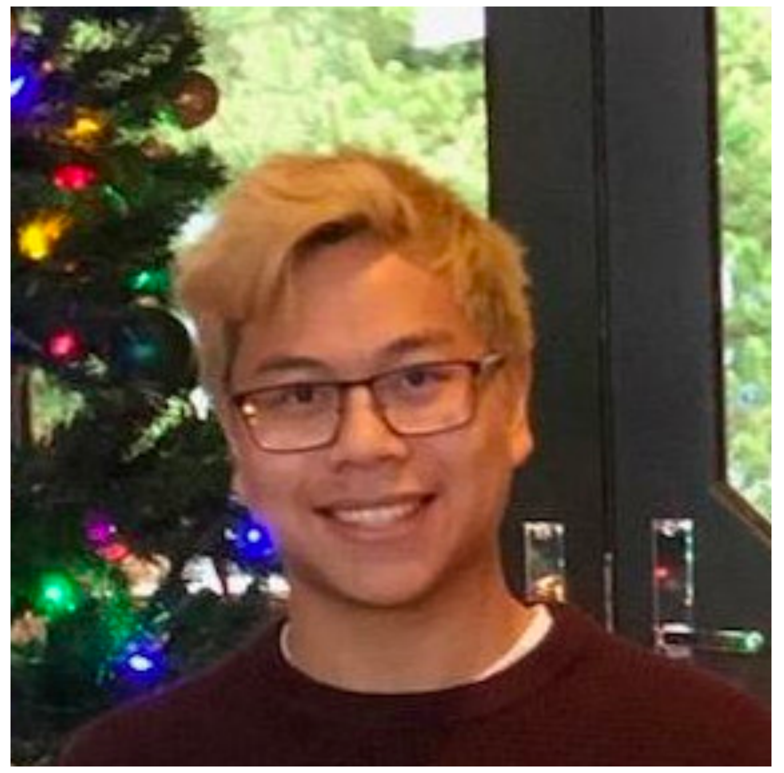
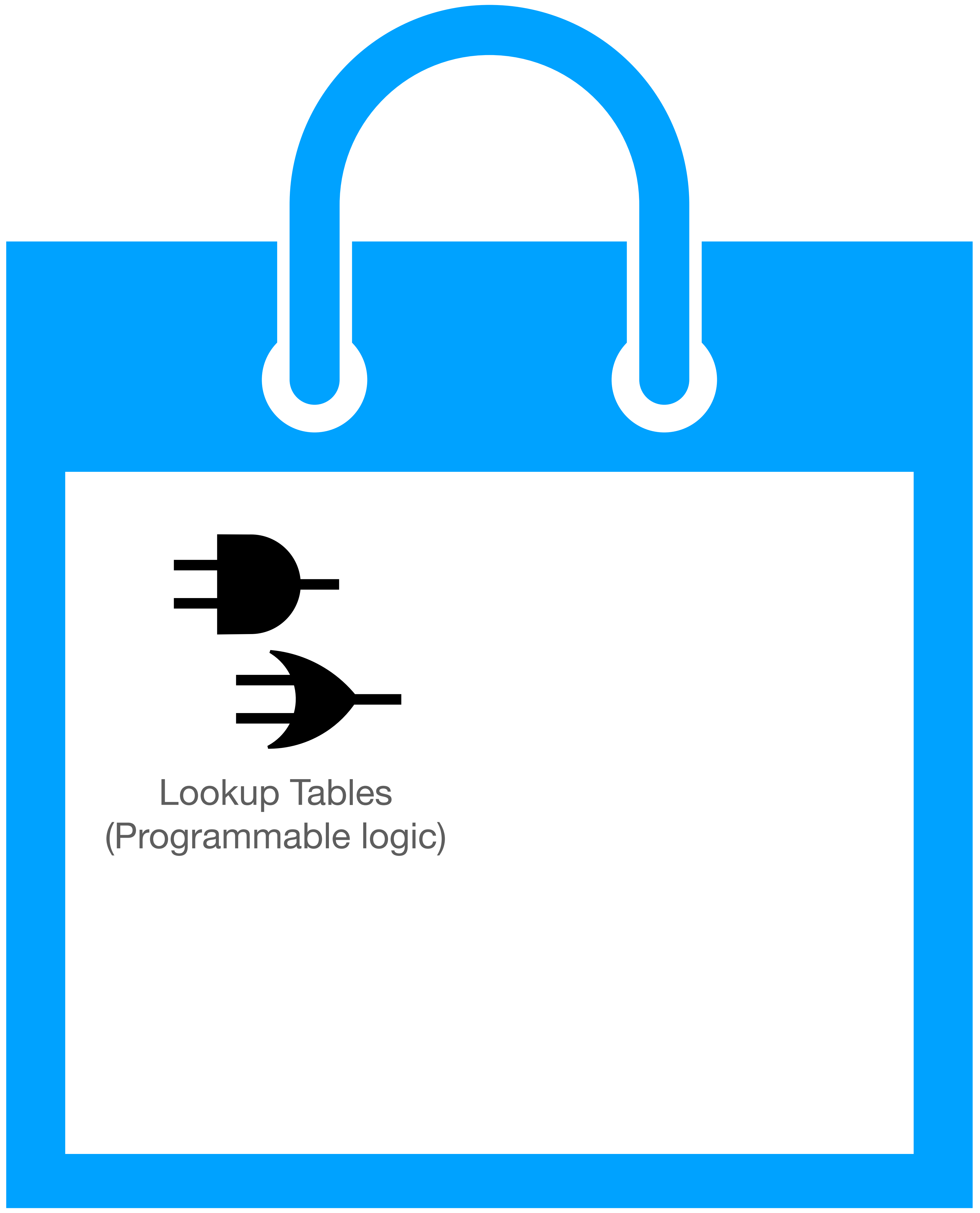
Lakeroad: Hardware Compilation via Program Synthesis

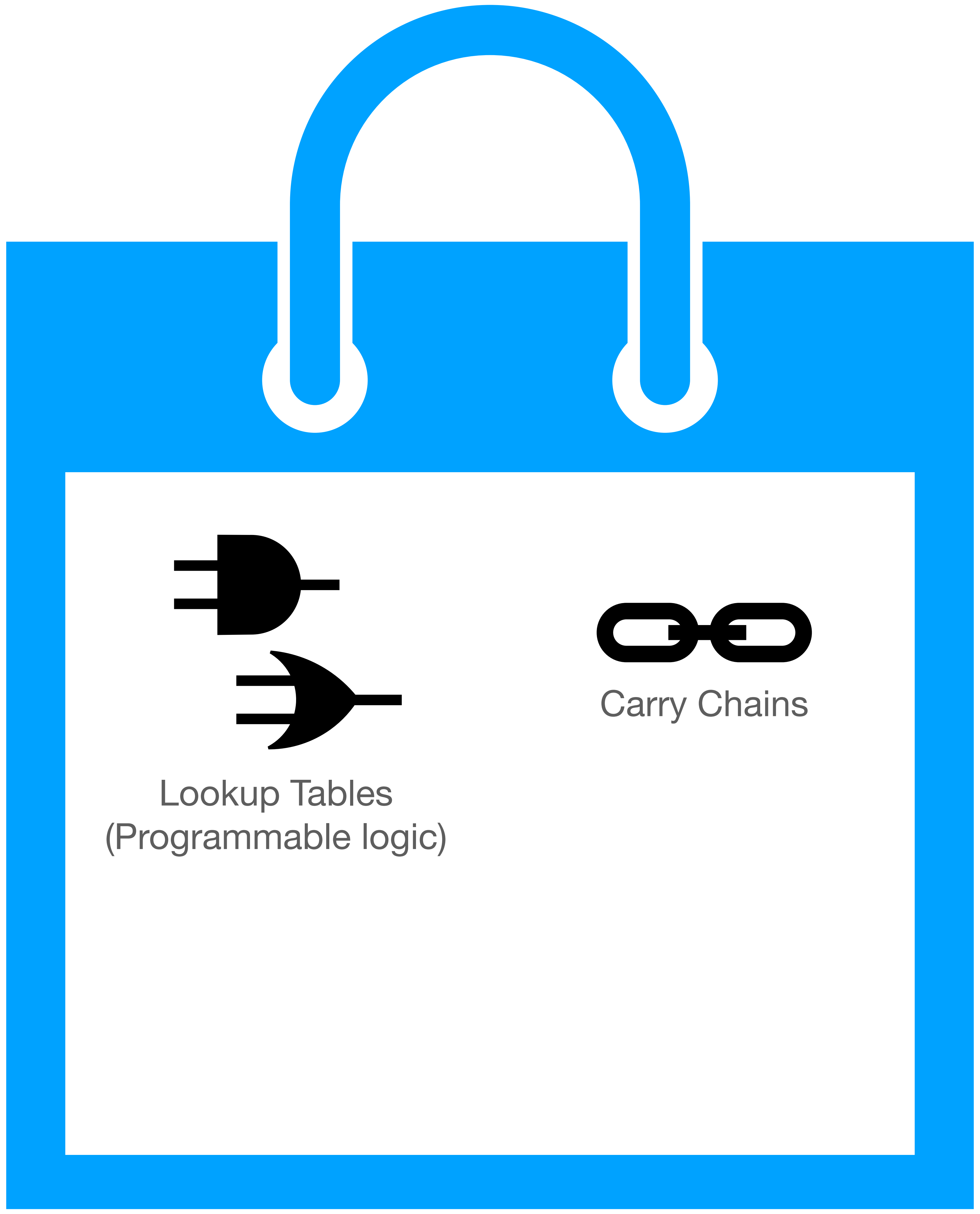
Gus Smith, PhD Candidate @ UW (https://justg.us)
PNW PLSE, May 9th 2023

