Formal Verification of Reverse Engineered FPGA Bitstreams

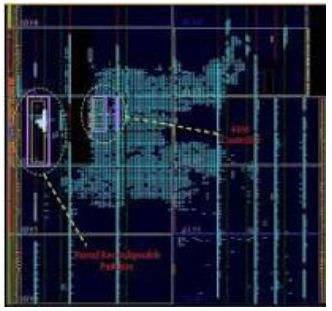
Russell Tessier, Andrew Hartnett, Anurag Muttur, and Jared Malone
Department of Electrical and Computer Engineering
University of Massachusetts Amherst

Maik Ender and Vincent Kraemer
Ruhr University Bochum
Max Planck Institute for Security and Privacy

Overview: Formal Verification of RTL Design

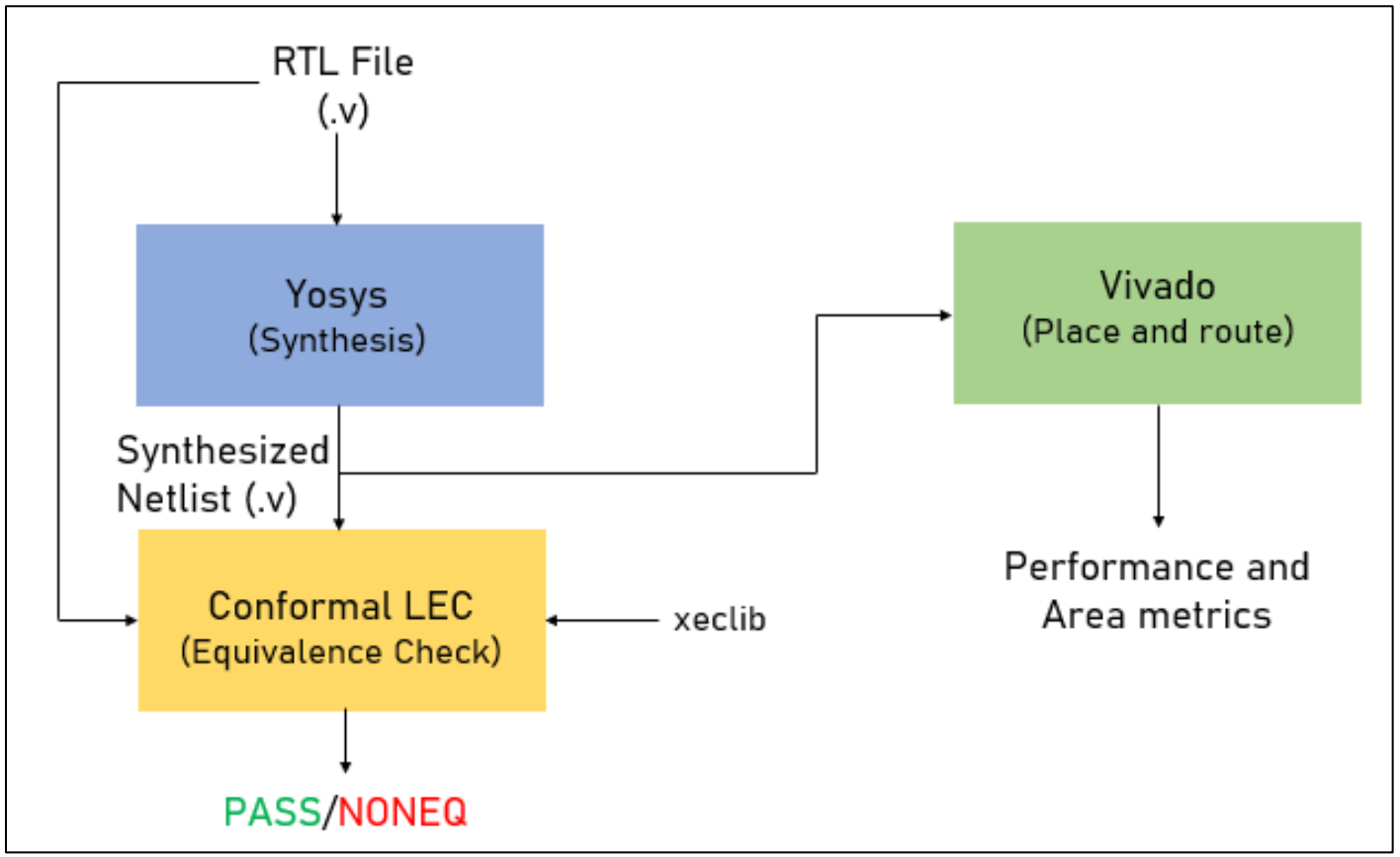
RTL Design ?= Synthesized Gate-level Design
RTL Design ?= Reverse-engineered Gate-level Design

```
always @(posedge clk)
result = in0 * in1
```



