Foundations of Embedded Systems

Fascicle 11: The YoSys, IceStorm, ArachnePNR and NextPNR Tools

(Video)

(~30 minutes)

Version 0.2020

Phillip **Stanley-Marbell**

Department of Engineering, University of Cambridge

http://physcomp.eng.cam.ac.uk

UNIVERSITY OF CAMBRIDGE

F(E)

(10_2)

Intended Learning Outcomes for Today

By the end of this session, you will be able to:

❶ Use the Yosys, ArachnePNR, and NextPNR tools for synthesis and place/route

❷ Use Yosys to simulate parts of a combinational circuit

❸ Use Yosys to check whether certain signal values are possible in your design

❹ Use Yosys to visualize your design at the gate and Verilog block levels

❺ Use the Yosys and ArachnePNR or NextPNR tools, along with icetime, icepack, and iceprog for synthesis and place/route, timing analysis bitstream conversion, and configuring the iCE40 MDP evaluation board

A Simple Example

Simple Example 1 of 2: Module Verilog

```
1  /*
2   *      Verilog uses C-style comments
3   */
4  module simple(
5                  /*
6                   *      The identifiers listed here are known
7                   *      as the "ports" of the Verilog module
8                   *      and this part of the module (in parenthesis)
9                   *      is known as the "port list".
10                  *
11                  *      We will follow a number of conventions
12                  *      for the stylistic layout of Verilog code.
13                  *      You can find the list of conventions here:
14                  *
15                  *      github.com/physical-computation/conventions/README-RTLCodingConvention.md
16                  */
17                 inSignal,
18                 outSignal
19                 );
20         /*
21          *      The first component in the module is typically the input/output (I/O) declarations.
22          */
23         input   inSignal;
24         output  outSignal;
25
26         /*
27          *      And finally the module's implementation.
28          */
29         assign  outSignal = inSignal;
30 endmodule
```

Use Yosys to Apply an Input to Combinational Logic


```
$ yosys -p "hierarchy -check; proc; opt; fsm; opt; memory; opt;
eval -set inSignal 1'b0 -show outSignal" simple.v
```

Use Yosys to Evaluate a Truth Table for a Combinational Block

```
$ yosys -p "hierarchy -check; proc; opt; fsm; opt; memory; opt;
eval -table inSignal" simple.v
```

Use Yosys to Check Satisfiability for a Set of Conditions

```
$ yosys -p "hierarchy -check; proc; opt; fsm; opt; memory; opt;
sat -set outSignal 1'b0" simple.v
```

Simple Example 2 of 2: Verilog Testbench


```
1  `timescale 1ns/1ns
2
3  module simpleTestbench;
4          reg     signalIn, signalOut;
5
6          simple simpleInstance(.inSignal(signalIn), .outSignal(singnalOut));
7
8          initial begin
9          /*
10         *       Delay for one time unit (1ns, as defined above), then set singalIn to 0
11         */
12         #1      signalIn = 0;
13
14         /*
15         *       Delay for one time unit (1ns, as defined above), then set singalIn to 1
16         */
17         #1      signalIn = 1;
18
19         /*
20         *       Delay for 10 time units (10ns, as defined above), then set singalIn to 0
21         */
22         #10     signalIn = 0;
23
24         /*
25         *       Delay for one time unit (1ns, as defined above), then set singalIn to 1
26         */
27         #1      signalIn = 1;
28         end
29         initial begin
30                 $dumpfile("simpleTestbench.vcd");
31                 $dumpvars;
32         end
33 endmodule
```

Running Testbench Using iverilog


```
$  iverilog -o simple simple.v simpleTestbench.v; ./simple
```

