

Advanced FPGA Design Methodologies with Xilinx Vivado



Lecturer: Alexander Jäger

Course of studies: Technische Informatik

Student number: 3158849

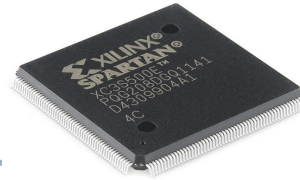
Date: 30.01.2015

Advanced FPGA Design Methodologies with Xilinx Vivado



- **What are FPGAs**
- Fields of applications
- Basic FPGA Design Flow
- Vivado Standard Design Flow
- Incremental Compile
- Test Setup & Results
- Conclusion

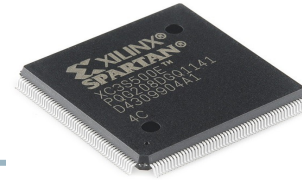
What are FPGAs



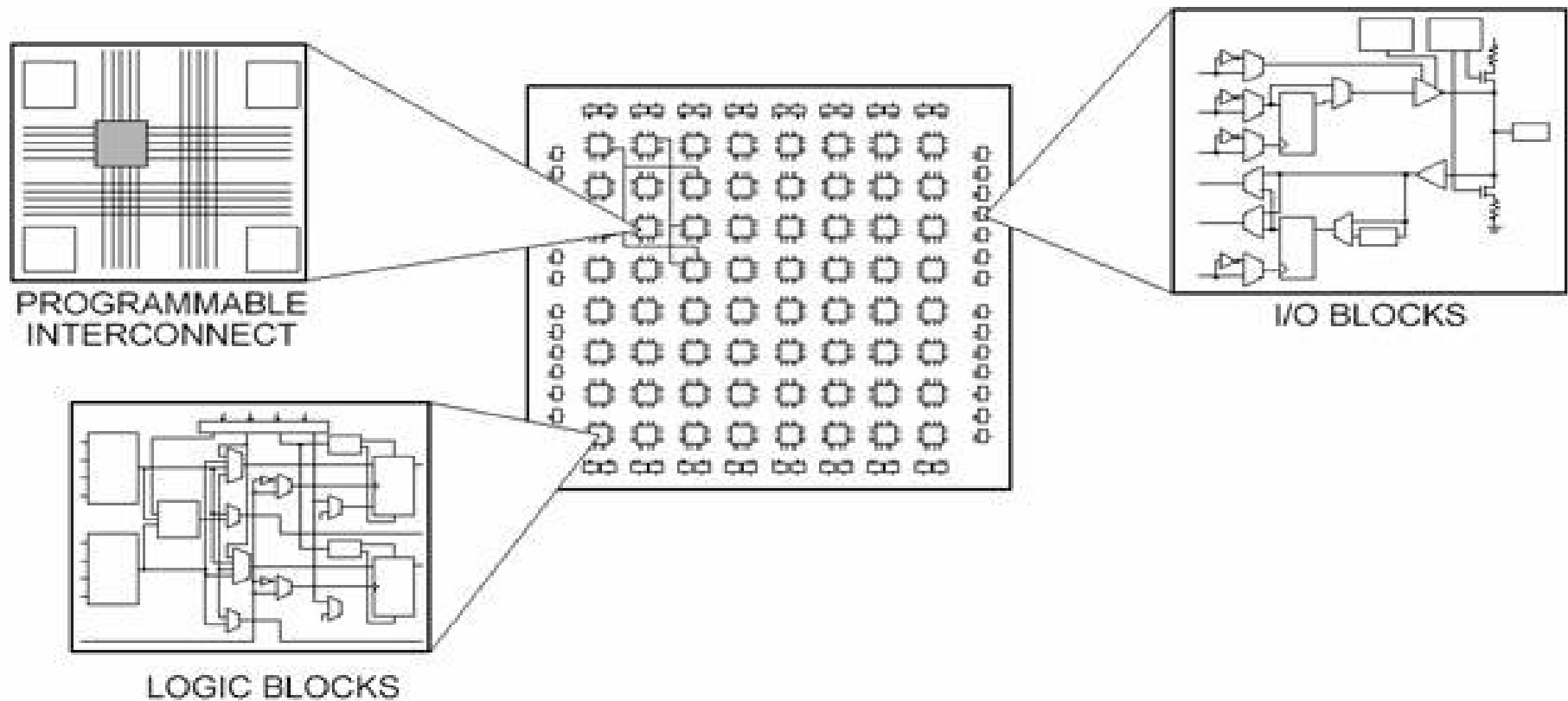
The field-programmable gate array (FPGA) is a semiconductor device that can be programmed **after manufacturing**. Instead of being restricted to any predetermined hardware function, an FPGA allows you to program product features and functions [...]. You can use an FPGA to implement **any logical function** ...

Altera Corporation - FPGAs , <http://www.altera.com/products/fpga.html> , 17.1.15, 19:21

What are FPGAs

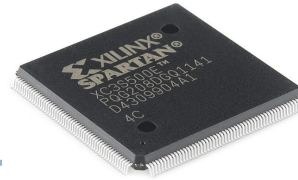


Structure:

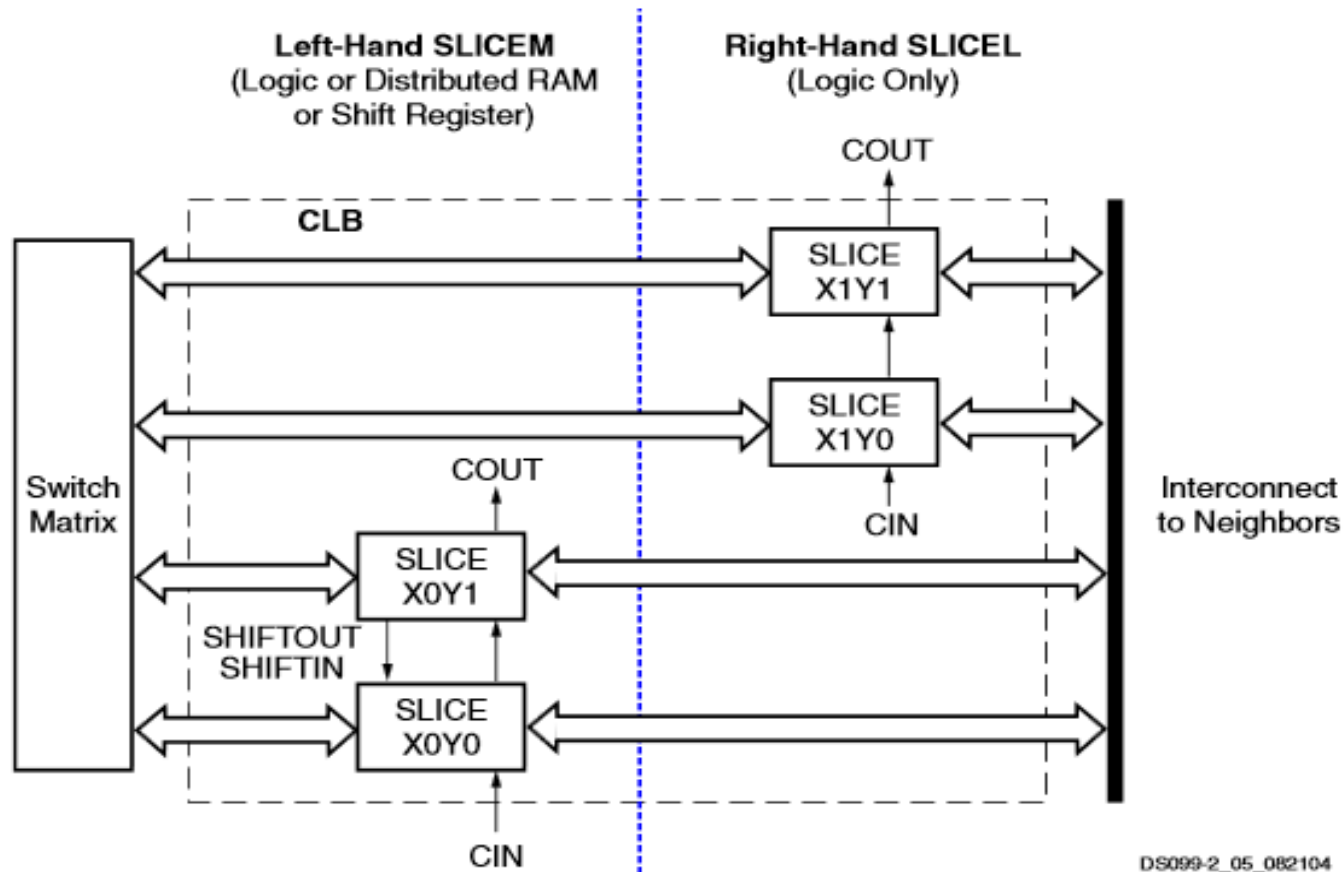


National Instruments Corporation - Wie funktionieren FPGAs?
<http://www.ni.com/white-paper/6983/de/> , 17.1.15, 19:33

