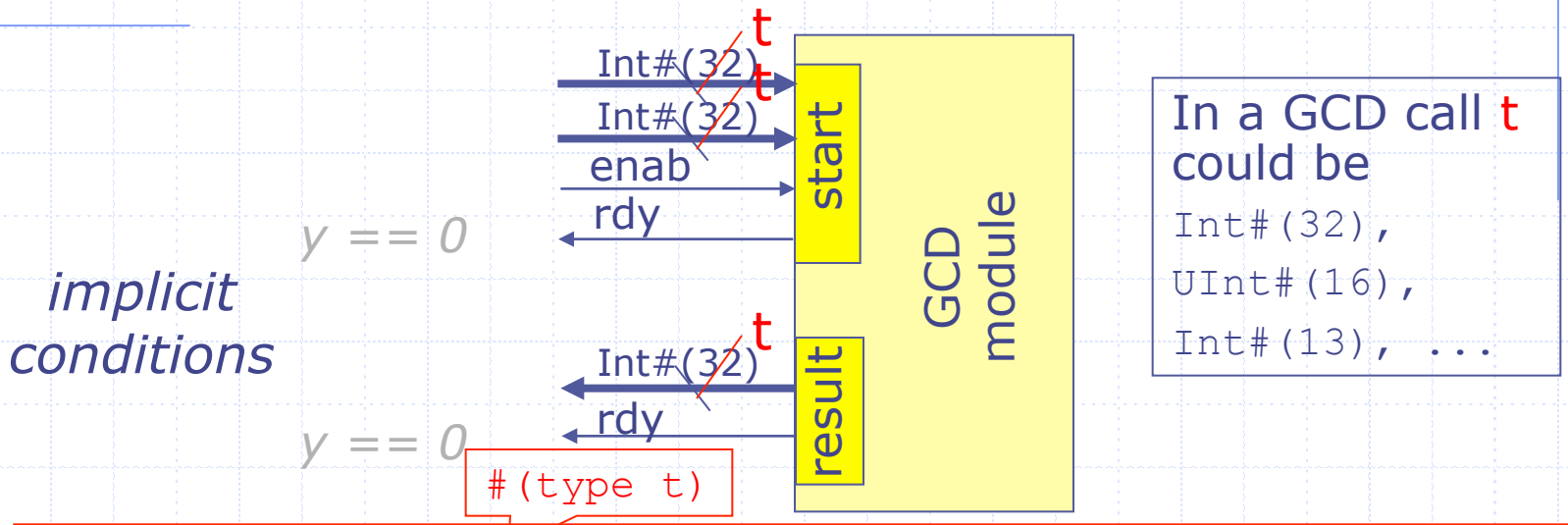
GCD: A simple example to explain hardware generation from Bluespec

Programming with rules: A simple example

Euclid's algorithm for computing the Greatest Common Divisor (GCD):

15		6	
9		6	*subtract*
3		6	*subtract*
6		3	*swap*
3		3	*subtract*
0	*answer:*	3	*subtract*

GCD Hardware Module



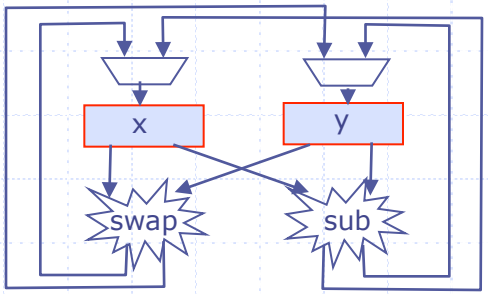
In a GCD call t could be

```
Int#(32),
UInt#(16),
Int#(13), ...
```

```
interface I_GCD;
    method Action start (Int#(32) a, Int#(32) b);
    method Int#(32) result();
endinterface
```

- The module can easily be made polymorphic
- Many different implementations can provide the same interface: `module mkGCD (I_GCD)`

GCD in BSV




```
module mkGCD (I_GCD);
    Reg#(Int#(32)) x <- mkRegU;
    Reg#(Int#(32)) y <- mkReg(0);

    rule swap ((x > y) &&  (y != 0));
        x <= y;  y <= x;
    endrule
    rule subtract ((x <= y) && (y != 0));
        y <= y - x;
    endrule

    method Action start(Int#(32) a, Int#(32) b)
                                        if (y==0);
       x <= a;  y <= b;
    endmethod
    method Int#(32) result() if (y==0);
        return x;
    endmethod
endmodule
```