A Larger Example

Larger Example 1 of 2


```
module e4Encoder_8bitData_1bitControl(dataIn, worstCaseDeviationCode, encoderOut);
	input		[7:0]	dataIn;
	input		[0:0]	worstCaseDeviationCode;
	output reg	[7:0]	encoderOut;

	reg		[7:0]	M1encoderOut;

	reg		[7:0]	M1dataIn;



	always @ (*)
	begin
		case (worstCaseDeviationCode)
			1'b0:
			begin
				M1dataIn = 8'h00;
			end

			1'b1:
			begin
				M1dataIn = dataIn;
			end
		endcase



		/*
		 *	M1dataIn => worst-case deviation is '10'
		 */
		case (M1dataIn)
			8'h00,8'h01,8'h02,8'h03,
			8'h04,8'h05,8'h06,8'h07,
			8'h08,8'h09,8'h0A		: M1encoderOut = 8'h00;
			8'h0B				: M1encoderOut = 8'h01;
			8'h0C,8'h0D			: M1encoderOut = 8'h03;
			8'h0E,8'h10,8'h11		: M1encoderOut = 8'h07;
			8'h0F,8'h12,8'h13,8'h14,
			8'h15,8'h16,8'h17,8'h18,
			8'h19				: M1encoderOut = 8'h0F;
			8'h1A,8'h1B,8'h1C,8'h1D,
			8'h1E,8'h1F,8'h20,8'h21,
			8'h22,8'h23,8'h24,8'h25,
			8'h26,8'h27,8'h28,8'h29		: M1encoderOut = 8'h1F;
			8'h2A				: M1encoderOut = 8'h20;
			8'h2B,8'h2C,8'h2D,8'h2E,
			8'h2F,8'h30,8'h31,8'h32,
			8'h33,8'h34			: M1encoderOut = 8'h30;
			8'h35,8'h36,8'h37,8'h38,
			8'h39,8'h3A,8'h3B,8'h3C,
			8'h3D,8'h3E,8'h3F,8'h40,
			8'h41,8'h42,8'h43,8'h44,
			8'h45,8'h46,8'h47,8'h48,
			8'h49				: M1encoderOut = 8'h3F;
			8'h4A				: M1encoderOut = 8'h40;
			8'h4B				: M1encoderOut = 8'h41;
			8'h4C,8'h4D			: M1encoderOut = 8'h43;
			8'h4E,8'h50,8'h51		: M1encoderOut = 8'h47;
			8'h4F,8'h52,8'h53,8'h54,
			8'h55				: M1encoderOut = 8'h4F;
			8'h56,8'h57,8'h58,8'h59,
			8'h5A,8'h5B,8'h5C,8'h5D,
			8'h5E,8'h5F,8'h60,8'h61,
			8'h62,8'h63,8'h64,8'h65,
			8'h66,8'h67,8'h68,8'h69,
			8'h6A				: M1encoderOut = 8'h60;
			8'h6B,8'h6C,8'h6D,8'h6E,
			8'h6F,8'h70,8'h71,8'h72,
			8'h73,8'h74			: M1encoderOut = 8'h70;
			8'h75,8'h76,8'h77,8'h78,