What are FPGAs

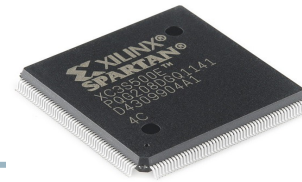


Structure of a logic block:

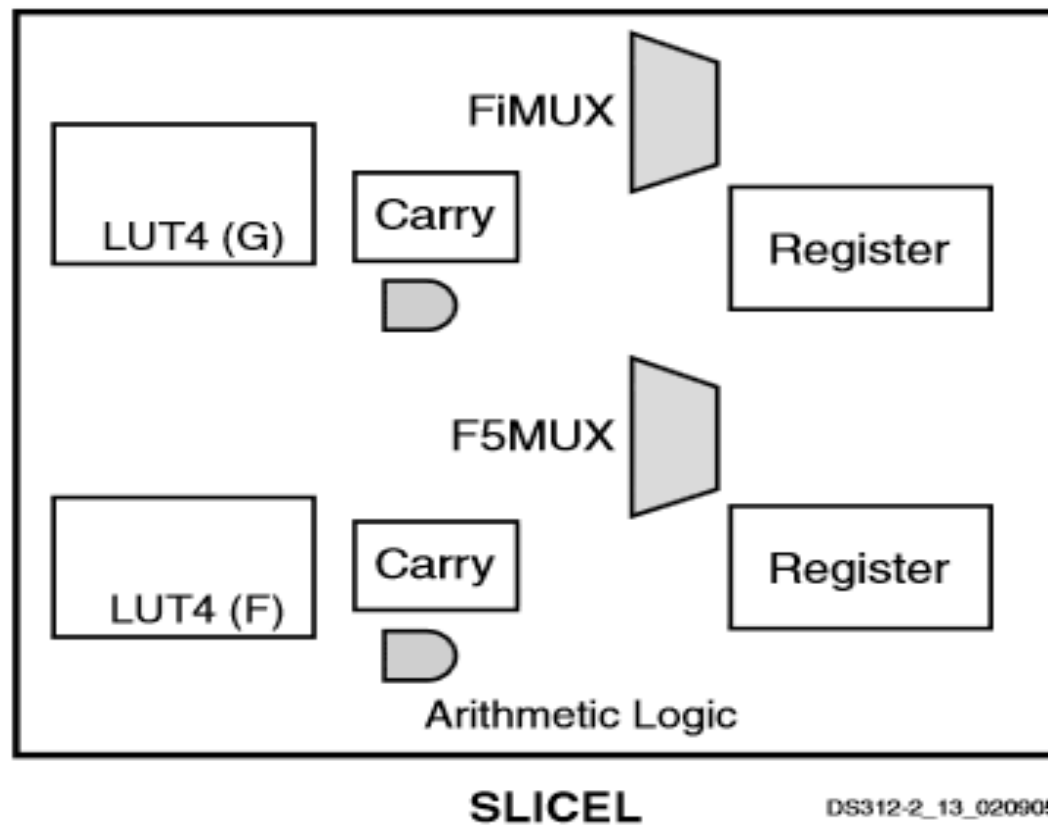


Xilinx Inc. - Spartan-3E FPGA Family Data Sheet, 19.7.2013 ,P.23

What are FPGAs



Structure of a SLICE:



Xilinx Inc. - Spartan-3E FPGA Family Data Sheet, 19.7.2013 ,P.23

Advanced FPGA Design Methodologies with Xilinx Vivado



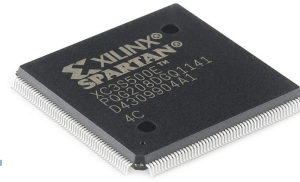
- What are FPGAs
- **Fields of applications**
- Basic FPGA Design Flow
- Vivado Standard Design Flow
- Incremental Compile
- Test Setup & Results
- Conclusion

Fields of applications



- Digital systems with small to medium quantities
- Prototyping of digital systems for evaluation and verification

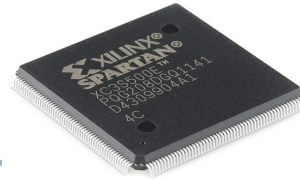
Fields of applications



Reasons:

- All possible digital functions can be implemented
- User programmable
- Easy changes of the Implementation
- No mask costs

Fields of applications



Disadvantages:

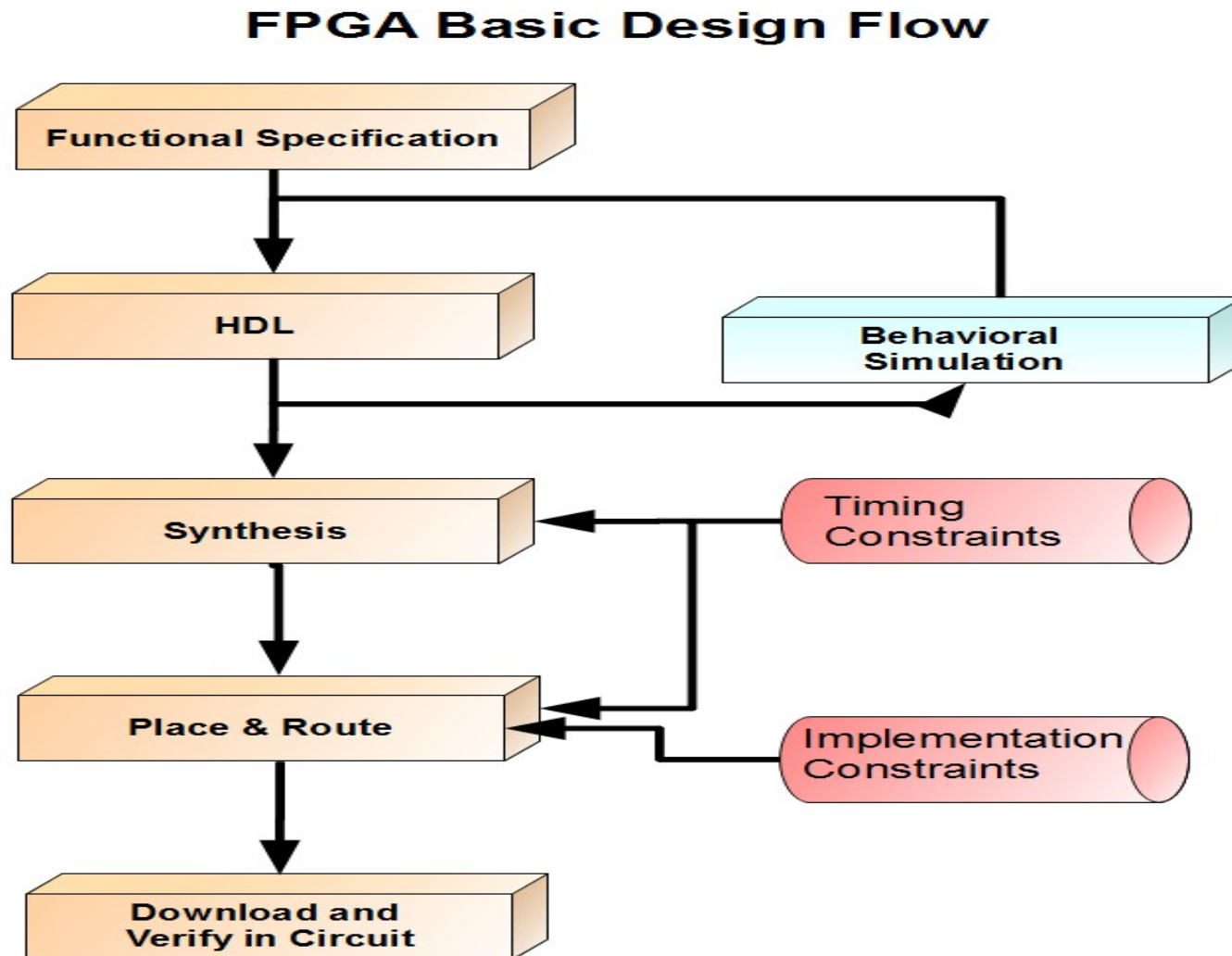
- No (flexible) analogue elements
- FPGA is slower and needs more power than an ASIC with same function
- Price per chip in High Volume Production relatively high

