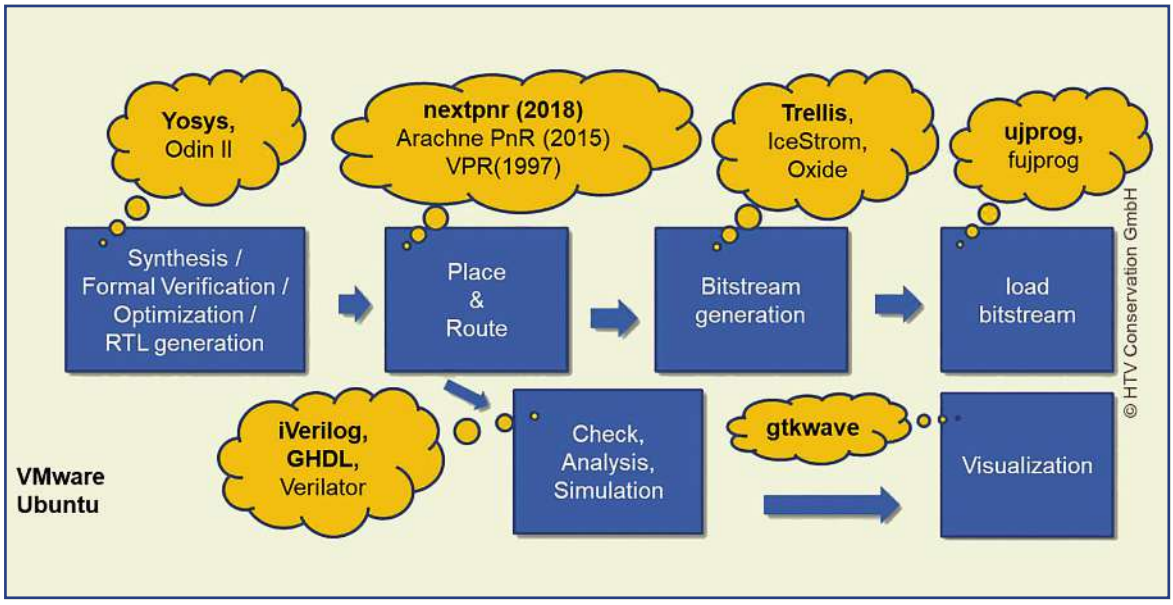


Bild: HTV

**Bild 1:** Open-Source-Tools zum Erzeugen eines FPGA-Designs für die ECP5-Architektur von Lattice.

QUELLOFFENE SOFTWAREWERKZEUGE

# Open-Source-Toolchain für FPGA-Entwicklung nutzen

Für das Programmieren von Field Programmable Gate Arrays, kurz FPGAs, sind immer mehr quelloffene Softwarewerkzeuge erhältlich. Diese können eine Alternative sein zu den proprietären Tools der Bausteinhersteller.

Bild: HTV

VERFASST VON
**Thomas Kuhn**

**Assistent der Geschäftsleitung, Leitung Forschungs- und Entwicklungsprojekte und HTV Akademie**
HTV Halbleiter-Test & Vertriebs-GmbH

FPGAs (Field Programmable Gate Arrays) sind integrierte Schaltkreise der Digitaltechnik und kommen in vielen Gebieten der digitalen Elektronik zum Einsatz. Im Gegensatz etwa zu Mikrocontrollern ist die Schaltung eines FPGAs nicht fest in Silizium gegossen, sondern lässt sich in weiten Teilen frei konfigurieren und für spezifische Einsatzzwecke optimieren. Realisiert wird dies mit so genannten Look Up Tables (LUTs). Diese LUTs sind neben Flip-Flop-Logikgattern die fundamentalen Building-Blocks beziehungsweise Grundelemente eines FPGAs. Durch diesen Aufbau sind FPGAs sehr flexibel einsetzbar und können die ihnen zugedachten Aufgaben sehr schnell abarbeiten. Die Konfiguration, die sich auch im Nachhinein – eben im Feld – noch ändern lässt, wird als sogenannter Bitstrom in den Baustein geladen.