State

Internal behavior

If (a==0) then 0 else b

External interface

Assume a/=0

GCD: Another implementation

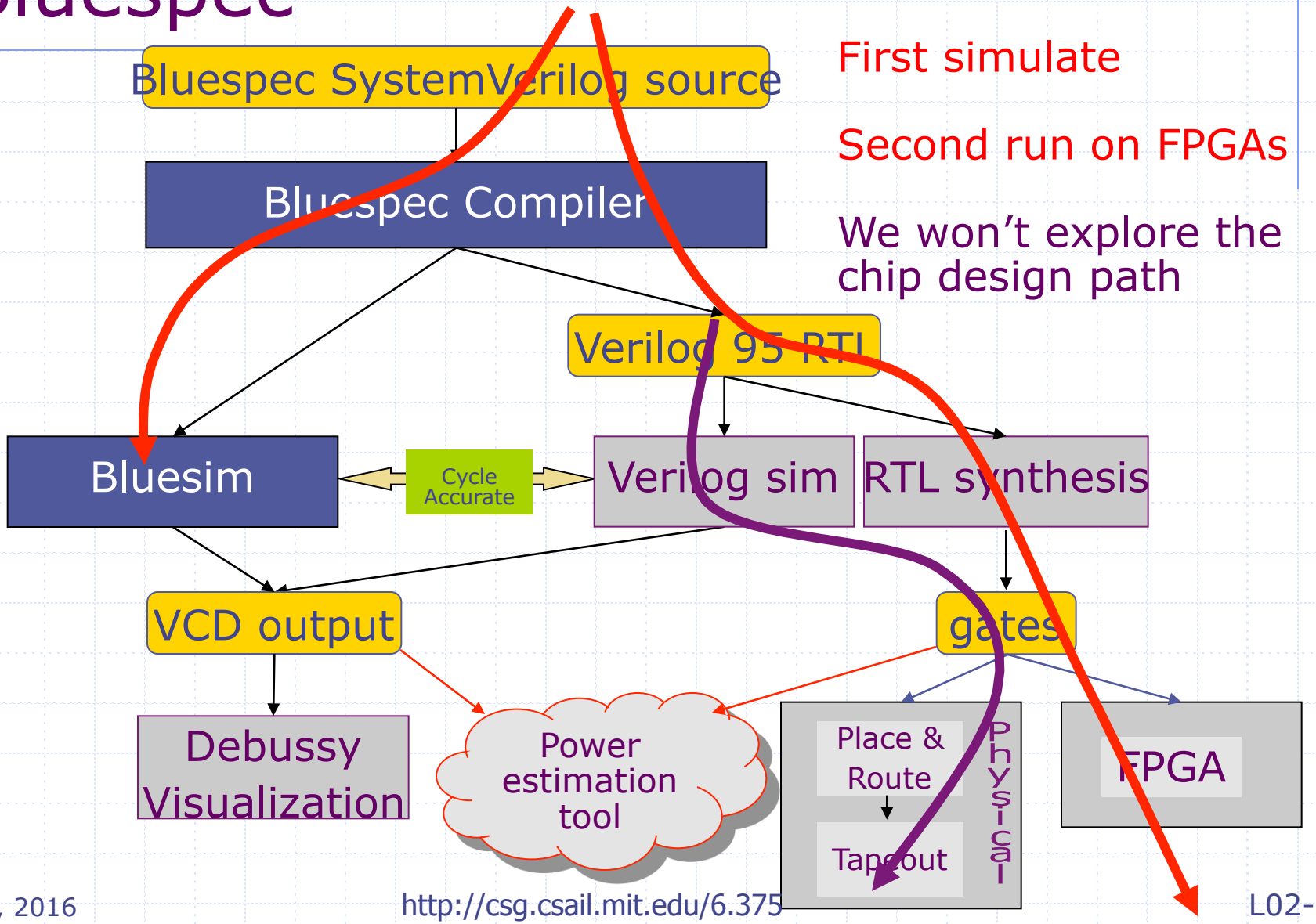
```
module mkGCD (I_GCD);
    Reg#(Int#(32)) x <- mkRegU;
    Reg#(Int#(32)) y <- mkReg(0);

    rule swapANDsub ((x > y) &&  (y != 0));
        x <= y;  y <= x - y;
    endrule
    rule subtract ((x<=y) && (y!=0));
        y <= y - x;
    endrule

    method Action start(Int#(32) a, Int#(32) b)
                                          if (y==0);
       x <= a;  y <= b;
    endmethod
    method Int#(32) result() if (y==0);
        return x;
    endmethod
endmodule
```

Combine swap and subtract rule

Does it compute faster ?

Does it take more resources ?