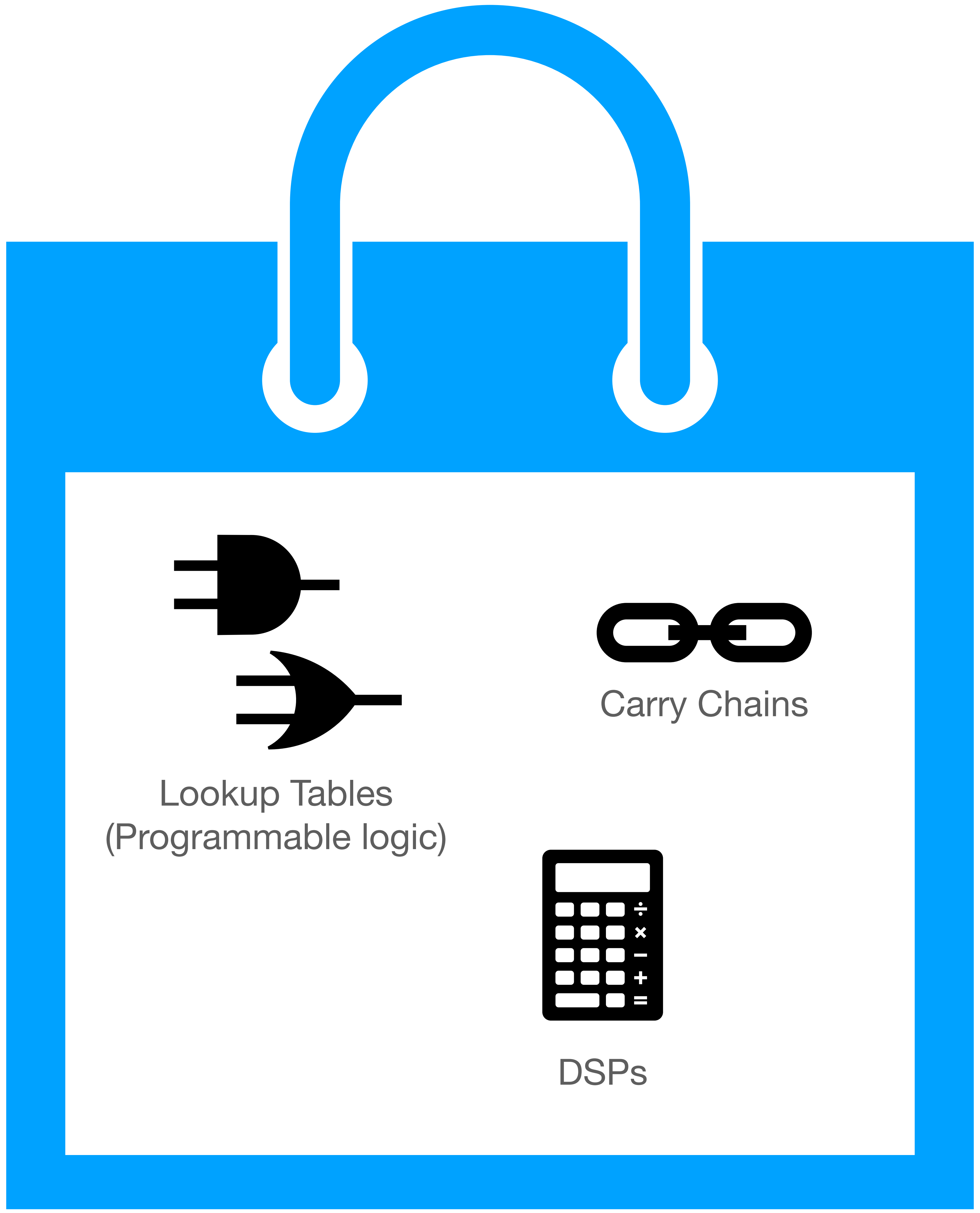


Lakeroad can compile a hardware design to an FPGA given only the Verilog models of the FPGA's primitives.

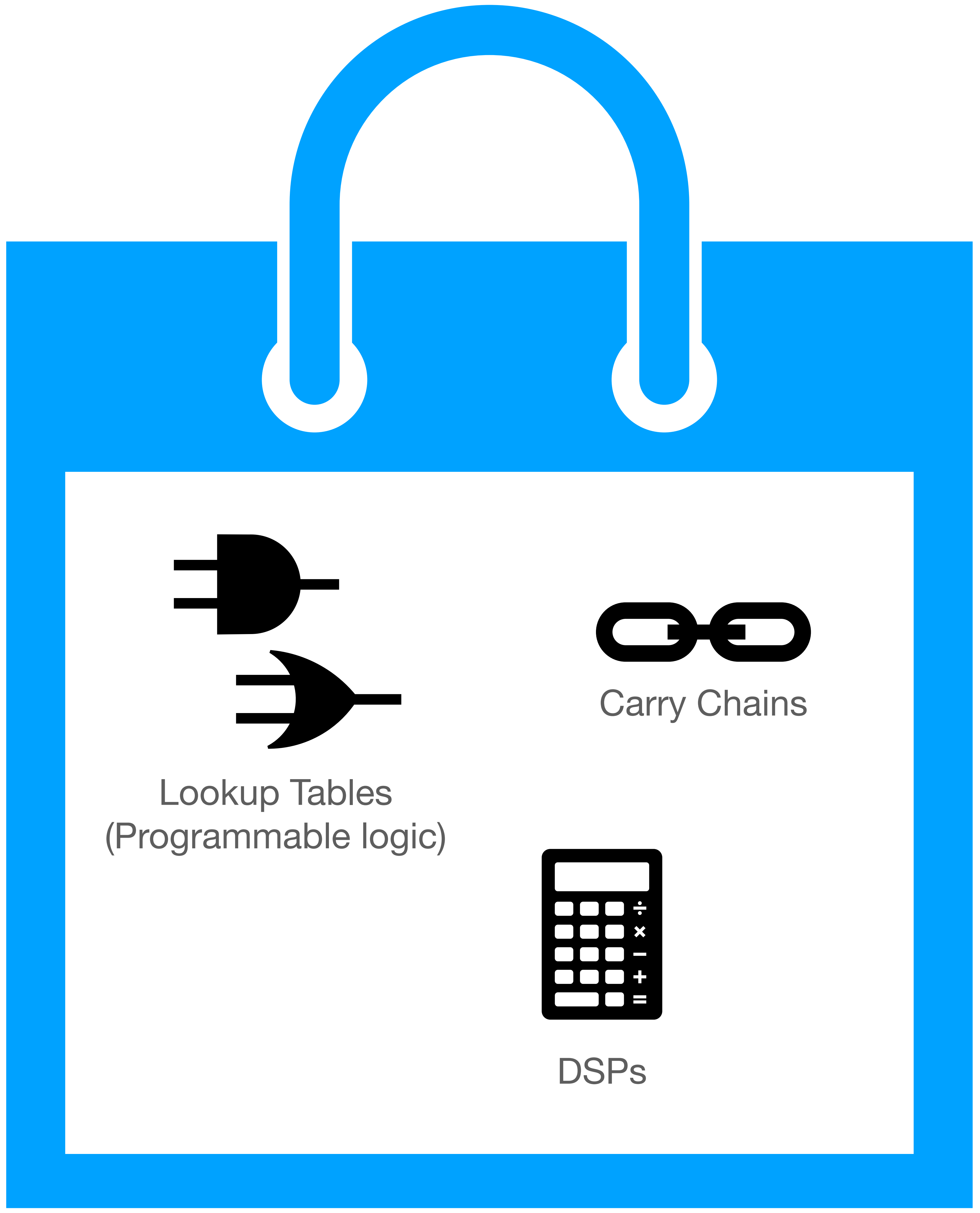
Lakeroad can compile a hardware design to an $\boxed{\text{FPGA}}$ given only the Verilog models of the FPGA’s primitives.


Lookup Tables
(Programmable logic)


Lookup Tables
(Programmable logic)
Carry Chains


Lookup Tables
(Programmable logic)
Carry Chains
DSPs

FPGA “primitives”



Lakeroad can compile a hardware design to an $\boxed{\text{FPGA}}$ given only the Verilog models of the FPGA’s primitives.

Lakeroad can compile a hardware design to an FPGA given only the Verilog models of the FPGA’s primitives.

