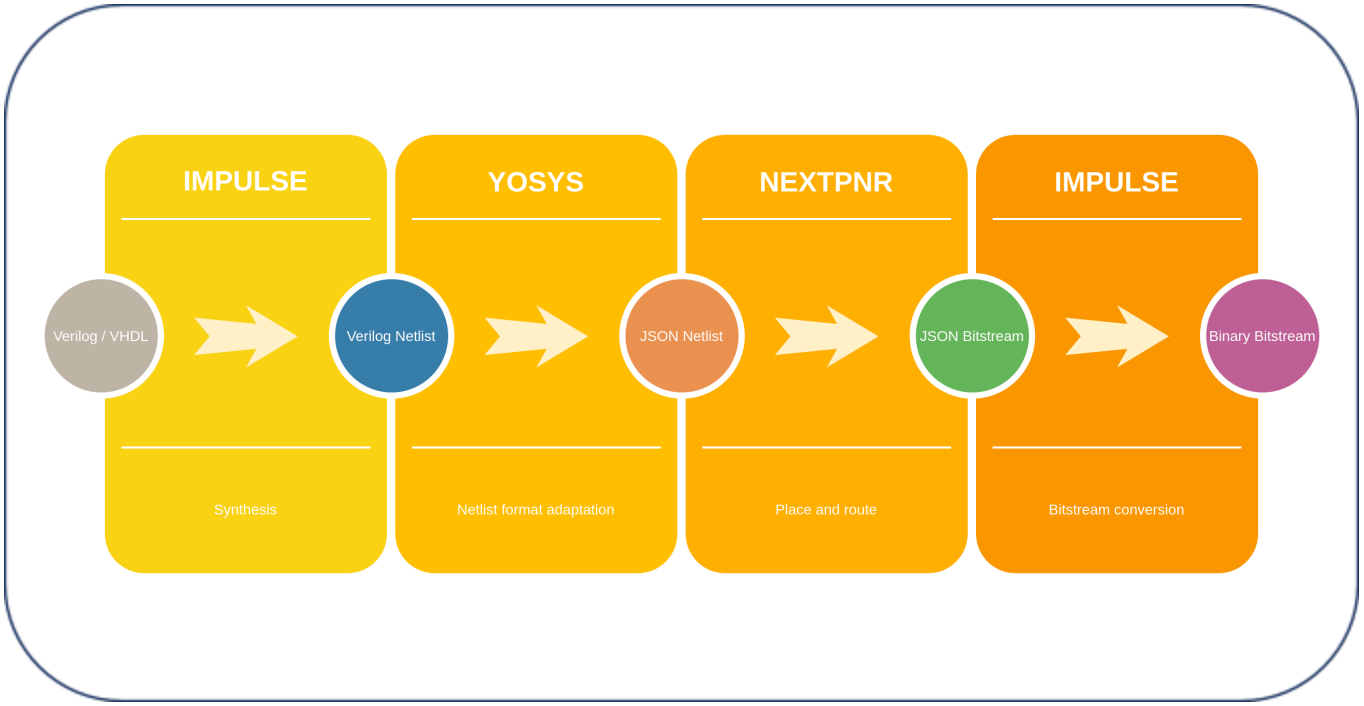

An open source FPGA toolchain for a European space-grade FPGA

Executive summary

Early Technology Development

Channel: Open Discovery Ideas Channel

Affiliation(s): YosysHQ GmbH

Activity summary:

This activity aimed to develop support for NanoXplore's NG-Ultra architecture within the open-source place and route tool nextpnr, as an alternative to the proprietary Impulse tool. Extensive benchmarking showed that in most cases the performance of the nextpnr prototype is comparable to Impulse but there are also a number of outliers with significant room for improvement. The results confirm that nextpnr is capable of supporting large FPGAs and overcoming architectural limitations by performing netlist transformations and placement optimisations. Future improvements may include support for high-speed I/O and SoC and further placement optimisations to get better quality of results.

