Open Source FPGA toolchains

FINAL REPORT

May 11, 2019

Ilias Daradimos

Contents

1 Scope 3

2 Platforms Tested 4

3 Tool-chains 5

3.1 IceStorm (iCE40) 5

3.2 Trellis (ECP5) 5

3.3 NextPNR 6

3.4 X-Ray (Xilinx 7) 6

3.5 Symbiflow 6

3.6 IceStudio 6

3.7 APIO 7

3.8 Intel Quartus Prime Lite Edition and ModelSim - Intel FPGA Starter Edition 8

3.9 CubicBoard 8

3.10 Vivado 8

4 Programming tasks 10

4.1 Simple programming tasks 10

4.2 Advanced programming tasks 11

5 Evaluation 12

5.1 Tool-chain capabilities 12

5.2 icestorm and prjtrellis 13

5.3 APIO 16

5.4 IceStudio 17

5.5 Quartus 19

5.6 Vivado 20

6 Version control 21
6.1 iCE40 21
6.2 ECP5 22
6.3 Vivado 22
6.4 Quartus 22

7 Performance 23

8 Conclusion 25

Chapter 1

Scope

The main scope of this activity is to act as a foundation for FPGA related sub-activities by testing available open source tool chains for FPGA programming with a selection of popular FPGA platforms.

Each tool chain will be evaluated against supported FPGA platforms in order to determine its capabilities. Focus is given on open source tool-chains while for platforms not yet supported by open source project, free or free license alternatives are explored.

During this process a basic set of FPGA programming tasks will be performed and evaluated. Simple tasks are used to evaluate tool chains on their simplest usage allowing the creation of common tasks across all the tool-chains. This will help evaluation by having a common reference design.

Chapter 2

Platforms Tested

For the purpose of this evaluation, the platforms selected should have at least 7K LUTs and having active or under development open-source tool-chains or, as a last resort, free toolchains.

1. Lattice ICE40
   Devboard: TinyFPGA BX
   FPGA: ICE40LP8K

2. Lattice ECP5
   Devboard: ECP5 Evaluation Board
   FPGA: LFE5UM5G-85F-8BG381

3. Xilinx Artix-7
   Devboard: Digilent Arty A7-100T Development Board
   FPGA: XC7A100T-1CSG324C

4. Intel Cyclone-V
   Devboard: DE0-CV
   FPGA: Cyclone V 5CEBA4F23C7N

Chapter 3

Tool-chains

3.1 IceStorm (iCE40)

IceStorm [1] is used for iCE40 bit-stream creation. It relies on yosys for synthesis and NextPNR for place and route.

Supports all package variants of LP1K, LP4K, LP8K and HX1K, HX4K, HX8K, LP384 and UltraPlus devices.

Does not support iCE40 LM, Ultra and UltraLite

3.2 Trellis (ECP5)

Project Trellis enables a fully open-source flow for ECP5 FPGAs using Yosys for Verilog synthesis and nextpnr for place and route. Project Trellis itself provides the device database and tools for bit-stream creation.

The following features are currently working in the Yosys -> nextpnr -> Trellis flow.

- Logic slice functionality, including carries
- Distributed RAM inside logic slices
- All internal interconnect
- Basic IO, including tristate, using TRELLIS_IO primitives. LPF files and DDR inputs/outputs
- Block RAM, using either inference in Yosys or manual instantiation of the DP16KD primitive

- Multipliers using manual instantiation of the MULT18X18D primitive. Inference and more advanced DSP features are not yet supported.
- Global networks (automatically promoted and routed in nextpnr)
- PLLs
- Transcievers (DCUs)

3.3 NextPNR

NextPNR is a portable FPGA place and route tool that aims to be a vendor neutral, timing driven, FOSS FPGA place and route tool.

Currently NextPNR supports:

- Lattice iCE40 devices through project IceStorm
- Lattice ECP5 devices through project Trellis, currently experimental
- "generic" back-end for user-defined architectures, currently experimental
- Xilinx 7 Series could be supported in the future through project X-Ray

3.4 X-Ray (Xilinx 7)

Project X-Ray documents the Xilinx 7-Series FPGA architecture to enable development of open-source tools.

3.5 Symbiflow

Symbiflow aims to be an open source flow for generating bit-streams from Verilog.

Since it uses Yosis, icestorm, trellis, xray and nextpnr for each of the supported platforms, device support is at the same level as the underlying projects.

