Open-Source tools for FPGA development

Marek Vašut <marex@denx.de>

July 14, 2016
Software engineer at DENX S.E. since 2011
  ▶ Embedded and Real-Time Systems Services, Linux kernel and driver development, U-Boot development, consulting, training

Versatile Linux kernel hacker
Custodian at U-Boot bootloader
Yocto (oe-core) contributor
Structure of the talk

- Introduction to FPGA technology
- Compiling the FPGA content, from HDL to bitstream:
  - Analysis and Synthesis tools
  - Place and Route tools
  - Assembler tools
  - Simulation/Visualisation tools
- Demonstration
- Why are open-source FPGA tools hard?
Field Programmable Gate Array
High-Speed Programmable logic
Plenty of I/O options
Extremely parallel architecture
Usually used for:
- Digital Signal Processing (DSP)
- Parallel data processing
- Custom hardware interfaces
- ASIC prototyping
- ...

Common vendors – Xilinx, Altera, Lattice, Microsemi...
Internal structure

**BLUE**  Global interconnect
**GREEN**  Local interconnect
**RED**    Logic element

Marek Vašut <marex@denx.de>  Open-Source tools for FPGA development
Programming the FPGA

- Each vendor has his own set of tools: Altera Quartus, Xilinx Vivado/ISE, Lattice Diamond, ...
- Tools are generally closed source
- Flow is very similar between tools:
  - Analysis and Synthesis → HDL → Netlist
  - Pack, Place and Route → Netlist → Technology
  - Assembler → Technology → Bitstream
  - Timing Analysis → Analyze design timing
  - Simulation and Visualisation → Check timing constraints
  - Simulate and analyze the design on the host

Marek Vašut <marex@denx.de> - Open-Source tools for FPGA development
Analysis and Synthesis

- HDL → Netlist
- Behavioral model → Circuit schematic
- Analysis – Parsing of HDLs, validation, ...
- Synthesis – Parsed HDL to Netlist
- Tools:
  - Icarus Verilog
  - Odin II
  - Yosys

Open-Source tools for FPGA development

Marek Vašut <marex@denx.de>
Icarus Verilog

- HDL simulation/translation/synthesis tool
- GPL license (with plugin exception)
- Plugin support
- Input:
  - Verilog 2005
    - Mostly supported
    - Widely used
    - Active development
  - System Verilog – Similar level of support as Verilog 2005
  - VHDL – Limited support
- Output:
  - VVP – Intermediate language used for simulation
  - Verilog – Minimization/Simplification
  - VHDL – Translation
  - Gate-level netlist – dropped in 0.9.1
- Website: http://iverilog.icarus.com/

Marek Vašut <marex@denx.de> Open-Source tools for FPGA development
Odin II

- HDL synthesis framework with visualisation support
- MIT license
- Input:
  - Verilog
  - BLIF netlist – from downstream stages
- Output: BLIF Netlist
  - Works directly with VPR
  - Usable for both FPGA and ASIC synthesis
- Links:
  - Website: https://code.google.com/archive/p/odin-ii/
  - Git: https://github.com/verilog-to-routing/vtr-verilog-to-routing/tree/master/ODIN_II
Logic optimization/minimization
Often coupled with synthesis tool
Input: BLIF netlist
Output: BLIF netlist
Yosys

- HDL synthesis suite
- ISC license
- Input:
  - Verilog 2005
  - BLIF netlist
- Output:
  - Simplified Verilog
  - BLIF/EDIF/… netlist
- Built-in logic optimization/minimization using abc
- Supports mapping (overlaps with PnR):
  - ASIC cell libraries
  - Xilinx 7-series FPGAs
  - Lattice iCE40 FPGAs
- Website: http://www.clifford.at/yosys/
Place and Route

- Netlist → Technology-mapped netlist
- Consists of multiple sub-steps:
  - Pack – Clump netlist elements into larger blocks
  - Place – Place the blocks in the FPGA
  - Route – Route the interconnect between blocks
- Tools:
  - Arachne PnR
  - VPR
Arachne PnR

- Place and Route tool specific to iCE40 FPGA
- Works specifically with Yosys
- Input:
  - Technology mapped netlist from Yosys
- Output:
  - Textual representation of bitstream
- Website: https://github.com/cseed/arachne-pnr
Versatile Placement and Routing
Pack, Place, Route tool
Now part of VtR (Verilog to Routing)
Extremely flexible
Works with any reasonable FPGA technology
Used extensively in FPGA research
Also works well with commercial FPGA tools
Website:
http://www.eecg.toronto.edu/~vaughn/vpr/vpr.html
Placed/Routed netlist → Bitstream

Technology is often undocumented "family gold"

This step has the least amount of tools

Tools:
- IcePack
IcePack

- Open-Source assembler for iCE40 FPGA
- Part of the IceStorm project
- Textual representation of bitstream → binary bitstream
- Website: http://www.clifford.at/icestorm/