Publishing Date: 2024-11-01
Contract Number: 4000141380
Implemented as ESA Initial Support for Innovation

This activity concerned the development and integration of the open-source place and route tool, nextpnr, for NanoXplore's NG-Ultra Field Programmable Gate Arrays (FPGAs). The NG-Ultra is a high-reliability, radiation-hardened FPGA intended for use in critical environments such as space. The primary aim of the project was to create an alternative place and route tool to NanoXplore's proprietary tool (Impulse) by implementing support for the NG-Ultra architecture inside nextpnr. To make the goal more realistic, the main focus was on implementing support for the basic set of primitives that are inferred by Impulse synthesis, and was extended to cover commonly used manually instantiated primitives. Due to the fact that some of the primitives that were not initially planned were quite simple to be added, the final implementation supported more than planned, but specific features that were not thoroughly tested were marked as such.

When converting a user's input design into a format suitable for running on the NanoXplore NG-Ultra, the design is first turned into a netlist of gates. This netlist is optimised, and then mapped into virtual cells representing the logic that the hardware provides, such as 4-input look-up tables ("LUT4s"; used to implement logic), D flip-flops ("DFFs"; single bits of memory distributed throughout the chip), carry chains (used for fast addition/subtraction), and register file blocks (high-capacity memories). This part of the process was out of scope for the project and we relied on the Impulse implementation for this step. Next, these virtual cells need to be assigned to physical locations on the FPGA ("placement"), and these placed cells need to be connected using the physical interconnect wires on the chip ("routing"). The steps of placement and routing are what Impulse and nextpnr perform to produce a physical representation of the input design that can then be converted into a configuration file the FPGA can load (the "bitstream"). Since the bitstream format is proprietary, nextpnr outputs a description of the placed and routed netlist in JSON format, which can be imported into Impulse for bitstream generation.

To perform the tasks of placement and routing, nextpnr needs to be aware of the available types and locations of logic, and how they are connected to each other. nextpnr makes a distinction between virtual logic ("cells"), and physical logic that cells can be placed into ("BELs", an abbreviation of "basic elements of logic"); placement is the assignment of cells to BELs. BELs are connected through a graph with nodes ("wires"), and edges ("PIPs", an abbreviation for "programmable interconnect points"); routing is the connection of wires and PIPs between BELs. Despite the large number of BELs and a complex routing graph that contains many PIPs, there are patterns of identical relative connections between logic. nextpnr can detect and deduplicate these patterns to store the routing graph compactly on disk.

As this kind of detailed internal architectural data is available only from vendors producing FPGAs, it was necessary that NanoXplore share this data. The complete set of data included not only a description of the specific building blocks of the NG-Ultra FPGA architecture and their connections but also the necessary timing information to accurately estimate propagation delays during the place and route process. This timing data is crucial for ensuring that signals meet the required timing constraints, particularly in high-performance designs where propagation delays can impact the overall functionality and reliability of the design. Using this detailed information, a chip database file was created. This database provides a compact representation of the NG-Ultra architecture, allowing nextpnr to perform placement and routing operations with reduced memory and processing overhead.

The implementation of a nextpnr backend for NG-Ultra ensures that designers have the option of validating their designs through an open-source tool, alongside the proprietary Impulse toolchain. This redundancy increases confidence in design accuracy, as a design which functions identically under two toolchains is significantly less likely to have been silently affected by toolchain issues. Furthermore, nextpnr's open-source nature allows for customization, enabling users to adapt the tool to meet specific needs. This is particularly important in space applications, where mission-critical reliability is essential. The nextpnr support currently includes key FPGA primitives,

with the potential for future expansion to more specialised components like high-speed I/O and SoC primitives.

This project represents a significant step forward in creating a flexible and open-source FPGA development toolchain for NG-Ultra. The adaptability and collaborative nature of nextpnr encourages innovation and improvements not only from within the developer community but also through ongoing partnerships with the academic world. By offering an alternative toolchain, the project ensures that developers can cross-verify their designs, which is important in critical applications like space missions. The project also proved that nextpnr, although being a generic implementation supporting multiple architectures, can be used for large FPGAs.