- Goal: Verify *Verilog RTL* design against reverse-engineering gate-level FPGA netlist (derived from bitstream)
- Current: Formal verification of RTL design against gate-level netlist generated by open-source synthesis tool
- Why? Locate Trojans and other tampering in design flow / implementation
- Formal verification of simple RTL designs against reverse engineered netlist

Formal Verification of Synthesized Designs

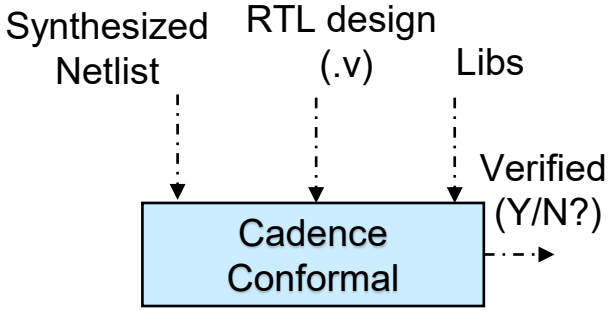


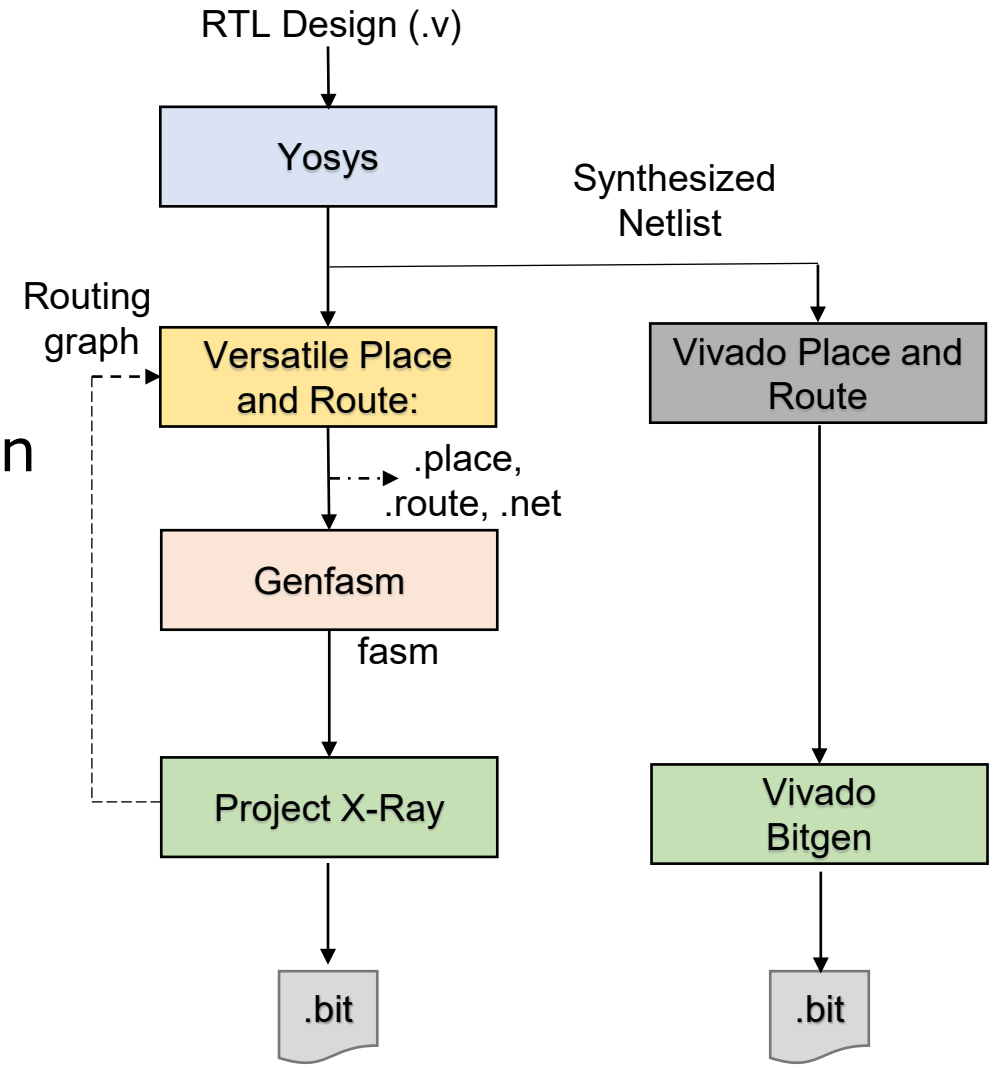
Question: Does it pass?

