Looking for a job!

FPGA Technology Mapping Using Sketch-Guided Program Synthesis

(aka Lakeroad)

Speaker: Gus Smith, University of Washington

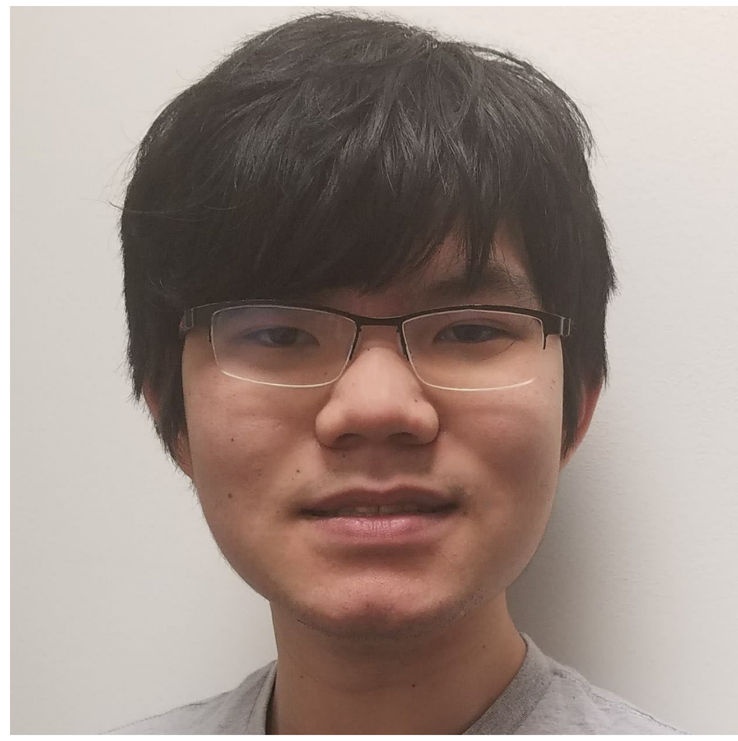
PAUL G. ALLEN SCHOOL

Ben Kushigian

Vishal Canumalla

Andrew Cheung

Steven Lyubomirsky



Sorawee Porncharoenwase

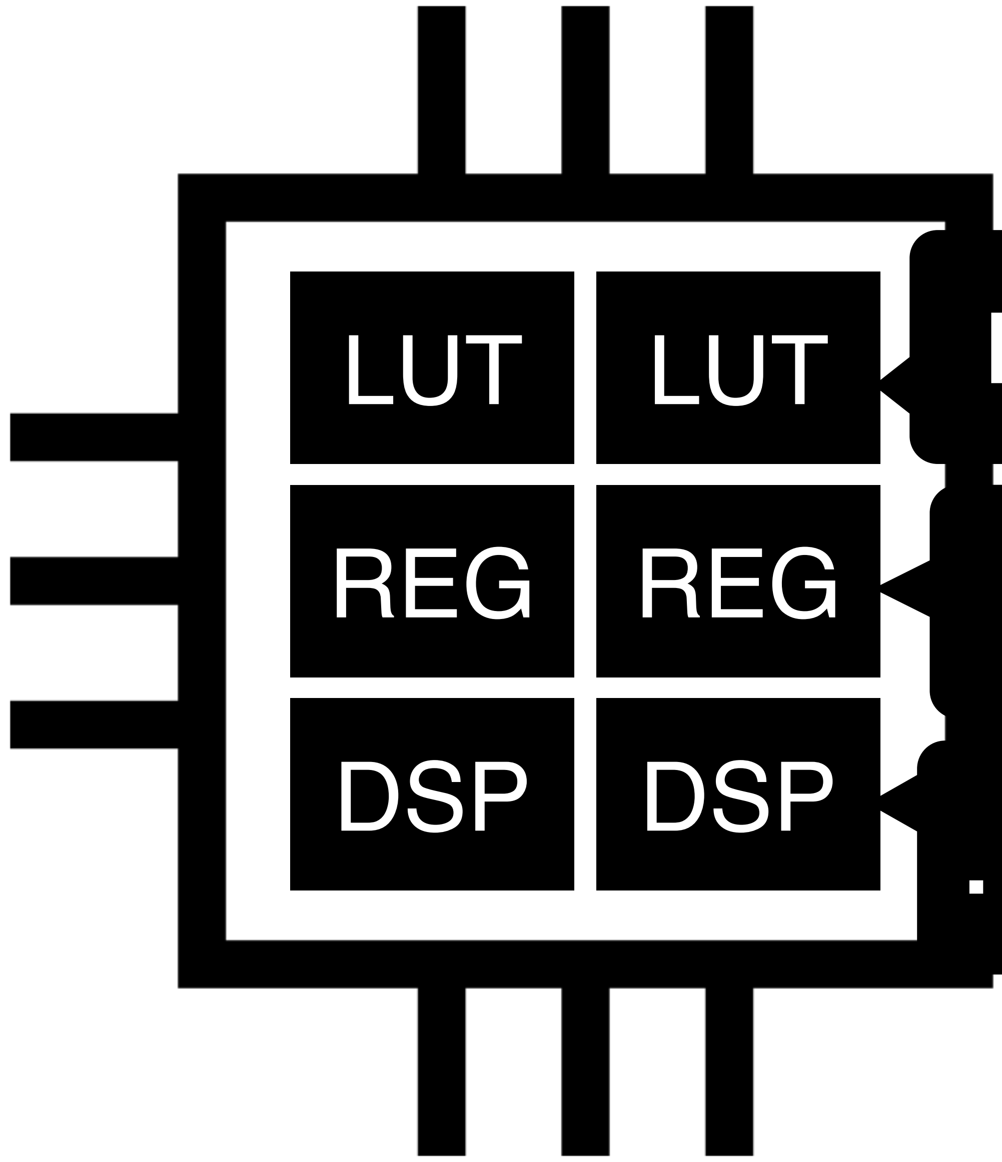
René Just

Gilbert Bernstein

Zachary Tatlock

PAUL G.
ALLEN
SCHOOL

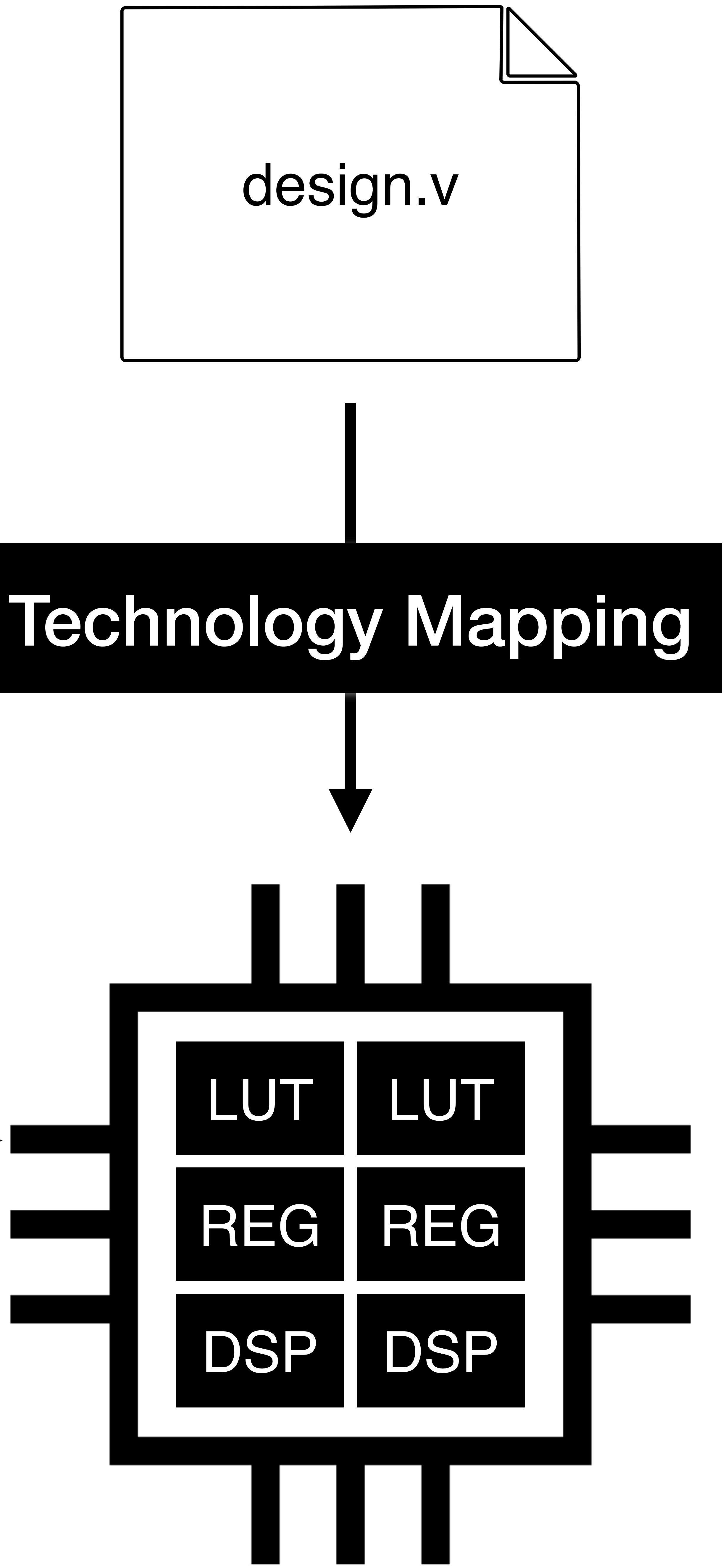
FPGAs are composed of small hardware *primitives*.



E.g. look-up tables implement logic functions…

…registers hold state…

…and digital signal processors accelerate arithmetic.



design.v

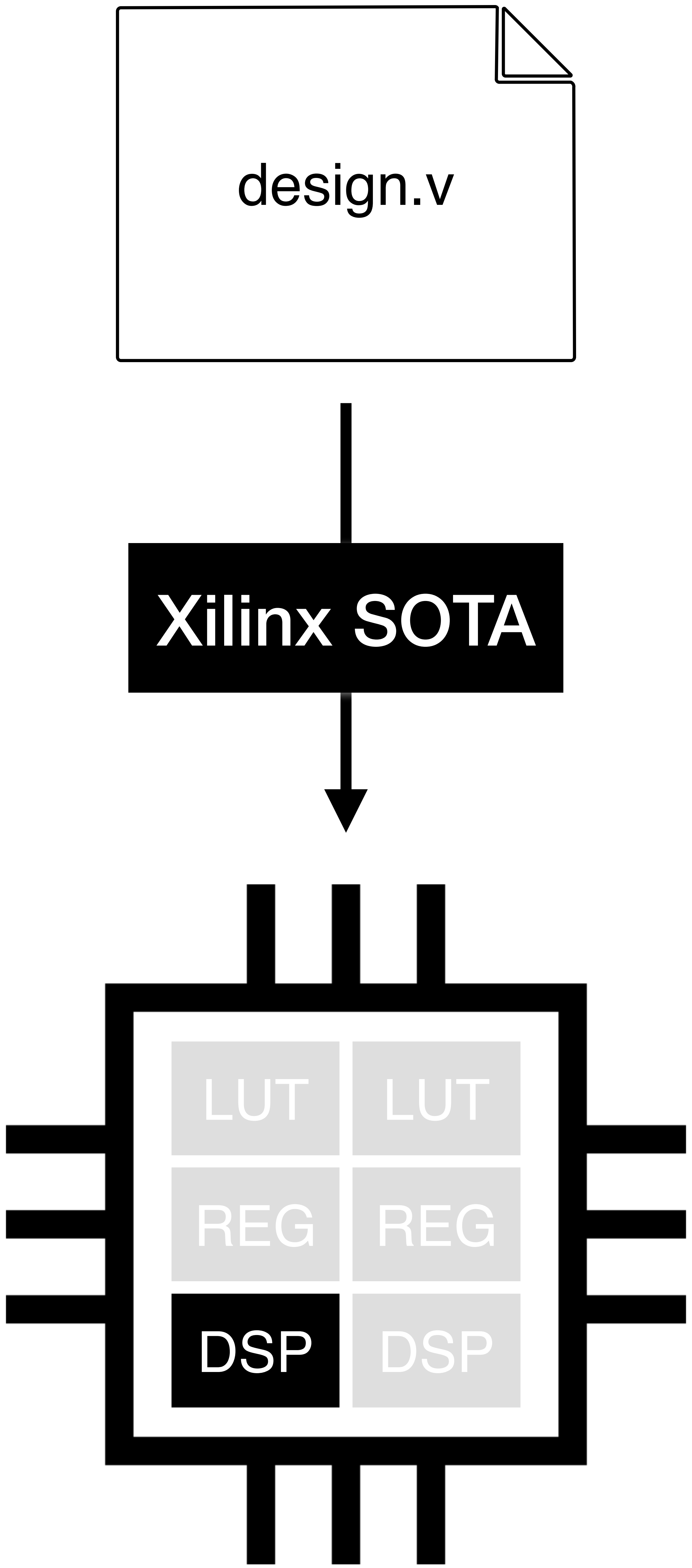
Technology Mapping

LUT LUT

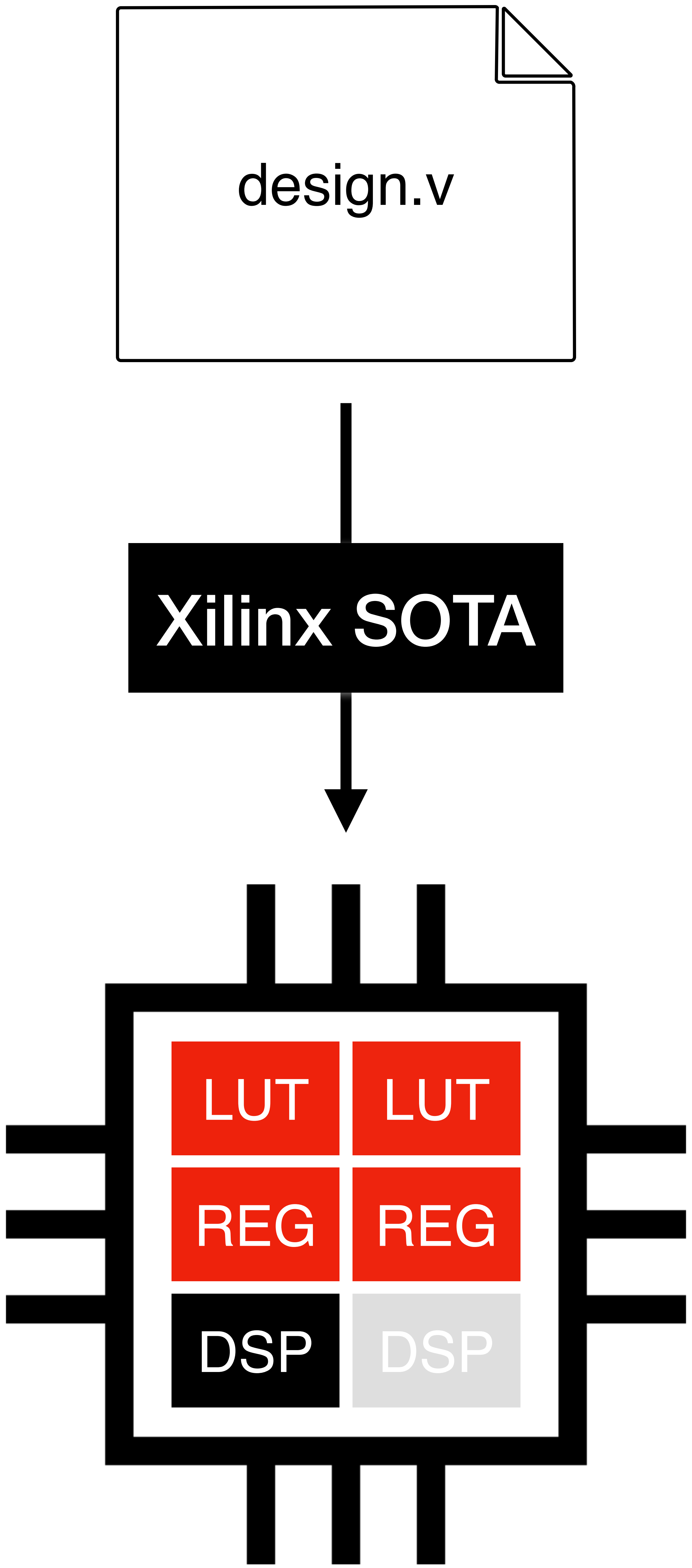
REG REG

DSP DSP

To produce an optimal compiled design, we must efficiently use primitives!



State of the art tools do not efficiently use primitives!



State of the art tools do not efficiently use primitives!

design.v

Xilinx SOTA

Lakeroad

State of the art tools do not efficiently use primitives!

LUT	LUT
REG	REG
DSP	DSP

Configuring DSPs is like writing programs, so use *program synthesis!*

LUT	LUT
REG	REG
DSP	DSP

More Complete

More Correct

More Extensible

Problem: existing tools fail to utilize programmable primitives.