Hardware compilation:
the process of compiling

Hardware compilation: the process of compiling

high-level (behavioral) Verilog

Hardware compilation: the process of compiling

high-level (behavioral) Verilog

```
module mul16(input [15:0] a,
             input [15:0] b,
             output [15:0] out);
  assign out = a * b;
endmodule
```

Hardware compilation: the process of compiling

high-level (behavioral) Verilog

```
module mul16(input [15:0] a,
             input [15:0] b,
             output [15:0] out);
  assign out = a * b;
endmodule
```

$\xrightarrow{\text{into}}$

Hardware compilation: the process of compiling

high-level (behavioral) Verilog

```
module mul16(input [15:0] a,
             input [15:0] b,
             output [15:0] out);
  assign out = a * b;
endmodule
```

$\xrightarrow{\text{into}}$

low-level (structural) Verilog

Hardware compilation: the process of compiling

high-level (behavioral) Verilog

```
module mul16(input [15:0] a,
             input [15:0] b,
             output [15:0] out);
  assign out = a * b;
endmodule
```

into →

low-level (structural) Verilog

```
module mul16
   (a,
    b,
    out);
  input [15:0]a;
  input [15:0]b;
  output [15:0]out;

  wire \<const0> ;
  wire \<const1> ;
  wire GND_2;
  wire VCC_2;
  wire [15:0]a;
  wire [15:0]b;
  wire [15:0]out;

  GND GND (.G(\<const0> ));
  GND GND_1 (.G(GND_2));
  VCC VCC (.P(\<const1> ));
  VCC VCC_1 (.P(VCC_2));

  (* METHODOLOGY_DRC_VIOS =
     "{SYNTH-13 {cell *THIS*}}" *)
  DSP48E2 #(
    .ACASCREG(0),
    .ALUMODEREG(0),
    ...
```

Hardware compilation: the process of compiling

high-level (behavioral) Verilog

```
module mul16(input [15:0] a,
             input [15:0] b,
             output [15:0] out);
  assign out = a * b;
endmodule
```

Doesn’t specify how multiplication is implemented

into

low-level (structural) Verilog

```
module mul16
   (a,
    b,
    out);
  input [15:0]a;
  input [15:0]b;
  output [15:0]out;

  wire \<const0> ;
  wire \<const1> ;
  wire GND_2;
  wire VCC_2;
  wire [15:0]a;
  wire [15:0]b;
  wire [15:0]out;

  GND GND (.G(\<const0> ));
  GND GND_1 (.G(GND_2));
  VCC VCC (.P(\<const1> ));
  VCC VCC_1 (.P(VCC_2));

  (* METHODOLOGY_DRC_VIOS =
     "{SYNTH-13 {cell *THIS*}}" *)
  DSP48E2 #(
    .ACASCREG(0),
    .ALUMODEREG(0),
    ...
```

Hardware compilation: the process of compiling

high-level (behavioral) Verilog

```
module mul16(input [15:0] a,
             input [15:0] b,
             output [15:0] out);
  assign out = a * b;
endmodule
```

Doesn't specify how multiplication is implemented

into →

low-level (structural) Verilog

```
module mul16
   (a,
    b,
    out);
  input [15:0]a;
  input [15:0]b;
  output [15:0]out;

  wire \<const0> ;
  wire \<const1> ;
  wire GND_2;
  wire VCC_2;
  wire [15:0]a;
  wire [15:0]b;
  wire [15:0]out;

  GND GND (.G(\<const0> ));
  GND GND_1 (.G(GND_2));
  VCC VCC (.P(\<const1> ));
  VCC VCC_1 (.P(VCC_2));

  (* METHODOLOGY_DRC_VIOS =
     "{SYNTH-13 {cell *THIS*}}" *)
  DSP48E2 #(
     .ACASCREG(0),
     .ALUMODEREG(0),
     ...
```

Hardware-specific implementation of multiplication

Lakeroad can $\boxed{\text{compile a hardware design}}$ to an FPGA given only the Verilog models of the FPGA's primitives.

Currently, in hardware compilation…

Currently, in hardware compilation…

- …automated tools only support simple primitives; otherwise, compilers use hand-written patterns

Currently, in hardware compilation…

- …automated tools only support simple primitives; otherwise, compilers use hand-written patterns
- …pattern-driven compilation is potentially buggy

Currently, in hardware compilation…

- …automated tools only support simple primitives; otherwise, compilers use hand-written patterns
- …pattern-driven compilation is potentially buggy
- …patterns fail to capture all ways to use primitives

Currently, in hardware compilation…

- …automated tools only support simple primitives; otherwise, compilers use hand-written patterns

❌ Automation

- …pattern-driven compilation is potentially buggy

- …patterns fail to capture all ways to use primitives

Currently, in hardware compilation…

- …automated tools only support simple primitives; otherwise, compilers use hand-written patterns
- …pattern-driven compilation is potentially buggy
- …patterns fail to capture all ways to use primitives

❌ Automation

❌ Correctness

Currently, in hardware compilation…

- …automated tools only support simple primitives; otherwise, compilers use hand-written patterns
- …pattern-driven compilation is potentially buggy
- …patterns fail to capture all ways to use primitives

❌ Automation

❌ Correctness

❌ Completeness

In contrast, Lakeroad:

In contrast, Lakeroad:

- Supports new architectures in <100 lines of YAML

In contrast, Lakeroad:

- Supports new architectures in <100 lines of YAML
- Provides strong correctness guarantees

In contrast, Lakeroad:

- Supports new architectures in <100 lines of YAML
- Provides strong correctness guarantees
- Effectively uses *all* primitives, even specialized units, without hand-written pattern matching

In contrast, Lakeroad:

- Supports new architectures in <100 lines of YAML
- Provides strong correctness guarantees
- Effectively uses *all* primitives, even specialized units, without hand-written pattern matching

✅ Automation

In contrast, Lakeroad:

- Supports new architectures in <100 lines of YAML
- Provides strong correctness guarantees
- Effectively uses *all* primitives, even specialized units, without hand-written pattern matching

✅ Automation

✅ Correctness

In contrast, Lakeroad:

- Supports new architectures in <100 lines of YAML
- Provides strong correctness guarantees
- Effectively uses *all* primitives, even specialized units, without hand-written pattern matching

✅ Automation

✅ Correctness

✅ Completeness

Lakeroad can compile a hardware design to an FPGA $\boxed{\text{given only the Verilog models}}$ of the FPGA’s primitives.

Using *program synthesis!*

∃Primitive, Config

$$\exists \text{Primitive}, \text{Config} \,.\, \forall I \,.$$

$$\exists \text{Primitive}, \text{Config} \,.\, \forall I \,.\, [\![\text{Primitive}(\text{Config})]\!](I)$$

$$\exists \text{Primitive}, \text{Config} \,.\, \forall I \,.\, [\![\text{Primitive}(\text{Config})]\!](I) = [\![\text{Module}]\!](I)$$

$$\exists \text{Primitive}, \text{Config} \,.\, \forall I \,.\, [\![\text{Primitive}(\text{Config})]\!](I) = [\![\text{Module}]\!](I)$$

Phrase it as a constraint problem
and query a solver (e.g. z3!)

$$\exists \text{Primitive}, \text{Config} \,.\, \forall I \,.\, \boxed{[[\text{Primitive}(\text{Config})]]}(I) = [[\text{Module}]](I)$$

$$\exists \text{Primitive}, \text{Config} \,.\, \forall I \,.\, \boxed{[\![\text{Primitive}(\text{Config})]\!]}(I) = [\![\text{Module}]\!](I)$$