Advanced FPGA Design Methodologies with Xilinx Vivado



- What are FPGAs
- Fields of applications
- **Basic FPGA Design Flow**
- Vivado Standard Design Flow
- Incremental Compile
- Test Setup & Results
- Conclusion

Basic FPGA Design Flow



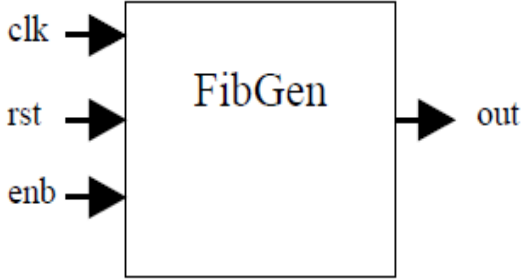
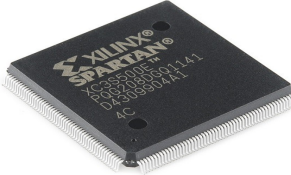
Basic FPGA Design Flow



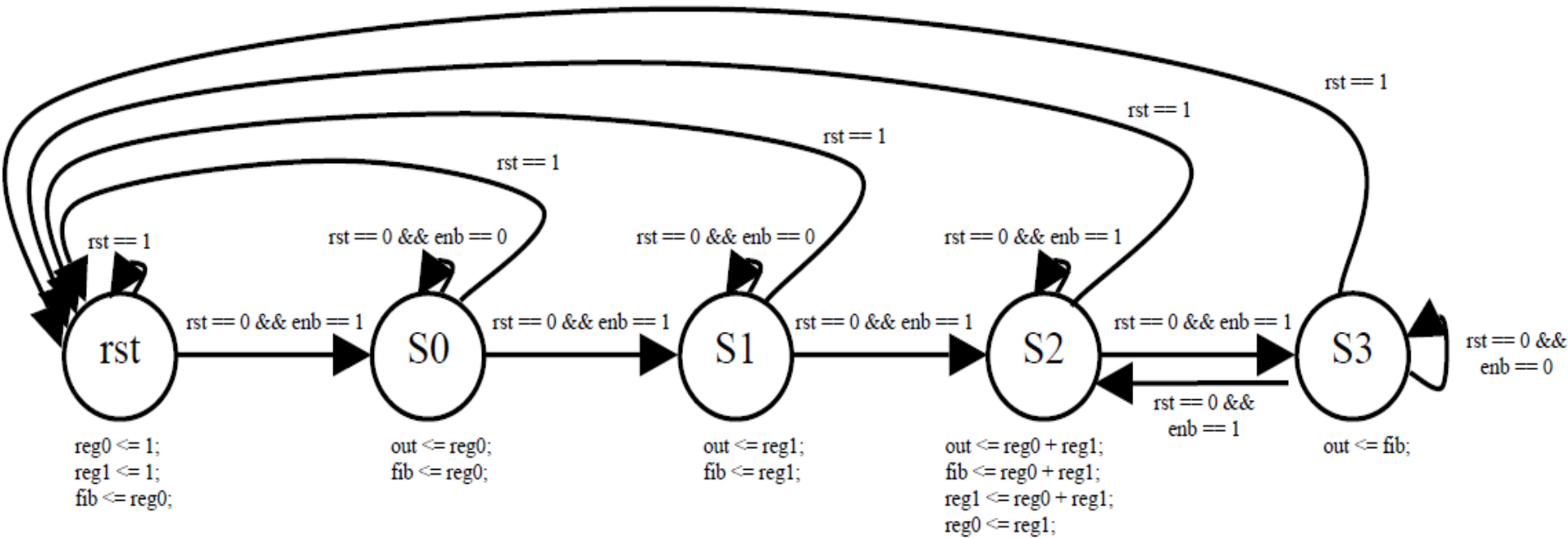
Functional Specification Minimal Example:

- Implement a Fibonacci number generator
- Inputs: Reset (positive), Enable, Clock
- Outputs: 16-Bit – Fibonacci Number
- Target Frequency: 300 Mhz
- Frequency of sampling device: 200 Mhz
- IO-Standard: LVCMOS25

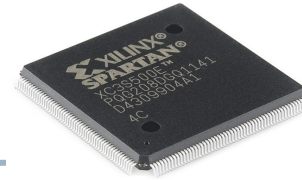
Basic FPGA Design Flow



$rst = 1$ - resets to the beginning of the Fibonacci sequence
 $enb = 1$ - FibGen outputs Fibonacci sequence on every clk cycle
 $enb = 0$ - FibGen stops and outputs the Fibonacci number last outputted

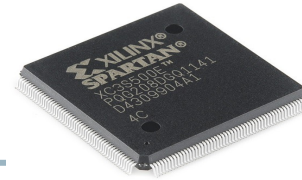


Basic FPGA Design Flow



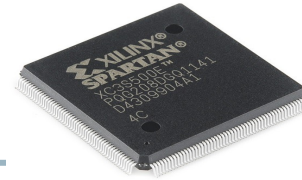
```
28 // -----
29 // Copyright (c) 2006 Susan Lysecky, University of Arizona
30 // Permission to copy is granted provided that this header remains
31 // intact. This software is provided with no warranties.
32 // -----
33
34 module FibGen(clk, rst, enb, out);
35
36     input clk, rst, enb;
37     output [16:0] out;
38     reg [16:0] out;
39
40     // states
41     parameter S0 = 3'b000;
42     parameter S1 = 3'b001;
43     parameter S2 = 3'b010;
44     parameter S3 = 3'b011;
45     parameter S4 = 3'b100;
46
47     // used to initialize registers
48     parameter Zero_16 = 16'b0000000000000000;
49     parameter One_16 = 16'b0000000000000001;
50
51     reg [16:0] reg_0 = Zero_16;
52     reg [16:0] reg_1 = One_16;
53     reg [16:0] fib = Zero_16;
54
55     reg [2:0] State;
56
57     always @ (posedge rst or posedge clk)
58     begin
59         if( rst == 1 )
60             begin
61                 reg_0 = Zero_16;
62                 reg_1 = One_16;
63                 fib = Zero_16;
64
65                 State <= S0;
66                 out <= Zero_16;
67             end
68     end
69
```

Basic FPGA Design Flow



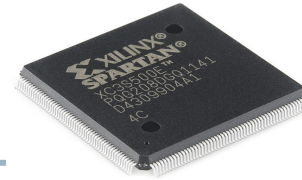
```
73     else
74     begin
75         case( State )
76             S0:
77                 begin
78                     // determine next state
79                     if( enb == 1 )
80                         State <= S1;
81                     else
82                         State <= S0;
83
84                     // assign output value
85                     out <= reg_0;
86                     fib <= reg_0;
87                 end
88
89             S1:
90                 begin
91                     // determine next state
92                     if( enb == 1 )
93                         State <= S2;
94                     else
95                         State <= S1;
96
97                     // assign output value
98                     out <= reg_1;
99                     fib <= reg_1;
100
101                 end
102
103             S2:
104                 begin
105                     // determine next state
106                     if( enb == 1 )
107                         State <= S2;
108                     else
109                         State <= S3;
110
111                     // update values and assign output value
112                     out <= reg_0 + reg_1;
113                     fib <= reg_0 + reg_1;
114                     reg_0 <= reg_1;
115                     reg_1 <= reg_0 + reg_1;
116                 end
```