Insight: Programmable primitives? Apply program synthesis!

We used this insight to build Lakeroad, which is

more complete, more correct, and more extensible.

Problem

Insight

Implementation

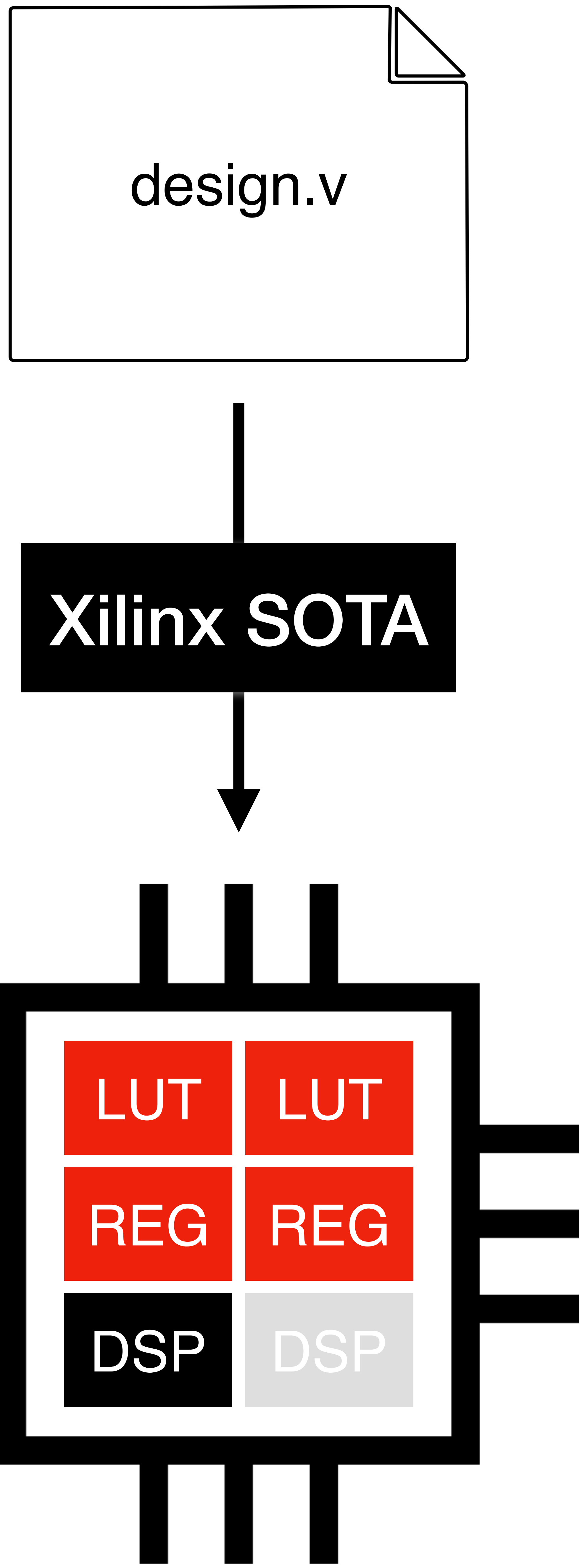
Evaluation

Problem

Insight

Implementation

Evaluation



design.v

Xilinx SOTA

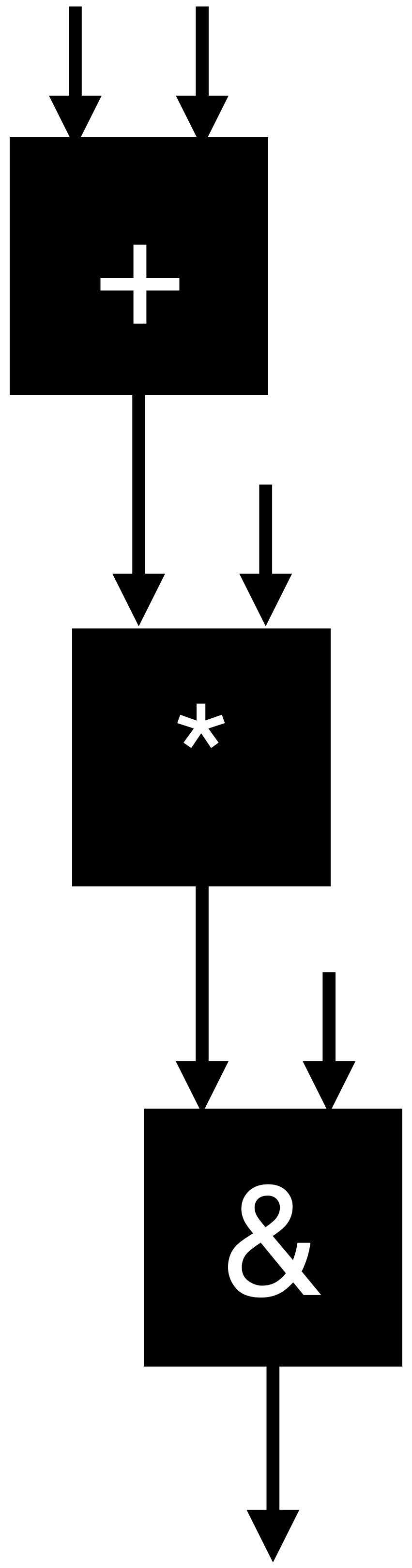
LUT LUT

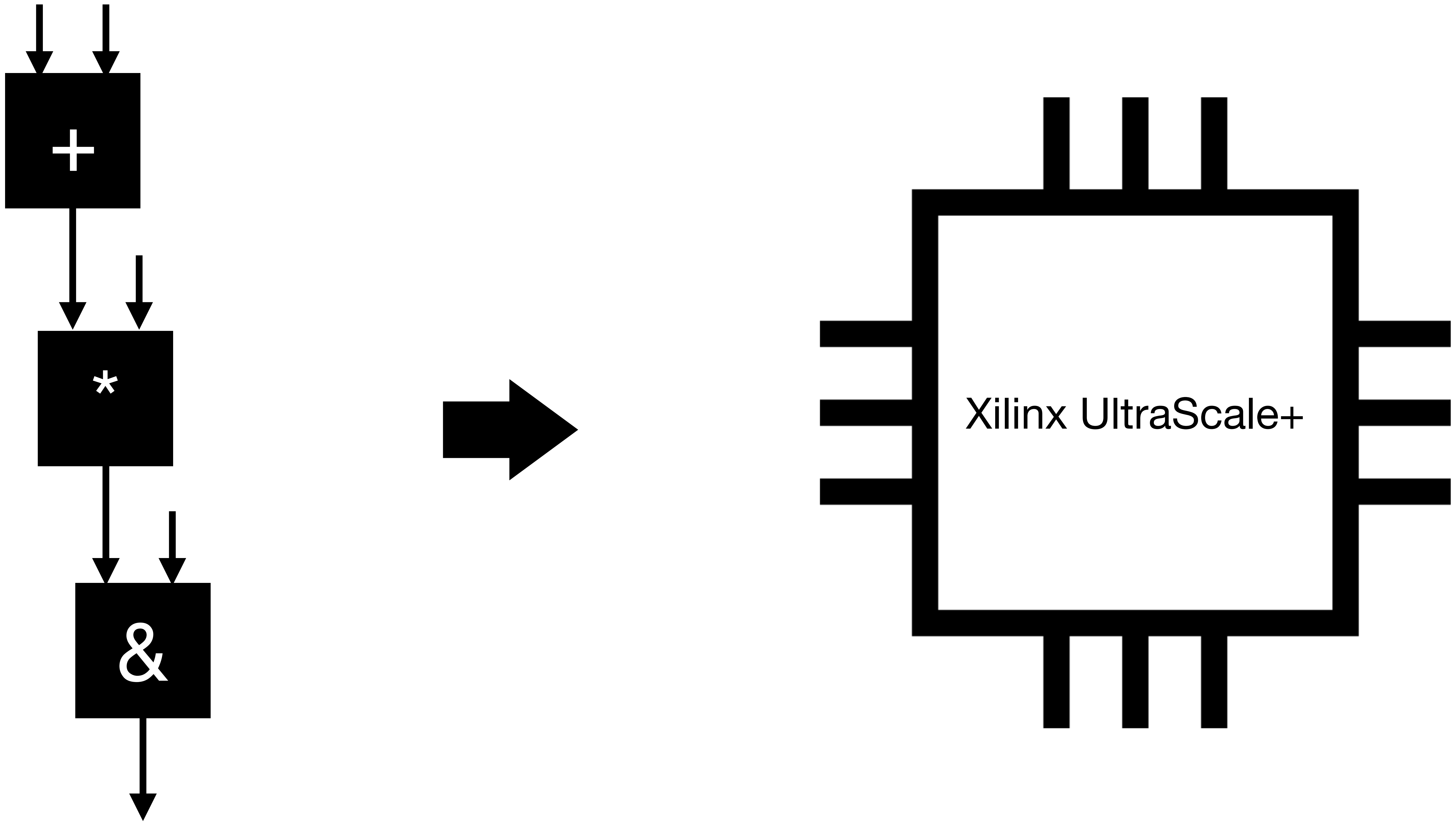
REG REG

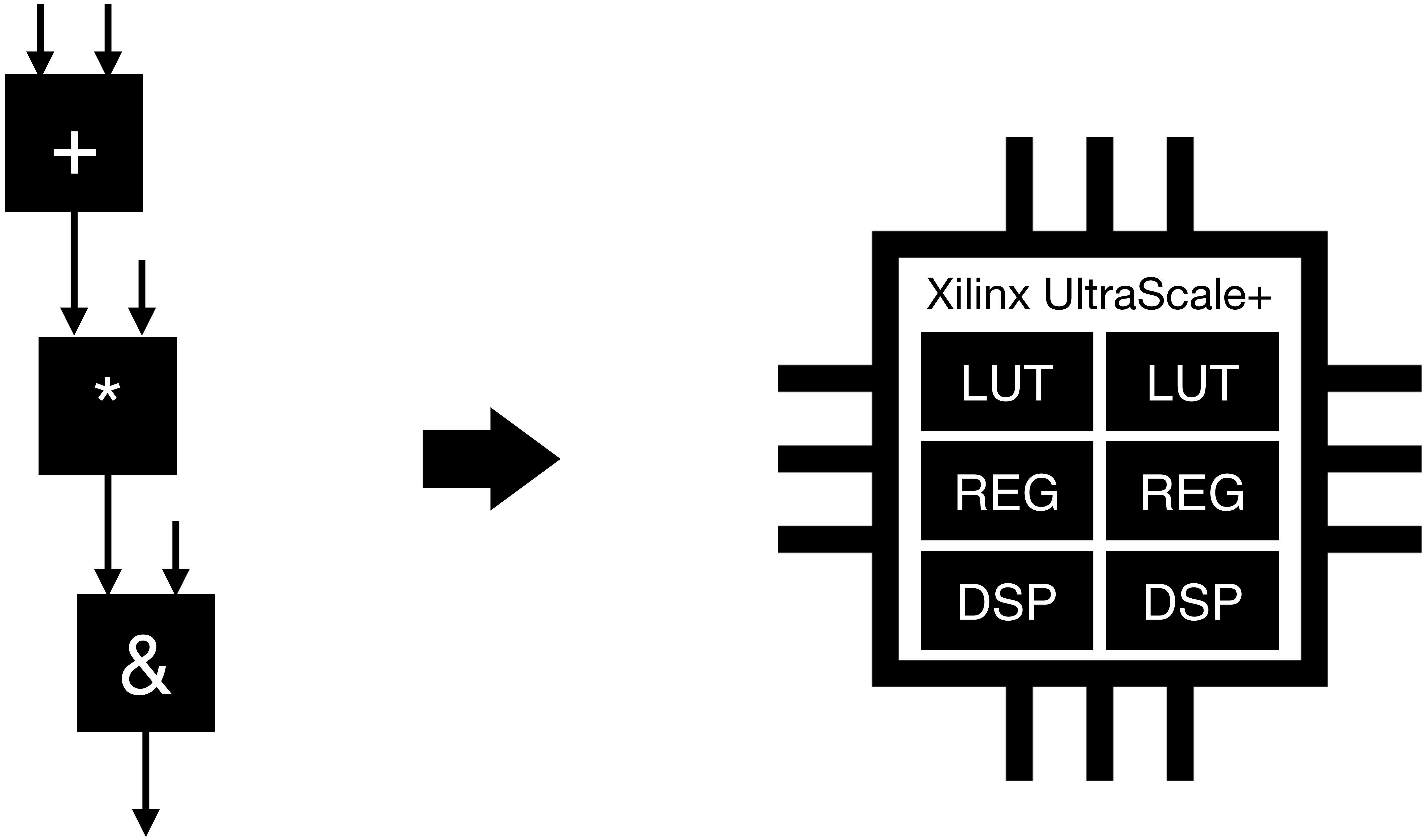
DSP DSP

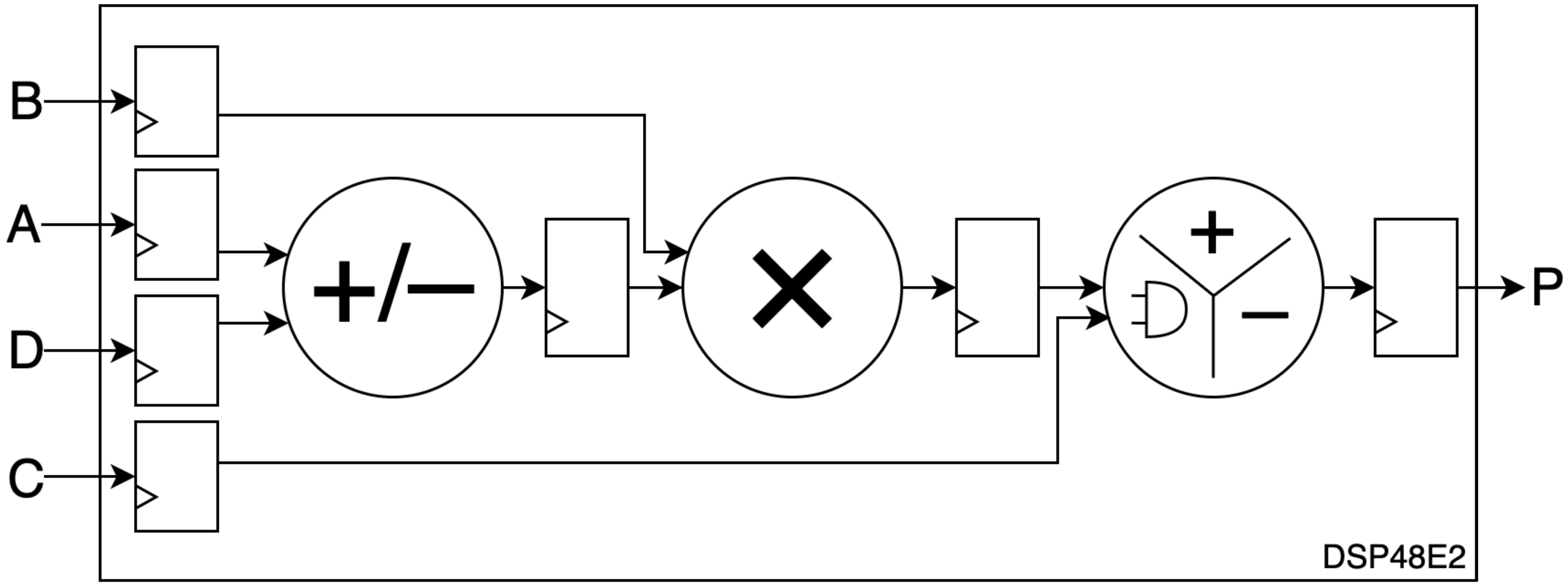
State of the art tools do not efficiently use primitives!