- Yosys – open-source logic synthesis tool (Claire Wolf)
- Perform formal verification using Cadence Conformal Logic Equivalence Checker (LEC)

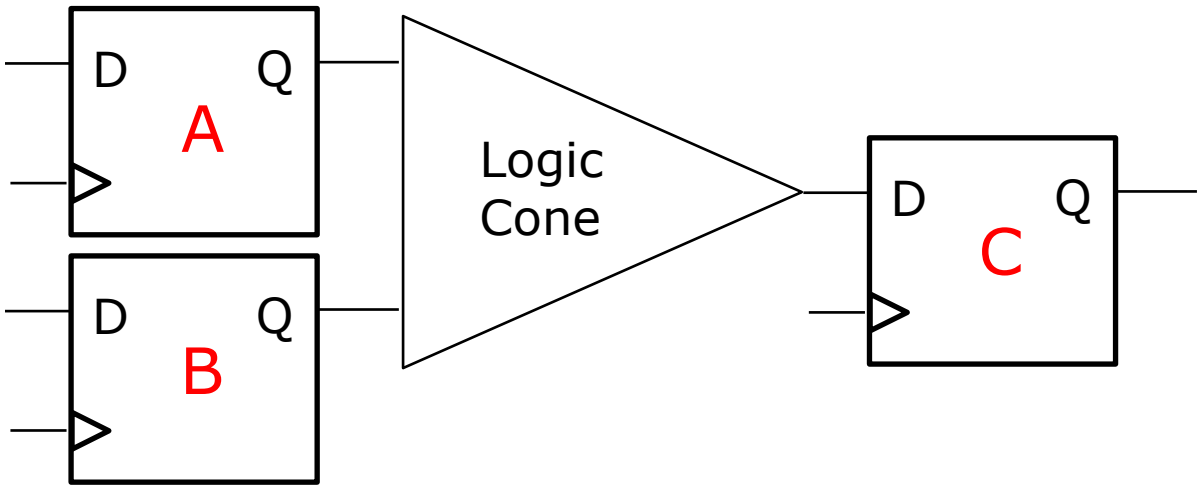
RTL Synthesis / Verification Design Flow – Artix 7

- Synthesis
  - Yosys
- Place and Route
  - VPR / Vivado
- Bitstream Generation
  - Project X-Ray / Vivado Bitgen
- Formal Verification
  - Cadence Conformal