Get the semantics directly from Verilog!

```
///////////////////////////////////////////////////////////////////////////////
/
//  Copyright (c) 1995/2017 Xilinx, Inc.
//  All Right Reserved.
///////////////////////////////////////////////////////////////////////////////
/
//   ____  ____
//  /   /\/   /
// /___/  \  /     Vendor      : Xilinx
// \   \   \/      Version     : 2017.3
//  \   \          Description : Xilinx Unified Simulation Library Component
//  /   /                        48-bit Multi-Functional Arithmetic Block
// /___/   /\      Filename    : DSP48E2.v
// \   \  /  \
//  ___\/___\
//
///////////////////////////////////////////////////////////////////////////////
/
//  Revision:
//  07/15/12 - Migrate from E1.
//  12/10/12 - Add dynamic registers
//  01/10/13 - 694456 - DIN_in/D_in connectivity issue
//  01/11/13 - DIN, D_DATA data width change (26/24) sync4 yml
//  02/13/13 - PCIN_47A change from internal feedback to PCIN(47) pin
//  03/06/13 - 701316 - A_B_reg no clk when REG=0
//  04/03/13 - yaml update
//  04/08/13 - 710304 - AREG, BREG, ACASCREG and BCASCREG dynamic registers
mis sized.
//  04/22/13 - 714213 - ACOUT, BCOUT wrong logic
//  04/22/13 - 713695 - Zero mult result on USE_SIMD
//  04/22/13 - 713617 - CARRYCASCOUT behaviour
//  04/23/13 - 714772 - remove sensitivity to negedge GSR
//  04/23/13 - 713706 - change P_PDBK connection
//  05/07/13 - 716896 - AREG, BREG, ACASCREG and BCASCREG localparams mis
sized.
//  05/07/13 - 716896 - ALUMODE/OPMODE_INV_REG mis sized
//  05/07/13 - 716896 - INMODE_INV_REG mis sized
//  05/07/13 - x_mac_cascd missing for sensitivity list.
//  10/22/14 - 808642 - Added #1 to $finish
//  End Revision:
///////////////////////////////////////////////////////////////////////////////
/

`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",
  parameter integer BREG = 1,
  parameter B_INPUT = "DIRECT",
  parameter integer CARRYINREG = 1,
  parameter integer CARRYINSELREG = 1,
  parameter integer CREG = 1,
  parameter integer DREG = 1,
  parameter integer INMODEREG = 1,
  parameter [3:0] IS_ALUMODE_INVERTED = 4'b0000,
  parameter [0:0] IS_CARRYIN_INVERTED = 1'b0,
  parameter [0:0] IS_CLK_INVERTED = 1'b0,
  parameter [4:0] IS_INMODE_INVERTED = 5'b00000,
  parameter [8:0] IS_OPMODE_INVERTED = 9'b000000000,
  parameter [0:0] IS_RSTALLCARRYIN_INVERTED = 1'b0,
  parameter [0:0] IS_RSTALUMODE_INVERTED = 1'b0,
  parameter [0:0] IS_RSTA_INVERTED = 1'b0,
  parameter [0:0] IS_RSTB_INVERTED = 1'b0,
  parameter [0:0] IS_RSTCTRL_INVERTED = 1'b0,
  parameter [0:0] IS_RSTC_INVERTED = 1'b0,
  parameter [0:0] IS_RSTD_INVERTED = 1'b0,
  parameter [0:0] IS_RSTINMODE_INVERTED = 1'b0,
  parameter [0:0] IS_RSTM_INVERTED = 1'b0,
  parameter [0:0] IS_RSTP_INVERTED = 1'b0,
  parameter [47:0] MASK = 48'h3FFFFFFFFFFF,
  parameter integer MREG = 1,
  parameter integer OPMODEREG = 1,
  parameter [47:0] PATTERN = 48'h000000000000,
  parameter PREADDINSEL = "A",
  parameter integer PREG = 1,
  parameter [47:0] RND = 48'h000000000000,
  parameter SEL_MASK = "MASK",
  parameter SEL_PATTERN = "PATTERN",
  parameter USE_MULT = "MULTIPLY",
  parameter USE_PATTERN_DETECT = "NO_PATDET",
  parameter USE_SIMD = "ONE48",
  parameter USE_WIDEXOR = "FALSE",
  parameter XORSIMD = "XOR24_48_96"
)(
  output [29:0] ACOUT,
  output [17:0] BCOUT,
  output CARRYCASCOUT,
  output [3:0] CARRYOUT,
  output MULTSIGNOUT,
  output OVERFLOW,
  output [47:0] P,
  output PATTERNBDETECT,
  output PATTERNDETECT,
  output [47:0] PCOUT,
  output UNDERFLOW,
  output [7:0] XOROUT,

  input [29:0] A,
  input [29:0] ACIN,
  input [3:0] ALUMODE,
  input [17:0] B,
  input [17:0] BCIN,
  input [47:0] C,
  input CARRYCASCIN,
  input CARRYIN,
  input [2:0] CARRYINSEL,
  input CEA1,
  input CEA2,
  input CEAD,
  input CEALUMODE,
  input CEB1,
  input CEB2,
  input CEC,
  input CECARRYIN,
  input CECTRL,
  input CED,
  input CEINMODE,
  input CEM,
  input CEP,
  input CLK,
  input [26:0] D,
  input [4:0] INMODE,
  input MULTSIGNIN,
  input [8:0] OPMODE,
  input [47:0] PCIN,
  input RSTA,
  input RSTALLCARRYIN,
  input RSTALUMODE,
  input RSTB,
  input RSTC,
  input RSTCTRL,
  input RSTD,
  input RSTINMODE,
  input RSTM,
  input RSTP
);

// define constants
  localparam MODULE_NAME = "DSP48E2";

// Parameter encodings and registers
  localparam AMULTSEL_A = 0;
  localparam AMULTSEL_AD = 1;
  localparam AUTORESET_PATDET_NO_RESET = 0;
  localparam AUTORESET_PATDET_RESET_MATCH = 1;
  localparam AUTORESET_PATDET_RESET_NOT_MATCH = 2;
  localparam AUTORESET_PRIORITY_CEP = 1;
  localparam AUTORESET_PRIORITY_RESET = 0;
  localparam A_INPUT_CASCADE = 1;
  localparam A_INPUT_DIRECT = 0;
  localparam BMULTSEL_AD = 1;
  localparam BMULTSEL_B = 0;
  localparam B_INPUT_CASCADE = 1;
  localparam B_INPUT_DIRECT = 0;
  localparam PREADDINSEL_A = 0;
  localparam PREADDINSEL_B = 1;
  localparam SEL_MASK_C = 1;
  localparam SEL_MASK_MASK = 0;
  localparam SEL_MASK_ROUNDING_MODE1 = 2;
  localparam SEL_MASK_ROUNDING_MODE2 = 3;
  localparam SEL_PATTERN_C = 1;
  localparam SEL_PATTERN_PATTERN = 0;
  localparam USE_MULT_DYNAMIC = 1;
  localparam USE_MULT_MULTIPLY = 0;
  localparam USE_MULT_NONE = 2;
  localparam USE_PATTERN_DETECT_NO_PATDET = 0;
  localparam USE_PATTERN_DETECT_PATDET = 1;
  localparam USE_SIMD_FOUR12 = 1;
  localparam USE_SIMD_ONE48 = 0;
  localparam USE_SIMD_TWO24 = 2;
  localparam USE_WIDEXOR_FALSE = 0;
  localparam USE_WIDEXOR_TRUE = 1;
  localparam XORSIMD_XOR12 = 1;
  localparam XORSIMD_XOR24_48_96 = 0;

  reg trig_attr = 1'b0;
// include dynamic registers - XILINX test only
`ifdef XIL_DR
  `include "DSP48E2_dr.v"
`else
  localparam [31:0] ACASCREG_REG = ACASCREG;
  localparam [31:0] ADREG_REG = ADREG;
  localparam [31:0] ALUMODEREG_REG = ALUMODEREG;
  localparam [16:1] AMULTSEL_REG = AMULTSEL;
  localparam [31:0] AREG_REG = AREG;
  localparam [120:1] AUTORESET_PATDET_REG = AUTORESET_PATDET;
  localparam [40:1] AUTORESET_PRIORITY_REG = AUTORESET_PRIORITY;
  localparam [56:1] A_INPUT_REG = A_INPUT;
  localparam [31:0] BCASCREG_REG = BCASCREG;
  localparam [16:1] BMULTSEL_REG = BMULTSEL;
  localparam [31:0] BREG_REG = BREG;
  localparam [56:1] B_INPUT_REG = B_INPUT;
  localparam [31:0] CARRYINREG_REG = CARRYINREG;
  localparam [31:0] CARRYINSELREG_REG = CARRYINSELREG;
  localparam [31:0] CREG_REG = CREG;
  localparam [31:0] DREG_REG = DREG;
  localparam [31:0] INMODEREG_REG = INMODEREG;
  localparam [3:0] IS_ALUMODE_INVERTED_REG = IS_ALUMODE_INVERTED;
  localparam [0:0] IS_CARRYIN_INVERTED_REG = IS_CARRYIN_INVERTED;
  localparam [0:0] IS_CLK_INVERTED_REG = IS_CLK_INVERTED;
  localparam [4:0] IS_INMODE_INVERTED_REG = IS_INMODE_INVERTED;
  localparam [8:0] IS_OPMODE_INVERTED_REG = IS_OPMODE_INVERTED;
  localparam [0:0] IS_RSTALLCARRYIN_INVERTED_REG = IS_RSTALLCARRYIN_INVERTED;