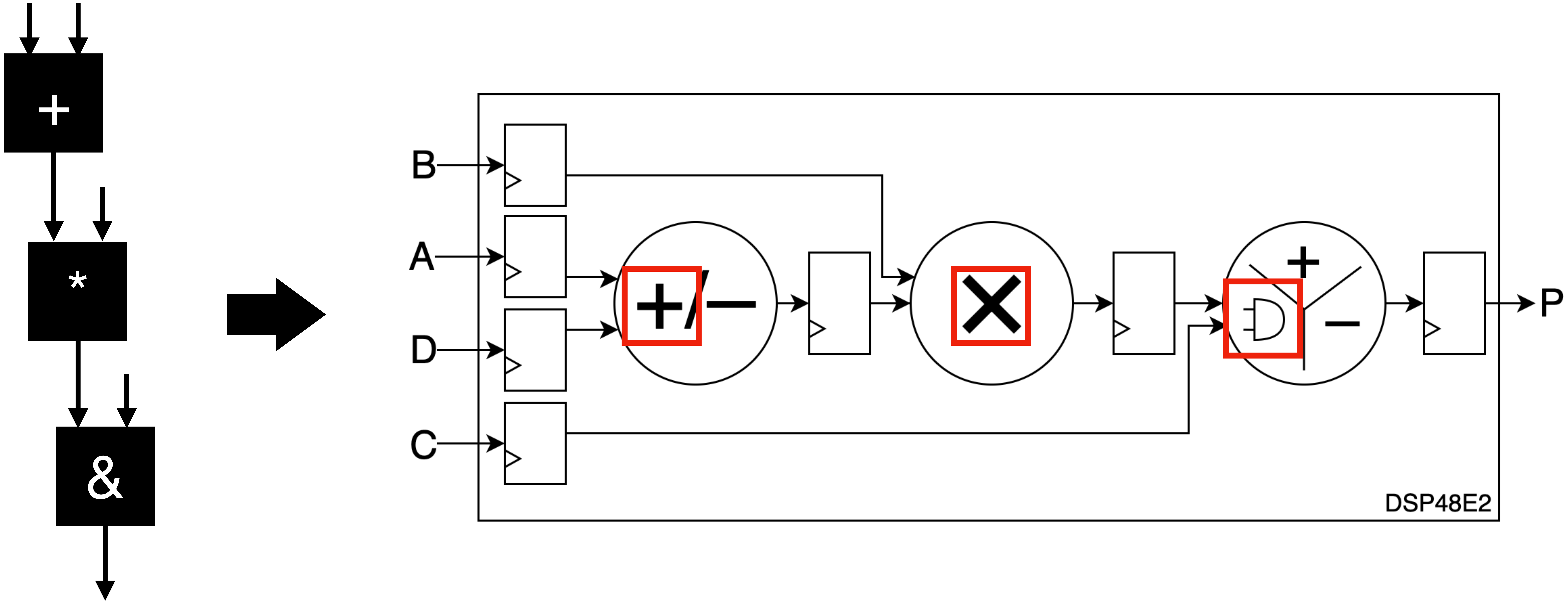
```
module add_mul_and(...);
  assign out = (d + a) * b & c;
endmodule
```

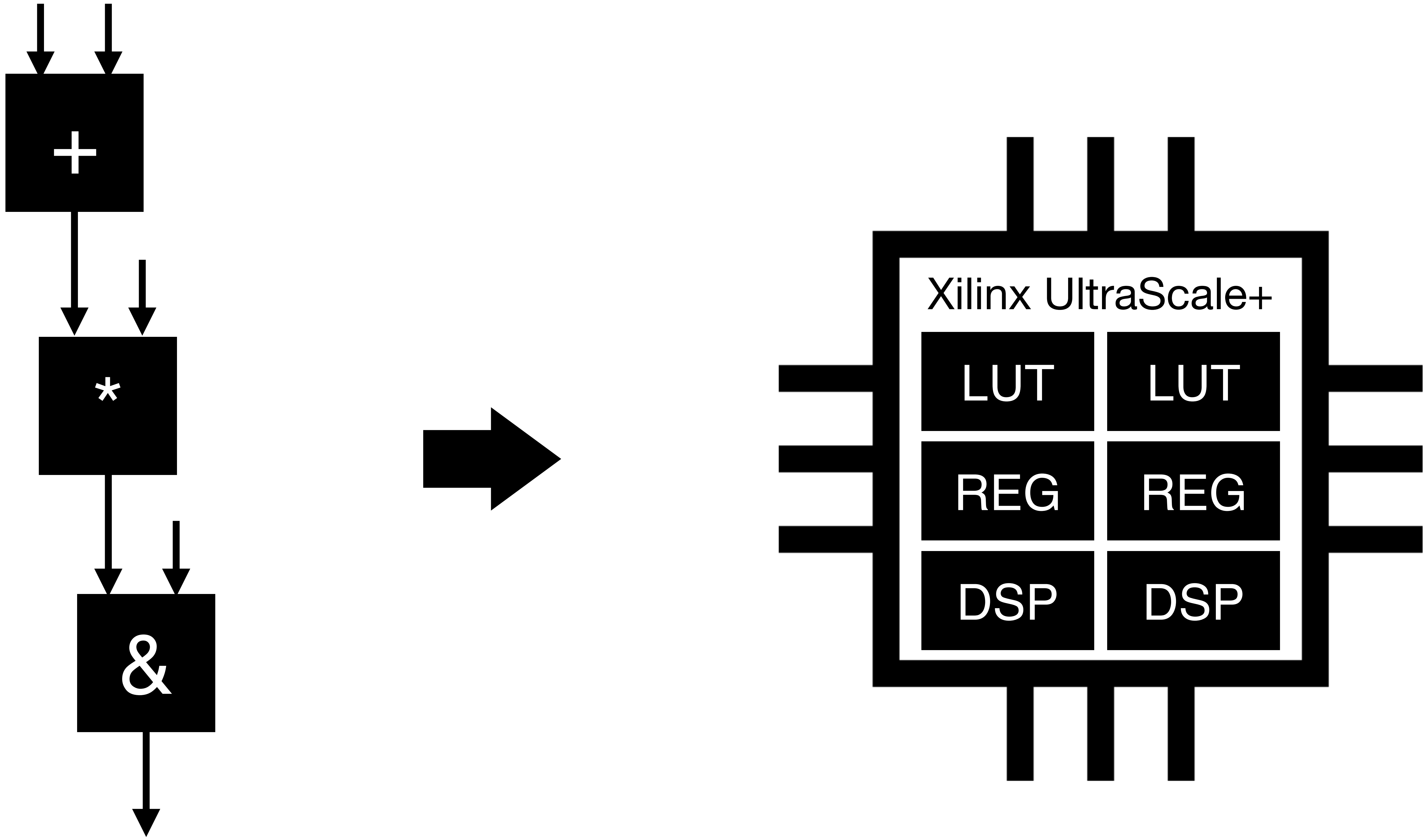



+
*
&
Xilinx UltraScale+


+
*
&
Xilinx UltraScale+
LUT
LUT
REG
REG
DSP
DSP


B
A
D
C
+/−
×
+
−
P
DSP48E2


+
*
&
B
A
D
C
+/−
×
+
−
P
DSP48E2



+

*

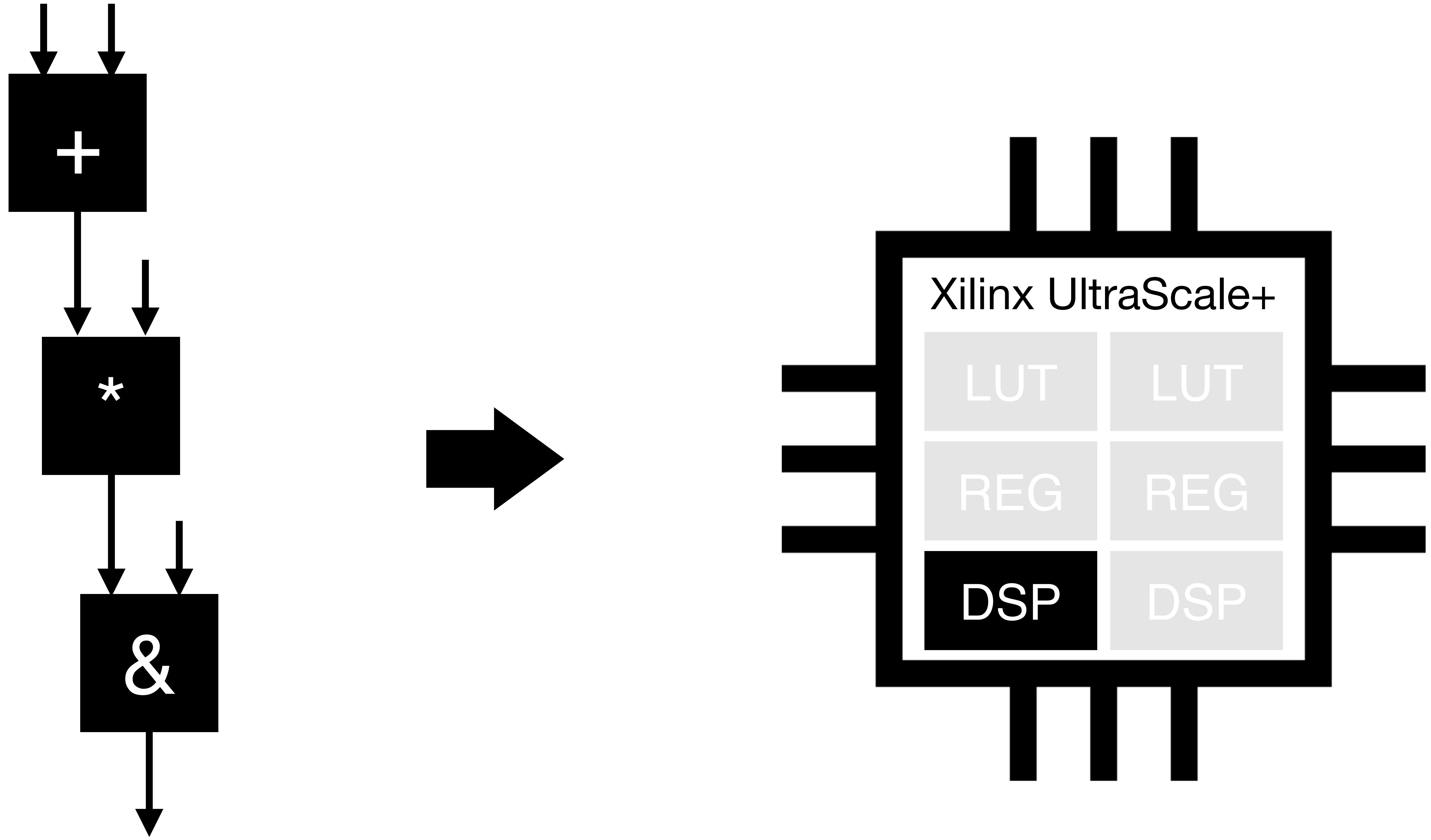
&

Xilinx UltraScale+

LUT LUT

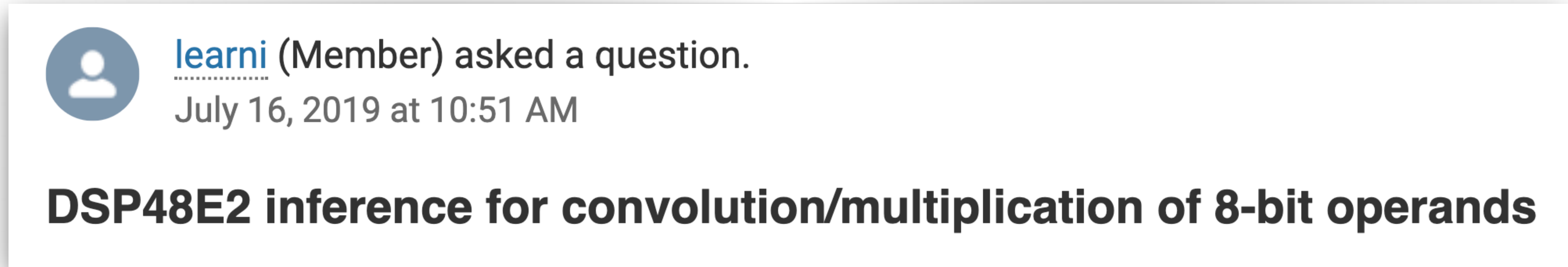
REG REG

DSP DSP


+
*
&
Xilinx UltraScale+
LUT
LUT
REG
REG
DSP
DSP

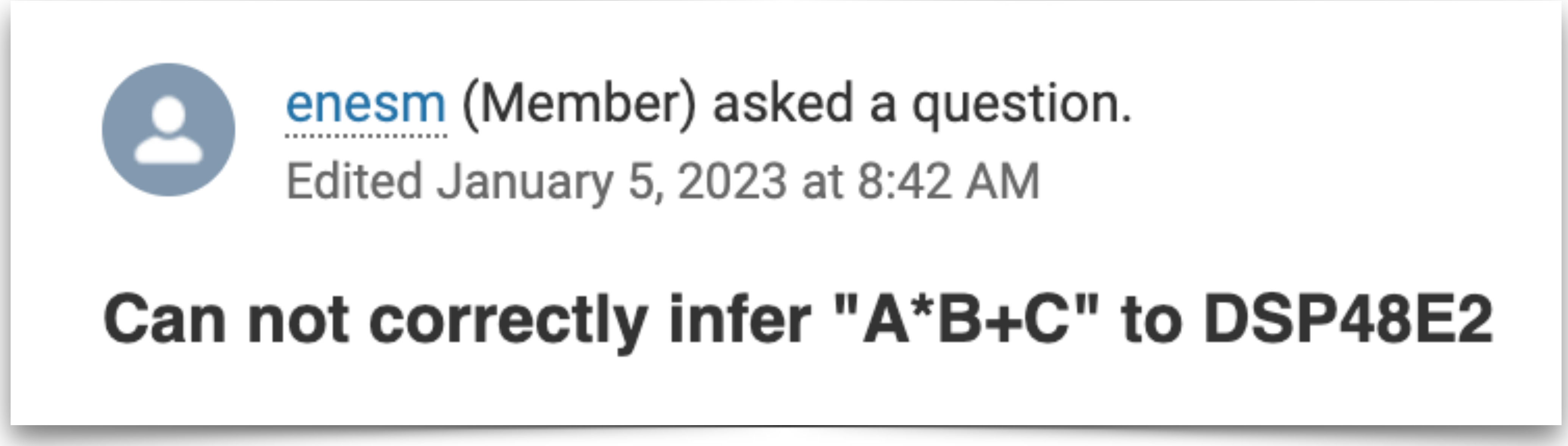
When we attempt to map the design with the proprietary state of the art compiler…

```
+-----------+------+
| Primitive | Used |
+-----------+------+
| DSP       |    1 |
| LUT       |    8 |
| Register  |    8 |
+-----------+------+
```



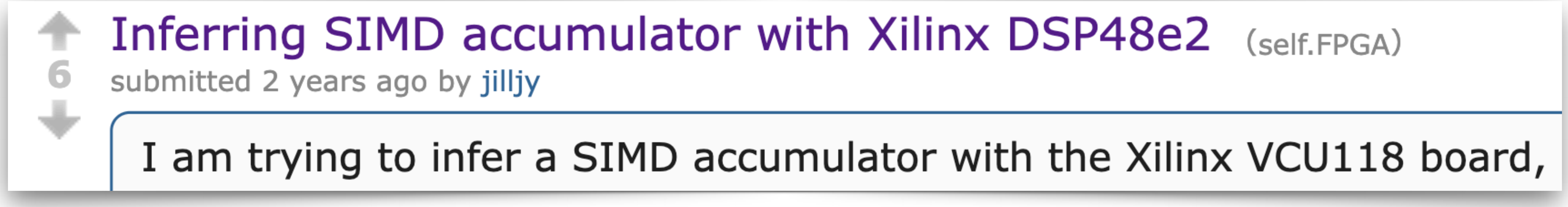
learni (Member) asked a question.
July 16, 2019 at 10:51 AM

DSP48E2 inference for convolution/multiplication of 8-bit operands



enesm (Member) asked a question.
Edited January 5, 2023 at 8:42 AM

Can not correctly infer "A*B+C" to DSP48E2



Inferring SIMD accumulator with Xilinx DSP48e2 (self.FPGA)
submitted 2 years ago by jilljy

I am trying to infer a SIMD accumulator with the Xilinx VCU118 board,

Now what?

Give up and accept suboptimal results?