			8'h79,8'h7A,8'h7B,8'h7C,
			8'h7D,8'h7E,8'h7F,8'h81,
			8'h82,8'h83,8'h84,8'h85,
			8'h86,8'h87,8'h88,8'h89		: M1encoderOut = 8'h7F;
			8'h80,8'h8A			: M1encoderOut = 8'h80;
			8'h8B				: M1encoderOut = 8'h81;
			8'h8C,8'h8D			: M1encoderOut = 8'h83;
			8'h8E,8'h90,8'h91		: M1encoderOut = 8'h87;
			8'h8F,8'h92,8'h93,8'h94,
			8'h95,8'h96,8'h97,8'h98,
			8'h99				: M1encoderOut = 8'h8F;
			8'h9A,8'h9B,8'h9C,8'h9D,
			8'h9E,8'h9F,8'hA0,8'hA1,
			8'hA2,8'hA3,8'hA4,8'hA5,
			8'hA6,8'hA7,8'hA8,8'hA9		: M1encoderOut = 8'h9F;
			8'hAA				: M1encoderOut = 8'hA0;
			8'hAB,8'hAC,8'hAD,8'hAE,
			8'hAF,8'hB0,8'hB1,8'hB2,
			8'hB3,8'hB4			: M1encoderOut = 8'hB0;
			8'hB5				: M1encoderOut = 8'hBF;
			8'hB6,8'hB7,8'hB8,8'hB9,
			8'hBA,8'hBB,8'hBC,8'hBD,
			8'hBE,8'hBF,8'hC0,8'hC1,
			8'hC2,8'hC3,8'hC4,8'hC5,
			8'hC6,8'hC7,8'hC8,8'hC9,
			8'hCA				: M1encoderOut = 8'hC0;
			8'hCB				: M1encoderOut = 8'hC1;
			8'hCC,8'hCD			: M1encoderOut = 8'hC3;
			8'hCE,8'hD0,8'hD1		: M1encoderOut = 8'hC7;
			8'hCF,8'hD2,8'hD3,8'hD4,
			8'hD5				: M1encoderOut = 8'hCF;
			8'hD6,8'hD7,8'hD8,8'hD9,
			8'hDA,8'hDB,8'hDC,8'hDD,
			8'hDE,8'hDF,8'hE0,8'hE1,
			8'hE2,8'hE3,8'hE4,8'hE5,
			8'hE6,8'hE7,8'hE8,8'hE9,
			8'hEA				: M1encoderOut = 8'hE0;
			8'hEB,8'hEC,8'hED,8'hEE,
			8'hEF,8'hF0,8'hF1,8'hF2,
			8'hF3,8'hF4			: M1encoderOut = 8'hF0;
			8'hF5,8'hF6,8'hF7,8'hF8,
			8'hF9,8'hFA,8'hFB,8'hFC,
			8'hFD,8'hFE,8'hFF		: M1encoderOut = 8'hFF;
			default				: M1encoderOut = 8'h00;
		endcase /* M1dataIn */



		/*
		 *	Select the final output from amongst the sub-encoder blocks.
		 */
		case (worstCaseDeviationCode)
			1'b0: encoderOut = dataIn;
			1'b1: encoderOut = M1encoderOut;
			default : encoderOut = dataIn;
		endcase
	end /* always @ (*) */
endmodule /* e4Encoder_8bitData_1bitControl */
```