  localparam [0:0] IS_RSTALUMODE_INVERTED_REG = IS_RSTALUMODE_INVERTED;
  localparam [0:0] IS_RSTA_INVERTED_REG = IS_RSTA_INVERTED;
  localparam [0:0] IS_RSTB_INVERTED_REG = IS_RSTB_INVERTED;
  localparam [0:0] IS_RSTCTRL_INVERTED_REG = IS_RSTCTRL_INVERTED;
  localparam [0:0] IS_RSTC_INVERTED_REG = IS_RSTC_INVERTED;
  localparam [0:0] IS_RSTD_INVERTED_REG = IS_RSTD_INVERTED;
  localparam [0:0] IS_RSTINMODE_INVERTED_REG = IS_RSTINMODE_INVERTED;
  localparam [0:0] IS_RSTM_INVERTED_REG = IS_RSTM_INVERTED;
  localparam [0:0] IS_RSTP_INVERTED_REG = IS_RSTP_INVERTED;
  localparam [47:0] MASK_REG = MASK;
  localparam [31:0] MREG_REG = MREG;
  localparam [31:0] OPMODEREG_REG = OPMODEREG;
  localparam [47:0] PATTERN_REG = PATTERN;
  localparam [8:1] PREADDINSEL_REG = PREADDINSEL;
  localparam [31:0] PREG_REG = PREG;
  localparam [47:0] RND_REG = RND;
  localparam [112:1] SEL_MASK_REG = SEL_MASK;
  localparam [56:1] SEL_PATTERN_REG = SEL_PATTERN;
  localparam [64:1] USE_MULT_REG = USE_MULT;
  localparam [72:1] USE_PATTERN_DETECT_REG = USE_PATTERN_DETECT;
  localparam [48:1] USE_SIMD_REG = USE_SIMD;
  localparam [40:1] USE_WIDEXOR_REG = USE_WIDEXOR;
  localparam [88:1] XORSIMD_REG = XORSIMD;
`endif

`ifdef XIL_XECLIB
  wire [1:0] ACASCREG_BIN;
  wire ADREG_BIN;
  wire ALUMODEREG_BIN;
  wire AMULTSEL_BIN;
  wire [1:0] AREG_BIN;
  wire [1:0] AUTORESET_PATDET_BIN;
  wire AUTORESET_PRIORITY_BIN;
  wire A_INPUT_BIN;
  wire [1:0] BCASCREG_BIN;
  wire BMULTSEL_BIN;
  wire [1:0] BREG_BIN;
  wire B_INPUT_BIN;
  wire CARRYINREG_BIN;
  wire CARRYINSELREG_BIN;
  wire CREG_BIN;
  wire DREG_BIN;
  wire INMODEREG_BIN;
  wire MREG_BIN;
  wire OPMODEREG_BIN;
  wire PREADDINSEL_BIN;
  wire PREG_BIN;
  wire [1:0] SEL_MASK_BIN;
  wire SEL_PATTERN_BIN;
  wire [1:0] USE_MULT_BIN;
  wire USE_PATTERN_DETECT_BIN;
  wire [1:0] USE_SIMD_BIN;
  wire USE_WIDEXOR_BIN;
  wire XORSIMD_BIN;
`else
  reg [1:0] ACASCREG_BIN;
  reg ADREG_BIN;
  reg ALUMODEREG_BIN;
  reg AMULTSEL_BIN;
  reg [1:0] AREG_BIN;
  reg [1:0] AUTORESET_PATDET_BIN;
  reg AUTORESET_PRIORITY_BIN;
  reg A_INPUT_BIN;
  reg [1:0] BCASCREG_BIN;
  reg BMULTSEL_BIN;
  reg [1:0] BREG_BIN;
  reg B_INPUT_BIN;
  reg CARRYINREG_BIN;
  reg CARRYINSELREG_BIN;
  reg CREG_BIN;
  reg DREG_BIN;
  reg INMODEREG_BIN;
  reg MREG_BIN;
  reg OPMODEREG_BIN;
  reg PREADDINSEL_BIN;
  reg PREG_BIN;
  reg [1:0] SEL_MASK_BIN;
  reg SEL_PATTERN_BIN;
  reg [1:0] USE_MULT_BIN;
  reg USE_PATTERN_DETECT_BIN;
  reg [1:0] USE_SIMD_BIN;
  reg USE_WIDEXOR_BIN;
  reg XORSIMD_BIN;
`endif

`ifdef XIL_ATTR_TEST
  reg attr_test = 1'b1;
`else
  reg attr_test = 1'b0;
`endif

  reg attr_err = 1'b0;
  reg glblGSR = 1'b0;
//  tri0 glblGSR = glbl.GSR;

  wire CARRYCASCIN_in;
  wire CARRYIN_in;
  wire CEA1_in;
  wire CEA2_in;
  wire CEAD_in;
  wire CEALUMODE_in;
  wire CEB1_in;
  wire CEB2_in;
  wire CECARRYIN_in;
  wire CECTRL_in;
  wire CEC_in;
  wire CED_in;
  wire CEINMODE_in;
  wire CEM_in;
  wire CEP_in;
  wire CLK_in;
  wire MULTSIGNIN_in;
  wire RSTALLCARRYIN_in;
  wire RSTALUMODE_in;
  wire RSTA_in;
  wire RSTB_in;
  wire RSTCTRL_in;
  wire RSTC_in;
  wire RSTD_in;
  wire RSTINMODE_in;
  wire RSTM_in;
  wire RSTP_in;
  wire [17:0] BCIN_in;
  wire [17:0] B_in;
  wire [26:0] D_in;
  wire [29:0] ACIN_in;
  wire [29:0] A_in;
  wire [2:0] CARRYINSEL_in;
  wire [3:0] ALUMODE_in;
  wire [47:0] C_in;
  wire [47:0] PCIN_in;
  wire [4:0] INMODE_in;
  wire [8:0] OPMODE_in;

  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 A_in[14] = (A[14] === 1'bx) || A[14]; // rv 1
  assign A_in[15] = (A[15] === 1'bx) || A[15]; // rv 1
  assign A_in[16] = (A[16] === 1'bx) || A[16]; // rv 1
  assign A_in[17] = (A[17] === 1'bx) || A[17]; // rv 1
  assign A_in[18] = (A[18] === 1'bx) || A[18]; // rv 1
  assign A_in[19] = (A[19] === 1'bx) || A[19]; // rv 1
  assign A_in[1] = (A[1] === 1'bx) || A[1]; // rv 1
  assign A_in[20] = (A[20] === 1'bx) || A[20]; // rv 1
  assign A_in[21] = (A[21] === 1'bx) || A[21]; // rv 1
  assign A_in[22] = (A[22] === 1'bx) || A[22]; // rv 1
  assign A_in[23] = (A[23] === 1'bx) || A[23]; // rv 1
  assign A_in[24] = (A[24] === 1'bx) || A[24]; // rv 1
  assign A_in[25] = (A[25] === 1'bx) || A[25]; // rv 1
  assign A_in[26] = (A[26] === 1'bx) || A[26]; // rv 1
  assign A_in[27] = (A[27] === 1'bx) || A[27]; // rv 1
  assign A_in[28] = (A[28] === 1'bx) || A[28]; // rv 1
  assign A_in[29] = (A[29] === 1'bx) || A[29]; // rv 1
  assign A_in[2] = (A[2] === 1'bx) || A[2]; // rv 1
  assign A_in[3] = (A[3] === 1'bx) || A[3]; // rv 1
  assign A_in[4] = (A[4] === 1'bx) || A[4]; // rv 1
  assign A_in[5] = (A[5] === 1'bx) || A[5]; // rv 1
  assign A_in[6] = (A[6] === 1'bx) || A[6]; // rv 1
  assign A_in[7] = (A[7] === 1'bx) || A[7]; // rv 1
  assign A_in[8] = (A[8] === 1'bx) || A[8]; // rv 1
  assign A_in[9] = (A[9] === 1'bx) || A[9]; // rv 1
  assign BCIN_in = BCIN;
  assign B_in[0] = (B[0] === 1'bx) || B[0]; // rv 1
  assign B_in[10] = (B[10] === 1'bx) || B[10]; // rv 1
  assign B_in[11] = (B[11] === 1'bx) || B[11]; // rv 1
  assign B_in[12] = (B[12] === 1'bx) || B[12]; // rv 1
  assign B_in[13] = (B[13] === 1'bx) || B[13]; // rv 1
  assign B_in[14] = (B[14] === 1'bx) || B[14]; // rv 1
  assign B_in[15] = (B[15] === 1'bx) || B[15]; // rv 1
  assign B_in[16] = (B[16] === 1'bx) || B[16]; // rv 1
  assign B_in[17] = (B[17] === 1'bx) || B[17]; // rv 1
  assign B_in[1] = (B[1] === 1'bx) || B[1]; // rv 1
  assign B_in[2] = (B[2] === 1'bx) || B[2]; // 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
  assign C_in[11] = (C[11] === 1'bx) || C[11]; // rv 1
  assign C_in[12] = (C[12] === 1'bx) || C[12]; // rv 1
  assign C_in[13] = (C[13] === 1'bx) || C[13]; // rv 1
  assign C_in[14] = (C[14] === 1'bx) || C[14]; // rv 1
  assign C_in[15] = (C[15] === 1'bx) || C[15]; // rv 1
  assign C_in[16] = (C[16] === 1'bx) || C[16]; // rv 1
  assign C_in[17] = (C[17] === 1'bx) || C[17]; // rv 1
  assign C_in[18] = (C[18] === 1'bx) || C[18]; // rv 1
  assign C_in[19] = (C[19] === 1'bx) || C[19]; // rv 1
  assign C_in[1] = (C[1] === 1'bx) || C[1]; // rv 1
  assign C_in[20] = (C[20] === 1'bx) || C[20]; // rv 1
  assign C_in[21] = (C[21] === 1'bx) || C[21]; // rv 1
  assign C_in[22] = (C[22] === 1'bx) || C[22]; // rv 1
  assign C_in[23] = (C[23] === 1'bx) || C[23]; // rv 1
  assign C_in[24] = (C[24] === 1'bx) || C[24]; // rv 1
  assign C_in[25] = (C[25] === 1'bx) || C[25]; // rv 1
  assign C_in[26] = (C[26] === 1'bx) || C[26]; // rv 1
  assign C_in[27] = (C[27] === 1'bx) || C[27]; // rv 1
  assign C_in[28] = (C[28] === 1'bx) || C[28]; // rv 1
  assign C_in[29] = (C[29] === 1'bx) || C[29]; // rv 1
  assign C_in[2] = (C[2] === 1'bx) || C[2]; // rv 1
  assign C_in[30] = (C[30] === 1'bx) || C[30]; // rv 1
  assign C_in[31] = (C[31] === 1'bx) || C[31]; // rv 1