Only one option—manually instantiate the DSP!

```
DSP48E2 #(
  .ACASCREG(32'd1),
  .ADREG(32'd0),
  .ALUMODEREG(32'd0),
  .AMULTSEL("AD"),
  .AREG(32'd1),
  .AUTORESET_PATDET("NO_RESET"),
  .AUTORESET_PRIORITY("RESET"),
  .A_INPUT("DIRECT"),
  .BCASCREG(32'd2),
  .BMULTSEL("B"),
  .BREG(32'd2),
  .B_INPUT("DIRECT"),
  .CARRYINREG(32'd0),
  .CARRYINSELREG(32'd0),
  .CREG(32'd1),
  .DREG(32'd1),
  .INMODEREG(32'd1),
  .IS_ALUMODE_INVERTED(4'h0),
  .IS_CARRYIN_INVERTED(1'h0),
  .IS_CLK_INVERTED(1'h0),
  .IS_INMODE_INVERTED(5'h00),
  .IS_OPMODE_INVERTED(9'h000),
  .IS_RSTALLCARRYIN_INVERTED(1'h0),
  .IS_RSTALUMODE_INVERTED(1'h0),
  .IS_RSTA_INVERTED(1'h0),
  .IS_RSTB_INVERTED(1'h0),
  .IS_RSTCTRL_INVERTED(1'h0),
  .IS_RSTC_INVERTED(1'h0),
  .IS_RSTD_INVERTED(1'h0),
  .IS_RSTINMODE_INVERTED(1'h0),
  .IS_RSTM_INVERTED(1'h0),
  .IS_RSTP_INVERTED(1'h0),
  .MASK(48'h000000000000),
  .MREG(32'd0),
  .OPMODEREG(32'd1),
  .PATTERN(48'h000000000000),
  .PREADDINSEL("A"),
  .PREG(32'd0),
  .RND(48'h000000000000),
  .SEL_MASK("MASK"),
  .SEL_PATTERN("PATTERN"),
  .USE_MULT("MULTIPLY"),
  .USE_PATTERN_DETECT("NO_PATDET"),
  .USE_SIMD("ONE48"),
  .USE_WIDEXOR("FALSE"),
  .XORSIMD("XOR12")
) DSP48E2_0 (
  .A(a),
  .ACIN(30'h00000000),
  .ALUMODE(4'h0),
  .B(b),
  .BCIN(18'h00000),
  .C({ 30'h00000000, c }),
  .CARRYCASCIN(1'h0),
  .CARRYIN(1'h0),
  .CARRYINSEL(3'h7),
  .CEA1(1'h1),
  .CEA2(1'h1),
  .CEAD(1'h1),
  .CEALUMODE(1'h1),
  .CEB1(1'h1),
  .CEB2(1'h1),
  .CEC(1'h1),
  .CECARRYIN(1'h1),
  .CECTRL(1'h1),
  .CED(1'h1),
  .CEINMODE(1'h1),
  .CEM(1'h1),
  .CEP(1'h1),
  .CLK(clk),
  .D(d),
  .INMODE(5'h14),
  .MULTSIGNIN(1'h0),
  .OPMODE(9'h185),
  .P({ P_0[47:18], out }),
  .PCIN(48'h000000000000),
  .RSTA(1'h0),
  .RSTALLCARRYIN(1'h0),
  .RSTALUMODE(1'h0),
  .RSTB(1'h0),
  .RSTC(1'h0),
  .RSTCTRL(1'h0),
  .RSTD(1'h0),
  .RSTINMODE(1'h0),
  .RSTM(1'h0),
  .RSTP(1'h0)
);
```

UltraScale Architecture DSP Slice

User Guide

UG579 (v1.7) June 4, 2018

XILINX
ALL PROGRAMMABLE.

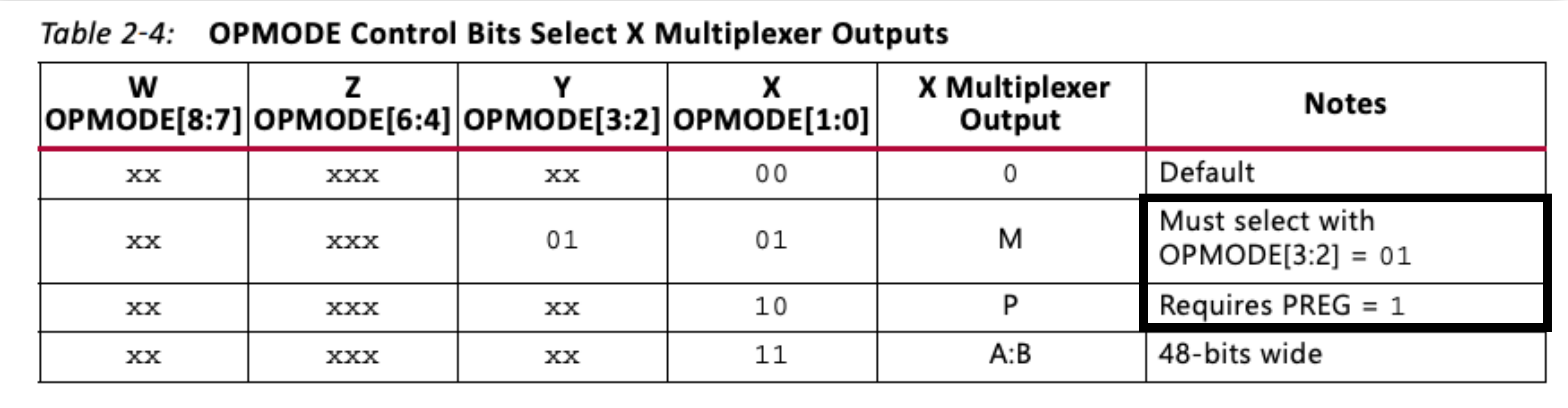
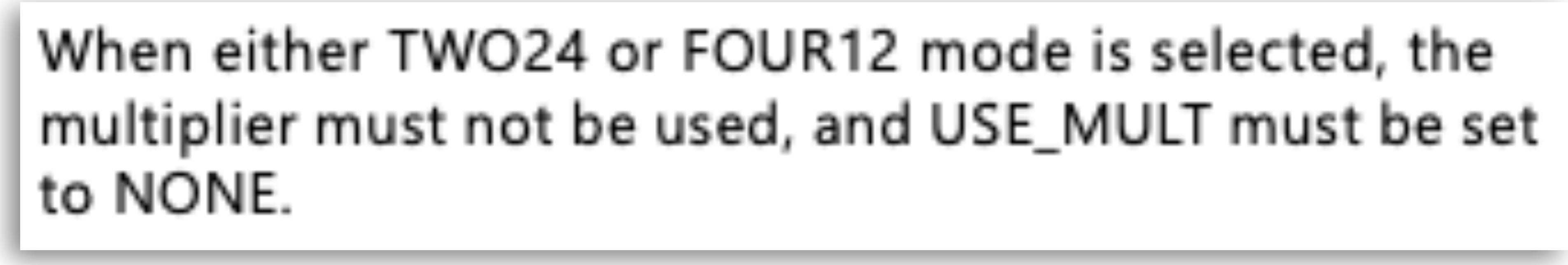
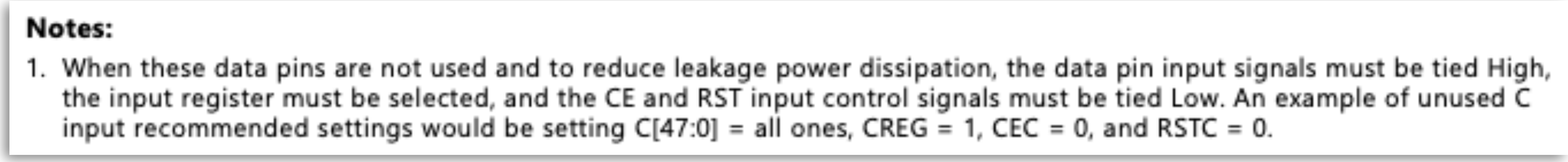


Table 2-4: **OPMODE Control Bits Select X Multiplexer Outputs**

W OPMODE[8:7]	Z OPMODE[6:4]	Y OPMODE[3:2]	X OPMODE[1:0]	X Multiplexer Output	Notes
xx	xxx	xx	00	0	Default
xx	xxx	01	01	M	Must select with OPMODE[3:2] = 01
xx	xxx	xx	10	P	Requires PREG = 1
xx	xxx	xx	11	A:B	48-bits wide



When either TWO24 or FOUR12 mode is selected, the multiplier must not be used, and USE_MULT must be set to NONE.



Notes:

1. When these data pins are not used and to reduce leakage power dissipation, the data pin input signals must be tied High, the input register must be selected, and the CE and RST input control signals must be tied Low. An example of unused C input recommended settings would be setting C[47:0] = all ones, CREG = 1, CEC = 0, and RSTC = 0.

Complex, interdependent constraints between parameter and port values

```
DSP48E2 #(
  .ACASCREG(32'd1),
  .ADREG(32'd0),
  .ALUMODEREG(32'd0),
  .AMULTSEL("AD"),
  .AREG(32'd1),
  .AUTORESET_PATDET("NO_RESET"),
  .AUTORESET_PRIORITY("RESET"),
  .A_INPUT("DIRECT"),
  .BCASCREG(32'd2),
  .BMULTSEL("B"),
  .BREG(32'd2),
  .B_INPUT("DIRECT"),
  .CARRYINREG(32'd0),
  .CARRYINSELREG(32'd0),
  .CREG(32'd1),
  .DREG(32'd1),
  .INMODEREG(32'd1),
  .IS_ALUMODE_INVERTED(4'h0),
  .IS_CARRYIN_INVERTED(1'h0),
  .IS_CLK_INVERTED(1'h0),
  .IS_INMODE_INVERTED(5'h00),
  .IS_OPMODE_INVERTED(9'h000),
  .IS_RSTALLCARRYIN_INVERTED(1'h0),
  .IS_RSTALUMODE_INVERTED(1'h0),
  .IS_RSTA_INVERTED(1'h0),
  .IS_RSTB_INVERTED(1'h0),
  .IS_RSTCTRL_INVERTED(1'h0),
  .IS_RSTC_INVERTED(1'h0),
  .IS_RSTD_INVERTED(1'h0),
  .IS_RSTINMODE_INVERTED(1'h0),
  .IS_RSTM_INVERTED(1'h0),
  .IS_RSTP_INVERTED(1'h0),
  .MASK(48'h000000000000),
  .MREG(32'd0),
  .OPMODEREG(32'd1),
  .PATTERN(48'h000000000000),
  .PREADDINSEL("A"),
  .PREG(32'd0),
  .RND(48'h000000000000),
  .SEL_MASK("MASK"),
         RN("PATTERN"),
          'MULTIPLY"),
         RN_DETECT("NO_PAT
          'ONE48"),
                 "),
         OR12")
  .ACIN(30'h00000000),
  .ALUMODE(4'h0)
  .B(b),
  .BCIN(
  .C({ 30'h00000000, c }),
  .CARRYCASCIN(1'h0),
  .CARRYIN(1'h0),
  .CARRYINSEL(3'h7),
  .CEA1(1'h1),
  .CEA2(1'h1),
  .CEAD(1'h1),
  .CEALUMODE(1'h1),
  .CEB1(1'h1),
  .CEB2(1'h1),
  .CEC(1'h1),
  .CECARRYIN(1'h1),
  .CECTRL(1'h1),
  .CED(1'h1),
  .CEINMODE(1'h1),
  .CEM(1'h1),
  .CEP(1'h1),
  .CLK(clk),
  .D(d),
  .INMODE(5'h14),
  .MULTSIGNIN(1'h0),
  .OPMODE(9'h185),
  .P({ P_0[47:18], out }),
  .PCIN(48'h000000000000),
  .RSTA(1'h0),
  .RSTALLCARRYIN(1'h0),
  .RSTALUMODE(1'h0),
             ),
             ),
  .RSTCTRL(1'h0),
  .RSTD(1'h0),
  .RSTINMODE(1'h0),
  .RSTM(1'h0),
  .RSTP(1'h0)
);
```

```
+-----------+------+
| Primitive | Used |
+-----------+------+
| DSP48E2   |    1 |
+-----------+------+
```

No formal guarantee of correctness!

State of the art tools failed to produce an efficient mapping.

It cost our designer a lot of time and effort to do it manually.

…and they don't even know if their output is correct!

```
DSP48E2 #(
  .ACASCREG(32'd1),
  .ADREG(32'd0),
  .ALUMODEREG(32'd0),
  .AMULTSEL("AD"),
  .AREG(32'd1),
  .AUTORESET_PATDET("NO_RESET"),
  .AUTORESET_PRIORITY("RESET"),
  .A_INPUT("DIRECT"),
  .BCASCREG(32'd2),
  .BMULTSEL("B"),
  .BREG(32'd2),
  .B_INPUT("DIRECT"),
  .CARRYINREG(32'd0),
  .CARRYINSELREG(32'd0),
  .CREG(32'd1),
  .DREG(32'd1),
  .INMODEREG(32'd1),
  .IS_ALUMODE_INVERTED(4'h0),
  .IS_CARRYIN_INVERTED(1'h0),
  .IS_CLK_INVERTED(1'h0),
  .IS_INMODE_INVERTED(5'h00),
  .IS_OPMODE_INVERTED(9'h000),
  .IS_RSTALLCARRYIN_INVERTED(1'h0),
  .IS_RSTALUMODE_INVERTED(1'h0),
  .IS_RSTA_INVERTED(1'h0),
  .IS_RSTB_INVERTED(1'h0),
  .IS_RSTCTRL_INVERTED(1'h0),
  .IS_RSTC_INVERTED(1'h0),
  .IS_RSTD_INVERTED(1'h0),
  .IS_RSTINMODE_INVERTED(1'h0),
  .IS_RSTM_INVERTED(1'h0),
  .IS_RSTP_INVERTED(1'h0),
  .MASK(48'h000000000000),
  .MREG(32'd0),
  .OPMODEREG(32'd1),
  .PATTERN(48'h000000000000),
  .PREADDINSEL("A"),
  .PREG(32'd0),
  .RND(48'h000000000000),
  .SEL_MASK("MASK"),
  .SEL_PATTERN("PATTERN"),
  .USE_MULT("MULTIPLY"),
  .USE_PATTERN_DETECT("NO_PATDET"),
  .USE_SIMD("ONE48"),
  .USE_WIDEXOR("FALSE"),
  .XORSIMD("XOR12")
) DSP48E2_0 (
  .A(a),
  .ACIN(30'h00000000),
  .ALUMODE(4'h0),
  .B(b),
  .BCIN(18'h00000),
  .C({ 30'h00000000, c }),
  .CARRYCASCIN(1'h0),
  .CARRYIN(1'h0),
  .CARRYINSEL(3'h7),
  .CEA1(1'h1),
  .CEA2(1'h1),
  .CEAD(1'h1),
  .CEALUMODE(1'h1),
  .CEB1(1'h1),
  .CEB2(1'h1),
  .CEC(1'h1),
  .CECARRYIN(1'h1),
  .CECTRL(1'h1),
  .CED(1'h1),
  .CEINMODE(1'h1),
  .CEM(1'h1),
  .CEP(1'h1),
  .CLK(clk),
  .D(d),
  .INMODE(5'h14),
  .MULTSIGNIN(1'h0),
  .OPMODE(9'h185),
  .P({ P_0[47:18], out }),
  .PCIN(48'h000000000000),
  .RSTA(1'h0),
  .RSTALLCARRYIN(1'h0),
  .RSTALUMODE(1'h0),
  .RSTB(1'h0),
  .RSTC(1'h0),
  .RSTCTRL(1'h0),
  .RSTD(1'h0),
  .RSTINMODE(1'h0),
  .RSTM(1'h0),
  .RSTP(1'h0)
);
```

UltraScale Architecture DSP Slice

User Guide

UG579 (v1.7) June 4, 2018

XILINX
ALL PROGRAMMABLE.

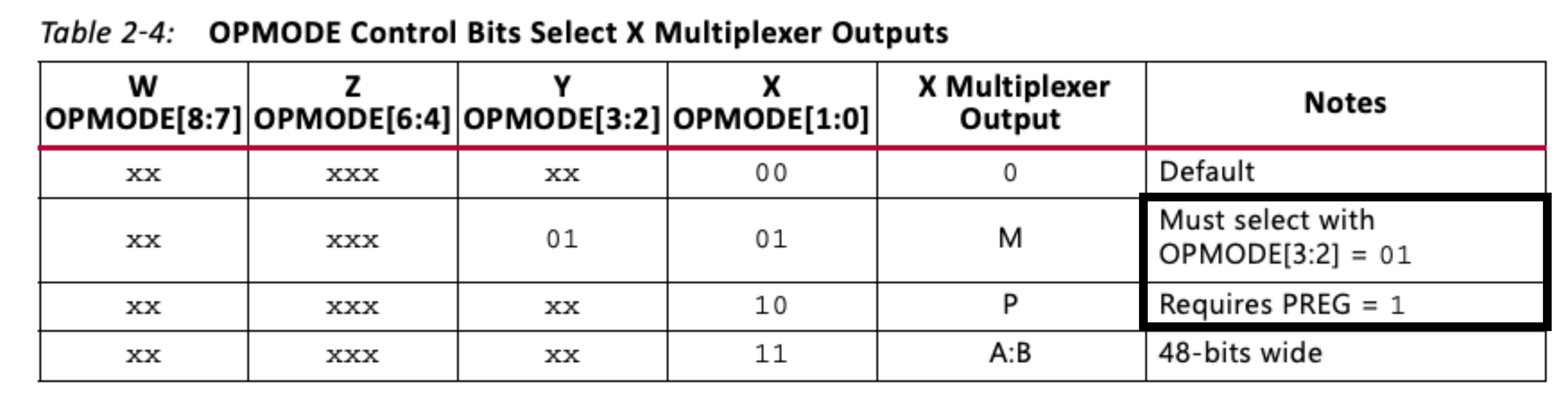
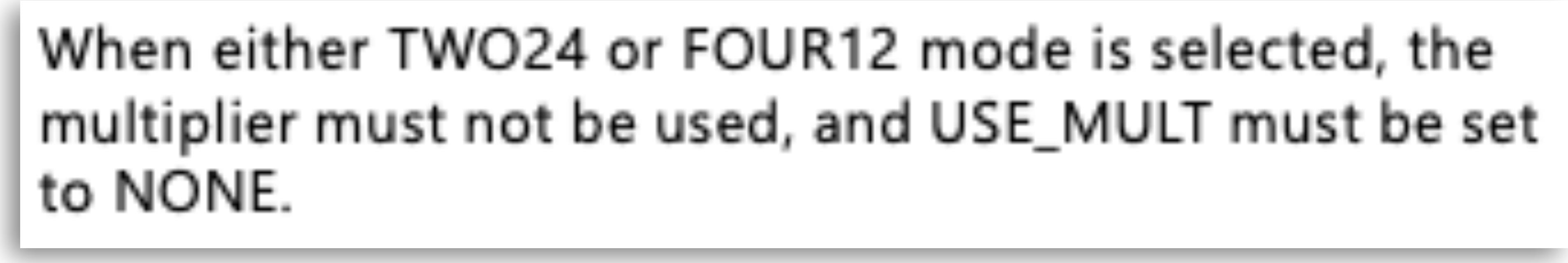
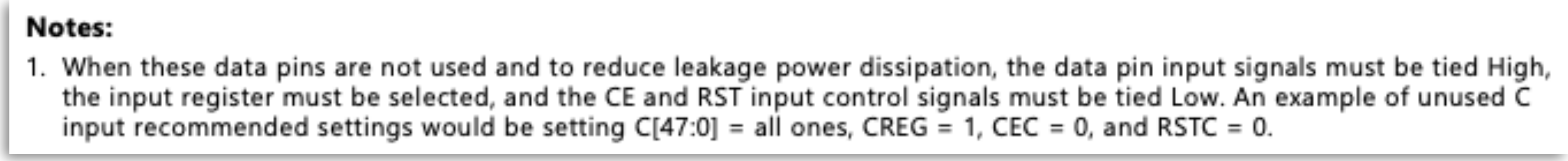


Table 2-4: **OPMODE Control Bits Select X Multiplexer Outputs**

W OPMODE[8:7]	Z OPMODE[6:4]	Y OPMODE[3:2]	X OPMODE[1:0]	X Multiplexer Output	Notes
xx	xxx	xx	00	0	Default
xx	xxx	01	01	M	Must select with OPMODE[3:2] = 01
xx	xxx	xx	10	P	Requires PREG = 1
xx	xxx	xx	11	A:B	48-bits wide



When either TWO24 or FOUR12 mode is selected, the multiplier must not be used, and USE_MULT must be set to NONE.