Die zum Programmieren von FPGAs notwendigen Tools sind zum überwiegenden Teil proprietär und auf bestimmte Bausteine des jeweiligen Herstellers abgestimmt. In den letzten Jahren sind jedoch immer mehr kostenlose Softwareprogramme veröffentlicht worden, die die FPGA-Entwicklung bei allen Entwicklungsschritten – Design Entry, Synthesis, Simulation, Mapping, Place & Route und Programming – als Open-Source-Toolchain unterstützen. Die Entwickler von Open-Source-Tools verfolgen dabei mehrere Ziele: Sie wollen etwa eine Plattform- beziehungsweise FPGA-Architektur-unabhängige Softwareentwicklung ermöglichen, die Entwicklungszeit verkürzen und die Entwicklungskosten senken.

Wie in der Open-Source-Welt üblich, arbeiten an der Neuschöpfung und Optimierung erforderlicher Tools nicht nur einzelne Entwickler. Vielmehr schließen sich einzelne Experten mit Firmen zu Gruppen zusammen, die gemeinsam definierte Arbeitspakete abarbeiten. Mit dabei sind auch führende FPGA-Hersteller wie Xilinx (seit Anfang 2022 Teil von AMD) und Altera (seit 2015 Teil von Intel). Beispiele für solche Zusammenschlüsse sind die CHIPS Allianz (Common Hardware for Interfaces, Processors and Systems) und die F4PGA Gemeinschaft (vgl. [1] und [3]).

Dieser Artikel zeigt am Beispiel der FPGA-Architektur ECP5 der Firma Lattice Semiconductor (Lattice) eine mögliche Open-Source-Toolchain für die Schaltungsentwick-

lung (Bild 1). Zum Einsatz kommt das FPGA-Board CSULX3S-01, für das bereits viele Programmbeispiele sowohl in den Hardware-Beschreibungssprachen (engl. Hardware Description Language, HDL) VHDL als auch Verilog veröffentlicht wurden ([2] und [8]). Dieses ist mit einem LFE5U-12F-6BG381I-FPGA mit 12k (12.000) LUTs ausgestattet und ist für etwa 140 Euro erhältlich.

Nach erfolgreicher Installation der Open-Source-Softwarewerkzeuge (vgl. Abschnitt 2) muss lediglich bei den Make-Dateien der Beispiele darauf geachtet werden, als verfügbare Anzahl an LUTs den Wert „12k“ dem RTL-Synthese-Werkzeug Yosys bekannt zu machen.

## Open-Source-Toolchain für die ECP5 FPGA-Architektur

Für die Ausführung der im Folgenden vorgestellten Open-Source-Tools wird eine Linux-Umgebung benötigt. Dafür eignet sich zum Beispiel eine virtuelle Maschine mit Ubuntu 20.04.4 LTS. Diese kann in der VMware Workstation 16 Player betrieben werden. Für die virtuelle Maschine wird ein Festplattenspeicher von 20 GB und ein Arbeitsspeicher von 8 GB für die fehlerfreie Installation der Open-Source-Tools empfohlen.

Eine FPGA-Hardware kann zum Beispiel mit den Hardwarebeschreibungssprachen VHDL oder Verilog modelliert werden. Hier kommt zunächst eine Abstraktionsebene in der Hardware-Modellierung von integrierten Schaltkreisen zum Einsatz, die Registertransferebene (englisch Register Transfer Level, RTL). Beim Entwurf auf dieser Ebene wird das System durch den Signalfluss zwischen Registern spezifiziert. VHDL und Verilog erzeugen daraus High-Level-Darstellungen der Schaltkreise. Von diesen lassen sich Darstellungen auf niedrigeren Ebenen und schließlich die konkrete Hardware synthetisieren.

