YosysHQ

Building alternative FPGA toolchains

ESA Contract # 4000141380/23/NL/GLC/ov

N. Engelhardt, M. Milanović

Presentation Overview

- Tools introduction
- NG-Ultra
- Implementation
- Conclusion
- Q&A

YosysHQ Open Source EDA Tools

- Yosys
  - Where it all started
  - Synthesis for FPGA and ASIC
  - Useful as a general netlist manipulation tool
- nextpnr
  - Place and route for FPGA
- Various others not relevant today
  - Formal verification
  - Equivalence checking
  - Mutation coverage

ESA Contract # 4000141380/23/NL/GLC/ov

Why Open Source Tools?

Synthesis (Yosys)

Pros:

- Generic tool - can be used for other architectures as well
- Easier integration with other tools
- Fine tuning of primitives used in output
- Adding new optimization passes for research
- Easy integration with LiteX and Amaranth

Cons:

- No full SystemVerilog support (possible with commercial libraries)
- Resource limitations and timing constraints are not known to synthesis
- Optimization quality can be lower than in vendor tools

Place and route (nextpnr)

Pros:

- Generic infrastructure, faster to develop
- Can integrate with other synthesis tools (then using Yosys for conversions)
- Placement and routing algorithms are generic
- Can use HDL attributes for placement
- Can be used for research projects as way to test various placement and routing algorithms

Cons:

- No feedback loop to synthesis
- So far rather small FPGAs were only supported
- Algorithms are not performance optimized
- Depending on architecture, possible that design becomes non-routable.
- Implementing good placement require more experience with architecture

Bitstream manipulation

Pros:

- Makes complete flow open source
- Enables binary bitstream back to netlist conversion
- Bitstream protection still requires same signing process so good security is not a problem
- More control over process, like enabling BRAM content or PLL configuration change without affecting logic

Cons:

- Require more knowledge of FPGA internals
- Reversing can be quite long and tedious process
- With vendor information is still a tedious process
- May expose security issues (but that can be good as well)

Implementing a flow for NG-Ultra

What is special about the NG-Ultra?

- Much larger than previously supported architectures
  - NG-Ultra has 500k LUT
- Unique structure
  - Very large tile compared to other FPGAs
- Comparatively low routing availability

ESA Contract # 4000141380/23/NL/GLC/ov

Creating support for new architecture

- Familiarization with architecture
- Synthesis:
  - Yosys support
    - Adding synth_xxx pass
- Place and route:
  - Chip database
    - Primitive description
    - Acquiring routing graph
  - Bitstream manipulation tools
    - Tools to extract bitstream into human readable files.
    - And vice versa
  - nextpnr support
    - Import of chip database
    - Implementation of architecture specifics

ESA Contract # 4000141380/23/NL/GLC/ov

Creating support for NG-Ultra

- Familiarization with architecture
- Synthesis:
  - Yosys support
    - Adding synth_xxx pass
- Place and route:
  - Chip database
    - ~~Primitive description~~
    - ~~Acquiring routing graph~~
  - Bitstream manipulation tools
    - ~~Tools to extract bitstream into human readable files.~~
    - ~~And vice versa~~
  - nextpnr support
    - Import of chip database
    - Implementation of architecture specifics

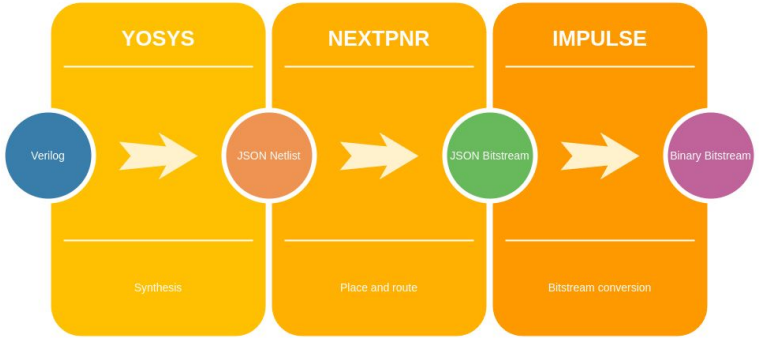
Not main project goal

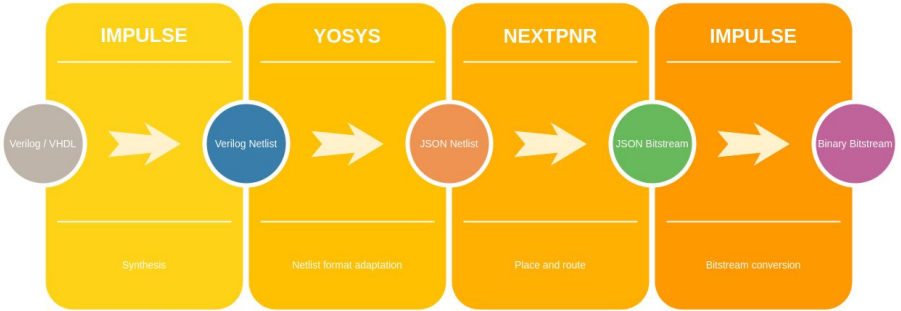
Data provided by vendor

Using vendor tools

NG-Ultra flow

- Impulse/Yosys for synthesis
- nextpnr for place and route
- Impulse to generate binary bitstream
- Vendor scripts to program board