Notes:

1. When these data pins are not used and to reduce leakage power dissipation, the data pin input signals must be tied High, the input register must be selected, and the CE and RST input control signals must be tied Low. An example of unused C input recommended settings would be setting C[47:0] = all ones, CREG = 1, CEC = 0, and RSTC = 0.

Configuring a DSP is like writing a program in a domain-specific language!

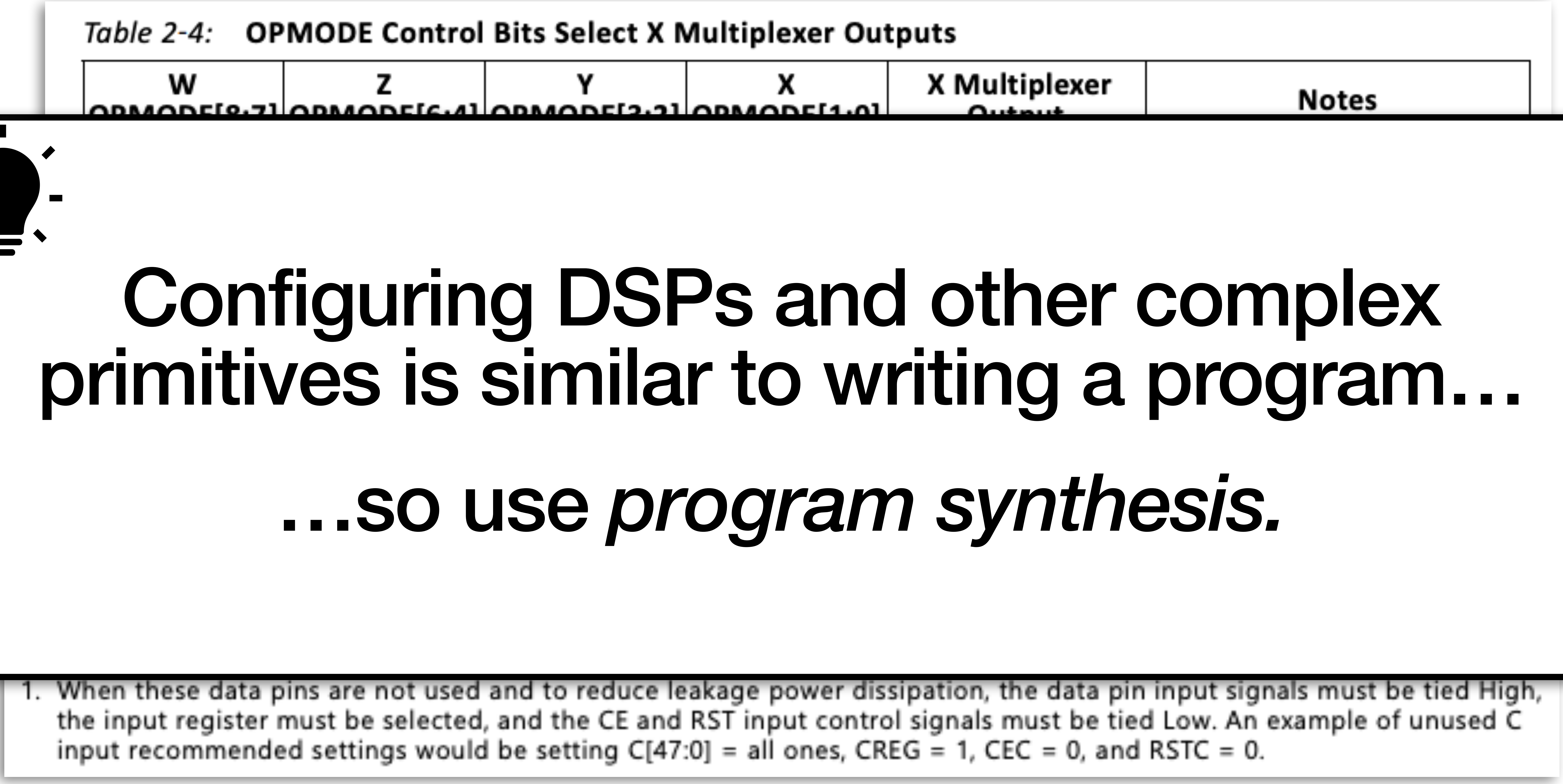


Table 2-4: **OPMODE Control Bits Select X Multiplexer Outputs**

W OPMODE[8:7]	Z OPMODE[6:4]	Y OPMODE[3:2]	X OPMODE[1:0]	X Multiplexer Output	Notes

1. When these data pins are not used and to reduce leakage power dissipation, the data pin input signals must be tied High, the input register must be selected, and the CE and RST input control signals must be tied Low. An example of unused C input recommended settings would be setting C[47:0] = all ones, CREG = 1, CEC = 0, and RSTC = 0.

Configuring DSPs and other complex primitives is similar to writing a program…

…so use *program synthesis.*

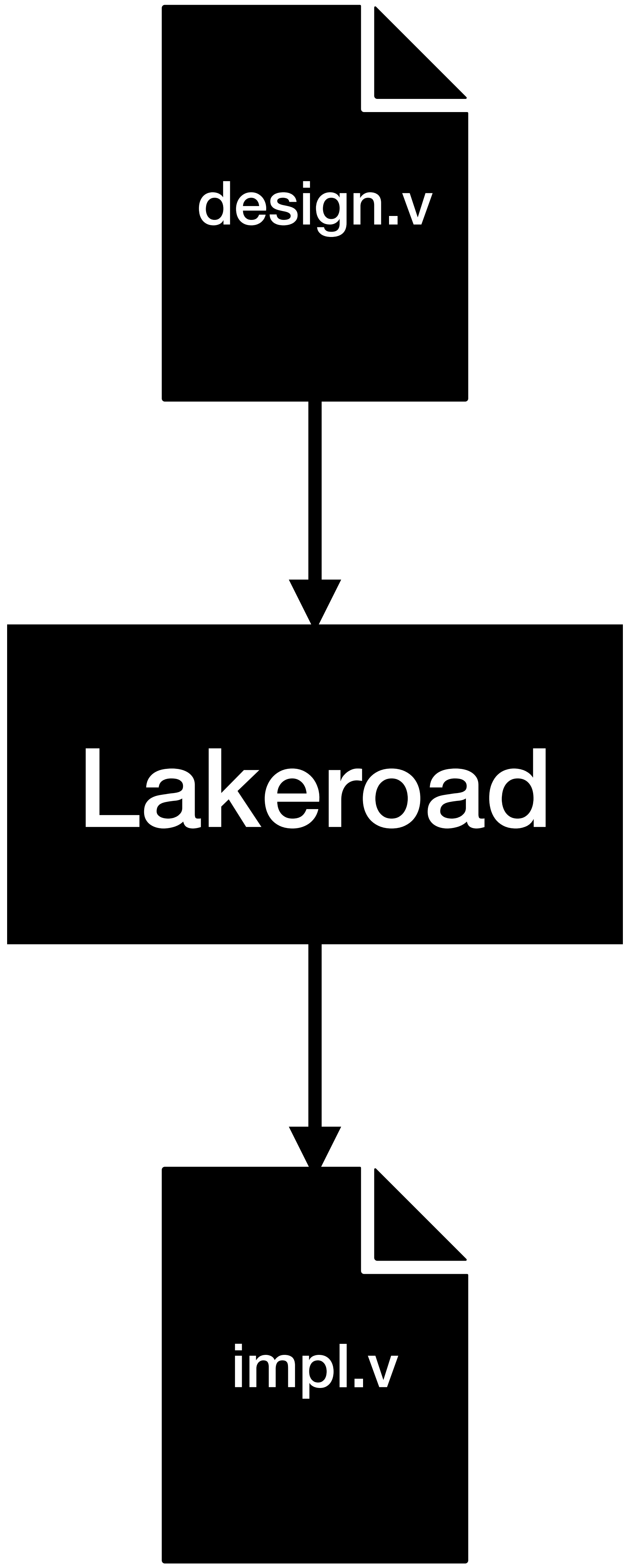
- Problem
- Insight
- Implementation
- Evaluation

Problem

Insight

Implementation

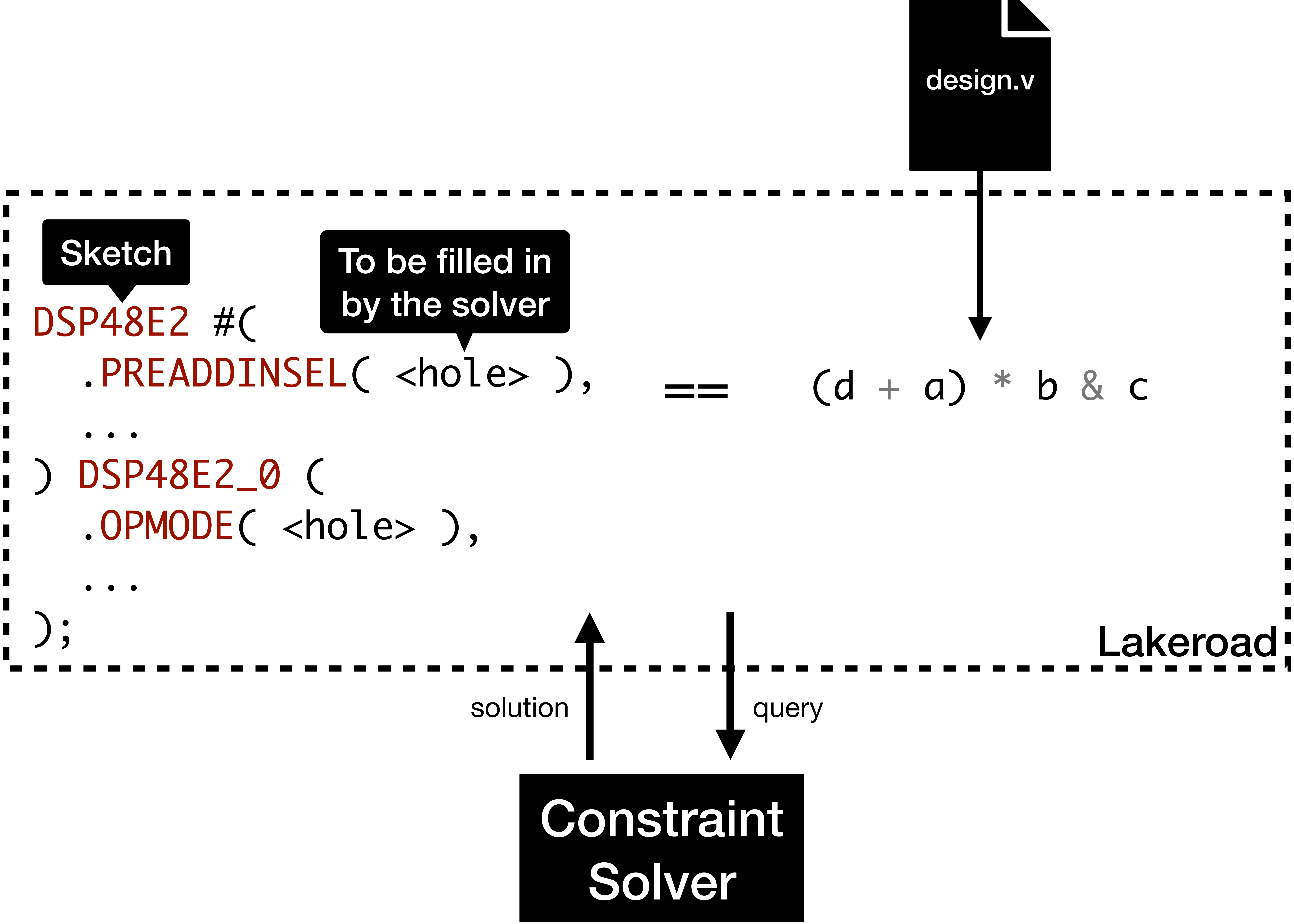
Evaluation



design.v

Lakeroad

impl.v



design.v

Sketch

To be filled in by the solver

```
DSP48E2 #(
  .PREADDINSEL( <hole> ),
  ...
) DSP48E2_0 (
  .OPMODE( <hole> ),
  ...
);
```