3.6 IceStudio

IceStudio is a visual editor for open FPGA boards. Built on top of the Icestorm project using Apio.

It implements Graphic design -> Verilog, PCF -> Bistream -> FPGA
It supports the following devices:

- HX1K
- HX8K
- LP8K
- UP5K

It is the easiest to setup as it handles all the tool-chain and device drivers downloads and it comes as an AppImage.

3.7 APIO

APIO is an Open source ecosystem for open FPGA boards inspired by PlatformIO It is built on top of Icestorm project and currently supports the ICE40 FPGA family A list supported boards is available on their github page, these include the following ICE40 families:

- HX1K
- HX8K
- LP8K
- UP5K

Documentation page: `http://apiodoc.readthedocs.io/`
GitHub Repo: `https://github.com/FPGAwars/apio`
APIO comes with a set of packages:

- Icestorm for iCE40 FPGA synthesis, place & route and configuration tools
- Simulation viewer. GTKWave project (only for Windows)
- Verilog simulation and synthesis tool. Icarus Verilog project
- Verilog HDL simulator. Verilator project

APIO also has an Atom editor package called apio-ide. Most of APIO functionality is available from related Atom menu items

- Verify
- Build
- Upload
- Simulation
- Time analysis

3.8 Intel Quartus Prime Lite Edition and ModelSim - Intel FPGA Starter Edition

Intel Quartus Prime is programmable logic device design software produced by Intel; prior to Intel's acquisition of Altera the tool was called Altera Quartus II. Quartus Prime enables analysis and synthesis of HDL designs, which enables the developer to compile their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer. Quartus Prime includes an implementation of VHDL and Verilog for hardware description, visual editing of logic circuits, and vector waveform simulation.[2]

3.9 CubicBoard

Cubicboard claims to be an open-source FPGA project for the Cyclone-V family. They do provide Open Hardware under Apache 2.0 license although source files are for Protel/Altium EDA. The provided software is a VirtualBox image of Ubuntu having Intel Quartus preinstalled.

3.10 Vivado

This is the official software from Xilinx. The HL WEBPack edition does not require a license and there is a Linux version.

Vivado Design Suite is a software suite produced by Xilinx for synthesis and analysis of HDL designs, superseding Xilinx ISE with additional features for system on a chip development and high-level synthesis. Vivado represents a ground-up rewrite and re-thinking of the entire design flow (compared to ISE), and has been described by reviewers as "well conceived, tightly integrated, blazing fast, scalable, maintainable, and intuitive"[3]

Chapter 4

Programming tasks

A set of programming tasks was planned in order to evaluate each platform. Tasks vary from simple tasks like LED blinking and counters to more advanced tasks like LVDS transceivers.

For each tasks the design complexity as well as the total time from source to bit-stream were recorded. Details regarding tool-chain performance can be found in chapter Performance

Verilog is used as a programming language although block design was preferred on tool-chains providing a block design editor.

A git repository was created in GitLab hosting all the code used in this activity. The repository is organised in per tool-chain folders with each folder containing implemented tasks.

4.1 Simple programming tasks

Simple tasks are used for having a similar design across all platforms. This helps to investigate the complexity required by each tool-chain to achieve the same task. For that reason a simple LED blinking task and a counter task was implemented. Small variations where introduced due to differences in the available development boards.

Project repository[4] is organised in per toolchain folders each containing implementation of the same task.

Under each tool-chain folder there is a LED_Blink and SimpleCounter projects, these are the basic reference project for each evaluation. An additional GeekCounter project is included for devboards having 7-segment displays. Detailed information can be found on each project readme.md file

4.2 Advanced programming tasks

Moving to a more advanced programming task, an LVDS transceiver was to be designed aiming in testing single device loop-back through LVDS as well as inter-device communication using a single LVDS pair. Due to time restriction this task is partly implemented.