  assign C_in[32] = (C[32] === 1'bx) || C[32]; // rv 1
  assign C_in[33] = (C[33] === 1'bx) || C[33]; // rv 1
  assign C_in[34] = (C[34] === 1'bx) || C[34]; // rv 1
  assign C_in[35] = (C[35] === 1'bx) || C[35]; // rv 1
  assign C_in[36] = (C[36] === 1'bx) || C[36]; // rv 1
  assign C_in[37] = (C[37] === 1'bx) || C[37]; // rv 1
  assign C_in[38] = (C[38] === 1'bx) || C[38]; // rv 1
  assign C_in[39] = (C[39] === 1'bx) || C[39]; // rv 1
  assign C_in[3] = (C[3] === 1'bx) || C[3]; // rv 1
  assign C_in[40] = (C[40] === 1'bx) || C[40]; // rv 1
  assign C_in[41] = (C[41] === 1'bx) || C[41]; // rv 1
  assign C_in[42] = (C[42] === 1'bx) || C[42]; // rv 1
  assign C_in[43] = (C[43] === 1'bx) || C[43]; // rv 1
  assign C_in[44] = (C[44] === 1'bx) || C[44]; // rv 1
  assign C_in[45] = (C[45] === 1'bx) || C[45]; // rv 1
  assign C_in[46] = (C[46] === 1'bx) || C[46]; // rv 1
  assign C_in[47] = (C[47] === 1'bx) || C[47]; // rv 1
  assign C_in[4] = (C[4] === 1'bx) || C[4]; // rv 1
  assign C_in[5] = (C[5] === 1'bx) || C[5]; // rv 1
  assign C_in[6] = (C[6] === 1'bx) || C[6]; // rv 1
  assign C_in[7] = (C[7] === 1'bx) || C[7]; // rv 1
  assign C_in[8] = (C[8] === 1'bx) || C[8]; // rv 1
  assign C_in[9] = (C[9] === 1'bx) || C[9]; // rv 1
  assign D_in[0] = (D[0] !== 1'bx) && D[0]; // rv 0
  assign D_in[10] = (D[10] !== 1'bx) && D[10]; // rv 0
  assign D_in[11] = (D[11] !== 1'bx) && D[11]; // rv 0
  assign D_in[12] = (D[12] !== 1'bx) && D[12]; // rv 0
  assign D_in[13] = (D[13] !== 1'bx) && D[13]; // rv 0
  assign D_in[14] = (D[14] !== 1'bx) && D[14]; // rv 0
  assign D_in[15] = (D[15] !== 1'bx) && D[15]; // rv 0
  assign D_in[16] = (D[16] !== 1'bx) && D[16]; // rv 0
  assign D_in[17] = (D[17] !== 1'bx) && D[17]; // rv 0
  assign D_in[18] = (D[18] !== 1'bx) && D[18]; // rv 0
  assign D_in[19] = (D[19] !== 1'bx) && D[19]; // rv 0
  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
  assign OPMODE_in[4] = (OPMODE[4] !== 1'bx) && (OPMODE[4] ^ IS_OPMODE_INVERTED_REG[4]); // rv 0
  assign OPMODE_in[5] = (OPMODE[5] !== 1'bx) && (OPMODE[5] ^ IS_OPMODE_INVERTED_REG[5]); // rv 0
  assign 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) && (RSTALLCARRYIN ^ IS_RSTALLCARRYIN_INVERTED_REG); //
rv 0
  assign RSTALUMODE_in = (RSTALUMODE !== 1'bx) && (RSTALUMODE ^ IS_RSTALUMODE_INVERTED_REG); // rv 0
  assign RSTA_in = (RSTA !== 1'bx) && (RSTA ^ IS_RSTA_INVERTED_REG); // rv 0
  assign RSTB_in = (RSTB !== 1'bx) && (RSTB ^ IS_RSTB_INVERTED_REG); // rv 0
  assign RSTCTRL_in = (RSTCTRL !== 1'bx) && (RSTCTRL ^ IS_RSTCTRL_INVERTED_REG); // rv 0
  assign RSTC_in = (RSTC !== 1'bx) && (RSTC ^ IS_RSTC_INVERTED_REG); // rv 0
  assign RSTD_in = (RSTD !== 1'bx) && (RSTD ^ IS_RSTD_INVERTED_REG); // rv 0
  assign RSTINMODE_in = (RSTINMODE !== 1'bx) && (RSTINMODE ^ IS_RSTINMODE_INVERTED_REG); // rv 0
  assign RSTM_in = (RSTM !== 1'bx) && (RSTM ^ IS_RSTM_INVERTED_REG); // rv 0
  assign RSTP_in = (RSTP !== 1'bx) && (RSTP ^ IS_RSTP_INVERTED_REG); // rv 0

`ifndef XIL_XECLIB
initial begin
  #1;
  trig_attr = ~trig_attr;
end
`endif

`ifdef XIL_XECLIB
  assign ACASCREG_BIN = ACASCREG_REG[1:0];

  assign ADREG_BIN = ADREG_REG[0];

  assign ALUMODEREG_BIN = ALUMODEREG_REG[0];

  assign AMULTSEL_BIN =
    (AMULTSEL_REG == "A") ? AMULTSEL_A :
    (AMULTSEL_REG == "AD") ? AMULTSEL_AD :
     AMULTSEL_A;

  assign AREG_BIN = AREG_REG[1:0];

  assign AUTORESET_PATDET_BIN =
    (AUTORESET_PATDET_REG == "NO_RESET") ? AUTORESET_PATDET_NO_RESET :
    (AUTORESET_PATDET_REG == "RESET_MATCH") ? AUTORESET_PATDET_RESET_MATCH :
    (AUTORESET_PATDET_REG == "RESET_NOT_MATCH") ? AUTORESET_PATDET_RESET_NOT_MATCH :
     AUTORESET_PATDET_NO_RESET;

  assign AUTORESET_PRIORITY_BIN =
    (AUTORESET_PRIORITY_REG == "RESET") ? AUTORESET_PRIORITY_RESET :
    (AUTORESET_PRIORITY_REG == "CEP") ? AUTORESET_PRIORITY_CEP :
     AUTORESET_PRIORITY_RESET;

  assign A_INPUT_BIN =
    (A_INPUT_REG == "DIRECT") ? A_INPUT_DIRECT :
    (A_INPUT_REG == "CASCADE") ? A_INPUT_CASCADE :
     A_INPUT_DIRECT;

  assign BCASCREG_BIN = BCASCREG_REG[1:0];

  assign BMULTSEL_BIN =
    (BMULTSEL_REG == "B") ? BMULTSEL_B :
    (BMULTSEL_REG == "AD") ? BMULTSEL_AD :
     BMULTSEL_B;

  assign BREG_BIN = BREG_REG[1:0];

  assign B_INPUT_BIN =
    (B_INPUT_REG == "DIRECT") ? B_INPUT_DIRECT :
    (B_INPUT_REG == "CASCADE") ? B_INPUT_CASCADE :
     B_INPUT_DIRECT;

  assign CARRYINREG_BIN = CARRYINREG_REG[0];

  assign CARRYINSELREG_BIN = CARRYINSELREG_REG[0];

  assign CREG_BIN = CREG_REG[0];

  assign DREG_BIN = DREG_REG[0];

  assign INMODEREG_BIN = INMODEREG_REG[0];

  assign MREG_BIN = MREG_REG[0];

  assign OPMODEREG_BIN = OPMODEREG_REG[0];

  assign PREADDINSEL_BIN =
    (PREADDINSEL_REG == "A") ? PREADDINSEL_A :
    (PREADDINSEL_REG == "B") ? PREADDINSEL_B :
     PREADDINSEL_A;

  assign PREG_BIN = PREG_REG[0];

  assign SEL_MASK_BIN =
    (SEL_MASK_REG == "MASK") ? SEL_MASK_MASK :
    (SEL_MASK_REG == "C") ? SEL_MASK_C :
    (SEL_MASK_REG == "ROUNDING_MODE1") ? SEL_MASK_ROUNDING_MODE1 :
    (SEL_MASK_REG == "ROUNDING_MODE2") ? SEL_MASK_ROUNDING_MODE2 :
     SEL_MASK_MASK;

  assign SEL_PATTERN_BIN =
    (SEL_PATTERN_REG == "PATTERN") ? SEL_PATTERN_PATTERN :
    (SEL_PATTERN_REG == "C") ? SEL_PATTERN_C :
     SEL_PATTERN_PATTERN;

  assign USE_MULT_BIN =
    (USE_MULT_REG == "MULTIPLY") ? USE_MULT_MULTIPLY :
    (USE_MULT_REG == "DYNAMIC") ? USE_MULT_DYNAMIC :
    (USE_MULT_REG == "NONE") ? USE_MULT_NONE :
     USE_MULT_MULTIPLY;

  assign USE_PATTERN_DETECT_BIN =
    (USE_PATTERN_DETECT_REG == "NO_PATDET") ? USE_PATTERN_DETECT_NO_PATDET :
    (USE_PATTERN_DETECT_REG == "PATDET") ? USE_PATTERN_DETECT_PATDET :
     USE_PATTERN_DETECT_NO_PATDET;

  assign USE_SIMD_BIN =
    (USE_SIMD_REG == "ONE48") ? USE_SIMD_ONE48 :
    (USE_SIMD_REG == "FOUR12") ? USE_SIMD_FOUR12 :
    (USE_SIMD_REG == "TWO24") ? USE_SIMD_TWO24 :
     USE_SIMD_ONE48;

  assign USE_WIDEXOR_BIN =
    (USE_WIDEXOR_REG == "FALSE") ? USE_WIDEXOR_FALSE :
    (USE_WIDEXOR_REG == "TRUE") ? USE_WIDEXOR_TRUE :
     USE_WIDEXOR_FALSE;

  assign XORSIMD_BIN =
    (XORSIMD_REG == "XOR24_48_96") ? XORSIMD_XOR24_48_96 :
    (XORSIMD_REG == "XOR12") ? XORSIMD_XOR12 :
     XORSIMD_XOR24_48_96;

`else
always @(trig_attr) begin
#1;
  ACASCREG_BIN = ACASCREG_REG[1:0];

  ADREG_BIN = ADREG_REG[0];

  ALUMODEREG_BIN = ALUMODEREG_REG[0];

  AMULTSEL_BIN =
    (AMULTSEL_REG == "A") ? AMULTSEL_A :
    (AMULTSEL_REG == "AD") ? AMULTSEL_AD :
     AMULTSEL_A;

  AREG_BIN = AREG_REG[1:0];

  AUTORESET_PATDET_BIN =
    (AUTORESET_PATDET_REG == "NO_RESET") ? AUTORESET_PATDET_NO_RESET :
    (AUTORESET_PATDET_REG == "RESET_MATCH") ? AUTORESET_PATDET_RESET_MATCH :
    (AUTORESET_PATDET_REG == "RESET_NOT_MATCH") ? AUTORESET_PATDET_RESET_NOT_MATCH :
     AUTORESET_PATDET_NO_RESET;