Für die RTL-Synthese von Verilog-Dateien eignet sich zum Beispiel die Open-Source-Software Yosys. Entwickler, die mit VHDL arbeiten, müssen dem Software-Tool noch ein spezielles GHDL-Plugin voranstellen (vgl. [5]). Yosys liefert als Ergebnis der Synthese eine vollständige Netzliste in unterschiedlichen Dateiformaten (zum Beispiel *.json, *.ebif, *.bif und *.edif) (vgl. Bild 3).

Das Tool Nextpnr ist anschließend in der Lage, die von Yosys gelieferte Netzliste zu verarbeiten. Die enthaltenen Elemente werden unter Berücksichtigung der gewählten FPGA-Architektur – im vorliegenden Fall ist die ECP5 – gepackt, platziert und miteinander verbunden. Nextpnr ist in der Lage, die Ergebnisse der einzelnen Schritte über ein GUI (Graphical User Interface) zu visualisieren (vgl. Bild 4). Das Tool erzeugt schließlich eine Konfigurationsdatei, aus der das Tool Trellis den finalen Bitstrom für das Gate-Array generiert. Das Übertragen dieses Bitstroms auf das

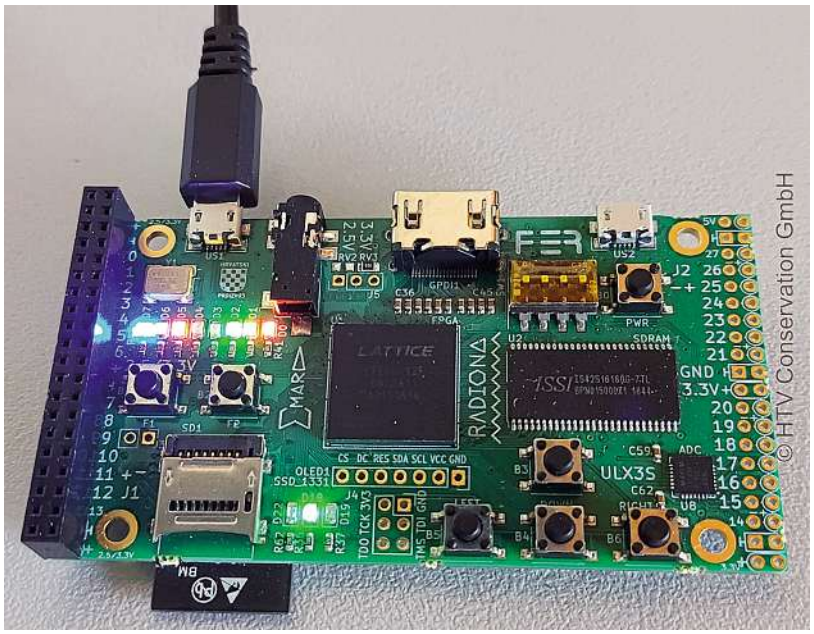


Bild: HTV

**Bild 2:**
FPGA-Board CS-ULX3S-01 mit ECP5 FPGA-Technologie der Firma Lattice.

FPGA können dann Softwarewerkzeuge wie ujprog und fujprog übernehmen. Soll das FPGA-Design näher untersucht und zudem visualisiert werden, steht für die grafische Ausgabe der Signale die Software GTKWave zur Verfügung (vgl. Bild 5). Für das Erzeugen der Signalverläufe können abhängig von der eingesetzten Hardwarebeschreibungssprache die Simulatoren GHDL, iVERILOG oder VERILATOR eingesetzt werden.

Praktisch: Die Open-Source-Tools liefern darüber hinaus passende Datenformate für die gängigen Softwarewerkzeuge der FPGA-Hersteller, um somit auch bei Zwischenschritten als externe Tools beim FPGA-Entwicklungsprozess eingebunden werden zu können.