Basic FPGA Design Flow



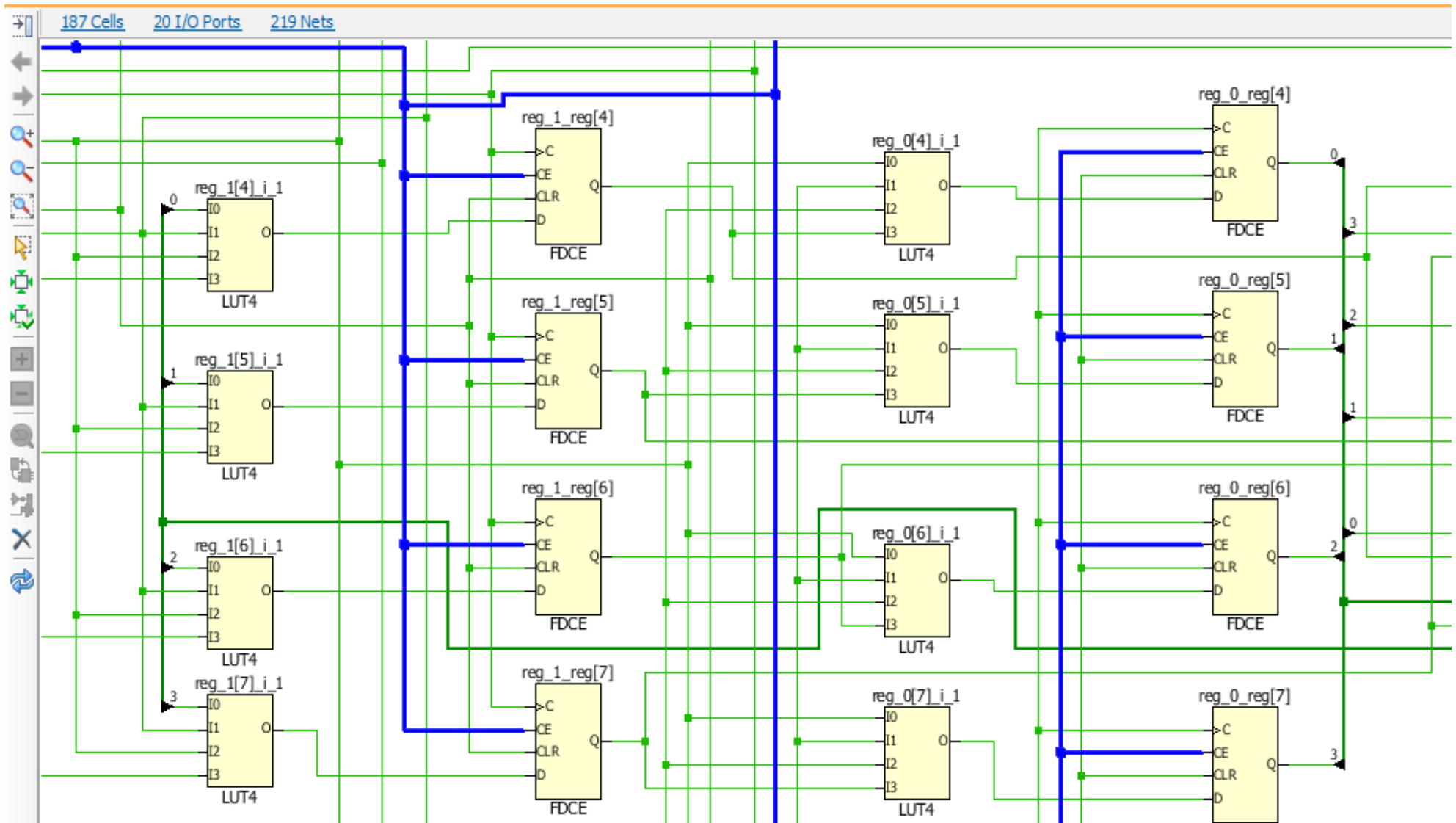
```
13 module Testbench;
14
15     reg clk_t, rst_t, enb_t;
16     wire [16:0] out_t;
17
18     FibGen FibGen_1(clk_t, rst_t, enb_t, out_t);
19
20     always
21     begin
22         clk_t <= 0;
23         #25;
24         clk_t <= 1;
25         #25;
26     end
27
28     initial
29     begin
30
31         // reset
32         rst_t <= 1; enb_t <= 0;
33         #100;
34
35         //case 0
36         rst_t <= 0; enb_t <= 0;
37         #100 $display("out_t = %b", out_t);
38
39         //case 1
40         enb_t <= 1;
41         #500 $display("out_t = %b", out_t);
42
43         //case 2
44         enb_t <= 0;
45         #100 $display("out_t = %b", out_t);
46
47         //case 3:
48         rst_t <= 1; enb_t <= 1;
49         #100 $display("out_t = %b", out_t);
50
51     end
52 endmodule
```

Basic FPGA Design Flow



```
1 create_clock -period 3.333 -name clk -waveform {0.000 1.667} [get_ports clk]
2 create_clock -period 5.000 -name clk_virt -waveform {0.000 2.500}
3 set_clock_groups -name out_clk -asynchronous -group [get_clocks clk_virt]
4
5 set_input_delay -clock [get_clocks clk] -min -add_delay 1.000 [get_ports enb]
6 set_input_delay -clock [get_clocks clk] -max -add_delay 1.500 [get_ports enb]
7 set_input_delay -clock [get_clocks clk] -min -add_delay 1.000 [get_ports rst]
8 set_input_delay -clock [get_clocks clk] -max -add_delay 1.500 [get_ports rst]
9 set_output_delay -clock [get_clocks clk_virt] -min -add_delay 0.900 [get_ports {out[*]}]
10 set_output_delay -clock [get_clocks clk_virt] -max -add_delay 1.000 [get_ports {out[*]}]
11
..
```

Basic FPGA Design Flow

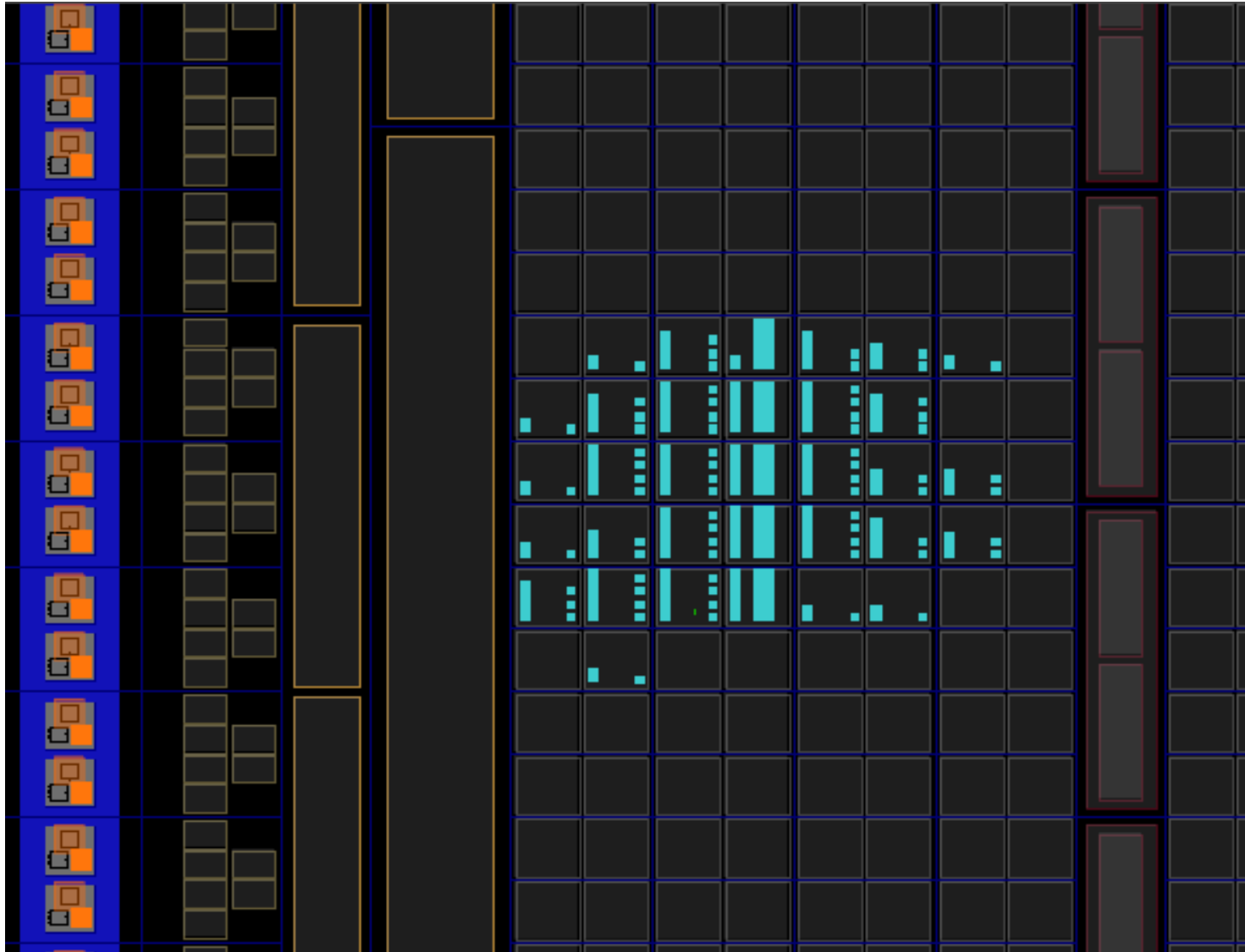
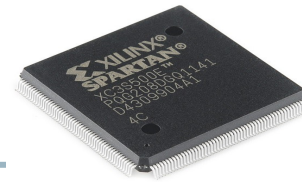