  AUTORESET_PRIORITY_BIN =
    (AUTORESET_PRIORITY_REG == "RESET") ? AUTORESET_PRIORITY_RESET :
    (AUTORESET_PRIORITY_REG == "CEP") ? AUTORESET_PRIORITY_CEP :
     AUTORESET_PRIORITY_RESET;

  A_INPUT_BIN =
    (A_INPUT_REG == "DIRECT") ? A_INPUT_DIRECT :
    (A_INPUT_REG == "CASCADE") ? A_INPUT_CASCADE :
     A_INPUT_DIRECT;

  BCASCREG_BIN = BCASCREG_REG[1:0];

  BMULTSEL_BIN =
    (BMULTSEL_REG == "B") ? BMULTSEL_B :
    (BMULTSEL_REG == "AD") ? BMULTSEL_AD :
     BMULTSEL_B;

  BREG_BIN = BREG_REG[1:0];

  B_INPUT_BIN =
    (B_INPUT_REG == "DIRECT") ? B_INPUT_DIRECT :
    (B_INPUT_REG == "CASCADE") ? B_INPUT_CASCADE :
     B_INPUT_DIRECT;

  CARRYINREG_BIN = CARRYINREG_REG[0];

  CARRYINSELREG_BIN = CARRYINSELREG_REG[0];

  CREG_BIN = CREG_REG[0];

  DREG_BIN = DREG_REG[0];

  INMODEREG_BIN = INMODEREG_REG[0];

  MREG_BIN = MREG_REG[0];

  OPMODEREG_BIN = OPMODEREG_REG[0];

  PREADDINSEL_BIN =
    (PREADDINSEL_REG == "A") ? PREADDINSEL_A :
    (PREADDINSEL_REG == "B") ? PREADDINSEL_B :
     PREADDINSEL_A;

  PREG_BIN = PREG_REG[0];

  SEL_MASK_BIN =
    (SEL_MASK_REG == "MASK") ? SEL_MASK_MASK :
    (SEL_MASK_REG == "C") ? SEL_MASK_C :
    (SEL_MASK_REG == "ROUNDING_MODE1") ? SEL_MASK_ROUNDING_MODE1 :
    (SEL_MASK_REG == "ROUNDING_MODE2") ? SEL_MASK_ROUNDING_MODE2 :
     SEL_MASK_MASK;

  SEL_PATTERN_BIN =
    (SEL_PATTERN_REG == "PATTERN") ? SEL_PATTERN_PATTERN :
    (SEL_PATTERN_REG == "C") ? SEL_PATTERN_C :
     SEL_PATTERN_PATTERN;

  USE_MULT_BIN =
    (USE_MULT_REG == "MULTIPLY") ? USE_MULT_MULTIPLY :
    (USE_MULT_REG == "DYNAMIC") ? USE_MULT_DYNAMIC :
    (USE_MULT_REG == "NONE") ? USE_MULT_NONE :
     USE_MULT_MULTIPLY;

  USE_PATTERN_DETECT_BIN =
    (USE_PATTERN_DETECT_REG == "NO_PATDET") ? USE_PATTERN_DETECT_NO_PATDET :
    (USE_PATTERN_DETECT_REG == "PATDET") ? USE_PATTERN_DETECT_PATDET :
     USE_PATTERN_DETECT_NO_PATDET;

  USE_SIMD_BIN =
    (USE_SIMD_REG == "ONE48") ? USE_SIMD_ONE48 :
    (USE_SIMD_REG == "FOUR12") ? USE_SIMD_FOUR12 :
    (USE_SIMD_REG == "TWO24") ? USE_SIMD_TWO24 :
     USE_SIMD_ONE48;

  USE_WIDEXOR_BIN =
    (USE_WIDEXOR_REG == "FALSE") ? USE_WIDEXOR_FALSE :
    (USE_WIDEXOR_REG == "TRUE") ? USE_WIDEXOR_TRUE :
     USE_WIDEXOR_FALSE;

  XORSIMD_BIN =
    (XORSIMD_REG == "XOR24_48_96") ? XORSIMD_XOR24_48_96 :
    (XORSIMD_REG == "XOR12") ? XORSIMD_XOR12 :
     XORSIMD_XOR24_48_96;

end
`endif

`ifndef XIL_XECLIB
  always @(trig_attr) begin
  #1;
    if ((attr_test == 1'b1) ||
        ((ACASCREG_REG != 1) &&
         (ACASCREG_REG != 0) &&
         (ACASCREG_REG != 2))) begin
      $display("Error: [Unisim %s-101] ACASCREG attribute is set to %d.  Legal values for this attribute are
1, 0 or 2. Instance: %m", MODULE_NAME, ACASCREG_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((ADREG_REG != 1) &&
         (ADREG_REG != 0))) begin
      $display("Error: [Unisim %s-102] ADREG attribute is set to %d.  Legal values for this attribute are 1
or 0. Instance: %m", MODULE_NAME, ADREG_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((ALUMODEREG_REG != 1) &&
         (ALUMODEREG_REG != 0))) begin
      $display("Error: [Unisim %s-103] ALUMODEREG attribute is set to %d.  Legal values for this attribute
are 1 or 0. Instance: %m", MODULE_NAME, ALUMODEREG_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((AMULTSEL_REG != "A") &&
         (AMULTSEL_REG != "AD"))) begin
      $display("Error: [Unisim %s-104] AMULTSEL attribute is set to %s.  Legal values for this attribute are
A or AD. Instance: %m", MODULE_NAME, AMULTSEL_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((AREG_REG != 1) &&
         (AREG_REG != 0) &&
         (AREG_REG != 2))) begin
      $display("Error: [Unisim %s-105] AREG attribute is set to %d.  Legal values for this attribute are 1, 0
or 2. Instance: %m", MODULE_NAME, AREG_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((AUTORESET_PATDET_REG != "NO_RESET") &&
         (AUTORESET_PATDET_REG != "RESET_MATCH") &&
         (AUTORESET_PATDET_REG != "RESET_NOT_MATCH"))) begin
      $display("Error: [Unisim %s-106] AUTORESET_PATDET attribute is set to %s.  Legal values for this
attribute are NO_RESET, RESET_MATCH or RESET_NOT_MATCH. Instance: %m", MODULE_NAME, AUTORESET_PATDET_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((AUTORESET_PRIORITY_REG != "RESET") &&
         (AUTORESET_PRIORITY_REG != "CEP"))) begin
      $display("Error: [Unisim %s-107] AUTORESET_PRIORITY attribute is set to %s.  Legal values for this
attribute are RESET or CEP. Instance: %m", MODULE_NAME, AUTORESET_PRIORITY_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((A_INPUT_REG != "DIRECT") &&
         (A_INPUT_REG != "CASCADE"))) begin
      $display("Error: [Unisim %s-108] A_INPUT attribute is set to %s.  Legal values for this attribute are
DIRECT or CASCADE. Instance: %m", MODULE_NAME, A_INPUT_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((BCASCREG_REG != 1) &&
         (BCASCREG_REG != 0) &&
         (BCASCREG_REG != 2))) begin
      $display("Error: [Unisim %s-109] BCASCREG attribute is set to %d.  Legal values for this attribute are
1, 0 or 2. Instance: %m", MODULE_NAME, BCASCREG_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((BMULTSEL_REG != "B") &&
         (BMULTSEL_REG != "AD"))) begin
      $display("Error: [Unisim %s-110] BMULTSEL attribute is set to %s.  Legal values for this attribute are
B or AD. Instance: %m", MODULE_NAME, BMULTSEL_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((BREG_REG != 1) &&
         (BREG_REG != 0) &&
         (BREG_REG != 2))) begin
      $display("Error: [Unisim %s-111] BREG attribute is set to %d.  Legal values for this attribute are 1, 0
or 2. Instance: %m", MODULE_NAME, BREG_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((B_INPUT_REG != "DIRECT") &&
         (B_INPUT_REG != "CASCADE"))) begin
      $display("Error: [Unisim %s-112] B_INPUT attribute is set to %s.  Legal values for this attribute are
DIRECT or CASCADE. Instance: %m", MODULE_NAME, B_INPUT_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((CARRYINREG_REG != 1) &&
         (CARRYINREG_REG != 0))) begin
      $display("Error: [Unisim %s-113] CARRYINREG attribute is set to %d.  Legal values for this attribute
are 1 or 0. Instance: %m", MODULE_NAME, CARRYINREG_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((CARRYINSELREG_REG != 1) &&
         (CARRYINSELREG_REG != 0))) begin
      $display("Error: [Unisim %s-114] CARRYINSELREG attribute is set to %d.  Legal values for this attribute
are 1 or 0. Instance: %m", MODULE_NAME, CARRYINSELREG_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((CREG_REG != 1) &&
         (CREG_REG != 0))) begin
      $display("Error: [Unisim %s-115] CREG attribute is set to %d.  Legal values for this attribute are 1 or
