m-labs / nmigen

A refreshed Python toolbox for building complex digital hardware

nmigen.org

View license

413 stars 43 forks

Star Watch

Code Issues 30 Pull requests 5 Actions Security Insights

master Go to file

sbourdeauducq Clifford -> Claire ... 13 days ago 826

View code

README.md

nMigen

A refreshed Python toolbox for building complex digital hardware

Although nMigen is incomplete and in active development, it can already be used for real-world designs. The nMigen language (`nmigen.hdl.ast` , `nmigen.hdl.dsl`) will not undergo incompatible changes. The nMigen standard library (`nmigen.lib`) and build system (`nmigen.build`) will undergo minimal changes before their design is finalized.

Despite being faster than schematics entry, hardware design with Verilog and VHDL remains tedious and inefficient for several reasons. The event-driven model introduces issues and manual coding that are unnecessary for synchronous circuits, which represent the lion's share of today's logic designs. Counterintuitive arithmetic rules result in steeper learning curves and provide a fertile ground for subtle bugs in designs. Finally, support for procedural generation of logic (metaprogramming) through "generate" statements is very limited and restricts the ways code can be made generic, reused and organized.

To address those issues, we have developed the *nMigen FHDL*, a library that replaces the event-driven paradigm with the notions of combinatorial and synchronous statements, has arithmetic rules that make integers always behave like mathematical integers, and most importantly allows the design's logic to be constructed by a Python program. This last point enables hardware designers to take advantage of the richness of the Python language—object oriented programming, function parameters, generators, operator overloading, libraries, etc.—to build well organized, reusable and elegant designs.

Other nMigen libraries are built on FHDL and provide various tools and logic cores. nMigen also contains a simulator that allows test benches to be written in Python.

See the doc/ folder for more technical information.

nMigen is based on Migen, a hardware description language developed by M-Labs. Although Migen works very well in production, its design could be improved in many fundamental ways, and nMigen reimplements Migen concepts from scratch to do so. nMigen also provides an extensive compatibility layer that makes it possible to build and simulate most Migen designs unmodified, as well as integrate modules written for Migen and nMigen.

The development of nMigen has been supported by M-Labs and LambdaConcept.

HLS?

nMigen is *not* a "Python-to-FPGA" conventional high level synthesis (HLS) tool. It will *not* take a Python program as input and generate a hardware implementation of it. In nMigen, the Python program is executed by a regular Python interpreter, and it emits explicit statements in the FHDL domain-specific language. Writing a conventional HLS tool that uses nMigen as an internal component might be a good idea, on the other hand :)

Installation

nMigen requires Python 3.6 (or newer), Yosys 0.9 (or newer), as well as a device-specific toolchain.

```
pip install git+https://github.com/m-labs/nmigen.git
pip install git+https://github.com/m-labs/nmigen-boards.git
```

Introduction

TBD

Supported devices

nMigen can be used to target any FPGA or ASIC process that accepts behavioral Verilog-2001 as input. It also offers extended support for many FPGA families, providing toolchain integration, abstractions for device-specific primitives, and more. Specifically:

- Lattice iCE40 (toolchains: **Yosys+nextpnr**, LSE-iCECube2, Synplify-iCECube2);
- Lattice MachXO2 (toolchains: Diamond);
- Lattice ECP5 (toolchains: **Yosys+nextpnr**, Diamond);
- Xilinx Spartan 3A (toolchains: ISE);
- Xilinx Spartan 6 (toolchains: ISE);
- Xilinx 7-series (toolchains: Vivado);
- Xilinx UltraScale (toolchains: Vivado);
- Intel (toolchains: Quartus).

FOSS toolchains are listed in **bold**.

Migration from Migen

If you are already familiar with Migen, the good news is that nMigen provides a comprehensive Migen compatibility layer! An existing Migen design can be synthesized and simulated with nMigen in three steps:

1. Replace all `from migen import <...>` statements with `from nmigen.compat import <...>`.
2. Replace every explicit mention of the default `sys` clock domain with the new default `sync` clock domain. E.g. `ClockSignal("sys")` is changed to `ClockSignal("sync")`.
3. Migrate from Migen build/platform system to nMigen build/platform system. nMigen does not provide a build/platform compatibility layer because both the board definition files and the platform abstraction differ too much.

Note that nMigen will **not** produce the exact same RTL as Migen did. nMigen has been built to allow you to take advantage of the new and improved functionality it has (such as producing hierarchical RTL) while making migration as painless as possible.

Once your design passes verification with nMigen, you can migrate it to the nMigen syntax one module at a time. Migen modules can be added to nMigen modules and vice versa, so there is no restriction on the order of migration, either.

Community

nMigen discussions take place on the M-Labs IRC channel, #m-labs at freenode.net. Feel free to join to ask questions about using nMigen or discuss ongoing development of nMigen and its related projects.

License

nMigen is released under the very permissive two-clause BSD license. Under the terms of this license, you are authorized to use nMigen for closed-source proprietary designs.

Even though we do not require you to do so, these things are awesome, so please do them if possible:

- tell us that you are using nMigen
- put the nMigen logo on the page of a product using it, with a link to https://nmigen.org
- cite nMigen in publications related to research it has helped
- send us feedback and suggestions for improvements
- send us bug reports when something goes wrong
- send us the modifications and improvements you have done to nMigen as pull requests on GitHub

See LICENSE file for full copyright and license info.

"Electricity! It's like magic!"

Releases

2 tags

Packages

No packages published

Contributors 24

+ 13 contributors

Languages

● **Python** 100.0%