An LVDS project can be found for Quartus and Vivado in the respective LVDS-Quartus[5] and LVDS-Vivado[6] branches of the project repository

Chapter 5

Evaluation

5.1 Tool-chain capabilities

The tool-chain support matrix depicts available platform support on current open-source projects. Projects that rely on these tool-chains like APIO or icestudio are not represented as their features are already covered. Based on the current project progress, implemented features are shown in tables 5.1, 5.2 and 5.3

	IceStorm	**Trellis**	**X-Ray**
Logic	Yes	Yes	Yes
Block RAM	Yes	Yes	Partial

Table 5.1: Basic Tiles

	IceStorm	**Trellis**	**X-Ray**
DSP	Yes	Yes	No
Hard Blocks	Yes	Yes	No
Clock Tiles	Yes	Yes	No
I/O Tiles	Yes	Yes	Partial

Table 5.2: Advanced Tiles

	IceStorm	**Trellis**	**X-Ray**
Logic	Yes	Yes	Yes
Clock	Yes	Yes	No

Table 5.3: Routing

Tool-chain features for open-source and free software is shown in Table 5.4

	IDE	Block Design	Timing Analysis	Simulation	Pin Mapping GUI	Floorplan GUI
APIO	Yes	No	Yes	Yes	No	No
icestorm	No	No	External	External		
icestudio	Yes	Yes	No	No	No	No
prjtrellis	No	No				
Quartus	Yes	Yes	Yes	Yes	Yes	Yes
Vivado	Yes	Yes	Yes	Yes	Yes	Yes

Table 5.4: Features

5.2 ICESTORM AND PRJTRELLIS

These are command line tools for ICE40 and ECP5 FPGA series. Each tool is responsible for a single step in the process of taking a verilog file and ending up with a programmed device. Since this is a multistep process, it is best to be carried out by using a Makefile as demonstrated in the project repository.

There is a PLL configuration tool for each platform (icepll, ecppll) that can be used for generating PLL code. As en example an 120MHz clock derived from a 12MHz clock is demonstrated:

```
$ icepll -m -f pll.v -i 12 -o 120

F_PLLIN:    12.000 MHz (given)
F_PLLOUT:  120.000 MHz (requested)
F_PLLOUT:  120.000 MHz (achieved)

FEEDBACK: SIMPLE
F_PFD:   12.000 MHz
F_VCO:  960.000 MHz

DIVR:  0 (4'b0000)
DIVF: 79 (7'b1001111)
DIVQ:  3 (3'b011)

FILTER_RANGE: 1 (3'b001)

PLL configuration written to: pll.v
```

This will also generate a pll module file that can be included in the project

```
/**
 * PLL configuration
 *
 * This Verilog module was generated automatically
 * using the icepll tool from the IceStorm project.
 * Use at your own risk.
 *
 * Given input frequency:        12.000 MHz
 * Requested output frequency:  120.000 MHz
 * Achieved output frequency:   120.000 MHz
 */

module pll(
input  clock_in,
output clock_out,
output locked
);

SB_PLL40_CORE #(
.FEEDBACK_PATH("SIMPLE"),
.DIVR(4'b0000),// DIVR =  0
.DIVF(7'b1001111),// DIVF = 79
.DIVQ(3'b011),// DIVQ =  3
.FILTER_RANGE(3'b001) // FILTER_RANGE = 1
) uut (
.LOCK(locked),
.RESETB(1'b1),
.BYPASS(1'b0),
.REFERENCECLK(clock_in),
.PLLOUTCORE(clock_out)
);

endmodule
```

Timing analysis is available for ICE40 using the icetime tool. It will report timing estimate as well as detailed timing report.

Simulation is possible by creating a testbench file and using iverilog combined with gtk-wave for visualization

5.3 APIO

APIO is a command line tool that wraps around icestorm toolchain and attempts to simplify development process. It provides all features of icestorm toolchain as well as simulation. It handles driver installation and project initialization for supported boards.