Larger Example 2 of 2


```
module e4Encoder_8bitData_1bitControl(dataIn, worstCaseDeviationCode, encoderOut);
        input       [7:0]   dataIn;
        input       [0:0]   worstCaseDeviationCode;
        output reg  [7:0]   encoderOut;

        reg         [7:0]   M1encoderOut;

        reg         [7:0]   M1dataIn;



        always @ (*)
        begin
                case (worstCaseDeviationCode)
                        1'b0:
                        begin
                                M1dataIn = 8'h00;
                        end

                        1'b1:
                        begin
                                M1dataIn = dataIn;
                        end
                endcase



                /*
                 *      M1dataIn => worst-case deviation is '10'
                 */
                case (M1dataIn)
                        8'h00,8'h01,8'h02,8'h03,
                        8'h04,8'h05,8'h06,8'h07,
                        8'h08,8'h09,8'h0A        : M1encoderOut = 8'h00;
                        8'h0B                    : M1encoderOut = 8'h01;
                        8'h0C,8'h0D              : M1encoderOut = 8'h03;
                        8'h0E,8'h10,8'h11        : M1encoderOut = 8'h07;
                        8'h0F,8'h12,8'h13,8'h14,
                        8'h15,8'h16,8'h17,8'h18,
                        8'h19                    : M1encoderOut = 8'h0F;
                        8'h1A,8'h1B,8'h1C,8'h1D,
                        8'h1E,8'h20,8'h21,8'h22,
                        8'h23,8'h24,8'h25,8'h26,
                        8'h27,8'h28,8'h29,8'h2A,
                        8'h2B,8'h2C,8'h2D,8'h2E,
                        8'h2F,8'h30,8'h31,8'h32,
                        8'h33,8'h34,8'h35,8'h36,
                        8'h37,8'h38,8'h39,8'h3A,
                        8'h3B,8'h3C,8'h3D,8'h3E,
                        8'h3F,8'h40,8'h41,8'h42,
                        8'h43,8'h44,8'h45,8'h46,
                        8'h47,8'h48,8'h49,8'h4A,
                        8'h4B,8'h4C,8'h4D,8'h4E,
                        8'h4F,8'h50,8'h51,8'h52,
                        8'h53,8'h54,8'h55,8'h56,
                        8'h57,8'h58,8'h59,8'h5A,
                        8'h5B,8'h5C,8'h5D,8'h5E,
                        8'h5F,8'h60,8'h61,8'h62,
                        8'h63,8'h64,8'h65,8'h66,
                        8'h67,8'h68,8'h69,8'h6A,
                        8'h6B,8'h6C,8'h6D,8'h6E,
                        8'h6F,8'h70,8'h71,8'h72,
                        8'h73,8'h74,8'h75,8'h76,
                        8'h77,8'h78,8'h79,8'h7A,
                        8'h7B,8'h7C,8'h7D,8'h7E,
                        8'h7F,8'h80,8'h81,8'h82,
                        8'h83,8'h84,8'h85,8'h86,
                        8'h87,8'h88,8'h89,8'h8A,
                        8'h8B,8'h8C,8'h8D,8'h8E,
                        8'h8F,8'h90,8'h91,8'h92,
                        8'h93,8'h94,8'h95,8'h96,
                        8'h97,8'h98,8'h99,8'h9A,
                        8'h9B,8'h9C,8'h9D,8'h9E,
                        8'h9F,8'hA0,8'hA1,8'hA2,
                        8'hA3,8'hA4,8'hA5,8'hA6,
                        8'hA7,8'hA8,8'hA9,8'hAA,
                        8'hAB,8'hAC,8'hAD,8'hAE,
                        8'hAF,8'hB0,8'hB1,8'hB2,
                        8'hB3,8'hB4,8'hB5,8'hB6,
                        8'hB7,8'hB8,8'hB9,8'hBA,
                        8'hBB,8'hBC,8'hBD,8'hBE,
                        8'hBF,8'hC0,8'hC1,8'hC2,
                        8'hC3,8'hC4,8'hC5,8'hC6,
                        8'hC7,8'hC8,8'hC9,8'hCA,
                        8'hCB,8'hCC,8'hCD,8'hCE,
                        8'hCF,8'hD0,8'hD1,8'hD2,
                        8'hD3,8'hD4,
                        8'hD5                    : M1encoderOut = 8'hCF;
                        8'hD6,8'hD7,8'hD8,8'hD9,
                        8'hDA,8'hDB,8'hDC,8'hDD,
                        8'hDE,8'hDF,8'hE0,8'hE1,
                        8'hE2,8'hE3,8'hE4,8'hE5,
                        8'hE6,8'hE7,8'hE8,8'hE9,
                        8'hEA                    : M1encoderOut = 8'hE0;
                        8'hEB,8'hEC,8'hED,8'hEE,
                        8'hEF,8'hF0,8'hF1,8'hF2,
                        8'hF3,8'hF4              : M1encoderOut = 8'hF0;
                        8'hF5,8'hF6,8'hF7,8'hF8,
                        8'hF9,8'hFA,8'hFB,8'hFC,
                        8'hFD,8'hFE,8'hFF        : M1encoderOut = 8'hFF;
                        default                  : M1encoderOut = 8'h00;
                endcase /* M1dataIn */



                /*
                 *      Select the final output from amongst the sub-encoder blocks.
                 */
                case (worstCaseDeviationCode)
                        1'b0: encoderOut = dataIn;
                        1'b1: encoderOut = M1encoderOut;
                        default : encoderOut = dataIn;
                endcase
        end /* always @ (*) */
endmodule /* e4Encoder_8bitData_1bitControl */
```