## Yosys – RTL-Synthese

Zum Erzeugen der benötigten Netzliste mit der Bezeichnung design.json wird dem Programm Yosys die Verilog-Datei (design.v) unter Angabe der gewünschten FPGA-Architektur – hier Lattice ECP5 – (synth_ecp5) und der erforderlichen Parameter übergeben (vgl. [10]):

*yosys -p 'read_verilog design.v; synth_ecp5 -noccu2 -nomux -nodram -json design.json'*

Liegen die Quelldateien für die FPGA-Entwicklung in VHDL vor, wird ein zusätzliches Plugin benötigt (vgl. [6]), das diese im Zusammenhang mit dem VHDL Simulator GHDL für Yosys passend konvertiert (vgl. [4]). Im Befehl zum Ausführen von Yosys muss das Plugin als Modul mit dem Befehlszusatz „-m ghdl" hinzugefügt werden:

*yosys -m ghdl -p 'read_verilog design.vhdl; synth_ecp5 -noccu2 -nomux -nodram -json design.json'*

**Bild 3:**
Mit der Kombination von Yosys und GHDL kann aus Verilog oder VHDL Beschreibungen eine RTL Synthese durchgeführt werden.

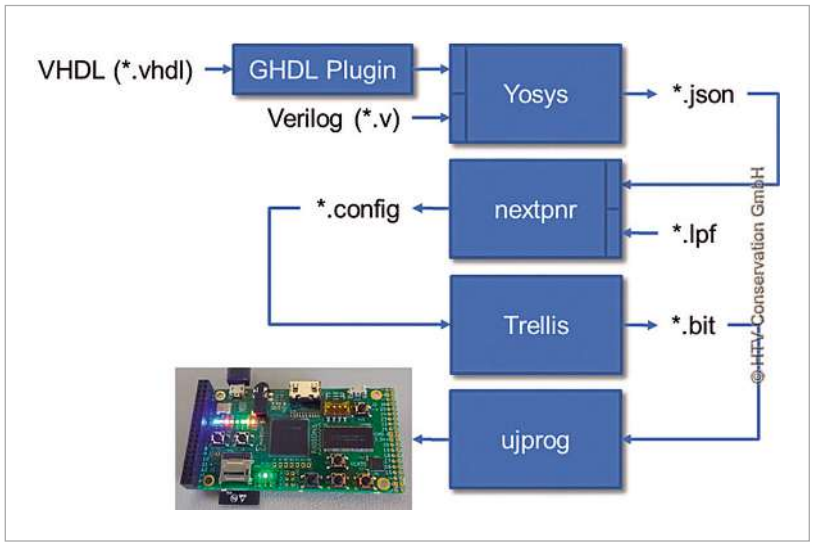


Bild: HTV

## Nextpnr – Mapping und Place & Route

Die von Yosys erzeuge Netzliste (design.json) wird nextpnr mit der Angabe der Architektur (-ecp5), der Anzahl der LUTs des verwendeten FPGAs (--12k) und einer Datei mit die Architektur betreffenden Einschränkungen (constraints.lpf) übergeben. Nextpnr führt anschließend das Packen, Platzieren und Verbinden der Elemente aus der Netzliste unter Berücksichtigung der gewünschten FPGA-Architektur durch:

*nextpnr-ecp5 --12k --json design.json --lpf constraints.lpf --textcfg out.config*

Ist es gewünscht, dass die Schritte für Place & Route auch grafisch ausgegeben werden, muss dem Befehl zum Aufruf von nextpnr noch der Zusatz „--gui" hinzugefügt werden (vgl. Bild 4):

*nextpnr-ecp5 --12k --json design.json --lpf constraints.lpf --textcfg out.config --gui*