Basic FPGA Design Flow

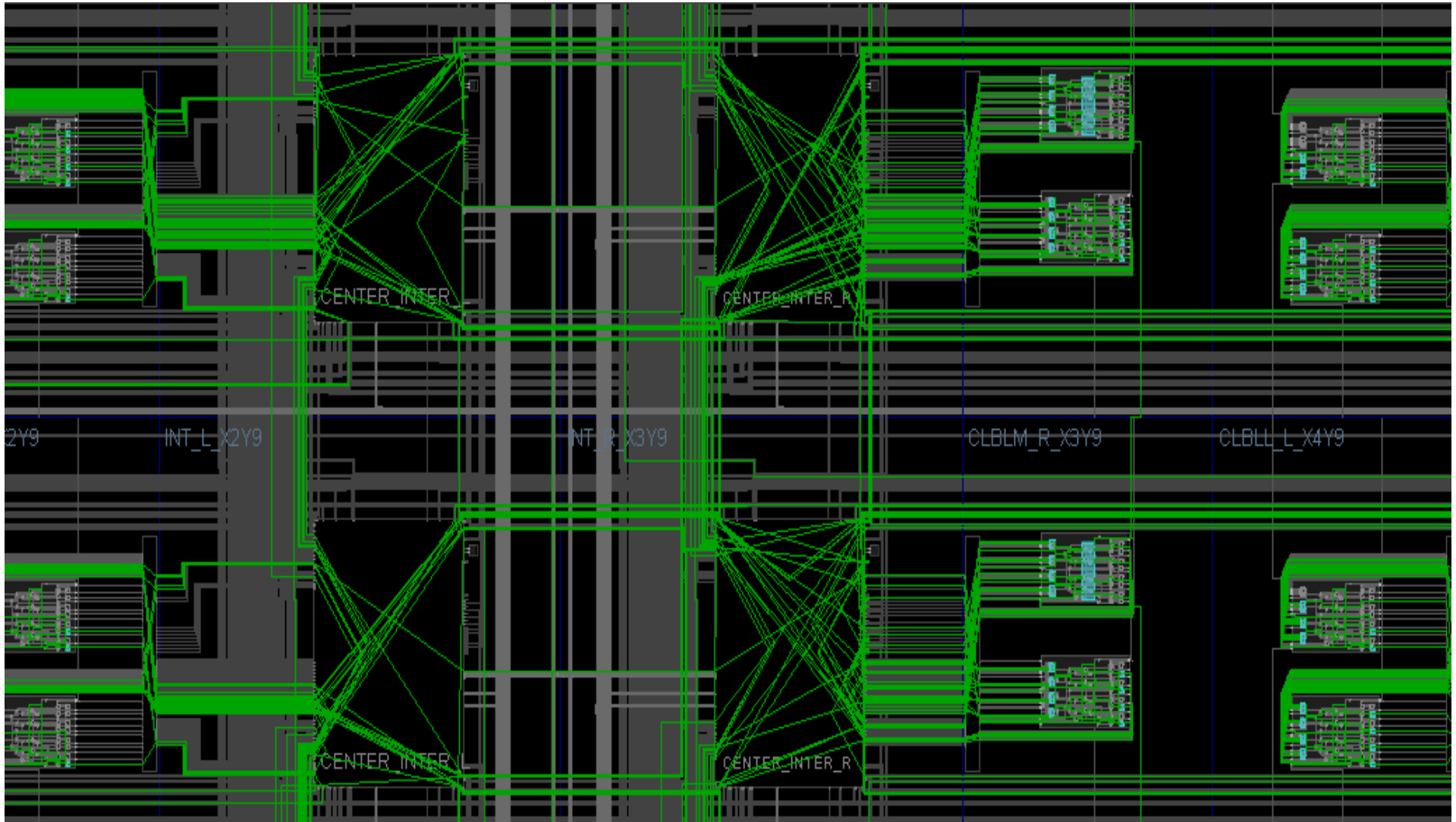
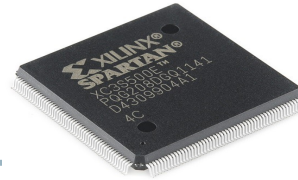


```
11 set_property PACKAGE_PIN R16 [get_ports {out[8]}]
12 set_property PACKAGE_PIN T16 [get_ports {out[7]}]
13 set_property PACKAGE_PIN W16 [get_ports {out[6]}]
14 set_property PACKAGE_PIN Y16 [get_ports {out[5]}]
15 set_property PACKAGE_PIN W14 [get_ports {out[4]}]
16 set_property PACKAGE_PIN Y14 [get_ports {out[3]}]
17 set_property PACKAGE_PIN V15 [get_ports {out[2]}]
18 set_property PACKAGE_PIN W15 [get_ports {out[1]}]
19 set_property PACKAGE_PIN T15 [get_ports {out[0]}]
20 set_property PACKAGE_PIN U15 [get_ports rst]
21 set_property IOSTANDARD LVCMOS25 [get_ports {out[16]}]
22 set_property IOSTANDARD LVCMOS25 [get_ports {out[15]}]
23 set_property IOSTANDARD LVCMOS25 [get_ports {out[14]}]
24 set_property IOSTANDARD LVCMOS25 [get_ports {out[13]}]
25 set_property IOSTANDARD LVCMOS25 [get_ports {out[12]}]
26 set_property IOSTANDARD LVCMOS25 [get_ports {out[11]}]
27 set_property IOSTANDARD LVCMOS25 [get_ports {out[10]}]
28 set_property IOSTANDARD LVCMOS25 [get_ports {out[9]}]
```