Use Yosys to Apply an Input to Combinational Logic


```
$ yosys -p "hierarchy -check; proc; opt; fsm; opt; memory; opt; eval
-set dataIn 8'h33 -set worstCaseDeviationCode 1'b1 -show M1dataIn -show
M1encoderOut -show encoderOut" e4Encoder_8bitData_1bitControl.v
```

Use Yosys to Evaluate a Truth Table for a Combinational Block

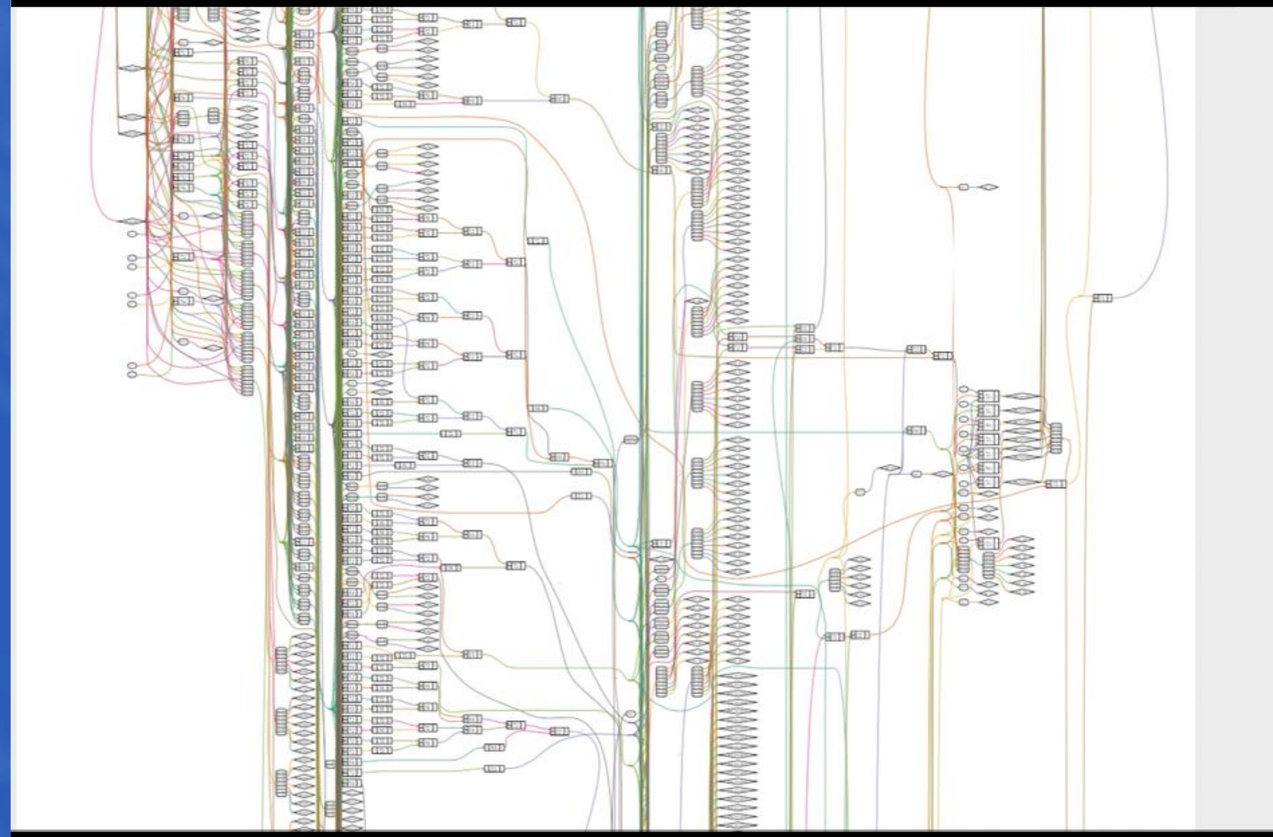
```
$ yosys -p "hierarchy -check; proc; opt; fsm; opt; memory; opt; eval
-table worstCaseDeviationCode,dataIn" e4Encoder_8bitData_1bitControl.v
```

Use Yosys to Check Satisfiability for a Set of Conditions


```
$ yosys -p "hierarchy -check; proc; opt; fsm; opt; memory; opt;
sat -set encoderOut 8'b01111111" e4Encoder_8bitData_1bitControl.v
```

Use Yosys to Visualize Design at the Logic Gate Level

```
$ yosys -p 'hierarchy -check; proc; opt; fsm; opt; memory; opt; techmap; opt;
splitnets -ports; show -lib ./simlib.v -format pdf -prefix
e4Encoder_8bitData_1bitControl-cmos -colors 33 -width -stretch'
e4Encoder_8bitData_1bitControl.v
```



Next, view the generated file `e4Encoder_8bitData_1bitControl-rtl.pdf` using a PDF viewer