Marek Vašut <marex@denx.de> Open-Source tools for FPGA development
Whole design flows

- Aforementioned tools can be assembled into complete flows
- Flows which take HDL and produce bitstream:
  - IceStorm
Verilog to Bitstream flow
Specific to Lattice iCE40 FPGA
Tools:
- Yosys – Analysis and Synthesis
- Arachne PnR – Place and Route
- IcePack – Bitstream generation
Additional tools:
- IceProg – Programming of the FPGA
- IceTime – Timing analysis
Website: http://www.clifford.at/icestorm/
Example of using IceStorm, Gray counter, Top module

```verilog
module top (  
        input hwclk,  
        output led1,  
        output led2,  
        output led3,  
        output led4,  
        output led5,  
        output led6,  
        output led7,  
        output led8
);  
```
Example of using IceStorm, Gray counter, Top module

/* Counter register */
reg [7:0] count = 8'b0;

/* Grey counter implementation */
assign led1 = count[0] ^ count[1];
assign led2 = count[1] ^ count[2];
assign led3 = count[2] ^ count[3];
assign led4 = count[3] ^ count[4];
assign led5 = count[4] ^ count[5];
assign led6 = count[5] ^ count[6];
assign led7 = count[6] ^ count[7];
assign led8 = count[7];

/* Increment counter */
always @(posedge hwclk)
  count <= count + 1;
endmodule
Example of using IceStorm, Gray counter, Pin map

1. set_io --warn-no-port led1 B5
2. set_io --warn-no-port led2 B4
3. set_io --warn-no-port led3 A2
4. set_io --warn-no-port led4 A1
5. set_io --warn-no-port led5 C5
6. set_io --warn-no-port led6 C4
7. set_io --warn-no-port led7 B3
8. set_io --warn-no-port led8 C3
9. set_io --warn-no-port hwclk J3
Example of using IceStorm, Building and Programming

1. $ yosys -p "synth_ice40 -top top -blif top.blif" top.v
2. $ arachne-pnr -d 8k -P ct256 \\n   -o top.txt -p pinmap.pcf top.blif
3. $ icepack top.txt top.bin
4. $ iceprog top.bin
HDL is simulated on the development host
Allows applying triggers and constraints
Tools:
  - gHDL
  - Icarus Verilog
  - Verilator
gHDL

▶ VHDL simulator
▶ Compiles VHDL into native code
▶ Uses GCC/LLVM/built-in backend for code generation
▶ Faster than interpreted simulator
▶ Output:
  ▶ VCD (Value Change Dump) – Verilog oriented
  ▶ gHDL waveform – Native format fit for VHDL
▶ Website: http://ghdl.free.fr/
Synthesis from Verilog to C++

Verilator does perform optimization during synthesis

Supported input:
- Verilog
- Verilog 2005 – Subset is supported
- System Verilog – Subset is supported

Website: http://www.veripool.org/wiki/verilator
Icarus Verilog

- Primarily a simulator/translator
- HDL is compiled to intermediate VVP code
- The `vvp` tool is used as VVP interpreter
- Extremely useful for writing testbenches
- Visualisation output: GTKWave
- Website: [http://iverilog.icarus.com/](http://iverilog.icarus.com/)
Example of using iVerilog, Gray counter, Testbench

```verilog
module top_tb ();

reg clk;
wire led1;
wire led2;
wire led3;
wire led4;
wire led5;
wire led6;
wire led7;
wire led8;
```
Example of using iVerilog, Gray counter, Testbench

```
top top (  
    .hwclk(clk),  
    .led1(led1),  
    .led2(led2),  
    .led3(led3),  
    .led4(led4),  
    .led5(led5),  
    .led6(led6),  
    .led7(led7),  
    .led8(led8)  
);  
```
Example of using iVerilog, Gray counter, Testbench

```verilog
initial begin
    $dumpfile("top_tb.lxt");
    $dumpvars(0, top);
    clk = 1'b0;
    repeat(1000) begin
        #1 clk = ~clk;
        #1 clk = ~clk;
    end
end
endmodule
```
Example of using iVerilog, Gray counter, Performing the test:

```
1  iverilog -o top.vvp top_tb.v top.v
2  vvp top.vvp -lxt2
3  gtkwave top_tb.lxt
```
GTKWave

- Visualisation tool
- Supports many formats – VCD, LXT, FST, ...
- Works well with gHDL, Icarus Verilog ...
- Website: http://gtkwave.sourceforge.net/
Why are open-source FPGA tools hard?

- Lack of documentation
- Fear of releasing proprietary algorithms
- Pushback from IP vendors
DeBit

- Attempt to document Altera and Xilinx FPGAs
- Appears inactive
- Textual documentation mostly missing
- Lots of cryptic C code
- Supports only old FPGAs
- Allows dumping bitstream of specific parts
Thank you for your attention!

Contact: Marek Vasut <marex@denx.de>