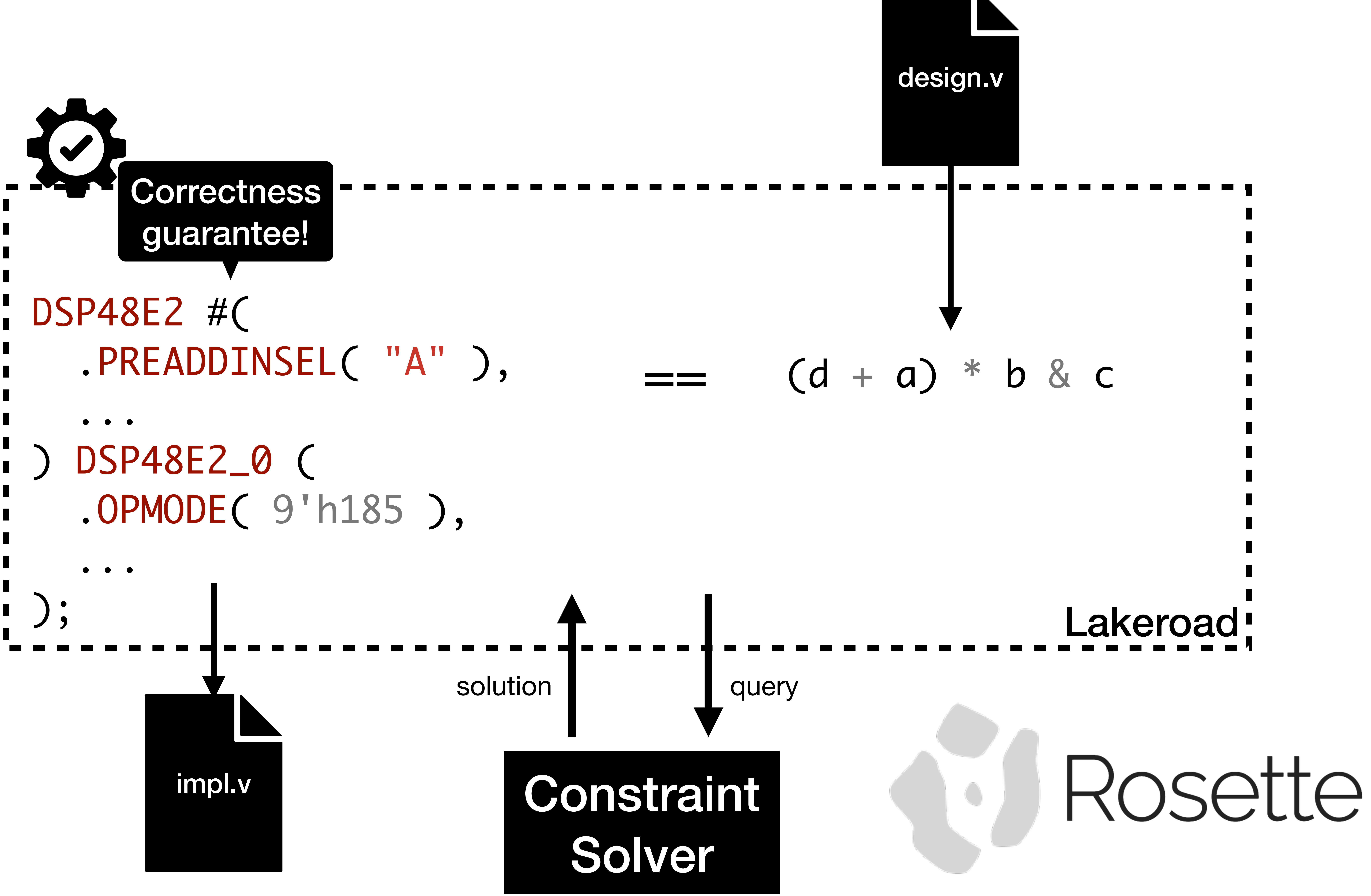
```
==    (d + a) * b & c
```

Lakeroad

solution

query

Constraint Solver

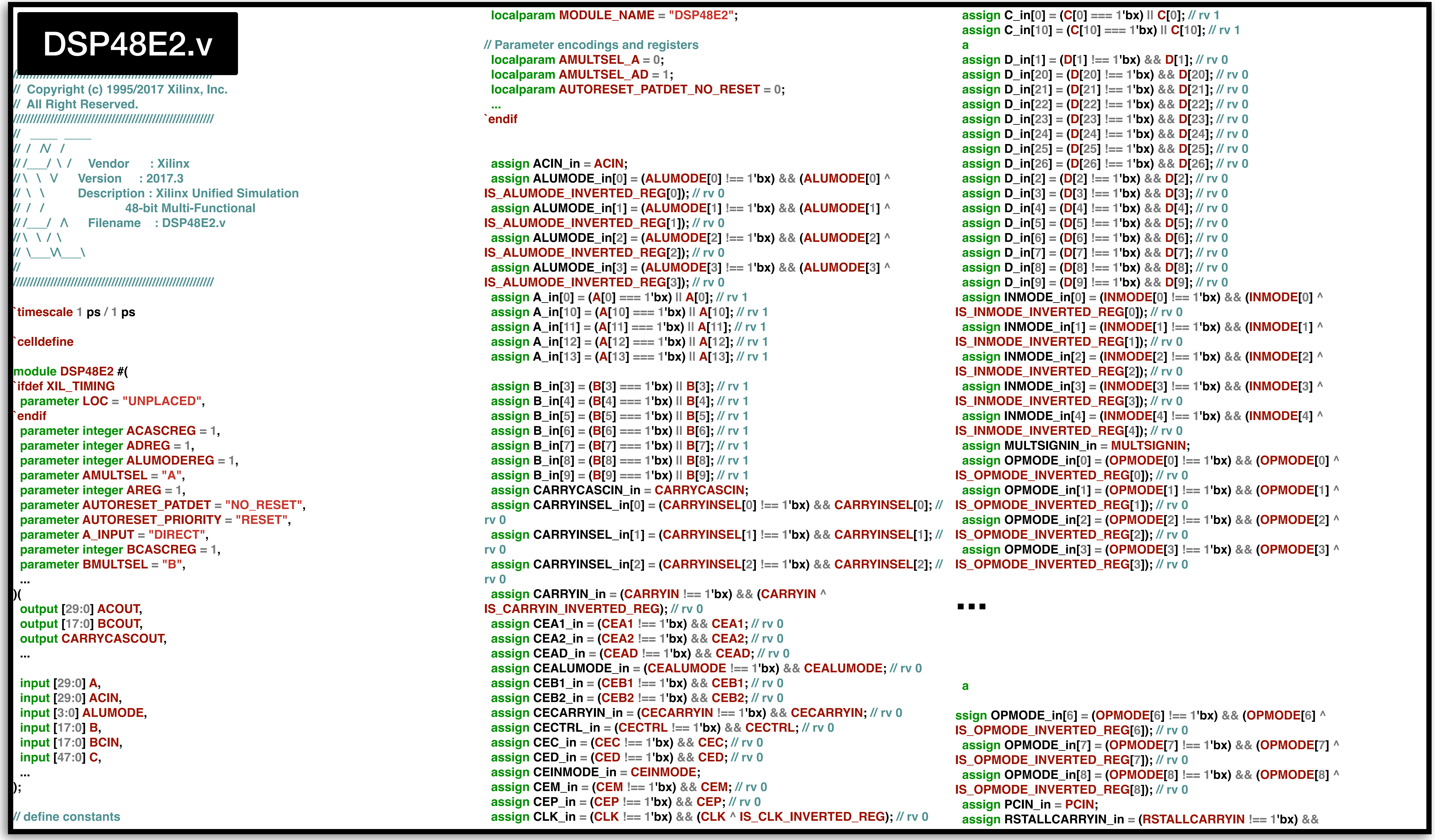


```
DSP48E2 #(
   .PREADDINSEL( "A" ),
   ...
) DSP48E2_0 (
   .OPMODE( 9'h185 ),
   ...
);
```

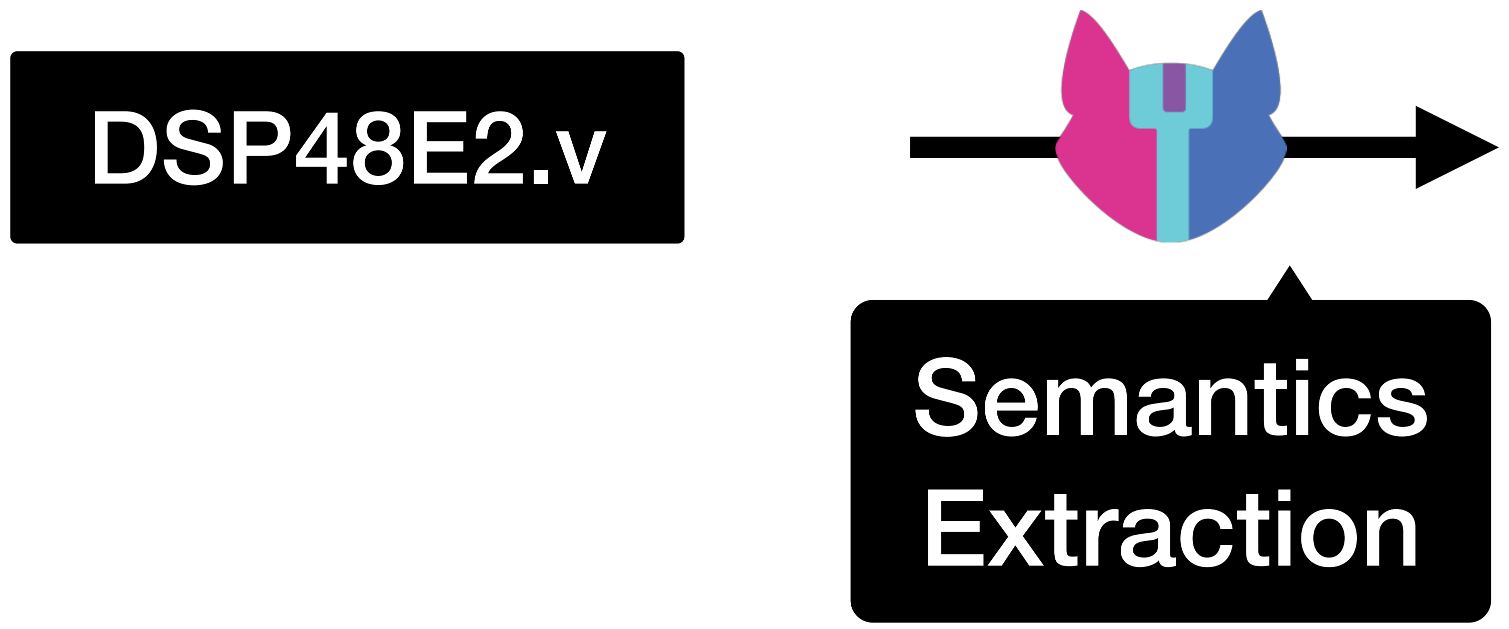
```
DSP48E2 #(
   .PREADDINSEL( <hole> ),
   ...
) DSP48E2_0 (
   .OPMODE( <hole> ),
   ...
);
```

The solver doesn’t know what this means!

DSP48E2.v



```
///////////////////////////////////////////////////////
//  Copyright (c) 1995/2017 Xilinx, Inc.
//  All Right Reserved.
///////////////////////////////////////////////////////
//   ____  ____
//  /   /\/   /
// /___/  \  /    Vendor      : Xilinx
// \   \   \/     Version     : 2017.3
//  \   \         Description : Xilinx Unified Simulation
//  /   /                       48-bit Multi-Functional
// /___/   /\     Filename    : DSP48E2.v
// \   \  /  \
//  \___\/\___\
//
///////////////////////////////////////////////////////

`timescale 1 ps / 1 ps

`celldefine

module DSP48E2 #(
`ifdef XIL_TIMING
  parameter LOC = "UNPLACED",
`endif
  parameter integer ACASCREG = 1,
  parameter integer ADREG = 1,
  parameter integer ALUMODEREG = 1,
  parameter AMULTSEL = "A",
  parameter integer AREG = 1,
  parameter AUTORESET_PATDET = "NO_RESET",
  parameter AUTORESET_PRIORITY = "RESET",
  parameter A_INPUT = "DIRECT",
  parameter integer BCASCREG = 1,
  parameter BMULTSEL = "B",
  ...
)(
  output [29:0] ACOUT,
  output [17:0] BCOUT,
  output CARRYCASCOUT,
  ...

  input [29:0] A,
  input [29:0] ACIN,
  input [3:0] ALUMODE,
  input [17:0] B,
  input [17:0] BCIN,
  input [47:0] C,
  ...
);

// define constants
 localparam MODULE_NAME = "DSP48E2";

// Parameter encodings and registers
 localparam AMULTSEL_A = 0;
 localparam AMULTSEL_AD = 1;
 localparam AUTORESET_PATDET_NO_RESET = 0;
 ...
`endif


 assign ACIN_in = ACIN;
 assign ALUMODE_in[0] = (ALUMODE[0] !== 1'bx) && (ALUMODE[0] ^
IS_ALUMODE_INVERTED_REG[0]); // rv 0
 assign ALUMODE_in[1] = (ALUMODE[1] !== 1'bx) && (ALUMODE[1] ^
IS_ALUMODE_INVERTED_REG[1]); // rv 0
 assign ALUMODE_in[2] = (ALUMODE[2] !== 1'bx) && (ALUMODE[2] ^
IS_ALUMODE_INVERTED_REG[2]); // rv 0
 assign ALUMODE_in[3] = (ALUMODE[3] !== 1'bx) && (ALUMODE[3] ^
IS_ALUMODE_INVERTED_REG[3]); // rv 0
 assign A_in[0] = (A[0] === 1'bx) || A[0]; // rv 1
 assign A_in[10] = (A[10] === 1'bx) || A[10]; // rv 1
 assign A_in[11] = (A[11] === 1'bx) || A[11]; // rv 1
 assign A_in[12] = (A[12] === 1'bx) || A[12]; // rv 1
 assign A_in[13] = (A[13] === 1'bx) || A[13]; // rv 1

 assign B_in[3] = (B[3] === 1'bx) || B[3]; // rv 1
 assign B_in[4] = (B[4] === 1'bx) || B[4]; // rv 1
 assign B_in[5] = (B[5] === 1'bx) || B[5]; // rv 1
 assign B_in[6] = (B[6] === 1'bx) || B[6]; // rv 1
 assign B_in[7] = (B[7] === 1'bx) || B[7]; // rv 1
 assign B_in[8] = (B[8] === 1'bx) || B[8]; // rv 1
 assign B_in[9] = (B[9] === 1'bx) || B[9]; // rv 1
 assign CARRYCASCIN_in = CARRYCASCIN;
 assign CARRYINSEL_in[0] = (CARRYINSEL[0] !== 1'bx) && CARRYINSEL[0]; //
rv 0
 assign CARRYINSEL_in[1] = (CARRYINSEL[1] !== 1'bx) && CARRYINSEL[1]; //
rv 0
 assign CARRYINSEL_in[2] = (CARRYINSEL[2] !== 1'bx) && CARRYINSEL[2]; //
rv 0
 assign CARRYIN_in = (CARRYIN !== 1'bx) && (CARRYIN ^
IS_CARRYIN_INVERTED_REG); // rv 0
 assign CEA1_in = (CEA1 !== 1'bx) && CEA1; // rv 0
 assign CEA2_in = (CEA2 !== 1'bx) && CEA2; // rv 0
 assign CEAD_in = (CEAD !== 1'bx) && CEAD; // rv 0
 assign CEALUMODE_in = (CEALUMODE !== 1'bx) && CEALUMODE; // rv 0
 assign CEB1_in = (CEB1 !== 1'bx) && CEB1; // rv 0
 assign CEB2_in = (CEB2 !== 1'bx) && CEB2; // rv 0
 assign CECARRYIN_in = (CECARRYIN !== 1'bx) && CECARRYIN; // rv 0
 assign CECTRL_in = (CECTRL !== 1'bx) && CECTRL; // rv 0
 assign CEC_in = (CEC !== 1'bx) && CEC; // rv 0
 assign CED_in = (CED !== 1'bx) && CED; // rv 0
 assign CEINMODE_in = CEINMODE;
 assign CEM_in = (CEM !== 1'bx) && CEM; // rv 0
 assign CEP_in = (CEP !== 1'bx) && CEP; // rv 0
 assign CLK_in = (CLK !== 1'bx) && (CLK ^ IS_CLK_INVERTED_REG); // rv 0

 assign C_in[0] = (C[0] === 1'bx) || C[0]; // rv 1
 assign C_in[10] = (C[10] === 1'bx) || C[10]; // rv 1
 a
 assign D_in[1] = (D[1] !== 1'bx) && D[1]; // rv 0
 assign D_in[20] = (D[20] !== 1'bx) && D[20]; // rv 0
 assign D_in[21] = (D[21] !== 1'bx) && D[21]; // rv 0
 assign D_in[22] = (D[22] !== 1'bx) && D[22]; // rv 0
 assign D_in[23] = (D[23] !== 1'bx) && D[23]; // rv 0
 assign D_in[24] = (D[24] !== 1'bx) && D[24]; // rv 0
 assign D_in[25] = (D[25] !== 1'bx) && D[25]; // rv 0
 assign D_in[26] = (D[26] !== 1'bx) && D[26]; // rv 0
 assign D_in[2] = (D[2] !== 1'bx) && D[2]; // rv 0
 assign D_in[3] = (D[3] !== 1'bx) && D[3]; // rv 0
 assign D_in[4] = (D[4] !== 1'bx) && D[4]; // rv 0
 assign D_in[5] = (D[5] !== 1'bx) && D[5]; // rv 0
 assign D_in[6] = (D[6] !== 1'bx) && D[6]; // rv 0
 assign D_in[7] = (D[7] !== 1'bx) && D[7]; // rv 0
 assign D_in[8] = (D[8] !== 1'bx) && D[8]; // rv 0
 assign D_in[9] = (D[9] !== 1'bx) && D[9]; // rv 0
 assign INMODE_in[0] = (INMODE[0] !== 1'bx) && (INMODE[0] ^
IS_INMODE_INVERTED_REG[0]); // rv 0
 assign INMODE_in[1] = (INMODE[1] !== 1'bx) && (INMODE[1] ^
IS_INMODE_INVERTED_REG[1]); // rv 0
 assign INMODE_in[2] = (INMODE[2] !== 1'bx) && (INMODE[2] ^
IS_INMODE_INVERTED_REG[2]); // rv 0
 assign INMODE_in[3] = (INMODE[3] !== 1'bx) && (INMODE[3] ^
IS_INMODE_INVERTED_REG[3]); // rv 0
 assign INMODE_in[4] = (INMODE[4] !== 1'bx) && (INMODE[4] ^
IS_INMODE_INVERTED_REG[4]); // rv 0
 assign MULTSIGNIN_in = MULTSIGNIN;
 assign OPMODE_in[0] = (OPMODE[0] !== 1'bx) && (OPMODE[0] ^
IS_OPMODE_INVERTED_REG[0]); // rv 0
 assign OPMODE_in[1] = (OPMODE[1] !== 1'bx) && (OPMODE[1] ^
IS_OPMODE_INVERTED_REG[1]); // rv 0
 assign OPMODE_in[2] = (OPMODE[2] !== 1'bx) && (OPMODE[2] ^
IS_OPMODE_INVERTED_REG[2]); // rv 0
 assign OPMODE_in[3] = (OPMODE[3] !== 1'bx) && (OPMODE[3] ^
IS_OPMODE_INVERTED_REG[3]); // rv 0


■■■


 a

ssign OPMODE_in[6] = (OPMODE[6] !== 1'bx) && (OPMODE[6] ^
IS_OPMODE_INVERTED_REG[6]); // rv 0
 assign OPMODE_in[7] = (OPMODE[7] !== 1'bx) && (OPMODE[7] ^
IS_OPMODE_INVERTED_REG[7]); // rv 0
 assign OPMODE_in[8] = (OPMODE[8] !== 1'bx) && (OPMODE[8] ^
IS_OPMODE_INVERTED_REG[8]); // rv 0
 assign PCIN_in = PCIN;
 assign RSTALLCARRYIN_in = (RSTALLCARRYIN !== 1'bx) &&
```



```
(ite (bveq (bv #b0 1) (extract 0 0 PREG))
     (bvxor
      (concat
       (extract 1 1
        (ite (bveq (bv #b0 1) (extract 0 0 ALUMODEREG))
             (bvxor ALUMODE IS_ALUMODE_INVERTED)
             (bv #x0 4)))
       (concat
        (extract 1 1
         (ite (bveq (bv #b0 1) (extract 0 0 ALUMODEREG))
              (bvxor ALUMODE IS_ALUMODE_INVERTED)
              (bv #x0 4))) ...)) ...) ...)
```