Basic FPGA Design Flow



Basic FPGA Design Flow

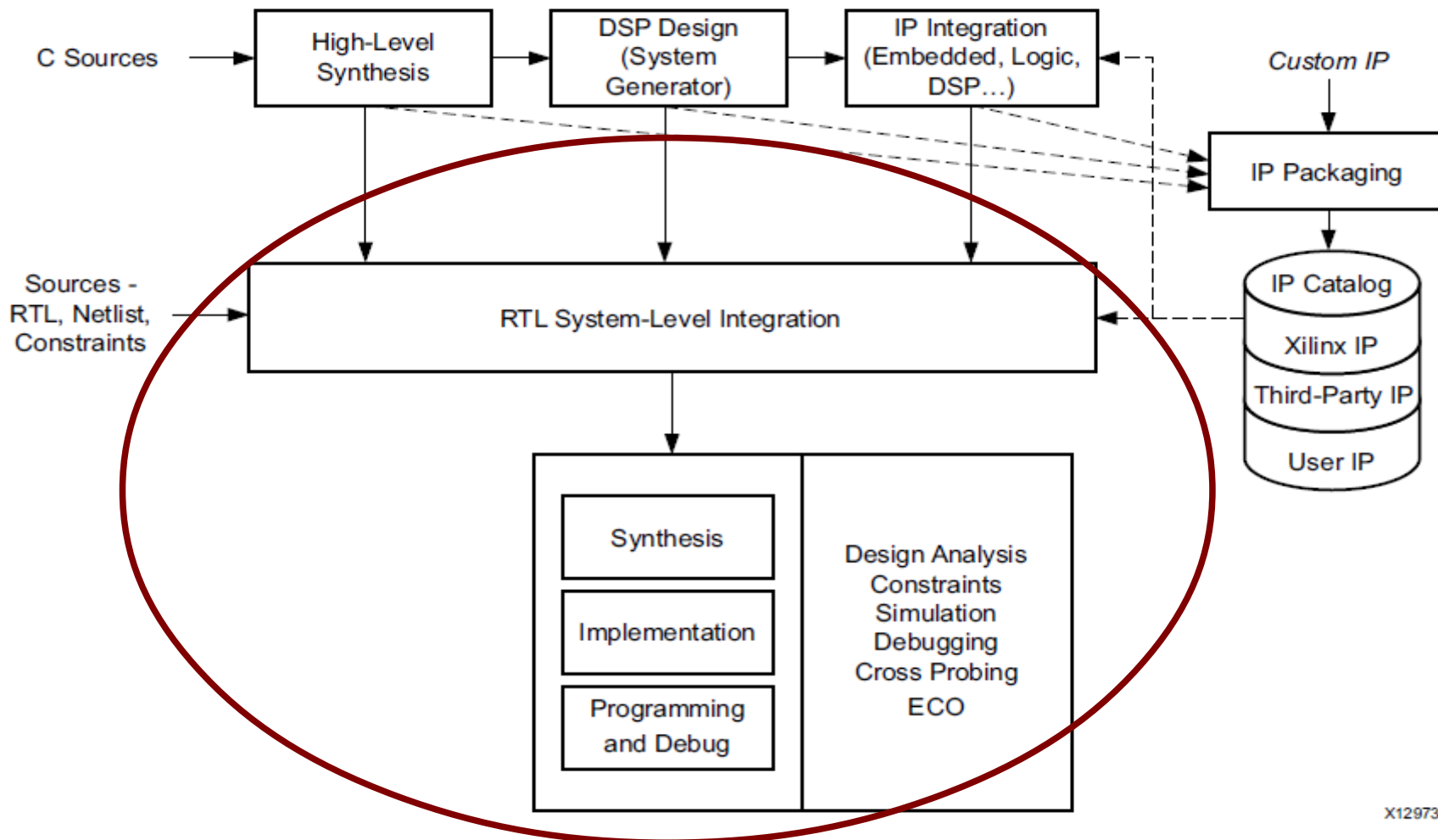


Advanced FPGA Design Methodologies with Xilinx Vivado



- What are FPGAs
- Fields of applications
- Basic FPGA Design Flow
- **Vivado Standard Design Flow**
- Incremental Compile
- Test Setup & Results
- Conclusion

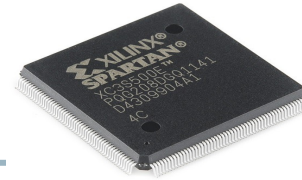
Vivado Standard Design Flow



X12973

Xilinx Inc. - Vivado Design Suite User Guide Design Flows Overview, 1.10.14, P.6

Vivado Standard Design Flow



Implementation Complete

Netlist

- vc709_top
 - Nets (2019)
 - Leaf Cells (95)
 - clk_gen_I (clk_gen_200_to_i2c)
 - dbg_hub (dbg_hub_CV)
 - hmc_controller_instance (hmc_controller_top)
 - i2c_slave (i2c_slave_wrapper)
 - pattern_gen_top_instance (pattern_gen_top)
 - res_rf_I (res_rf)

Implementation Run Properties

Name: impl_9_entries96
 Part: xc7vx690tffg1761-2 (active)
 Description: Vivado Implementation Defaults

Design Runs

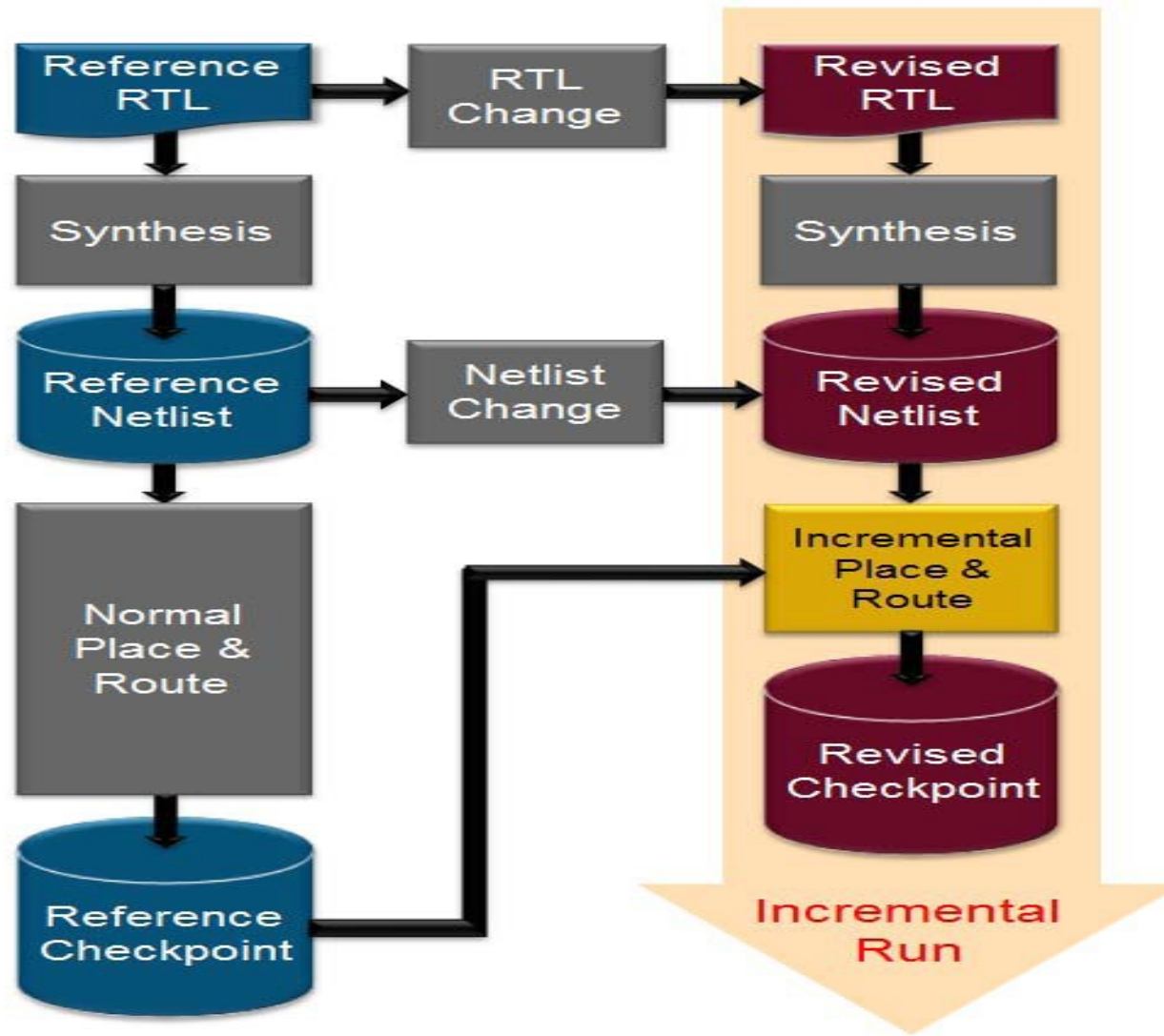
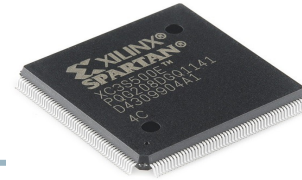
Name	Constraints	WNS	TNS	WHS	THS	TPWS	Failed Routes	LUT	FF	BRAM	DSP	Start
impl_8_entries256	constrs_1	0.06	0.00	0.01	0.00	0.00	0	5.31	3.07	17.14	0.00	1/5/15 6:4
synth_8_entries256_ic	constrs_1							4.28	2.02	1.09	0.00	1/5/15 4:5
impl_8_entries256_ic	constrs_1	0.10	0.00	0.05	0.00	0.00	0	5.31	3.07	17.14	0.00	1/5/15 8:0
synth_9_entries96	constrs_1							4.15	1.79	1.09	0.00	1/6/15 11:3
impl_9_entries96	constrs_1	0.11	0.00	0.01	0.00	0.00	0	5.17	2.85	17.14	0.00	1/6/15 11:4
synth_9_entries96_ic (active)	constrs_1							4.15	1.79	1.09	0.00	1/6/15 12:21
impl_9_entries96_ic (acti...)	constrs_1	0.10	0.00	0.06	0.00	0.00	0	5.17	2.85	17.14	0.00	1/6/15 12:31
Out-of-Context Module Runs												
clk_gen_200_to_i2c_synth_1	clk_gen_200_to_i2c							0.00	0.00	0.00	0.00	11/4/14 4:1
ila_1_synth_1	ila_1							0.95	0.77	16.05	0.00	11/4/14 4:2
hmc_transceiver_8x_synth_1	hmc_transceiver_8x							0.54	0.28	0.00	0.00	11/17/14 10:4