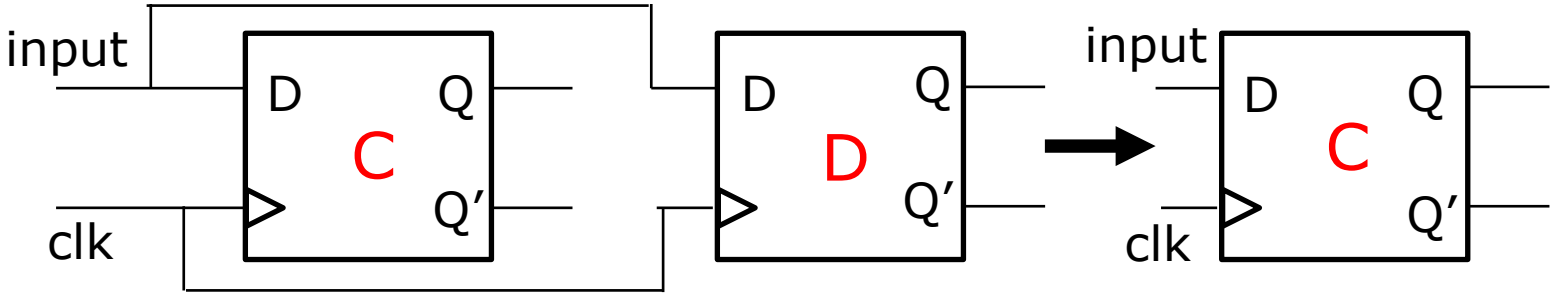
Formal Verification with Cadence Conformal



- **Conformal effectively performs synthesis of RTL design**
- Compare "golden netlist" (RTL) to gate-level design
- Checks logic from FF to FF or FF to I/O
- Typically, must have at least I/O names in common
  - Gate library needed
  - Limited popularity with FPGAs

Yosys Optimizations

- Conformal must "understand" optimizations in Yosys
  1. Flip flop elimination



  2. FSM optimization – binary to one-hot encoding
     - 000 -> 00001
     - 001 -> 00010
     - 010 -> 00100
     - 011 -> 01000
     - 100 -> 10000

Create "hints" file for Conformal

Flip flop naming should match

Unsupported Yosys Optimizations

- Some optimizations in Yosys must be shut off
  - Memory block remapping
  - Retiming
  - Unused I/O pruning
  - Some don’t care optimization / logic restructuring
  - Incompletely defined FSMs

```
case (state)
    s1: state = s2
    s2: state = s3;
    s3: state = s1;
endcase
```

```
s1 -> 2’b00
s2 -> 2’b01
s3 -> 2’b10
```

Results Summary

- 168 designs evaluated using Yosys and Conformal, 155 pass.
  - 124 in yosys-simple, 9 in yosys-bigsim, 18 in vtr-benchmarks, and 4 local designs.

- Designs from yosys-simple, yosys-bigsim, and VTR suites
  - Simple designs (a few gates, small number of FFs) – all but 8 passed checking
  - Modest designs (thousands of gates) – a few failed due to Verilog coding issues
  - Often took a lot of "coaxing" to get Conformal to "try harder"
  - Conformal ignores dead logic

Designs With Thousands of LUTs

- 9 out of 11 yosys-bigsim and 18 of 19 vtr-benchmarks pass
- Small microprocessors, hardware used in ray-tracing, hardware used in AES encryption for 5 stages etc.
- Designs with multipliers greater than 24x24 not supported.
- Verification times range from a few seconds to 2 hours.

```
Undriven key points                          0              176
   Unmapped                                  0              176
      Unreachable                            0              176

Primary outputs                            128              128
   Mapped                                  128              128
      Equivalent              128

State key points                           896              896
   Mapped                                  128              128
      Equivalent              128
   Unmapped                                768              768
      Unreachable                          768              768
================================================================================
// Command: usage
CPU time     : 4084.24 seconds
Memory usage : 332.07  M bytes
// Command: report verification -compare_result
Compare Results:                                                        PASS
// Command: set log file
// Command: exit -f
```