Problem

Insight

Implementation

Evaluation

Problem

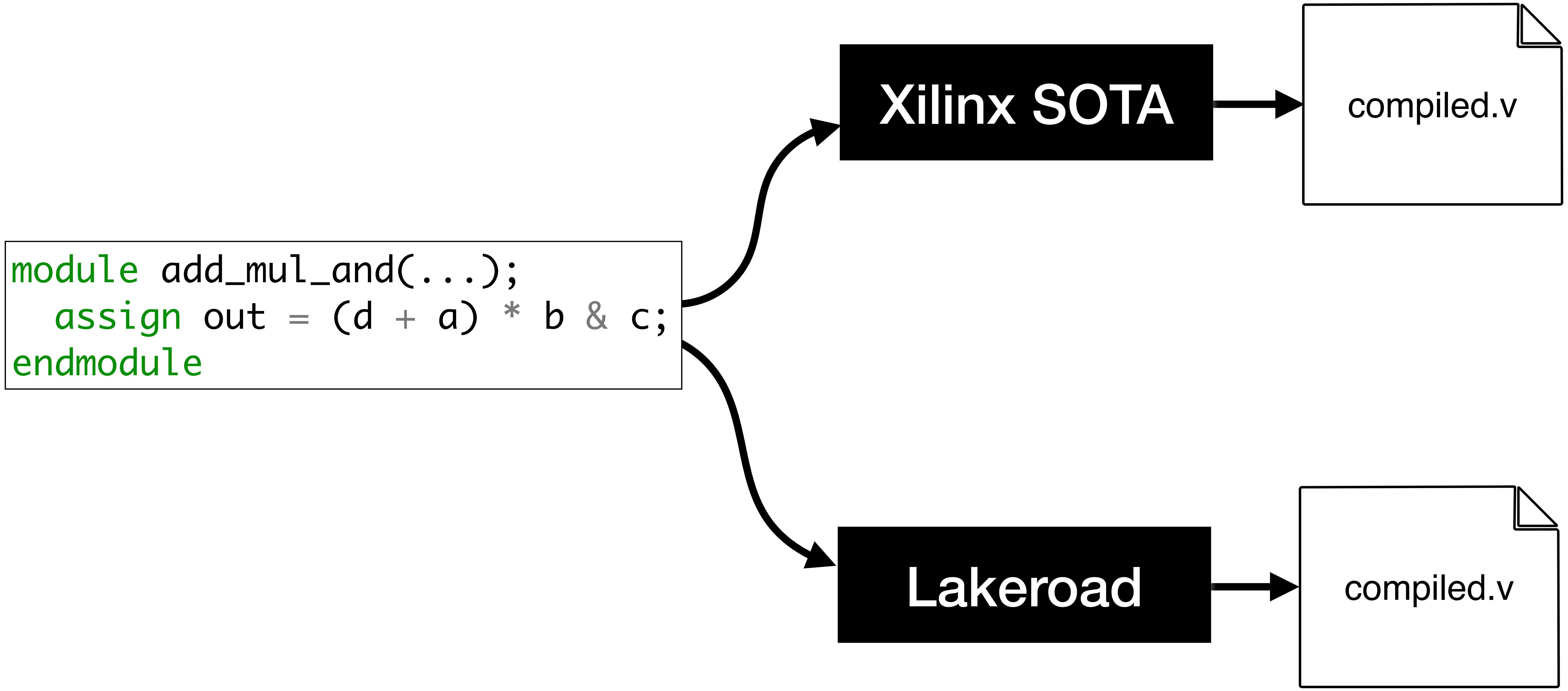
Insight

Implementation

Evaluation

Completeness: Lakeroad finds more mappings than other tools.

Completeness: Lakeroad finds more mappings than other tools.



```
module add_mul_and(...);
  assign out = (d + a) * b & c;
endmodule
```

```
+----------+-------+
| Ref Name | Used |
+----------+-------+
| DSP48E2  |    1 |
| LUT2     |    8 |
| Register |    8 |
+----------+-------+
```

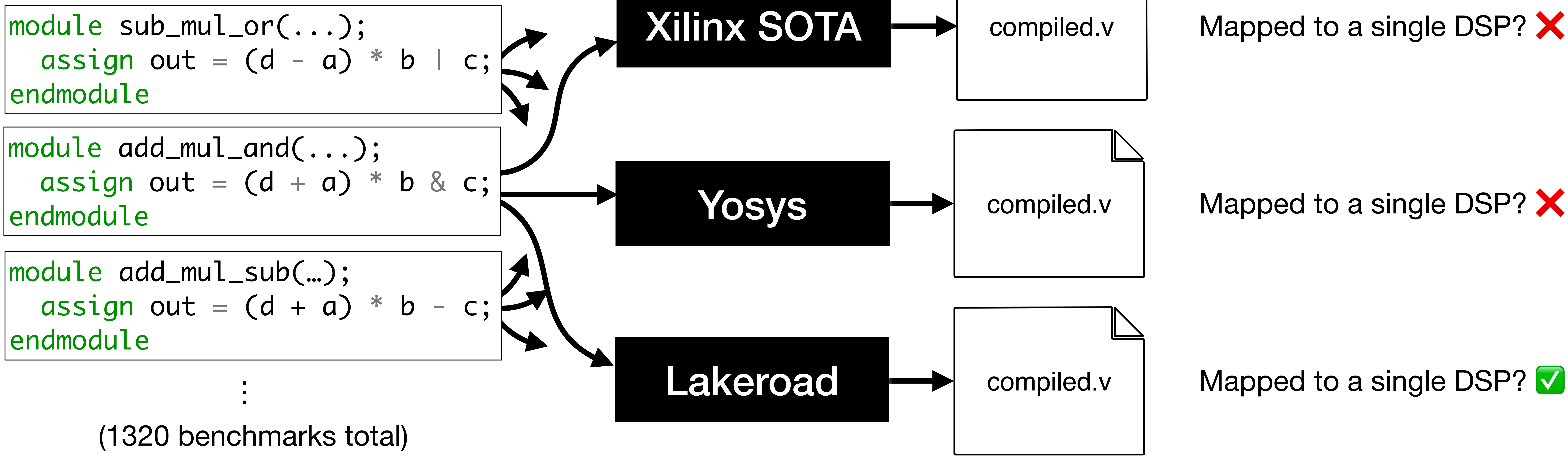
Mapped to a single DSP? ❌

```
+----------+-------+
| Ref Name | Used |
+----------+-------+
| DSP48E2  |    1 |
+----------+-------+
```

Mapped to a single DSP? ✅

Completeness: Lakeroad finds more mappings than other tools.

Each *should* go on a single DSP (according to DSP documentation)



```
module sub_mul_or(...);
  assign out = (d - a) * b | c;
endmodule
```

```
module add_mul_and(...);
  assign out = (d + a) * b & c;
endmodule
```

```
module add_mul_sub(…);
  assign out = (d + a) * b - c;
endmodule
```

⋮

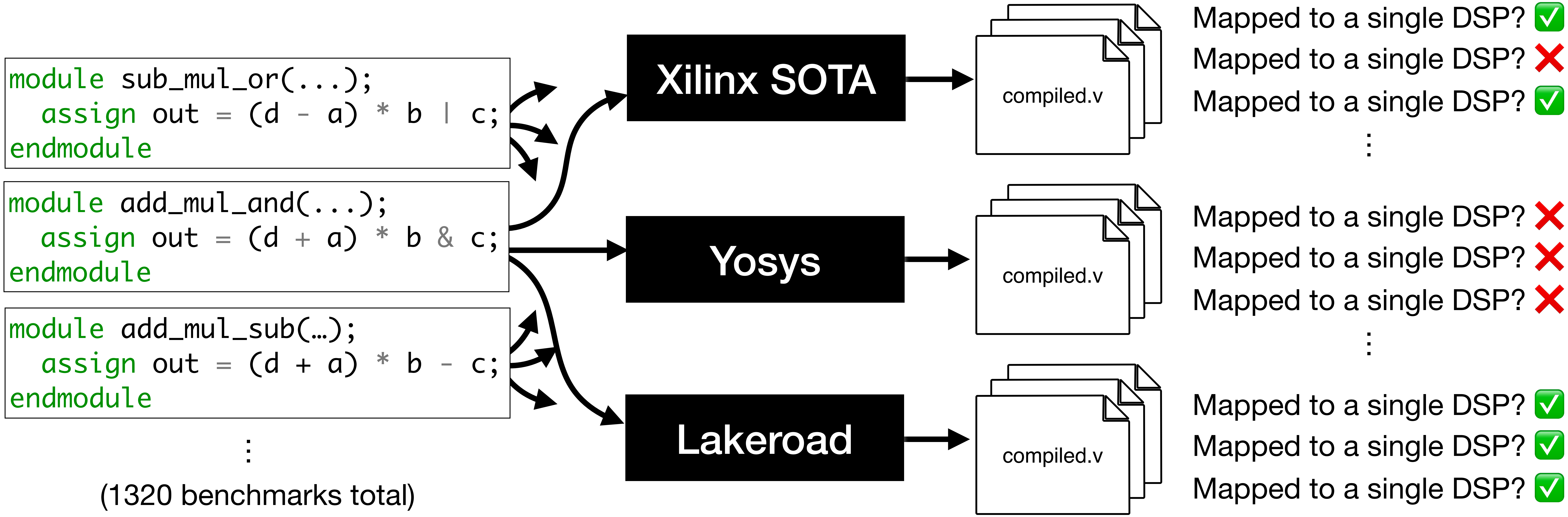
(1320 benchmarks total)

Xilinx SOTA → compiled.v — Mapped to a single DSP? ❌

Yosys → compiled.v — Mapped to a single DSP? ❌

Lakeroad → compiled.v — Mapped to a single DSP? ✅

Completeness: Lakeroad finds more mappings than other tools.



```
module sub_mul_or(...);
  assign out = (d - a) * b | c;
endmodule
```

```
module add_mul_and(...);
  assign out = (d + a) * b & c;
endmodule
```

```
module add_mul_sub(…);
  assign out = (d + a) * b - c;
endmodule
```

⋮

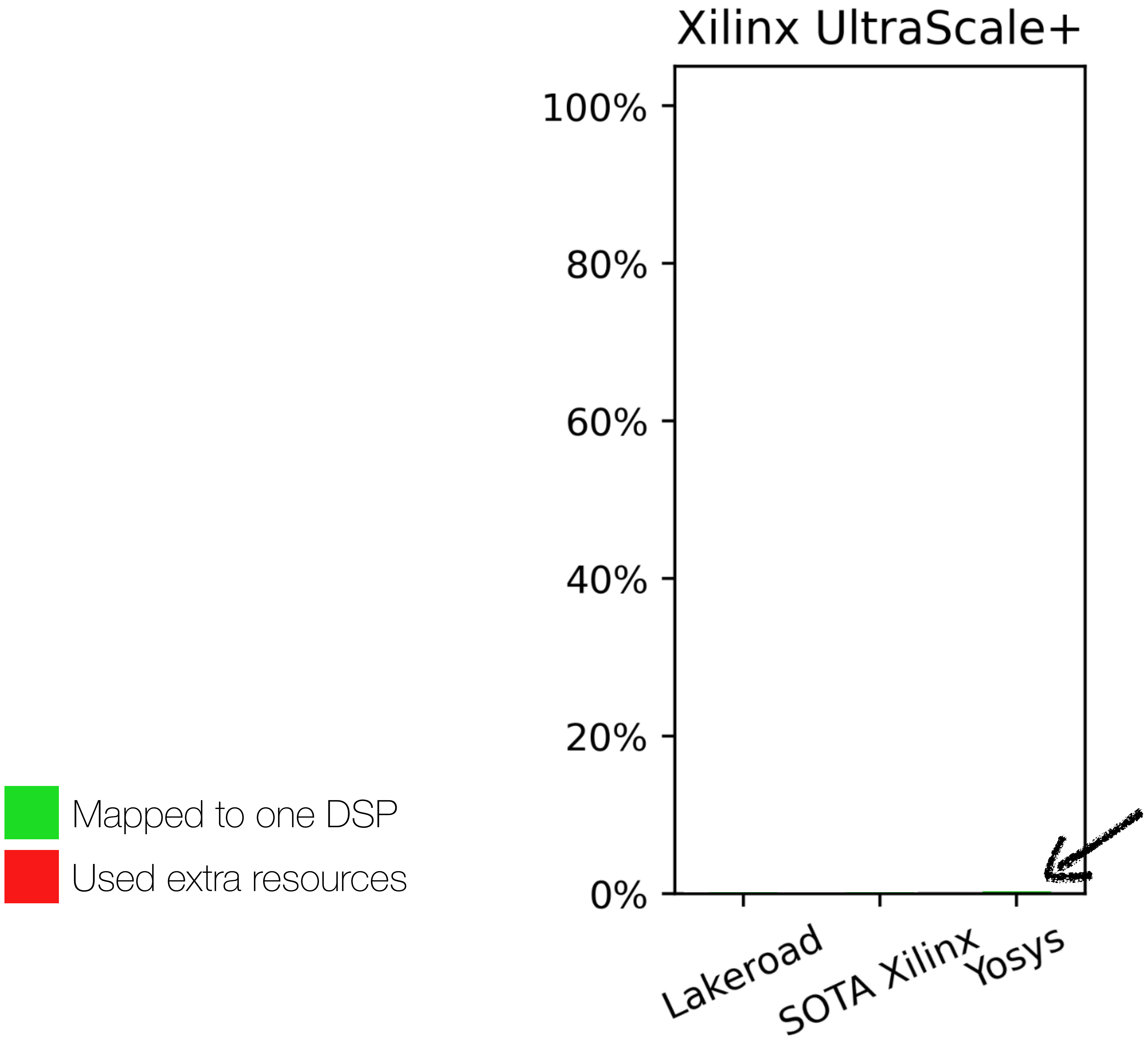
(1320 benchmarks total)

Mapped to a single DSP? ✅
Mapped to a single DSP? ❌
Mapped to a single DSP? ✅
⋮

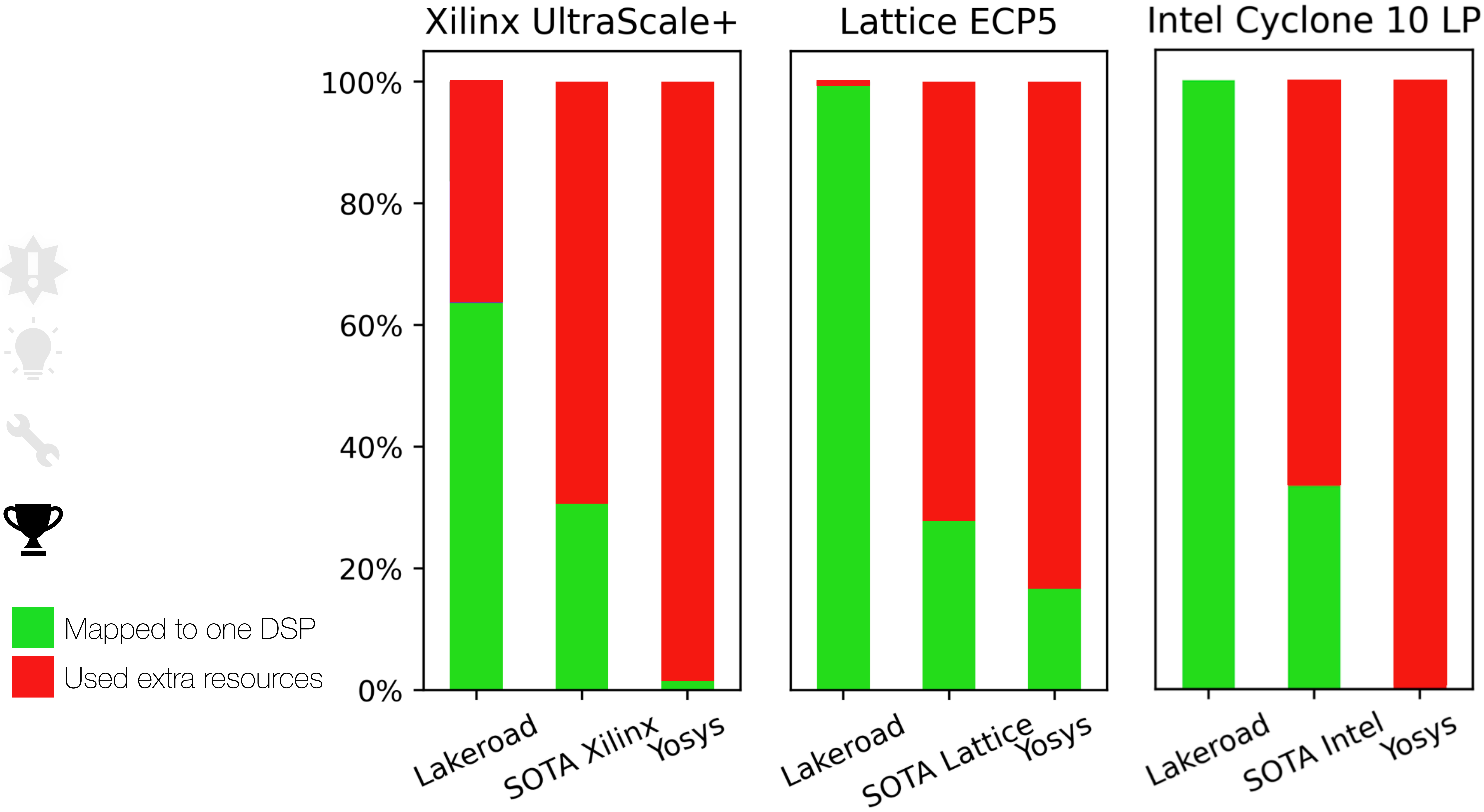
Mapped to a single DSP? ❌
Mapped to a single DSP? ❌
Mapped to a single DSP? ❌
⋮

Mapped to a single DSP? ✅
Mapped to a single DSP? ✅
Mapped to a single DSP? ✅
⋮

Completeness: Lakeroad finds more mappings than other tools.