Advanced FPGA Design Methodologies with Xilinx Vivado



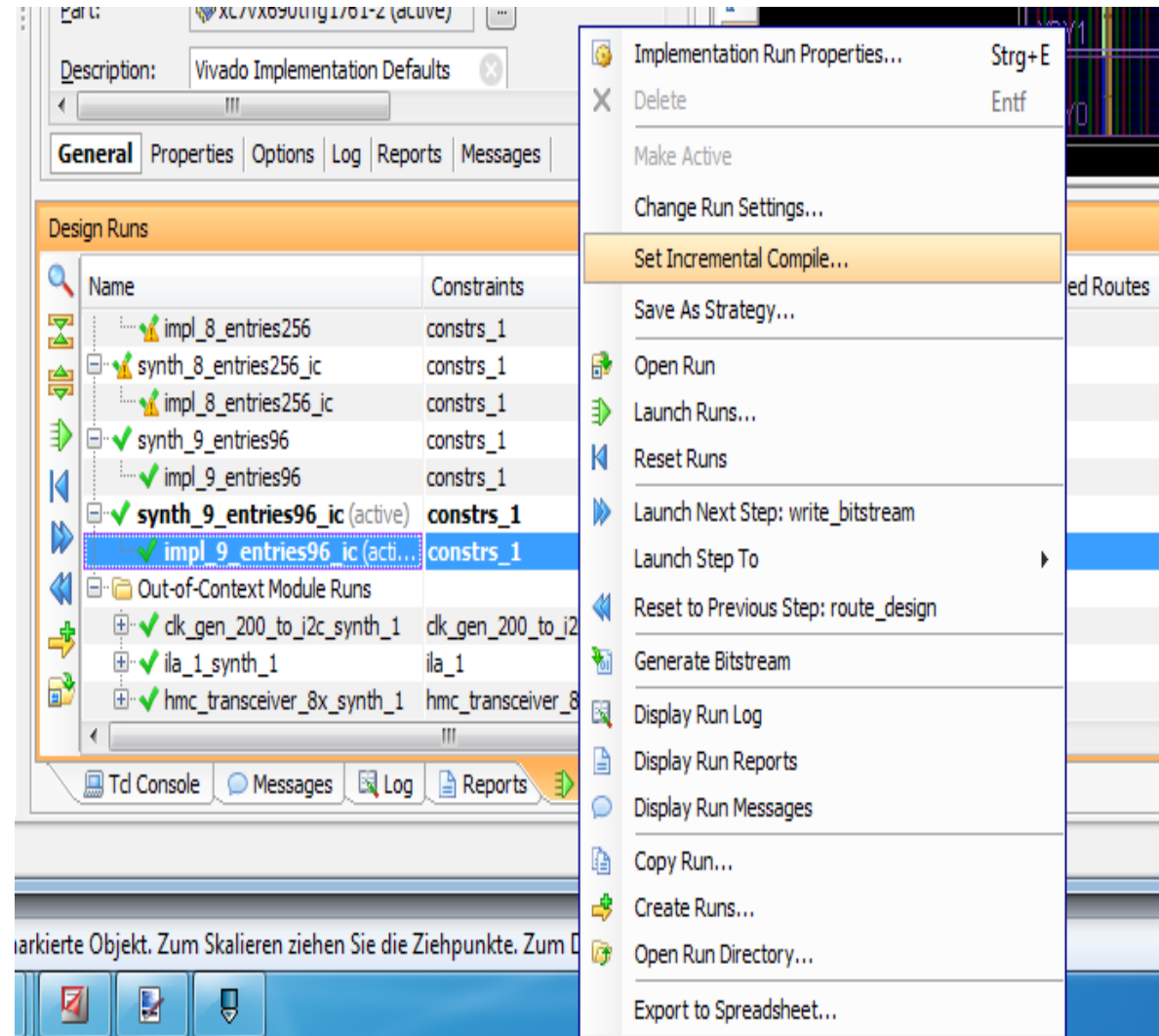
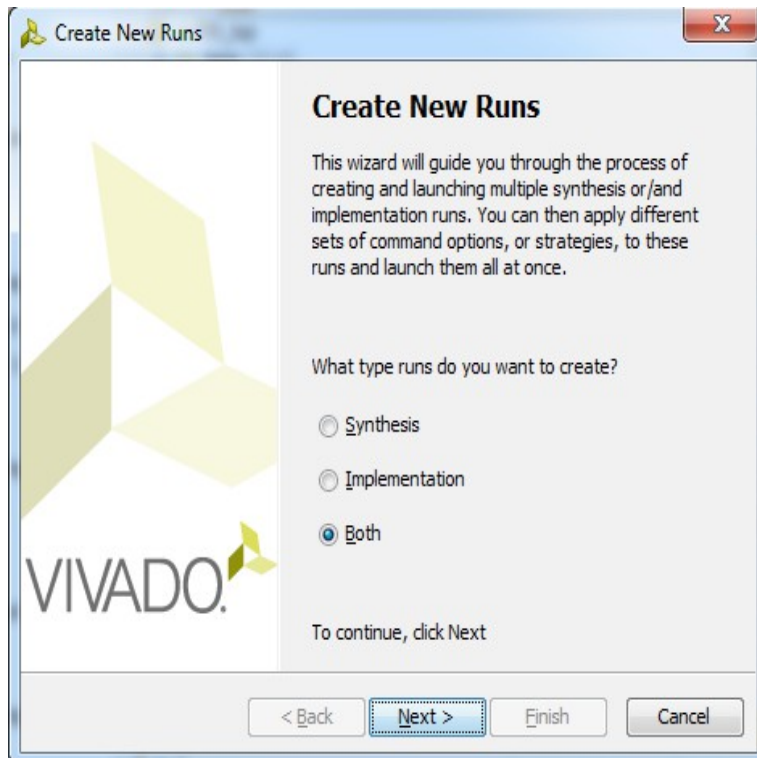
- What are FPGAs
- Fields of applications
- Basic FPGA Design Flow
- Vivado Standard Design Flow
- **Incremental Compile**
- Test Setup & Results
- Conclusion

Incremental Compile



Xilinx Inc. -
Vivado Design Suite
User Guide
Implementation
15.10.14
P.83

Incremental Compile



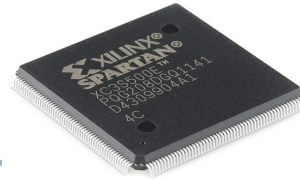
Incremental Compile



The screenshot displays the Vivado IDE interface. A 'Set Incremental Compile' dialog box is open, showing the 'Use checkpoint:' field with a file path: `_pci_800mv/vc709_1hmc_8lane/vc709_1hmc_8lane.runs/impl_2/vc709_top_routed.dcp`. A red circle highlights the file selection icon (three dots) next to the path. Below this, a 'Vivado Checkpoint Files' dialog box is open, showing the 'Look in:' field set to 'impl_2', which is also circled in red. The file list shows `vc709_top_routed` selected. The file preview shows details for `vc709_top_routed.dcp`, including its directory, creation and modification dates (Friday 19.12.14 12:03 PM), size (12.1 MB), type (Checkpoint design), and owner (Bubu-PC/Jäsch).

AM	DSP
17.14	0.0
1.09	0.0
17.14	0.0
1.09	0.0
17.14	0.0
1.09	0.0
17.14	0.0

Incremental Compile



Short Summary:

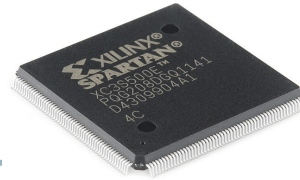
- A minimum of 85% match between original and new netlist required
- Design Checkpoint from previous Implementation needed
- Checkpoint can be (partially) placed or (partially) placed and (partially) routed