A summary of available features is presented by command usage

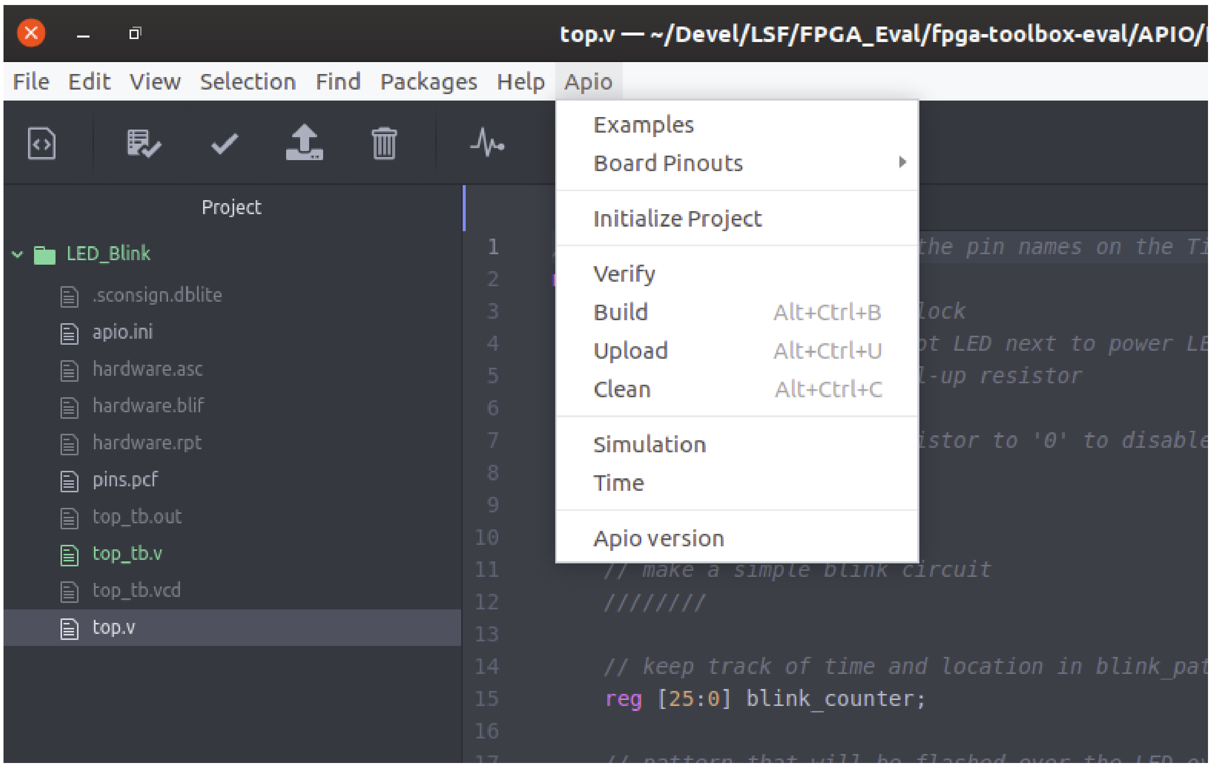
```
Project commands:
  build      Synthesize the bitstream.
  clean      Clean the previous generated files.
  lint       Lint the verilog code.
  sim        Launch the verilog simulation.
  time       Bitstream timing analysis.
  upload     Upload the bitstream to the FPGA.
  verify     Verify the verilog code.

Setup commands:
  drivers    Manage FPGA boards drivers.
  init       Manage apio projects.
  install    Install packages.
  uninstall  Uninstall packages.

Utility commands:
  boards     Manage FPGA boards.
  config     Apio configuration.
  examples   Manage verilog examples.
  raw        Execute commands using Apio packages.
  system     System tools.
  upgrade    Check the latest Apio version.
```

One drawback is that it uses the now deprecated arachne-pnr instead of its replacement nextpnr.

An Atom editor plugin is available that integrates APIO functionality and provides the same functionality from within the editor.



```
the pin names on the Ti

lock
ot LED next to power LE
l-up resistor

istor to '0' to disable



// make a simple blink circuit
////////

// keep track of time and location in blink_pat
reg [25:0] blink_counter;
```

Figure 5.1: APIO plugin for Atom

5.4 IceStudio

Icestudio is built on top of APIO and Icestorm projects and currently supports the ICE40 FPGA family. Icestudio comes as an AppImage and handles all toolchain and "Driver" (udev rules) requirements through its GUI It is a block design editor (Figure 5.2) and comes with a basic set of blocks like