Completeness: Lakeroad finds more mappings than other tools.



Completeness: Lakeroad finds more mappings than other tools.

Problem

Insight

Implementation

Evaluation

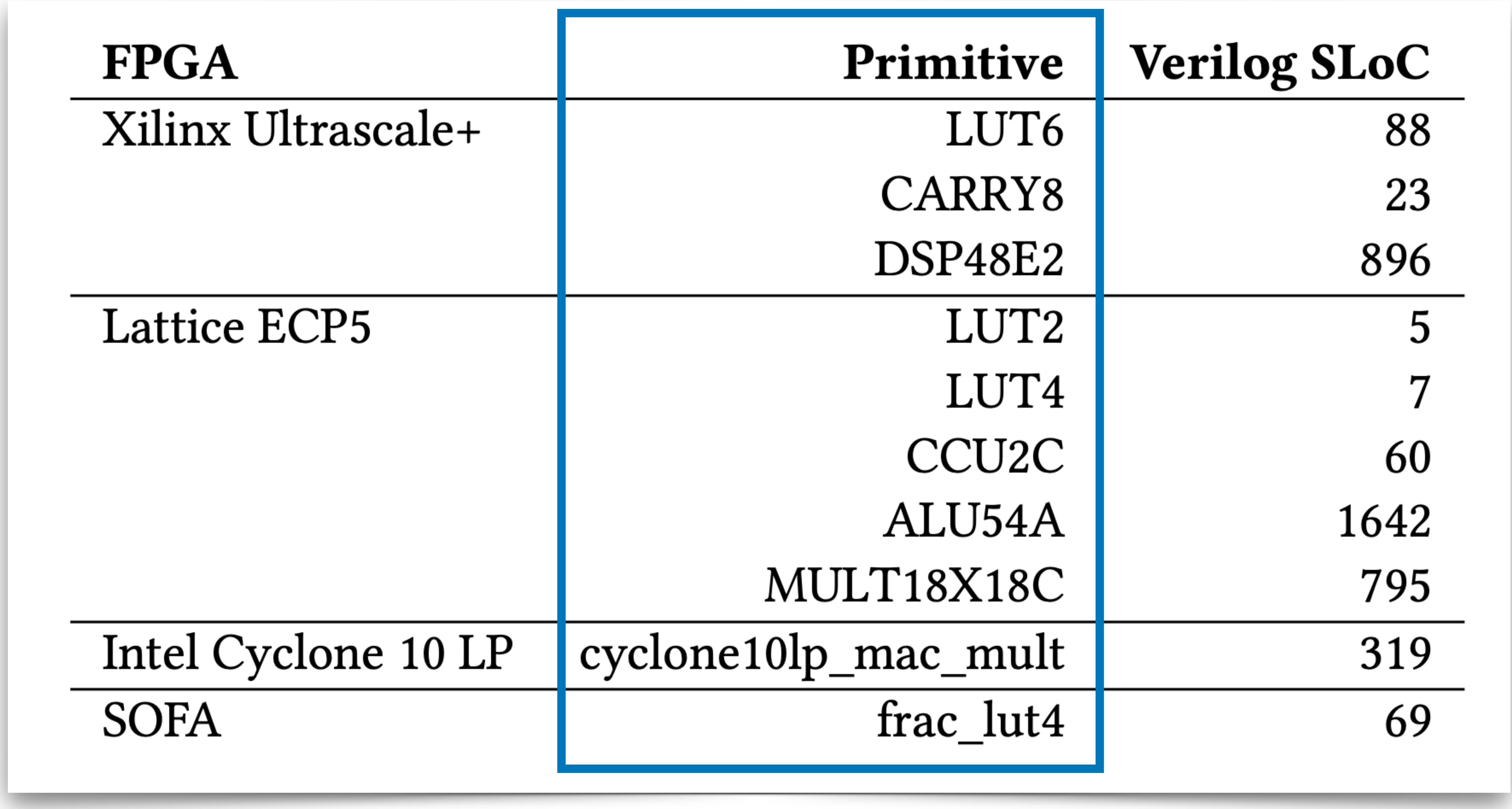
The tools we compare against provide no correctness guarantees!

Correctness: Lakeroad provides stronger correctness guarantees.

Lakeroad provides correctness guarantees due to its use of synthesis!

Extensibility: Lakeroad more easily supports new FPGAs.

Extensibility: Lakeroad more easily supports new FPGAs.



FPGA	Primitive	Verilog SLoC
Xilinx Ultrascale+	LUT6	88
	CARRY8	23
	DSP48E2	896
Lattice ECP5	LUT2	5
	LUT4	7
	CCU2C	60
	ALU54A	1642
	MULT18X18C	795
Intel Cyclone 10 LP	cyclone10lp_mac_mult	319
SOFA	frac_lut4	69

Problem

Insight

Implementation

Evaluation

Future direction:
scaling up to larger designs

Go use it!

github.com/uwsampl/lakeroad

Problem: existing tools fail to utilize programmable primitives.

Insight: Programmable primitives? Apply program synthesis!

We used this insight to build Lakeroad, which is

more complete, more correct, and more extensible.

gussmith@cs.washington.edu
justg.us

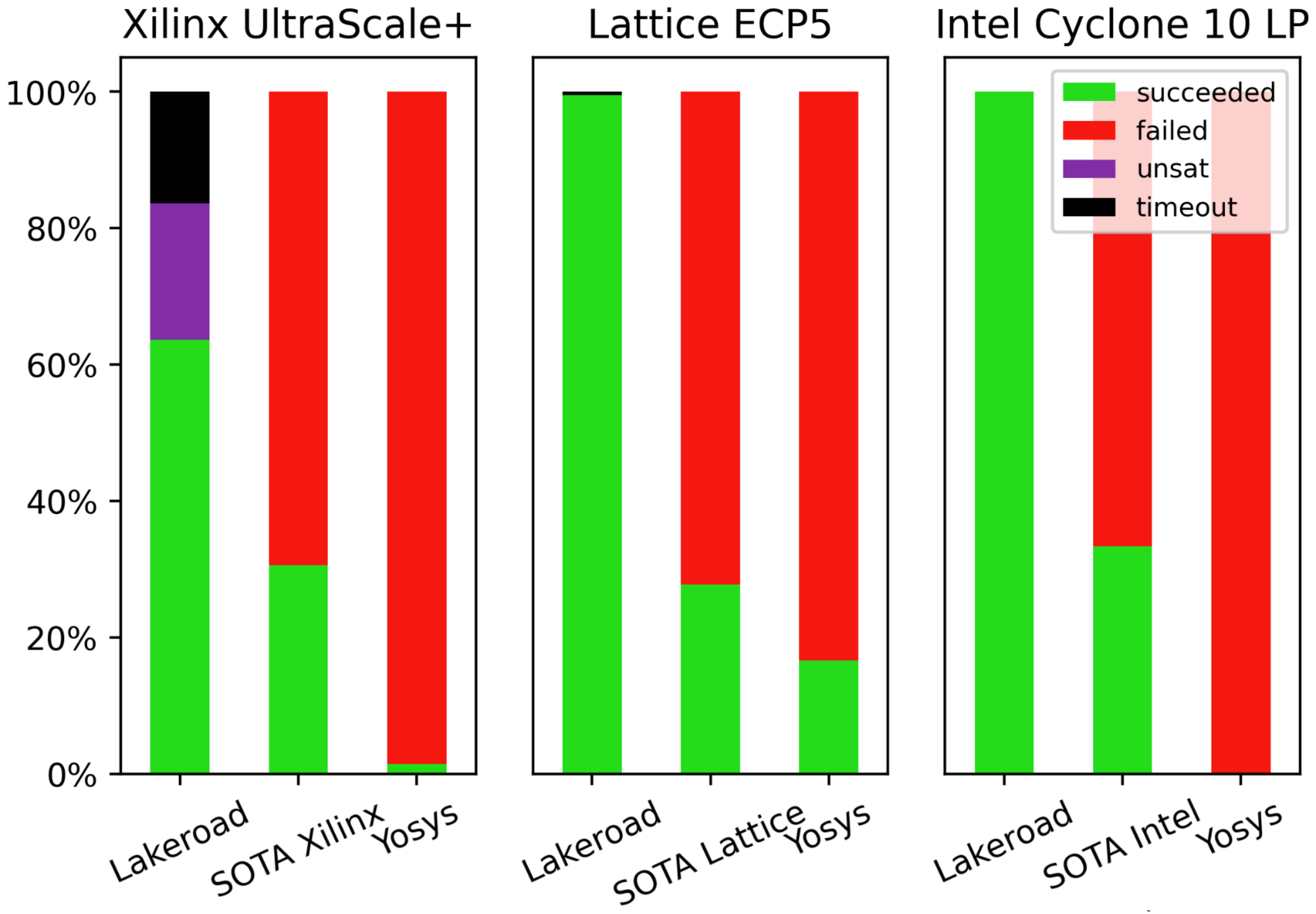
github.com/uwsampl/lakeroad

Thank you!

gussmith@cs.washington.edu
justg.us

github.com/uwsampl/lakeroad

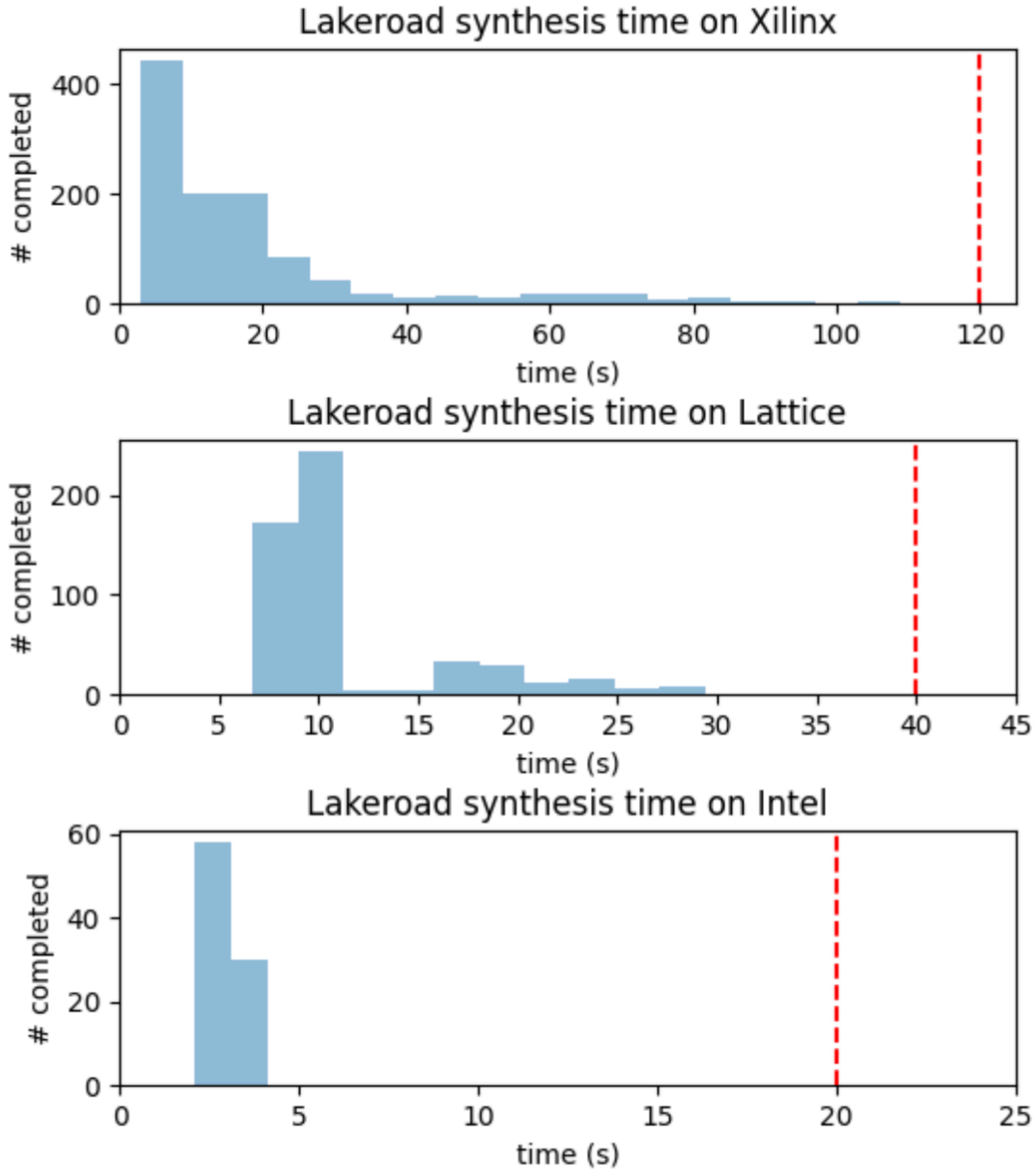
Extra slides



Xilinx UltraScale+
Lattice ECP5
Intel Cyclone 10 LP
100%
80%
60%
40%
20%
0%
Lakeroad
SOTA Xilinx
Yosys
Lakeroad
SOTA Lattice
Yosys
Lakeroad
SOTA Intel
Yosys
🟩 succeeded
🟥 failed
unsat
timeout

Lakeroad's greater completeness directly translates into resource reduction. On average, for each microbenchmark, Lakeroad uses 3.9 fewer LEs (logic elements: LUTs, muxes, or carry chains) and 7.5 fewer registers than the Xilinx SOTA, 7.2 fewer LEs/11.9 fewer registers than the Lattice SOTA, 8.2 fewer LEs/14.3 fewer registers than the Intel SOTA, and 33.3 fewer LEs/11.4 fewer registers than Yosys. In the real world, the small modules captured by our microbenchmarks may be reused dozens if not hundreds of times across a large design. Thus, the sizable resource reduction Lakeroad provides on a single microbenchmark will be multiplied significantly for an entire design.

Tool	Median Time (s)	Min / Max Time (s)	
Xilinx			
Lakeroad	14.99	2.99	127.70
SOTA Xilinx	261.61	227.82	598.67
Yosys	14.97	6.66	21.10
Lattice			
Lakeroad	9.49	6.70	55.23
SOTA Lattice	2.32	0.95	4.52
Yosys	2.31	0.90	4.01
Intel			
Lakeroad	2.92	2.12	4.13
SOTA Intel	38.73	19.11	43.49
Yosys	0.96	0.48	1.88


Lakeroad synthesis time on Xilinx
# completed
0
200
400
0
20
40
60
80
100
120
time (s)
Lakeroad synthesis time on Lattice
# completed
0
100
200
0
5
10
15
20
25
30
35
40
45
time (s)
Lakeroad synthesis time on Intel
# completed
0
20
40
60
0
5
10
15
20
25
time (s)