Advanced FPGA Design Methodologies with Xilinx Vivado



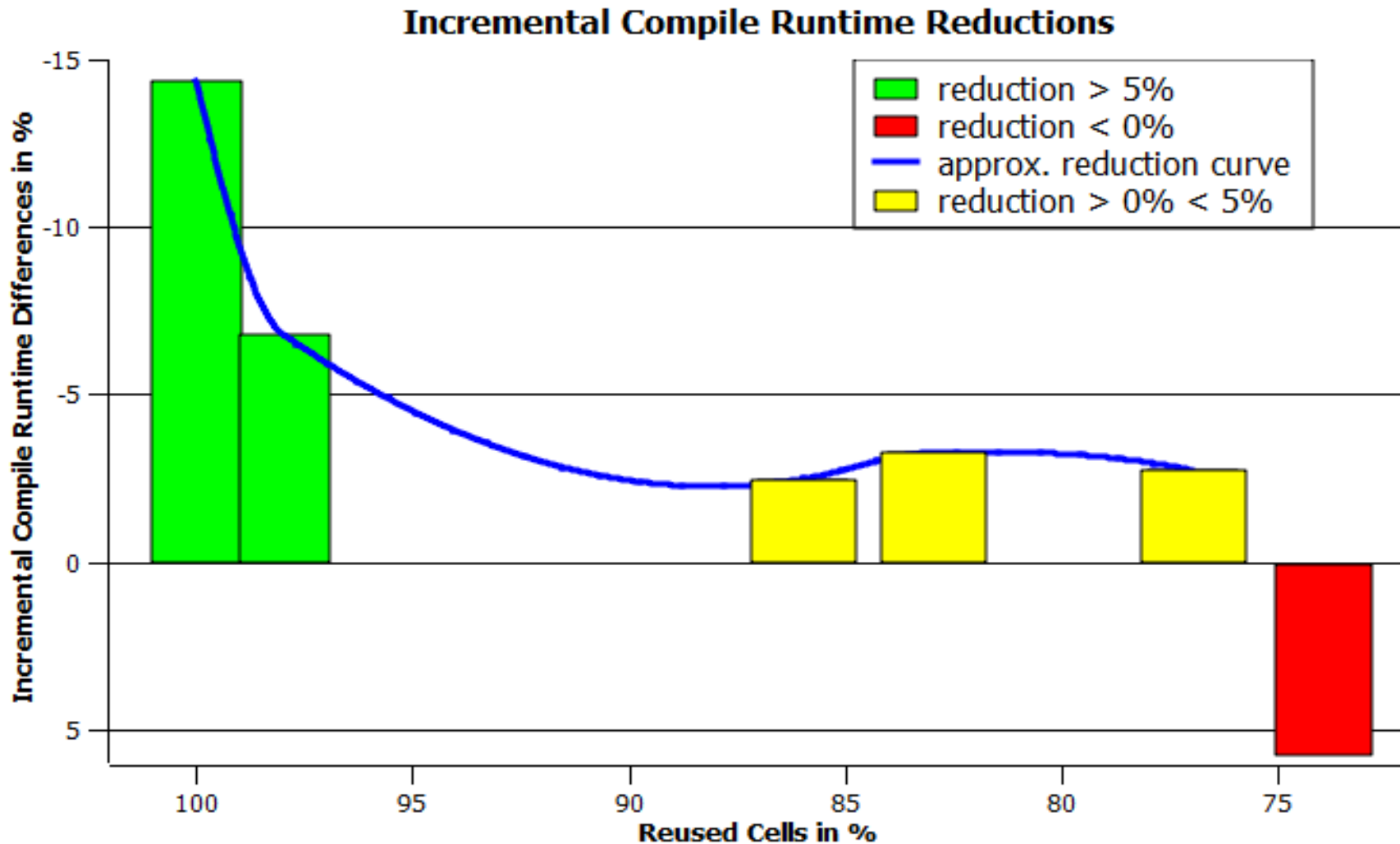
- What are FPGAs
- Fields of applications
- Basic FPGA Design Flow
- Vivado Standard Design Flow
- Incremental Compile
- **Test Setup & Results**
- Conclusion

Test Setup & Results

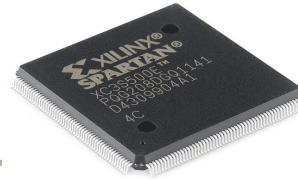


- Starting Point: Fully implemented design with checkpoints
- List of small to big changes
- 2 runs for every list entry – One with Standard Flow, One with Incremental Compile
- Compare: Runtime, Timing, Resource Utilization

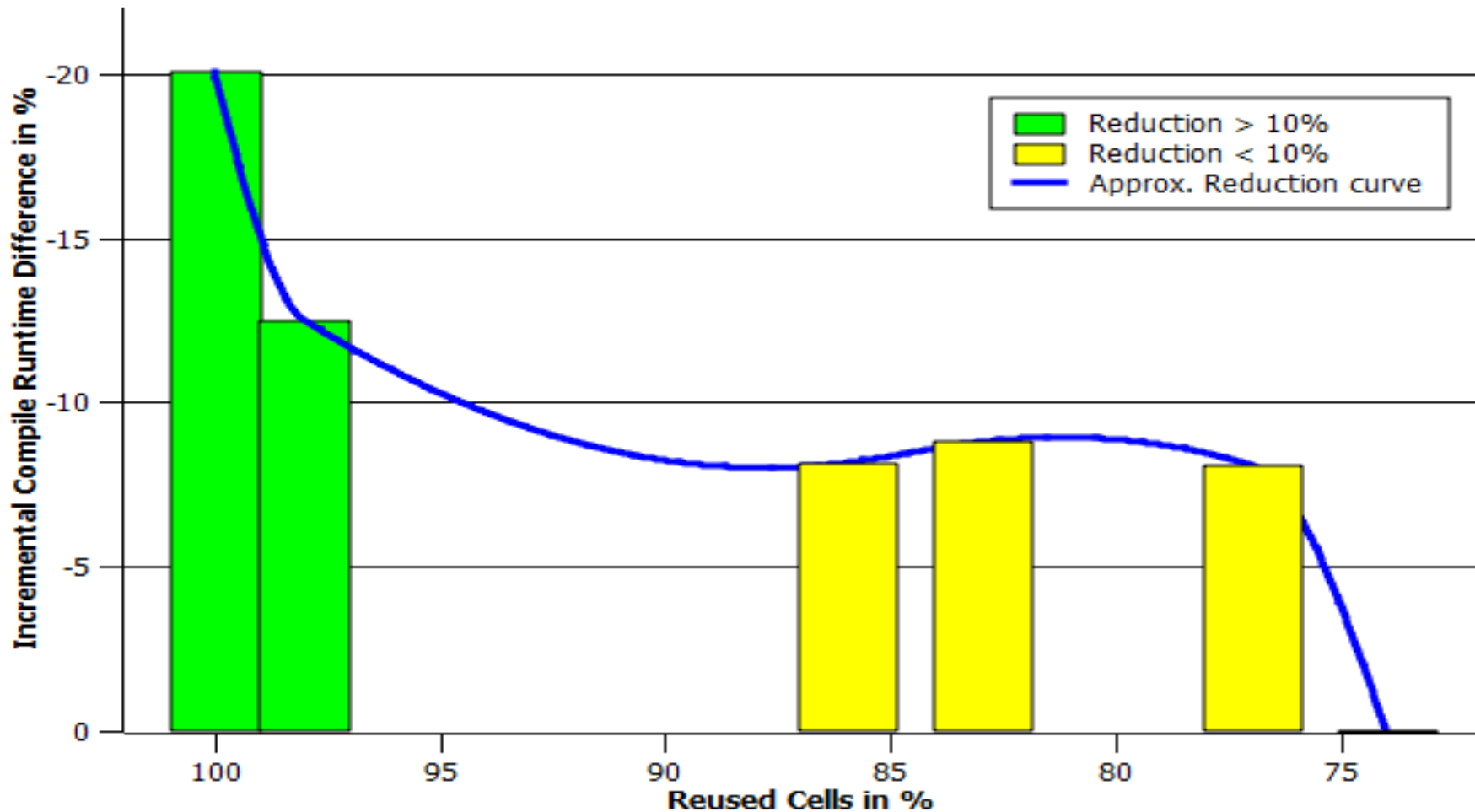
Test Setup & Results



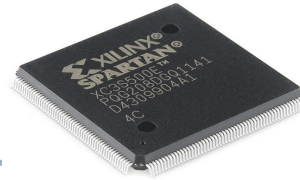
Test Setup & Results



Calculated Theoretical Runtime Reductions



Test Setup & Results



Runtime Facts:

- Best measured runtime reduction: 21.22 %
- Best theoretical reduction: 26.81 %
- Average runtime reduction: 6.19 %
- Additional runtime inducted through Incremental Compile: 1:45 min

Test Setup & Results



Timing results:

- No influence, all required timings met

Test Setup & Results



Resource utilization:

- No influence on used Block Ram and Slice Registers
- Utilization of Slice LUTs, LUT FF-Pairs and used Slices stayed same or dropped a bit

Advanced FPGA Design Methodologies with Xilinx Vivado



- What are FPGAs
- Fields of applications
- Basic FPGA Design Flow
- Vivado Standard Design Flow
- Incremental Compile
- Test Setup & Results
- **Conclusion**

Conclusion

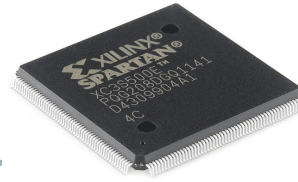


Incremental Compile:

- Easy to use
- Overall small runtime reductions
- Sometimes small resource utilization reductions

=> Only minor improvements, still recommended for usage

Discussion



Any Questions?

Xilinx IC: http://en.wikipedia.org/wiki/Xilinx#mediaviewer/File:Xilinx_Spartan-3E_%28XC3S500E%29.jpg