- I/O
- Memory
- Multiplexers
- Gates

- Flip-flops
- Prescalers/Counters

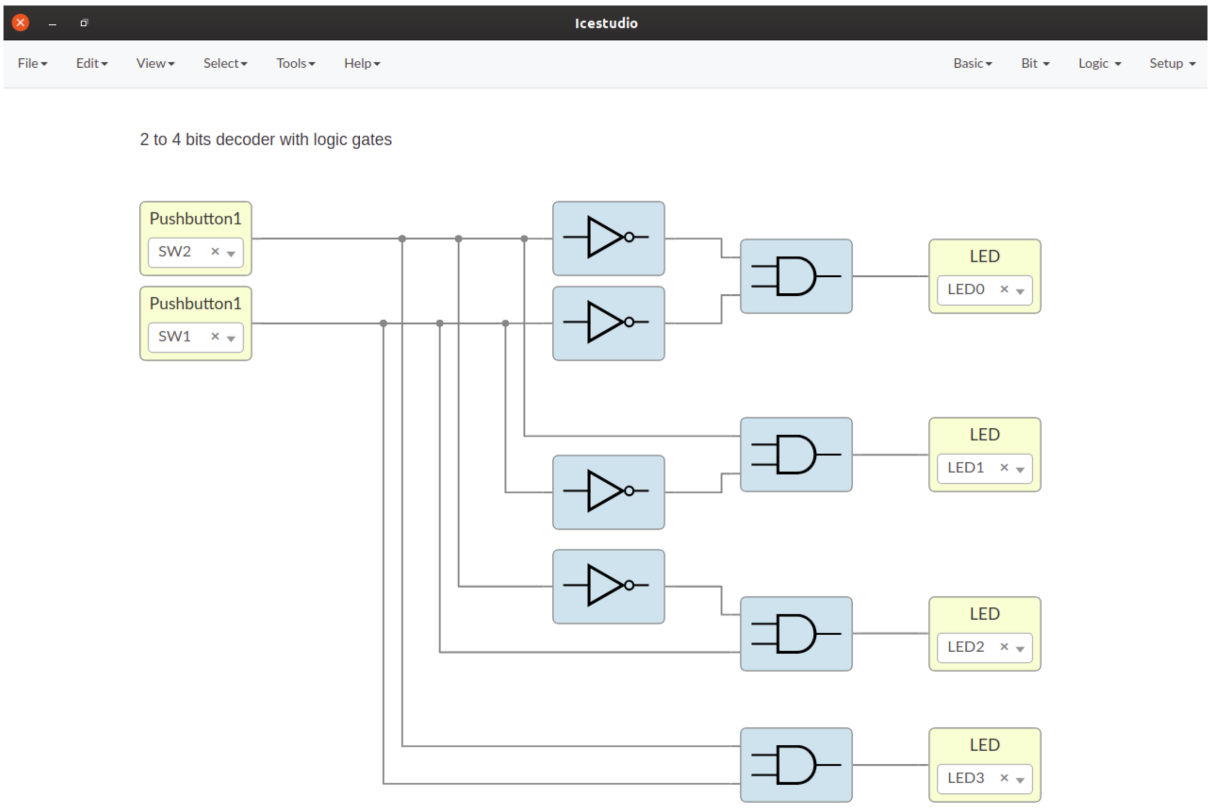


Figure 5.2: Icestudio block design

Custom blocks can be created by implementing the underlying functionality in Verilog (Figure 5.3) and provide input and output ports in order to connecto to other blocks