## Trellis – Erzeugung des Bitstroms

libtrellis ist eine C++-Bibliothek für ECP5-FPGAs. Die Dienstprogramme von libtrellis und zugehörige Datenbanken ermöglichen die Erzeugung und Bearbeitung von ECP5-Bitströmen. Aus den Funktionen eines FPGAs (Routing und Konfiguration der programmierbaren Logikblöcke) kann sowohl der zugehörige Bitstrom erzeugt werden, als auch aus einer Bitstromdatei die zugehörigen Funktionen wieder extrahiert werden. Obwohl libtrellis als Standardbibliothek verwendet werden kann, wird es im Projekt Trellis hauptsächlich als Python-Modul (pytrellis) verwendet. Trellis erzeugt in der OpenSource-Toolchain aus der von nextpnr erzeugten Konfigurationsdatei (out.config) den gewünschten Bitstrom (top.bit):

ecppack out.config top.bit

## Konfiguration des FPGAs

Der FPGA wird dann mit Hilfe von ujprog konfiguriert. Das Linux-Tool überträgt den Bitstrom über die USB-Schnittstelle des CS-ULX3S-01 Boards zum FPGA.

*ujprog top.bit*

## GTKWave – graphische Ausgabe der Signalverläufe

Für die graphische Ausgabe von Signalverläufen steht GTKWave zur Verfügung. Diese Open-Source-Software erhält die Daten für die Darstellung von Signalverläufen aus Simulatoren. GHDL wird für VHDL-Dateien verwendet (vgl. [4]), iVERILOG für Verilog-Dateien (vgl. [9]) und der VERILATOR für Verilog- und SystemVerilog-Dateien und Testumgebungen in C++ (vgl. [7]).

Nach der Installation von GHDL und GTKWave, werden der VHDL-Quelltext (design.vhdl) und auch die VHDL-Testumgebung (design_tb.vhdl) auf syntaktische Fehler geprüft:

*ghdl -s design.vhdl*
*ghdl -s design_tb.vhdl*

Anschließend erfolgt die Analyse der Dateien:

*ghdl -a design.vhdl*
*ghdl -a design_tb.vhdl*

Danach wird alles synthetisiert und die simulierten Signalverläufe im VCD-Format abgelegt:

*ghdl -e design_tb*
*ghdl -r design_tb --vcd=waveform.vcd*

Der Aufruf von GTKWare zur Anzeige der simulierten Signale erfolgt dann mit dem Befehl:

*gtkwave waveform.vcd*

## Installation der Open-Source-Softwarewerkzeuge

Die folgenden Codezeilen installieren die oben verwendeten Tools Yosys, Nextpnr, Trellis, Ujprog:

```
sudo apt-get install build-essential clang bison flex \libreadline-dev gawk tcl-dev libffi-dev git \graphviz xdot pkg-config python3 libboost-system-dev \libboost-python-dev libboost-filesystem-dev zlib1g-dev
git clone https://github.com/YosysHQ/yosys.git
cd yosys
make
sudo make install
git clone https://github.com/YosysHQ/nextpnr.git
cd nextpnr
sudo apt install libeigen3-dev
cmake -DARCH=ecp5 -DBUILD_GUI=ON -DTRELLIS_INSTALL_PREFIX=/usr/local -DCMAKE_INSTALL_PREFIX=/usr/local .
make -j 4
sudo make install
git clone --recursive https://github.com/YosysHQ/prjtrellis
cd prjtrellis/libtrellis
sudo apt install libboost-all-dev
sudo apt install cmake
cmake -DCMAKE_INSTALL_PREFIX=/usr/local .
make
sudo make install
git clone https://github.com/emard/ulx3s-bin
sudo cp ulx3s-bin/usb-jtag/linux-amd64/ujprog /usr/local/bin
```

Für den Zugriff unter Linux auf das FPGA-Board muss noch die Datei „/etc/udev/rules.d/80-fpga-ulx3s.rules“ mit folgendem Inhalt erstellt werden:

```
# this is for usb-serial tty device
SUBSYSTEM=="tty", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6015", \
 MODE="664", GROUP="dialout"
# this is for ujprog libusb access
ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6015", \
 GROUP="dialout", MODE="666"
```