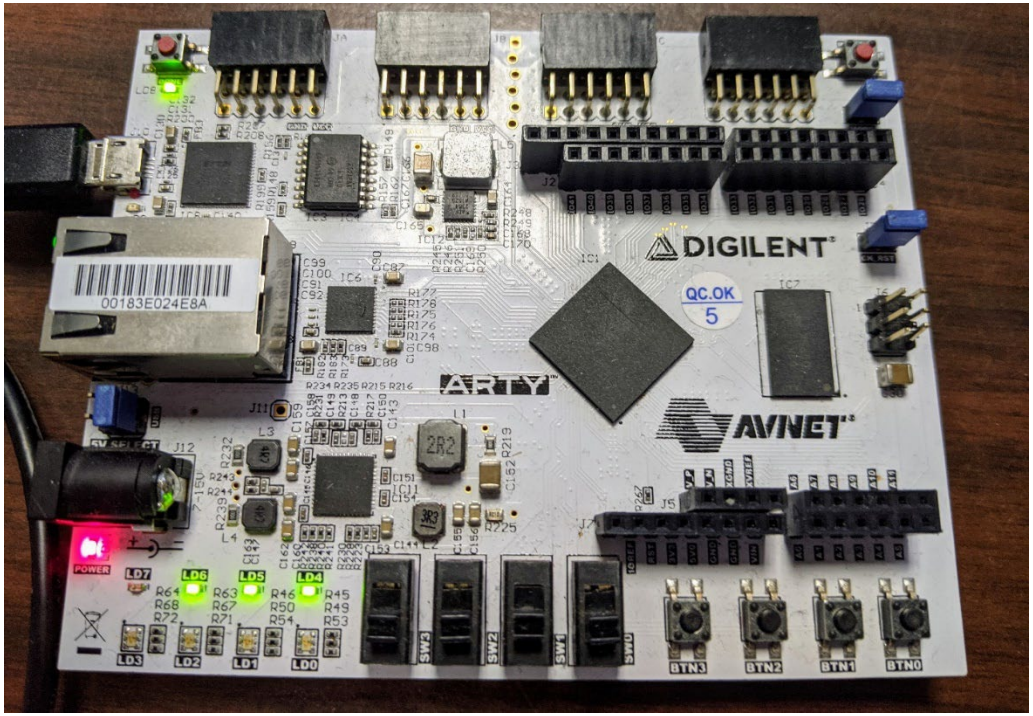
Design Clock Speed Performance Results

- Optimizations can make a substantial performance difference (FSM recoding, FF merging, logic pruning)