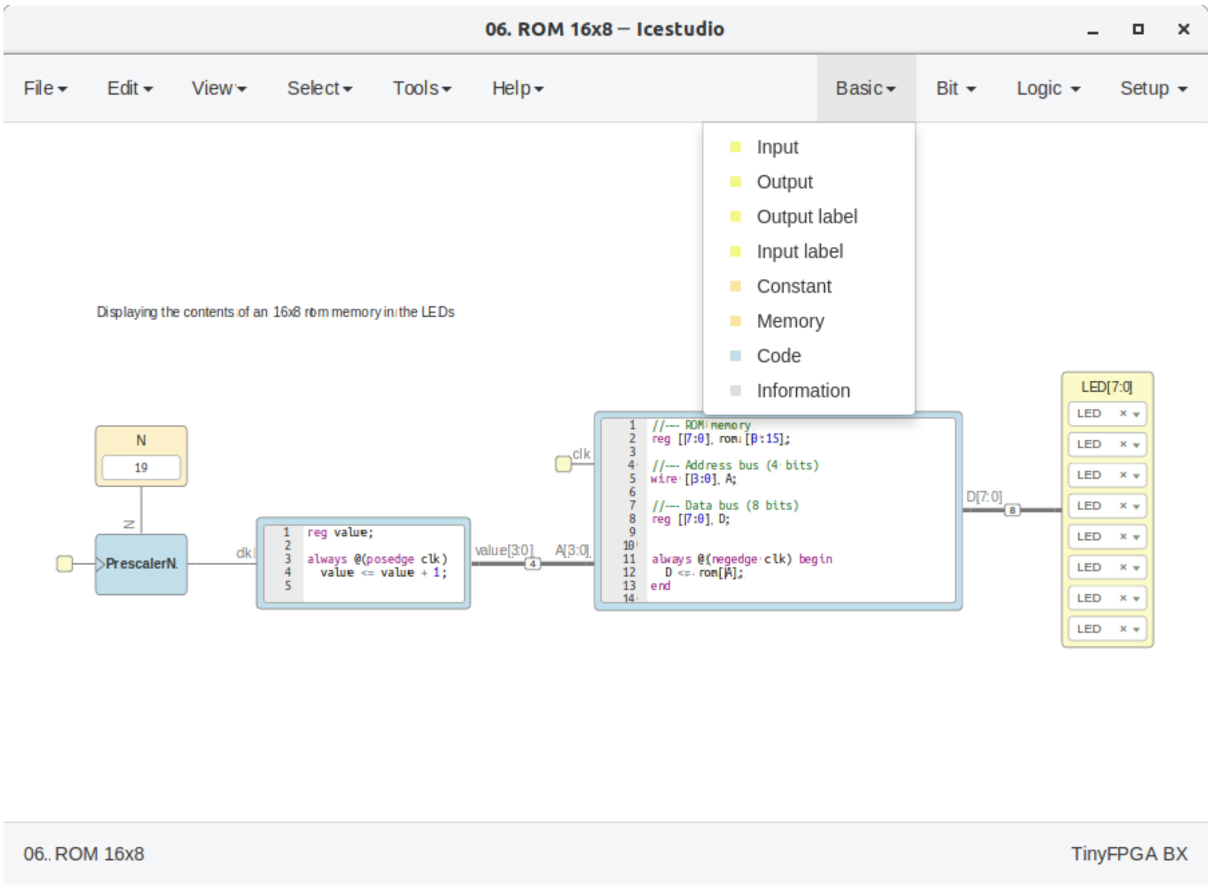
Sets of blocks can be organised in collections that are ZIP files of specific file structure. Collections can be added in Icestudio extending its functionality. Collections also include example projects that can be loaded into the block editor

A list of opensource collections can be found at

`https://github.com/FPGAwars/icestudio-collections`

`https://fpgawars.github.io/`

For creating custom collections, a tool is available assisting in creating the basic file and folder structure: `https://github.com/FPGAwars/icm`



```
reg value;

always @(posedge clk)
  value <= value + 1;
```

```
//--- ROM memory
reg [7:0] rom [0:15];

//--- Address bus (4 bits)
wire [3:0] A;

//--- Data bus (8 bits)
reg [7:0] D;


always @(negedge clk) begin
  D <= rom[A];
end
```

Figure 5.3: Icestudio custom block

5.5 Quartus

Quartus supports VHDL and Verilog source files as well as block design. It comes with a variety of IP core re-configurable generators

Quartus Prime software features include [2]:

- SOPC Builder, a tool in Quartus Prime software that eliminates manual system integration tasks by automatically generating interconnect logic and creating a test-bench to verify functionality
- Qsys, a system-integration tool that is the next generation of SOPC Builder. It uses an FPGA-optimized network-on-chip architecture that doubles the fMAX performance vs. SOPC Builder.
- SoCEDS, a set of development tools, utility programs, run-time software, and application examples to help you develop software for SoC FPGA embedded systems.

- DSP Builder, a tool that creates a seamless bridge between the MATLAB/Simulink tool and Quartus Prime software, so FPGA designers have the algorithm development, simulation, and verification capabilities of MATLAB/Simulink system-level design tools
- External memory interface toolkit, which identifies calibration issues and measures the margins for each DQS signal.
- Generation of JAM/STAPL files for JTAG in-circuit device programmers.

A comparison of available features at this time is shown in

5.6 Vivado

Vivado HL WebPACK Edition was used for this evaluation. It is provided at no cost but with device limitations.

Features available in this edition are:

- Synthesis and Place and Route
- Dynamic Function eXchange
- Simulator
- Device Programmer
- Logic Analyzer
- Serial I/O Analyzer
- Debug IP (ILA/VIO/IBERT)
- High-Level Synthesis
- IP Integrator

Non free editions also include System Generator for DSP and Model Composer

Chapter 6

Version control

Version control is a vital tool for complex projects requiring the collaboration of several persons. Version control tools provide the means for creating milestones in code, develop new or experimental features as well as revert changes that introduced bugs in an organized and structured way.