Dann folgen noch die Installationen von GHDL, des benötigten Yosys-Plugins und von GTKWave:

```
git clone https://github.com/ghdl/ghdl.git
./configure –enable-synth
Make
sudo make install
git clone https://github.com/ghdl/ghdl-yosys-plugin.git
make
sudo make install
sudo apt install gtkwave
```

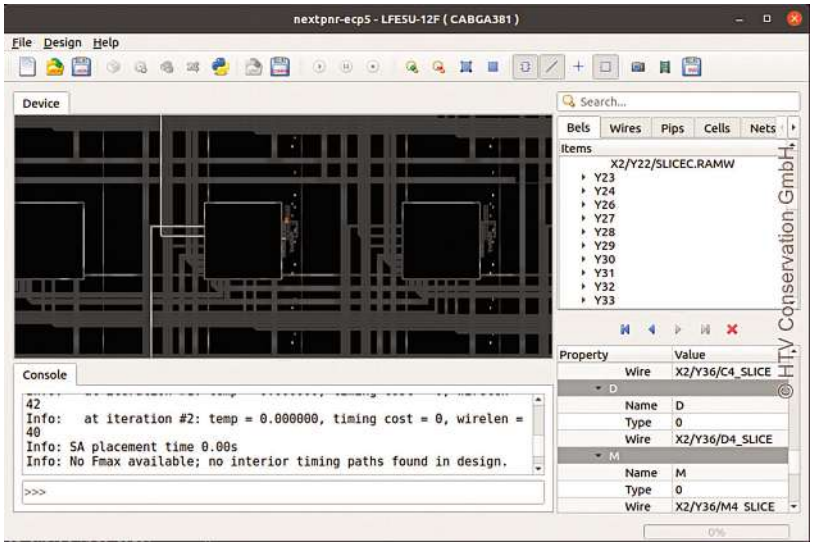


Bild: HTV

**Bild 4:** Grafische Ausgabe der Schritte Place & Rout mit der Open-Source-Software nextpnr.

## Fazit

Die HTV Halbleiter-Test & Vertriebs-GmbH unterstützt seine Kunden bei der Entwicklung von FPGA-Hardware. Um Entwicklungskosten und -zeit einzusparen, werden dabei gerne Open-Source-Tools eingesetzt. In Zukunft geht HTV von einer weiter steigenden Anzahl unterstützter FPGA-Architekturen und einer vollständigeren Dokumentation bestehender Open-Source-Tools aus, da deren Entwicklung durch die FPGA-Hersteller und Universitäten unterstützt und von einer wachsenden Gemeinschaft von Entwicklern begleitetet wird. *(ME)*

### Literatur


[1] CHIPS Alliance. CHIPS (Common Hardware for Interfaces, Processors and Systems) Alliance harnesses the energy of open source collaboration to accelerate hardware development. https://chipsalliance.org/, 2022.

[2] Emard. ULX3S miscellaneous examples. https://github.com/emard/ulx3s-misc, 2022.

[3] F4PGA. Innovate by reaching for the open source FPGA tooling. https://f4pga.org/, 2022.

[4] GHDL. GHDL - the open-source analyzer, compiler, simulator and synthesizer for VHDL. https://github.com/ghdl/ghdl.git, 2022.

[5] GHDL. GHDL-YOSYS-PLUGIN. https://github.com/ghdl/ghdl-yosys-plugin, 2022.

[6] GHDL. GHDL-YOSYS-PLUGIN: VHDL synthesis (based on GHDL and Yosys). https://github.com/ghdl/ghdl-yosys-plugin.git, 2022.

[7] Norbert Kremeris. Why use Verilator? https://www.itsembedded.com/dhd/verilator_1/, 2022.

[8] ULX3S. link ULX3S LED. https://github.com/ulx3s/blink.git, 2022.

[9] Stephen Williams. Icarus Verilog. http://iverilog.icarus.com/, 2022.

[10] YosysHQ. Yosys Open SYnthesis Suite. https://github.com/YosysHQ/yosys.git, 2022.