High-level Synthesis from Bluespec



First simulate

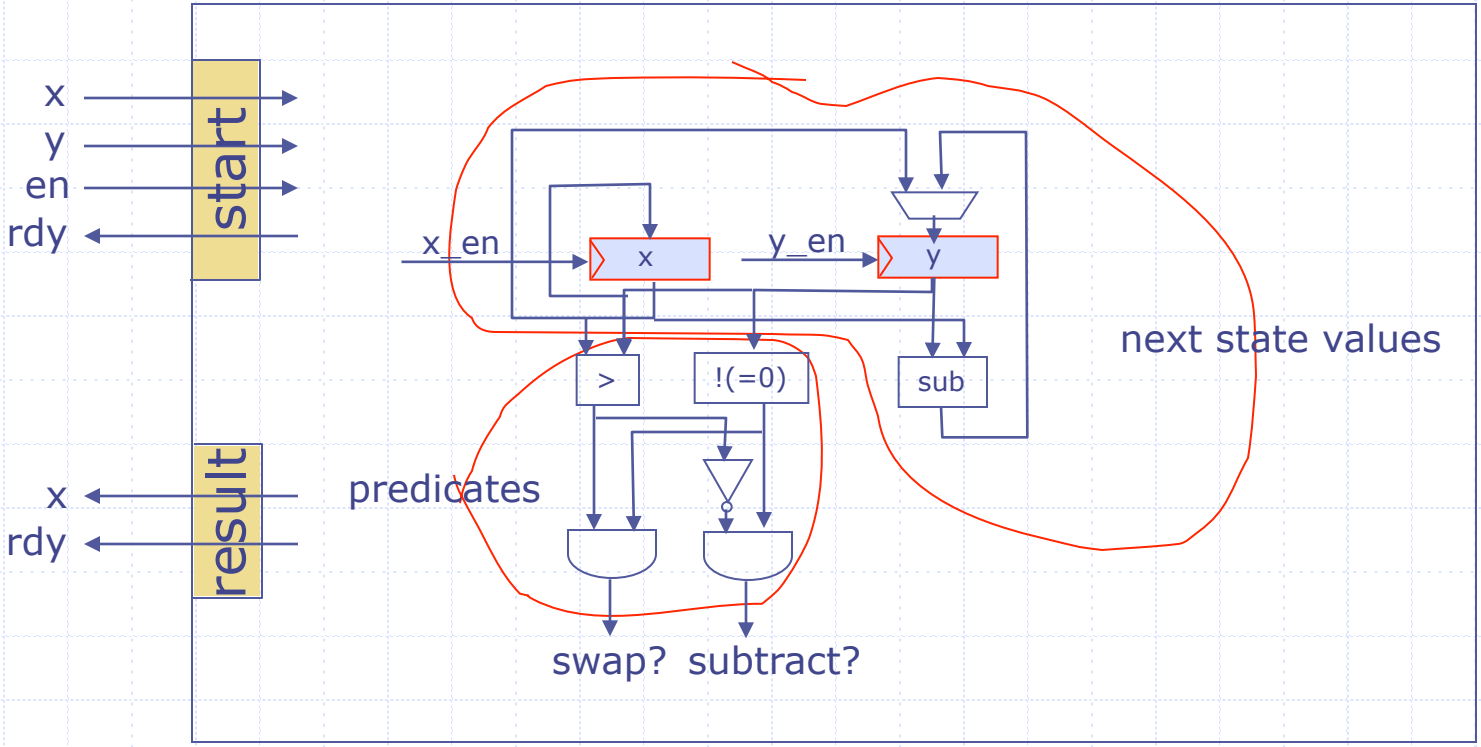
Second run on FPGAs

We won't explore the chip design path

Generated Verilog RTL: GCD

```
module mkGCD(CLK,RST_N,start_a,start_b,EN_start,RDY_start,
             result,RDY_result);
  input  CLK; input  RST_N;
// action method start
  input [31 : 0] start_a; input [31 : 0] start_b; input EN_start;
  output RDY_start;
// value method result
  output [31 : 0] result; output RDY_result;
// register x and y
  reg [31 : 0] x;
  wire [31 : 0] x$D_IN; wire x$EN;
  reg [31 : 0] y;
  wire [31 : 0] y$D_IN; wire y$EN;
...
// rule RL_subtract
  assign WILL_FIRE_RL_subtract = x_SLE_y___d3 && !y_EQ_0___d10 ;
// rule RL_swap
  assign WILL_FIRE_RL_swap = !x_SLE_y___d3 && !y_EQ_0___d10 ;
...
```