When several persons need to modify the content of the same file at the same time, all changes need to be merged into the final result. Such a task, done manually, would require tremendous effort, introduce new errors and in some cases is almost impossible.

Version control allows several persons to interact and modify the same files and provides the means for all the changes to be merged in a functional outcome. Parts of the content are assembled automatically and any conflicts in content can be handled efficiently.

6.1 iCE40

iCE40 projects can easily use git for version control. An example .gitignore file is presented. All files generated by Makefile are ignored.

icestorm and APIO .gitignore file contents

```
*.out
*.blif
*.bin
*.asc
*.vcd
*.rpt
abc.history
.sconsign.dblite
```

6.2 ECP5

prjtrellis projects can easily use git for version control. All files generated by Makefile are ignored

An example .gitignore file is presented

```
*.bit
*.svf
*.config
*.json
```

6.3 Vivado

Version control methods for Vivado are explained in application note XAPP1165 where version control for project mode, non-project mode and IP are described.

An example .gitignore is provided by xilinx and was extended for latest Vivado version. It can be found in Vivado folder in the project repo.

6.4 Quartus

For Quartus an effort was made to determine the bare minimum of files needed in order for a project to be able to regenerated. A .gitignore file was created keeping all files necessary for project creation and IP regeneration. It can be found under Quartus folder in the project repository.

Chapter 7

Performance

Performance evaluation for each tool-chain is based on time taken for all steps needed to go from source to bit-stream. Each test is run twice to also measure total time on consecutive executions. This was needed as some tool-chains generate IP cores the first time used and that contributes to overall performance time.

IceStudio is using APIO under the hood so performance should be almost identical

Performance times are shown for LED Blinker project (Figure 7.1) and SimpleCounter project (Figure 7.2)