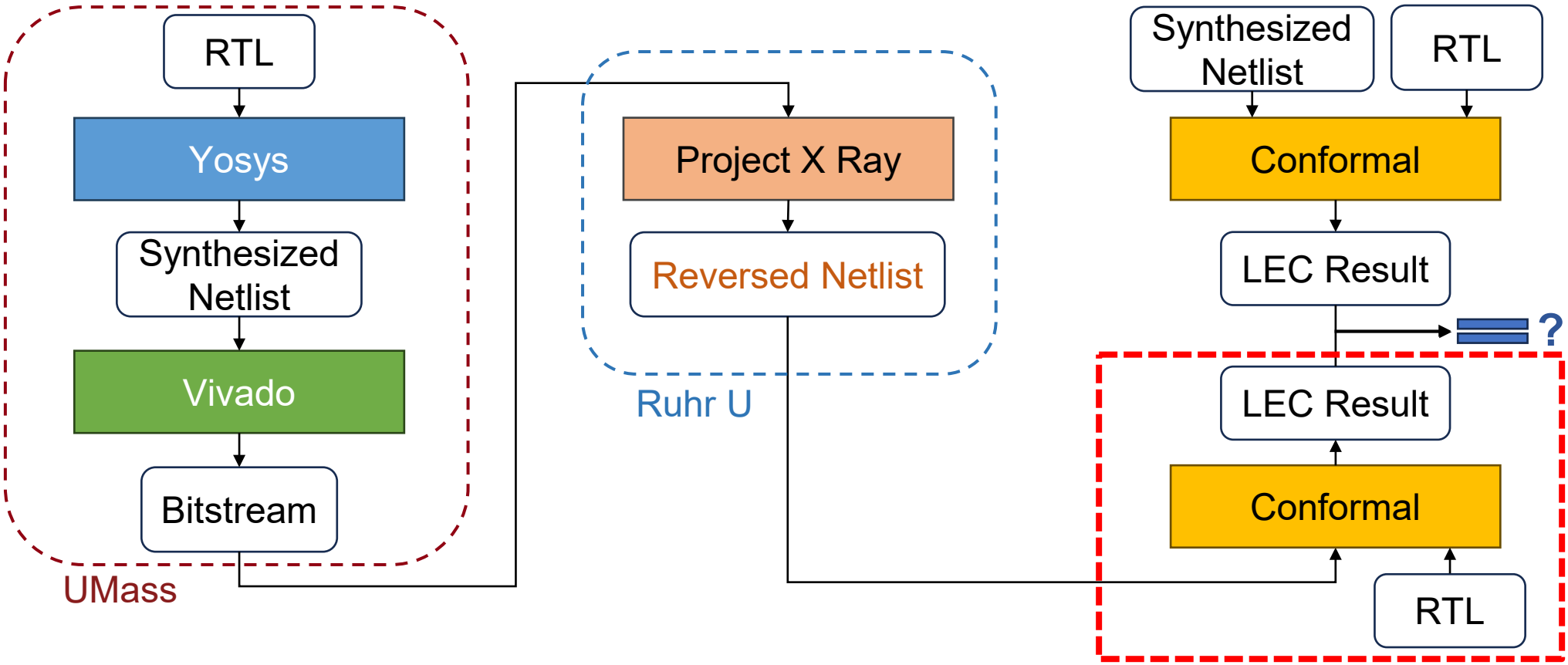
Design Name	Without Yosys Optimizations (Formally Verifiable)			With Yosys Optimizations (Formally Verifiable)			Vivado (Not Formally Verifiable)		
	LUT	Flip Flop	Clock Speed (MHz)	LUT	Flip Flop	Clock Speed (MHz)	LUT	Flip Flop	Clock Speed (MHz)
softusb_navre	1018	340	73.56	1105	353	80.46	877	340	92.91
blob_merge	3461	552	48.48	3659	575	49.04	5457	575	69.74
stereovision0	10396	13258	129.05	3611	9069	200.80	3359	8081	221.43
ch_intrinsiscs	209	496	265.25	92	211	270.71	29	82	360.23
sha	1400	910	122.02	1479	893	120.29	1245	903	128.65

FPGA Hardware Verification Flow

- Sixteen benchmarks are verified in hardware (Artix A7-35T)
- Test vectors generated for design using RTL simulation
- Generated results compared to expected results (stored in BRAM)
- Entire setup, including test harness, synthesized via Yosys



Bitstream Verification Workflow (in progress)



- New flow: use netlist generated by reverse engineering for comparison against RTL

Formal Verification of Reverse Engineering Design

- Issues:
  - Internal nodes (including flip flops) contain no naming information
  - Reverse-engineered netlist includes information for entire logic block (including unused portions)

RTL for 2-input and gate

```
input a;
input b;
output y;

assign y = a & b;
```

Synthesized gate level

```
LUT2 #(
   .INIT(4'h8)
) _2_ (
   .I0(_1_[0]),
   .I1(_1_[1]),
   .O(_0_)
);
```