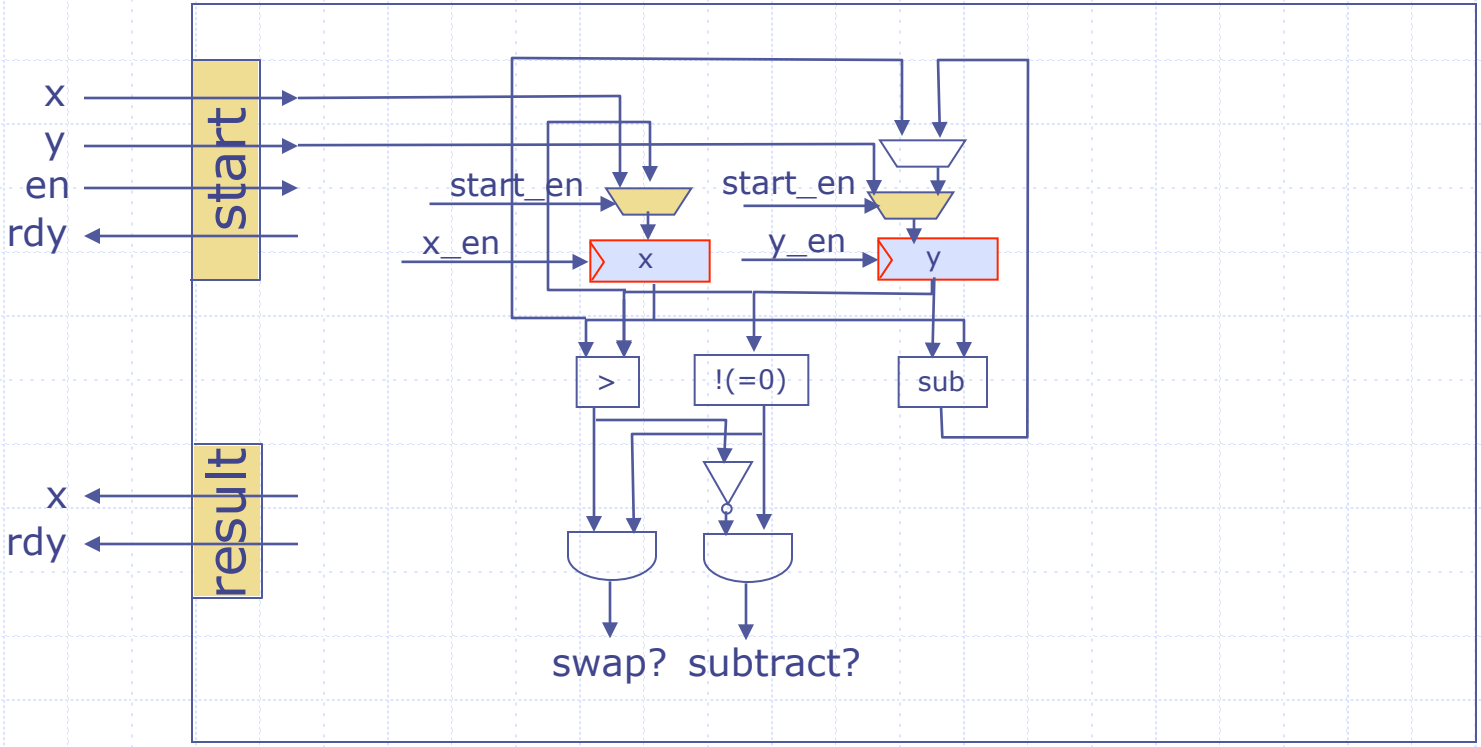
Generated Hardware




```
rule swap ((x>y)&&(y!=0));
   x <= y;  y <= x; endrule
rule subtract ((x<=y)&&(y!=0));
   y <= y - x; endrule
```

x_en = swap?
y_en = swap? OR subtract?

Generated Hardware Module



x_en = swap? OR start_en

y_en = swap? OR subtract? OR start_en

rdy = (y==0)

GCD: A Simple Test Bench

```
module mkTest ();
  Reg#(Int#(32)) state <- mkReg(0);
  I_GCD      gcd   <- mkGCD();

  rule go (state == 0);
    gcd.start (423, 142);
    state <= 1;
  endrule

  rule finish (state == 1);
    $display ("GCD of 423 & 142 =%d",gcd.result());
    state <= 2;
  endrule
endmodule
```

Why do we need the state variable?

Is there any timing issue in displaying the result?

No. Because the `finish` rule cannot execute until `gcd.result` is ready

GCD: Test Bench

```
module mkTest ();
   Reg#(Int#(32)) state <- mkReg(0);
   Reg#(Int#(4))     c1 <- mkReg(1);
   Reg#(Int#(7))     c2 <- mkReg(1);
   I_GCD            gcd <- mkGCD();

   rule req (state==0);
     gcd.start(signExtend(c1), signExtend(c2));
     state <= 1;
   endrule

   rule resp (state==1);
     $display ("GCD of %d & %d =%d", c1, c2,
gcd.result());
     if (c1==7) begin c1 <= 1; c2 <= c2+1; end
                else  c1 <= c1+1;
     if (c1==7 && c2==63) state <= 2 else state <= 0;
   endrule
endmodule
```

Feeds all pairs (c1,c2)

$1 < c1 < 7$

$1 < c2 < 63$

to GCD