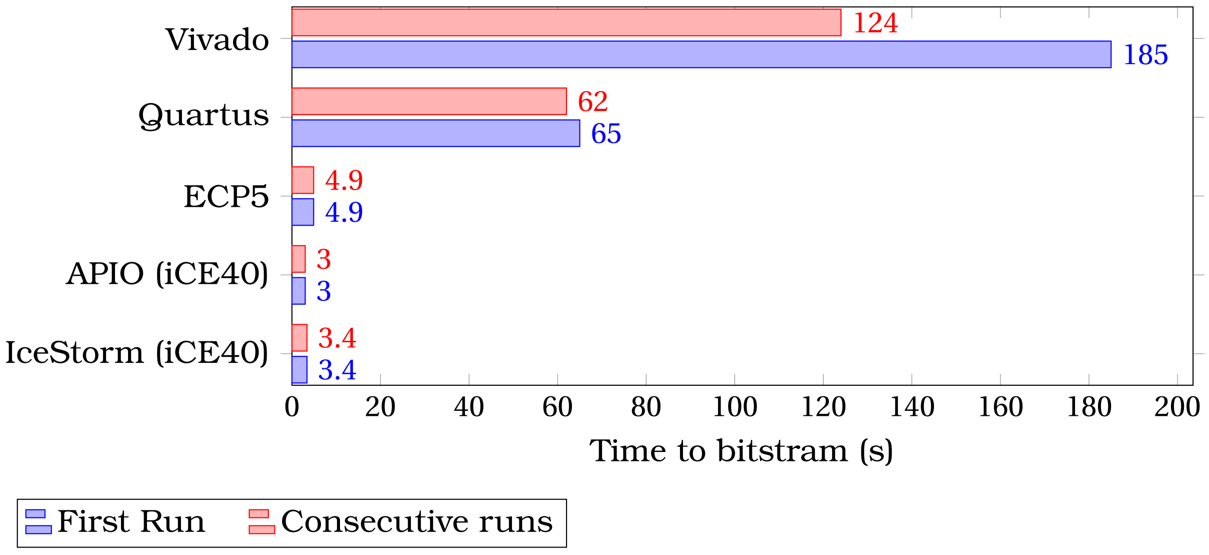


Figure 7.1: LED Blinker time

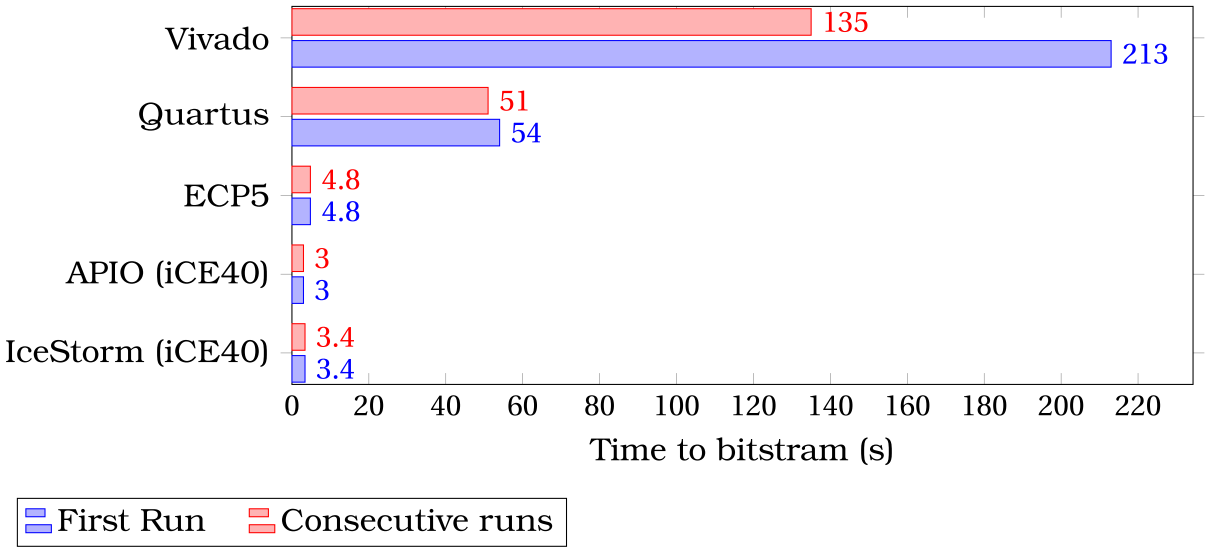


Figure 7.2: Counter time

Chapter 8

Conclusion

Open source ice40 and ECP5 tool-chains are mature enough for code-based development. Tools for timing and some code generation scripts are already available. Simulation visualization is possible through other open-source projects,

APIO is a step forward towards an IDE easing development. Although it currently supports only ice40 devices the foundations for ecp5 support are well laid.

IceStudio is the only application available allowing for block design development for the ice40 family. It is still lacking many features of the commercial equivalents but being an active project one can expect that it will be on par with commercial solutions. As with APIO, an ECP5 version is likely to be possible.

Moving on to commercial solutions, Intel Quartus although initially appeared straightforward to use, there were some shortcomings. Being a project inherited from Altera, all code had to be refactored for re-branding. As it turns out this was done poorly and in some cases, users where required to alter platforms source files to overcome errors.

User experience left mixed feelings, although steps from block design to bit-stream are very clear, block design editor needs some work as it requires some effort to keep the design clean and tidy.

Xilinx Vivado was a pleasant surprise regarding its block design editor. the interface is intuitive, blocks and connections can be managed in a way that always produces a clean and tidy design. Rearrange and align tools further help in that aspect. A design assistant tool is also available to further accelerate design by suggesting missing elements of a newly placed IP.

What felt a bit peculiar is the use of IPs even for simple tasks like selecting bits from a bus but it still promotes clear design UX wise. Finally, when it comes to choosing a device family for an open-source project, options are quite clear as ice40 and ECP5 are the first candidates.

For more demanding applications an investment on Xilinx devices is suggested since an effort for Xilinx open-source tool-chains is in the works

Bibliography

[1] "Project icestorm." [Online]. Available: https://github.com/cliffordwolf/icestorm.git

[2] "Wikipedia quartus." [Online]. Available: https://en.wikipedia.org/wiki/Intel_Quartus_Prime

[3] "Wikipedia vivado." [Online]. Available: https://en.wikipedia.org/wiki/Xilinx_Vivado

[4] "Fpga toolbox evaluation repository." [Online]. Available: https://gitlab.com/librespacefoundation/sdrmakerspace/fpga-toolbox-eval

[5] "Quartus lvds branch." [Online]. Available: https://gitlab.com/librespacefoundation/sdrmakerspace/fpga-toolbox-eval/-/tree/LVDS-Quartus

[6] "Vivado lvds branch." [Online]. Available: https://gitlab.com/librespacefoundation/sdrmakerspace/fpga-toolbox-eval/-/tree/LVDS-Vivado