Throughout the project, the team conducted extensive testing and validation by running output bitstreams on a test board to ensure that nextpnr is reliable and performs well with NG-Ultra devices. A series of functional, integration, and performance tests were carried out, comparing results between nextpnr and NanoXplore's Impulse tool. Metrics such as place-and-route (PnR) times, maximum operating frequency, and resource utilisation were analysed to gauge performance and efficiency. Initial tests focused on basic FPGA components like LUTs and DFFs, while subsequent tests incorporated more complex design elements, including memory blocks and DSP components. As these tests needed to confirm a completely working workflow they were also including bitstream generation and manual testing of design on the development board, which were both time consuming but also necessary to validate results.

Several challenges emerged during the development process. One of the primary difficulties was matching the performance of highly optimised proprietary tools in runtime and timing closure. While benchmarking highlighted areas of weakness in nextpnr compared to Impulse, it was not always clear how to resolve them. Achieving comparable performance required ongoing tuning and testing, which informed what features were to be included in nextpnr to reduce runtime. Overall goal was not to perform better but have comparable performance to Impulse, but most importantly to have a reliable place and route that produces a bitstream that works on the development board as expected.

Collecting a set of benchmarks which fully exercised nextpnr's support for various NG-Ultra features also proved difficult. Many publicly-available architecture-generic designs are intended for FPGAs much smaller than the NG-Ultra, while larger designs usually focus on a specific FPGA, utilising architecture-specific primitives that do not generalise. As NG-Ultra is quite a new architecture we were not able to find existing open big projects utilising architecture specific primitives, and these capabilities were mostly covered with individual tests.

Another challenge involved maintaining compatibility with NanoXplore's proprietary toolchain, as Impulse received frequent updates to the JSON netlist import functionality which was being co-developed simultaneously for nextpnr integration. Despite these challenges, close collaboration with NanoXplore and the use of detailed vendor documentation were crucial in ensuring that the integration progressed smoothly. For example, differences in how Impulse and nextpnr represent the routing graph could result in the Impulse importer going into an infinite loop when attempting to load the output of nextpnr; discussion with NanoXplore identified the cause of this and a fix was implemented in nextpnr.

The benchmark results highlighted several important improvements, but also pointed to pitfalls and limitations of NG-Ultra architecture. Most impactful changes were the introduction of the optimised bounding box prediction and CSC insertion, that both helped dramatically reduce time needed to finish place and route for most of the designs. Many low effort optimisations, like DFF and LUT chains and XLUT use, have proven useful, and helped routing by generating more efficient placements utilising local routing resources and reducing congestion. While some experimental

features, such as electrostatic placement and net partitioning, showed potential, they were not included in the final version due to instability or poorer results. Overall, all of the included improvements have resulted in faster place and route times, and in most cases, gave comparable performance relative to the Impulse, with occasional trade offs in maximum frequency and critical path lengths, especially for more complex designs.

Looking ahead, future work could focus on expanding support within nextpnr to cover additional FPGA primitives, such as high-speed I/O blocks and SoC integration making it more useful for high complexity designs. Developing a tool that translates the JSON output from nextpnr into a binary bitstream format would enable a fully open-source toolchain, eliminating reliance on proprietary tools, which may be useful for auditable and reproducible builds. As the benchmark results demonstrated, there are still placement improvements that could be implemented to achieve better execution time, but also to make sure that each design that can be processed with Impulse can also be placed and routed with nextpnr.

In conclusion, the integration of nextpnr with NG-Ultra represents a major achievement in open-source FPGA development, as it is by far the largest FPGA supported by open-source tooling. The tool provides developers with a flexible and customizable option for designing for NG-Ultra FPGAs, offering increased reliability through the ability to cross-verify designs using two independent tools. Although there are challenges ahead in terms of optimising performance and expanding tool functionality, this project has laid the foundation for ongoing innovation and collaboration in the FPGA development community.