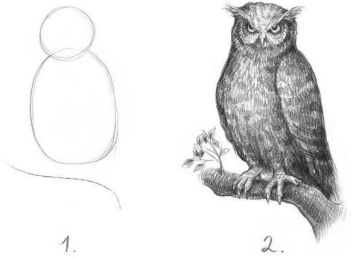




Familiarization with architecture

- Obtaining knowledge about architecture of target FPGA
- Find description of HDL primitives
- Familiarize with vendor tools and naming conventions
- Obtain hardware (development board and accessories)
- Create small examples that do work on actual hardware
- Test complete flow using vendor tools

Yosys support



- Start by adding LUT and FF mapping support
  - Write simulation models for basic primitives
  - Use Yosys equivalence checks to confirm mapping is done right
  - Confirm using place and route using vendor tools
- Continue by adding io pad mapping, carry chains, block ram, DSPs …
  - Custom passes sometimes needs to be added
  - Add techmap rules for wrapper primitives
  - Add black box definitions for primitives not planned to be supported
- Make parts of the flow optional and configurable
- **NOTE**: For NG-Ultra this was not part of the project, support is not complete (mostly DSP related parts are missing)

ESA Contract # 4000141380/23/NL/GLC/ov

Chip database

- Routing information was provided by vendor in text format
- Extracting more information
  - Recognize tiles and their types
  - Model crossbars and muxes
  - Validate signal direction for primitives
- Changes that can reflect in performance or QoR improvements:
  - Placement:
    - BEYOND_FE mapped to LUT + DFF pair
    - Store metadata for special features (CSC, SCC) and lobe
  - Routing:
    - LUT permutations
    - LUT and DFF propagation
    - GCK and WFG propagation
    - Store metadata of mapping to original data

ESA Contract # 4000141380/23/NL/GLC/ov

Bitstream manipulation tools

- For this project we relied on vendor provided tools
- Usually a reverse engineering task
- Defines a mapping between bitstream bits and FPGA configuration
- nextpnr outputs data in human readable format that needs to be converted to a binary file
- Usually we provide two-way conversion, going back from binary to text is required to validation

nextpnr architecture support

- PnR workflow
  - Load netlist -> Pack -> Place -> Route -> Generate bitstream
- Generic algorithms built on the Arch API
- Himbächel API as additional layer on top
- Packing
  - Mapping between HDL and hardware primitives
  - Checking input parameters
  - Optimizing depending on input parameters
- Pre-placement (IO constraints, PLL and WFG pre-placement)
- Post-placement (CSC, SCC and GCK insertion)
- Post-routing
  - Recalculating for LUT permutations
  - Propagation routing to primitive allocation
  - Fix crossbar configurations
  - Exporting bitstream

ESA Contract # 4000141380/23/NL/GLC/ov

Conclusion

Use cases

- As a user:
  - Small portable tools requiring no license to be used
  - All is open-source which makes full audit possible
  - Faster iteration of full workflow
  - Easier porting existing HDL designs to new architecture
  - Able to test design in different conditions
  - Easy to integrate with other open source tools
- As a vendor:
  - Getting more potential hardware customers
  - Potential larger user base
  - Profit from general improvements in Yosys/nextpnr

ESA Contract # 4000141380/23/NL/GLC/ov

Current state of Yosys NanoXplore support

- Only NG-Ultra supported (no Ultra300 specifics implemented)
- NG-Medium and NG-Large are just placeholders
- Block RAM mapping does not efficiently use complete memory block
- DSP mapping not implemented
- Wrappers for DSP are missing
- Simulation models only for basic primitives

ESA Contract # 4000141380/23/NL/GLC/ov

nextpnr for NG-Ultra

- Able to place and route designs of varying complexity with a comparable time to Impulse.
  - Of the designs we tested, some of the larger designs fail to place and route.
- Placement is crucial due to lower routability of the NG-Ultra architecture compared to other FPGAs we've worked on.
- Complex designs that include usage of high speed I/Os and SoC are not supported yet.
- Plenty of room to improve, specially placement optimizations.

ESA Contract # 4000141380/23/NL/GLC/ov

Q&A

Thank you