Reverse engineered netlist

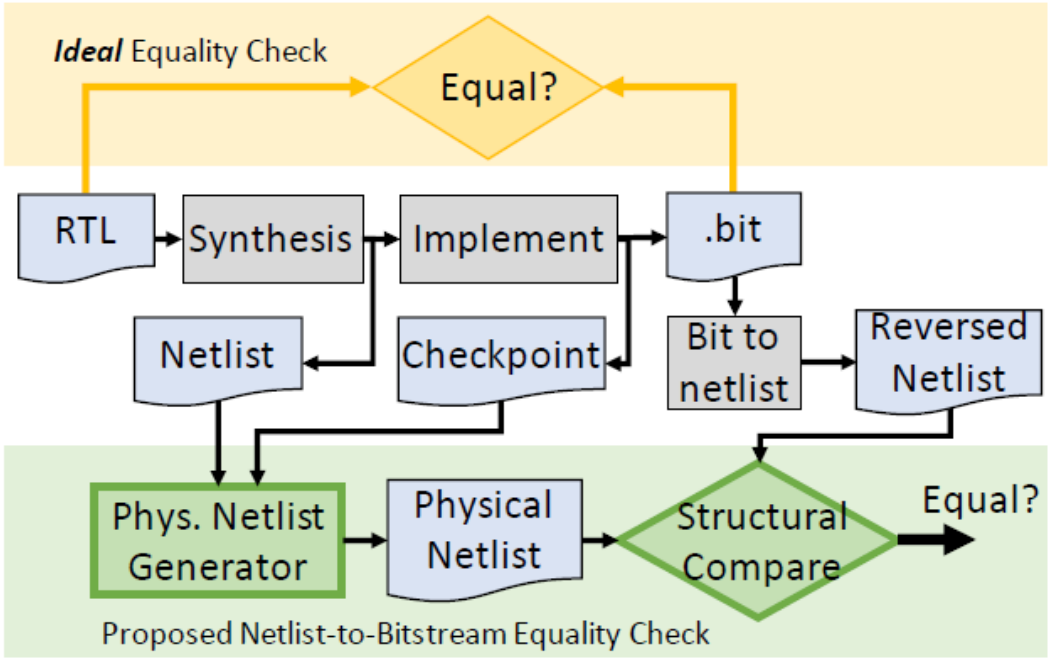
```
LUT6_2 #(
    .INIT(64'hF0F0F0F000000000)
) CLBLL_L_X2Y125_SLICE_X0Y125_ALUT (
    .I0(const_1 ),
    .I1(const_1 ),
    .I2(\LIOB33_X0Y125_IOB_X0Y125_I(0) ),
    .I3(const_1 ),
    .I4(const_1 ),
    .I5(\LIOB33_X0Y125_IOB_X0Y126_I(0) ),
    .O5(\CLBLL_L_X2Y125_SLICE_X0Y125_AO5(0) ),
    .O6(\CLBLL_L_X2Y125_SLICE_X0Y125_AO6(0) )
);
```

Verification of Reversed Bitstream Status

- Successfully verified four small RTL designs against reverse-engineered netlists
  - Single-gate design, flip flop with inverter (2 variants), four bit carry adder
- Bitstream reversal of design in form usable for formal verification is important
- Next steps:
  - Larger designs
  - Memory-based circuits
  - User interfaces

Comparison to Recent Work

- Starts with synthesized netlist, not RTL
- Netlist restructured to aid verification
- Very fast structural comparison (<10s for 13,000 LUTs)
- Next: Correlate RTL with Physical netlist



McKendrick, Faulkner, and Goeders, *Assuring Netlist-to-Bitstream Equivalence using Physical Netlist Generation and Structural Comparison,* FPT'24

Conclusion

- Presented a robust flow of synthesis using Yosys and formal verification using Cadence Conformal.
- Formally verified 155 designs including designs over 10,000 LUTs
- Current formal verification approach is slow (minutes for thousands of logic gates). Open-source tools are needed
- Proof of concept / flow in place for RTL / reversed bitstream formal verification

```
================================================================================
// Command: usage
CPU time      : 4084.24 seconds
Memory usage : 332.07  M bytes
// Command: report verification -compare_result
Compare Results:                                                          PASS
// Command: set log file
// Command: exit -f
```