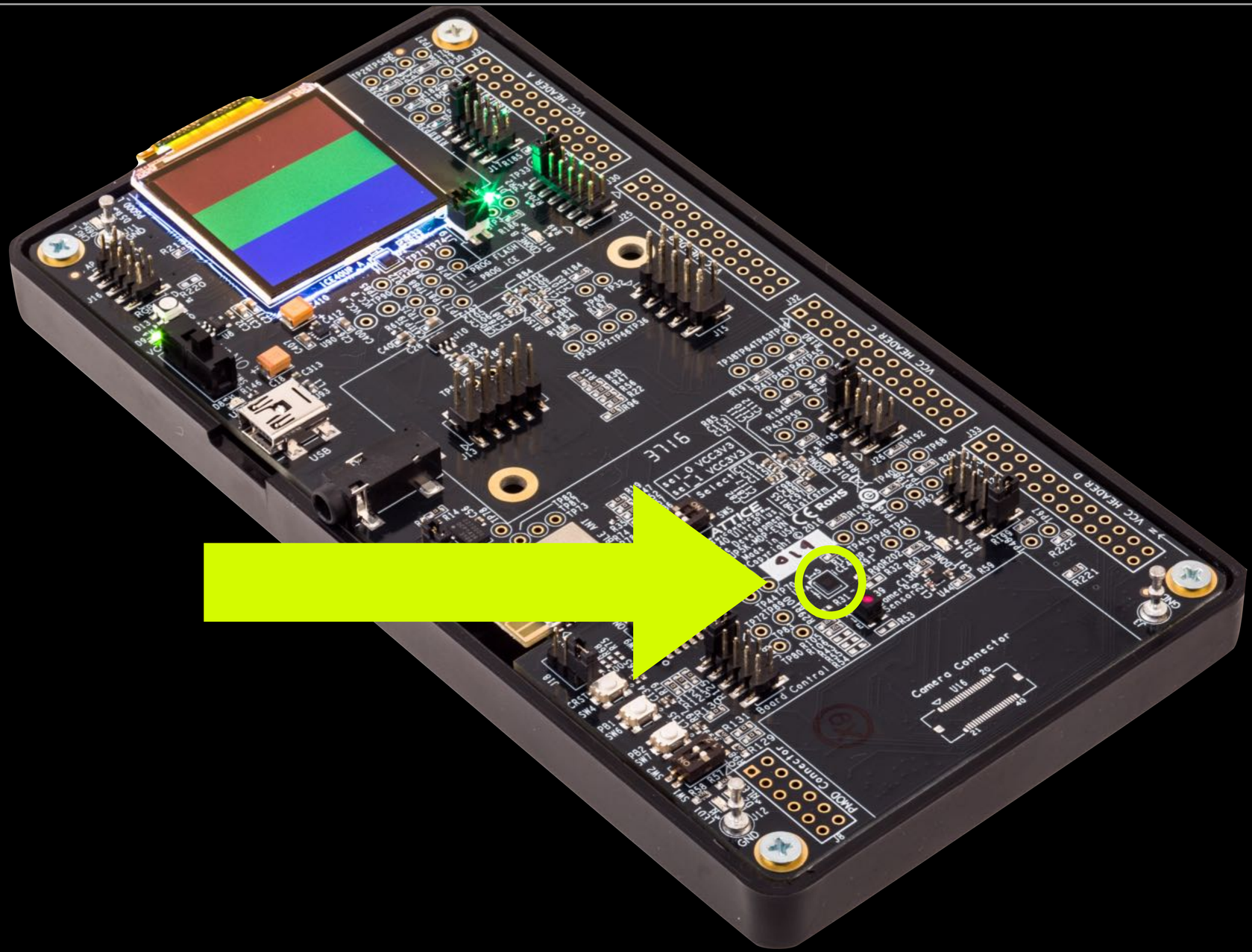
Use Yosys to Visualize Design at the RTL-Block Level

```
$ yosys -p 'hierarchy -check; proc; opt; fsm; opt; memory; opt; show
-format pdf -prefix e4Encoder_8bitData_1bitControl-rtl -colors 33 -width
-stretch' e4Encoder_8bitData_1bitControl.v
```

Next, view the generated file `e4Encoder_8bitData_1bitControl-rtl.pdf` using a PDF viewer

