
vivado-risc-v

AMD/Xilinx Vivado block designs for FPGA RISC-V SoC running Debian Linux distro.

This repository contains FPGA prototype of fully functional RISC-V Linux server with networking, on-line Linux package repository and daily package updates. It includes scripts and sources to generate RISC-V SoC HDL, AMD/Xilinx Vivado project, FPGA bitstream, and bootable SD card. The SD card contains RISC-V Open Source Supervisor Binary Interface (OpenSBI), U-Boot, Linux kernel and Debian root FS. Linux package repositories and regular updates are provided by Debian. Over 90% of packages of the whole Debian collection are available for download.

Also can be used to run bare-metal or RTOS software.

The project is used as reference design to validate RISC-V support in Eclipse TCF.

AMD/Xilinx tools support debugging of RISC-V software over JTAG.

Prerequisites

Hardware

AMD VC707 or AMD KC705 or Digilent Genesys 2 or Digilent Nexys Video or Digilent Nexys A7 100T or Digilent Arty A7 100T board.

VC707 allows to prototype more powerful system: up to 8 64-bit RISC-V cores, up to 100MHz clock speed, 1GB RAM.

KC705 and Genesys 2 are as fast as VC707, but have slightly smaller FPGA - up to 4 cores.

Nexys Video is several times less expensive, academic discount is available. It supports up to 2 cores, up to 50MHz clock speed.

Nexys A7 100T and Arty A7 100T are least expensive supported boards. They have small and slow FPGA, barely enough to run Linux on a single core RISC-V at 50MHz.

Workstation

Ubuntu 20 LTS machine with min 32GB RAM is recommended. sudo access required.

Alternatively, a Windows 10 machine with Ubuntu on Windows can be used to run the tools, see Running RISC-V tools on Windows.

Software

Download and install AMD/Xilinx Vitis. Supported Vitis versions are 2020.2, 2021.1, 2021.2, 2022.1, 2022.2, 2023.1, 2023.2. Vitis installation includes Vivado Design Suite - there is no need to install Vivado separately.

Nexys Video, Nexys A7 100T and Arty A7 100T are supported by free version of Vivado. KC705, VC707 and Genesys 2 require Vivado license.

If using a Digilent board, install Vivado Board Files for Digilent FPGA Boards.

Usage

Checkout the repository, install required packages and update submodules

```
1 sudo apt install git make
2 git clone https://github.com/eugene-tarassov/vivado-risc-v.git
3 cd vivado-risc-v
4 make apt-install
5 make update-submodules
```

Build FPGA bitstream

```
1 source /opt/Xilinx/Vivado/2022.2/settings64.sh
2 make CONFIG=rocket64b2 BOARD=nexys-video bitstream
```

For KC705, use `BOARD=kc705`

For VC707, use `BOARD=vc707`

For Genesys 2 use `BOARD=genesys2`

For Nexys A7 100T use `BOARD=nexys-a7-100t`

For Arty A7 100T use `BOARD=arty-a7-100t`

Some of available CONFIG values (See rocket.scala): * 64-bit big RISC-V cores, Linux capable: * `rocket64b1` - 1 core * `rocket64b2` - 2 cores * `rocket64b2l2` - 2 cores with 512KB level 2 cache * `rocket64b2gem` - 2 cores with 512KB level 2 cache and Gemmini accelerator * `rocket64b4l2w` - 4 cores with 512KB level 2 cache and wide 256-bit memory bus * `rocket64b4` - 4 cores * `rocket64b8` - 8 cores * 64-bit Sonic BOOM cores, Linux capable: * `rocket64w1` - 1-wide Small BOOM, 1 core * `rocket64x1` - 2-wide superscalar Medium BOOM, 1 core * `rocket64y1` - 3-wide superscalar Large BOOM, 1 core * `rocket64z1` - 4-wide superscalar Mega BOOM, 1 core * 32-bit small

RISC-V cores, Linux not supported: * [rocket32s1](#) - 1 core * [rocket32s2](#) - 2 cores * [rocket32s4](#) - 4 cores * [rocket32s8](#) - 8 cores * [rocket32s16](#) - 16 cores

FPGA utilization, LUTs: * 32-bit small RISC-V: 10,800 + 6,100 per core * 64-bit big RISC-V: 10,800 + 27,500 per core * 2-wide superscalar Medium BOOM, 1 core, L2 cache: 148,500 * 3-wide superscalar Large BOOM, 1 core, L2 cache: 252,700

Prepare the SD card

Use USB SD card reader to connect SD card to the workstation, and run:

```
1 ./mk-sd-card
```

The script looks for USB memory device and asks confirmation before using it. Make sure to confirm right SD card device - all old data will be erased.

Booting Linux with QEMU (optional)

In some cases when Linux runs slow on the FPGA, especially designs with lower clock speeds or no ethernet access), it might be worth it to first install the dependencies quickly before running it on FPGA.

You can run the following:

```
1 ./qemu/boot_qemu.sh
```

The script will check for the existence of a debian image under [debian-riscv64/](#) and run [./mk-sd-image](#) as needed. Then it will clone and make a suitable version of u-boot and opensbi for QEMU before finally booting Linux.