0. Instance: %m", MODULE_NAME, CREG_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((DREG_REG != 1) &&
         (DREG_REG != 0))) begin
      $display("Error: [Unisim %s-116] DREG attribute is set to %d.  Legal values for this attribute are 1 or
0. Instance: %m", MODULE_NAME, DREG_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((INMODEREG_REG != 1) &&
         (INMODEREG_REG != 0))) begin
      $display("Error: [Unisim %s-117] INMODEREG attribute is set to %d.  Legal values for this attribute are
1 or 0. Instance: %m", MODULE_NAME, INMODEREG_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((MREG_REG != 1) &&
         (MREG_REG != 0))) begin
      $display("Error: [Unisim %s-134] MREG attribute is set to %d.  Legal values for this attribute are 1 or
0. Instance: %m", MODULE_NAME, MREG_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((OPMODEREG_REG != 1) &&
         (OPMODEREG_REG != 0))) begin
      $display("Error: [Unisim %s-135] OPMODEREG attribute is set to %d.  Legal values for this attribute are
1 or 0. Instance: %m", MODULE_NAME, OPMODEREG_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((PREADDINSEL_REG != "A") &&
         (PREADDINSEL_REG != "B"))) begin
      $display("Error: [Unisim %s-137] PREADDINSEL attribute is set to %s.  Legal values for this attribute
are A or B. Instance: %m", MODULE_NAME, PREADDINSEL_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((PREG_REG != 1) &&
         (PREG_REG != 0))) begin
      $display("Error: [Unisim %s-138] PREG attribute is set to %d.  Legal values for this attribute are 1 or
0. Instance: %m", MODULE_NAME, PREG_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((SEL_MASK_REG != "MASK") &&
         (SEL_MASK_REG != "C") &&
         (SEL_MASK_REG != "ROUNDING_MODE1") &&
         (SEL_MASK_REG != "ROUNDING_MODE2"))) begin
      $display("Error: [Unisim %s-140] SEL_MASK attribute is set to %s.  Legal values for this attribute are
MASK, C, ROUNDING_MODE1 or ROUNDING_MODE2. Instance: %m", MODULE_NAME, SEL_MASK_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((SEL_PATTERN_REG != "PATTERN") &&
         (SEL_PATTERN_REG != "C"))) begin
      $display("Error: [Unisim %s-141] SEL_PATTERN attribute is set to %s.  Legal values for this attribute
are PATTERN or C. Instance: %m", MODULE_NAME, SEL_PATTERN_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((USE_MULT_REG != "MULTIPLY") &&
         (USE_MULT_REG != "DYNAMIC") &&
         (USE_MULT_REG != "NONE"))) begin
      $display("Error: [Unisim %s-142] USE_MULT attribute is set to %s.  Legal values for this attribute are
MULTIPLY, DYNAMIC or NONE. Instance: %m", MODULE_NAME, USE_MULT_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((USE_PATTERN_DETECT_REG != "NO_PATDET") &&
         (USE_PATTERN_DETECT_REG != "PATDET"))) begin
      $display("Error: [Unisim %s-143] USE_PATTERN_DETECT attribute is set to %s.  Legal values for this
attribute are NO_PATDET or PATDET. Instance: %m", MODULE_NAME, USE_PATTERN_DETECT_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((USE_SIMD_REG != "ONE48") &&
         (USE_SIMD_REG != "FOUR12") &&
         (USE_SIMD_REG != "TWO24"))) begin
      $display("Error: [Unisim %s-144] USE_SIMD attribute is set to %s.  Legal values for this attribute are
ONE48, FOUR12 or TWO24. Instance: %m", MODULE_NAME, USE_SIMD_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((USE_WIDEXOR_REG != "FALSE") &&
         (USE_WIDEXOR_REG != "TRUE"))) begin
      $display("Error: [Unisim %s-145] USE_WIDEXOR attribute is set to %s.  Legal values for this attribute
are FALSE or TRUE. Instance: %m", MODULE_NAME, USE_WIDEXOR_REG);
      attr_err = 1'b1;
    end

    if ((attr_test == 1'b1) ||
        ((XORSIMD_REG != "XOR24_48_96") &&
         (XORSIMD_REG != "XOR12"))) begin
      $display("Error: [Unisim %s-146] XORSIMD attribute is set to %s.  Legal values for this attribute are
XOR24_48_96 or XOR12. Instance: %m", MODULE_NAME, XORSIMD_REG);
      attr_err = 1'b1;
    end

    if (attr_err == 1'b1) #1 $finish;
  end
`endif

// begin behavioral model

  always @(trig_attr) begin
    #1;
    case (AREG_REG)
      0, 1 : if(AREG_REG != ACASCREG_REG) begin
      $display("Error: [Unisim %s-2] AREG attribute is set to %s and ACASCREG attribute is set to %s. When
AREG is 0 or 1, ACASCREG must be set to the same value. Instance: %m", MODULE_NAME, AREG_REG, ACASCREG_REG);
        attr_err = 1'b1;
        end
      2 : if(ACASCREG_REG == 0) begin
      $display("Error: [Unisim %s-3] AREG attribute is set to %s and ACASCREG attribute is set to %s. When
AREG is 2, ACASCREG must be set to 1 or 2. Instance: %m", MODULE_NAME, AREG_REG, ACASCREG_REG);
        attr_err = 1'b1;
        end
    endcase

    case (BREG_REG)
      0, 1 : if(BREG_REG != BCASCREG_REG) begin
      $display("Error: [Unisim %s-4] BREG attribute is set to %s and BCASCREG attribute is set to %s. When
BREG is 0 or 1, BCASCREG must be set to the same value. Instance: %m", MODULE_NAME, BREG_REG, BCASCREG_REG);
        attr_err = 1'b1;
        end
      2 : if(BCASCREG_REG == 0) begin
      $display("Error: [Unisim %s-5] BREG attribute is set to %s and BCASCREG attribute is set to %s. When
BREG is 2, BCASCREG must be set to 1 or 2. Instance: %m", MODULE_NAME, BREG_REG, BCASCREG_REG);
        attr_err = 1'b1;
        end
    endcase

    if (attr_err == 1'b1) #1 $finish;
  end

  always @(trig_attr) begin
    #1;
    if ((USE_MULT_REG == "NONE") && (MREG_REG !== 0)) begin
        $display("Error : [Unisim %s-6] : Attribute USE_MULT is set to \"NONE\" and MREG is set to %d. MREG
must be set to 0 when the multiplier is not used. Instance %m", MODULE_NAME, MREG_REG);
        attr_err = 1'b1;
    end

  if (attr_err == 1'b1) #1 $finish;
  end

// Connections between atoms
  wire [44:0] U_DATA;
  wire [44:0] V_DATA;
  reg [26:0] A2A1;
  reg [17:0] B2B1;
  wire AMULT26_in;
  wire BMULT17_in;
  wire ALUMODE10_in;
  wire CCOUT_in;
  wire MULTSIGN_ALU_in;
  wire P_FDBK_47_in;
  wire [26:0] AD_DATA;
  wire [26:0] AD_in;
  wire [26:0] D_DATA;
  wire [26:0] PREADD_AB;
  wire [3:0] COUT_in;
  wire [44:0] U_in;
  wire [44:0] V_in;
  wire [44:0] U_DATA_in;
  wire [44:0] V_DATA_in;
  wire [47:0] ALU_OUT_in;
  wire [47:0] C_DATA_in;
  wire [47:0] P_FDBK_in;
  wire [7:0] XOR_MX_in;

// DSP_ALU wires
  localparam MAX_ALU_FULL   = 48;
  localparam MAX_CARRYOUT   = 4;
  localparam A_WIDTH   = 30;
  localparam B_WIDTH   = 18;
  localparam C_WIDTH   = 48;
```

FPGA	Primitive	SystemVerilog
Xilinx Ultrascale+	LUT6	88
	CARRY8	23
	DSP48E2	1426
Lattice ECP5	LUT2	5
	LUT4	7
	CCU2C	60
	ALU24B	672
	MULT18X18D	985
SOFA	frac_lut4	69
Intel Cyclone	altmult_accum	1460

Lakeroad can compile a hardware design to an FPGA given only the Verilog models of the FPGA’s primitives.

Thank You!