A Hardware Example

Context: The **MDP Evaluation Board** and **iCE40 FPGA**



Example on Hardware: Toggling an I/O Pin in `blink.v`


```
2
3  module blink(led);
4          output          led;
5
6          wire            clk;
7          reg             LEDstatus = 1;
8          reg [31:0]      count = 0;
9
10         /*
11         *       Creates a 48MHz clock signal from
12         *       internal oscillator of the iCE40
13         */
14         SB_HFOSC OSCInst0 (
15                 .CLKHFPU(1'b1),
16                 .CLKHFEN(1'b1),
17                 .CLKHF(clk)
18         );
19
20         /*
21         *       Blinks LED at approximately 1Hz. The constant kFofE_CLOCK_DIVIDER_FOR_1Hz
22         *       (defined above) is calibrated to yield a blink rate of about 1Hz.
23         */
24         always @(posedge clk) begin
25                 if (count > `kFofE_CLOCK_DIVIDER_FOR_1Hz) begin
26                         LEDstatus <= !LEDstatus;
27                         count <= 0;
28                 end
29                 else begin
30                         count <= count + 1;
31                 end
32         end
33
34         /*
35         *       Assign output led to value in LEDstatus register
36         */
37         assign  led = LEDstatus;
38 endmodule
```

The PCF

PCF

- Physical constraints file
- Specifies which signals in top-level are connected to which pins in package

PCF file syntax

`set_io` *<signal name> <package pin name>*

PCF for `blink.v` Example

```
#
#       Pin configuration file (PCF) for place and route
#
#       Sets output led to pin D3 (connected to LED D14) of the
#       version of the package on the MDP board.
#
#       More details on pinouts here:
#
#               http://www.latticesemi.com/en/Products/FPGAandCPLD/iCE40UltraPlus
#
set_io  led D3
```

Example: Connect signal `led[0]` to pin `D3` of the uwg30 package on the MDP board

```
set_io led[0] D3
```

Use ArachnePNR to Place and Route the Design


```
$  yosys -p "synth_ice40 -blif blink.blif; write_json blink.json" blink.v

$  arachne-pnr -d 5k -P uwg30 -p blink.pcf blink.blif -o blink.asc
   --asc blink.asc

$  icetime -p blink.pcf -P uwg30 -d up5k -t blink.asc

$  icepack blink.asc blink.bin

$  sudo iceprog -S blink.bin
```

Use NextPNR to Place and Route the Design


```
$  yosys -p "synth_ice40 -blif blink.blif; write_json blink.json" blink.v

$  nextpnr-ice40 --up5k --package uwg30 --json blink.json --pcf blink.pcf
   --asc blink.asc

$  icetime -p blink.pcf -P uwg30 -d up5k -t blink.asc

$  icepack blink.asc blink.bin

$  sudo iceprog -S blink.bin
```

How Many Gates Does it Take to Blink an LED?

Using a custom circuit implemented di

```
$  yosys -p "synth_ice40 -blif b
            write_json blinkDat

$  nextpnr-ice40 --up5k --packag
                 --pcf blink.pcf
```

Using a C program, which runs on a processor, which is in turn implemented using FPGA logic

```
$  yosys -q ../../../yscripts/sail.ys

$  nextpnr-ice40 --up5k --package uwg30 --json $(DESIGN).json
                 --pcf pcf/sail.pcf --asc sail.asc
```

```c
#include "softwareblink.h"

enum
{
        kSpinDelay = 400000,
};

int
main(void)
{
        /*
         *      Reading from the special address pointed to by
         *      kDebugLedsMemoryMappedRegister will cause the processor to
         *      set the value of 8 of the FPGA's pins to the byte written
         *      to the address. See the PCF file for how those 8 pins are
         *      mapped.
         */
        while (1)
        {
                *kDebugLedsMemoryMappedRegister = 0xFF;

                /*
                 *      Spin
                 */
                for (int j = 0; j < kSpinDelay; j++);

                *kDebugLedsMemoryMappedRegister = 0x00;

                /*
                 *      Spin
                 */
                for (int j = 0; j < kSpinDelay; j++);
        }
```

Things to Do

❶ Complete a **“muddiest point”** 2-question survey using this link