Once Linux has booted successfully on QEMU, you can also easily ssh and scp too:

```
1 ssh -p2222 debian@localhost
```

Once all this is done, you can make the sd card without making the image:

```
1 ./mk-sd-card skip_mk_img
```

Program the FPGA flash memory

```
1 source /opt/Xilinx/Vivado/2022.2/settings64.sh
2 make CONFIG=rocket64b2 BOARD=nexys-video flash
```

Alternatively, flash memory can be programmed using Vivado GUI.

Linux login

Host name: debian

User login and password: debian debian

Root login and password: root root

You can login over UART console:

```
1 sudo miniterm /dev/ttyUSB0 115200
```

or, after Linux boot, over SSH:

```
1 ssh debian@debian
```

Modding the design (optional): adding a peripheral device

Use Vivado Block Design to add an IP

Open Vivado:

```
1 source /opt/Xilinx/Vivado/2022.2/settings64.sh
2 make CONFIG=rocket64b2 BOARD=nexys-video vivado-gui
```

The IO block in the design is the best place to add device controllers, like GPIO. See AXI Uartlite as an example, connect your IP to AXI interconnect and interrupts. Validate and synthesize the design, but don't build bitstream yet - device tree and RISC-V HDL need to be updated first.

Close Vivado.

Check the device driver is enabled in patches/linux.config

For example, for AMD/Xilinx GPIO, the config should contain line:

```
1 CONFIG_GPIO_XILINX=y
```

If necessary, change config, then rebuild Linux kernel and bootloader:

```
1 make linux bootloader
2 ./mk-sd-image -r debian-riscv64-boot
```

Copy `debian-riscv64-boot/extlinux` directory to the SD card.

Note: don't change files in the project submodules: `linux-stable`, `u-boot`, `opensbi` or `rocket-chip`. Such changes are lost when the project is rebuilt.

For details on AMD/Xilinx drivers, see [Linux Drivers](#).

Edit `board/nexys-video/bootrom.dts`

Add device description in the “`io-bus {...}`” section. For example, GPIO description can look like this:

```
1      gpio: gpio@60030000 {
2          #gpio-cells = <2>;
3          compatible = "xlnx,xps-gpio-1.00.a";
4          gpio-controller ;
5          interrupt-parent = <&L2>;
6          interrupts = <4>;
7          reg = < 0x60030000 0x10000 >;
8          xlnx,all-inputs = <0x0>;
9          xlnx,dout-default = <0x0>;
10         xlnx,gpio-width = <0x8>;
11         xlnx,interrupt-present = <0x1>;
12         xlnx,is-dual = <0>;
13         xlnx,tri-default = <0xffffffff>;
14     };
```

Make sure the description matches your design. In particular, check addresses and interrupt numbers.

Rebuild FPGA bitstream

```
1 make CONFIG=rocket64b2 BOARD=nexys-video bitstream
```

Program the FPGA or the board flash memory.

Prebuilt images

Prebuilt FPGA bitstream and SD card image are available in the [releases area](#).

Notes

Rocket Chip is used as RISC-V implementation: UC Berkeley Architecture Research - Rocket Chip Generator. Rocket Chip is configured to include virtual memory, instruction and data caches, coherent interconnect, floating point, and all the relevant infrastructure. See `rocket.scala` for Rocket Chip configuration classes.

RISC-V SoC in this repo contains bootrom, which differ from original Rocket Chip bootrom. The modified bootrom contains SD card boot loader and extended device tree.

RISC-V SoC in this repo contains DDR, UART, SD and Ethernet controllers. DDR is provided by Vivado. UART, SD and Ethernet are open source Verilog.

SD controller implements SD HS (High Speed) specs, 25MB/s read/write speed.

Ethernet controller is based on Verilog Ethernet Components project, which is a collection of Ethernet-related components for gigabit, 10G, and 25G packet processing.

Linux kernel and U-Boot use device tree, which is stored in RISC-V bootrom in FPGA. So, same SD card should boot OK on any board or RISC-V configuration.

Nexys Video and Nexys A7 boards can be configured to load FPGA bitstream from SD card.

The device tree contains Ethernet MAC address, which is not unique. It might be necessary to rebuild bitstream with different MAC, see Makefile for details.

If not using provided SD card image: the bootrom loads and executes `boot.elf` file from SD card DOS partition. `boot.elf` is regular executable ELF, it can contain any software suitable for RISC-V RV64 M mode. In case of Linux boot, `boot.elf` contains OpenSBI and U-Boot.

The Makefile creates Vivado project directory, e.g. `workspace/rocket64b2/vivado-nexys-video-riscv`. You can open the project in Vivado GUI to see RISC-V SoC structure, make changes, add peripherals, rebuild the bitstream. The SoC occupies portion of FPGA, leaving plenty of space for experiments and developing additional hardware.

RISC-V SoC in this repo uses BSCAN block to support both RISC-V debugging and FPGA access over same JTAG cable.