RISC-V on an FPGA, pt. 1

A4SD 16-FEB-2016 13:20
P/N 1286-1081-ND
410-292
BOARD FPGA NEXYS4 DDR ARTIX-7
RECEIPT OF GOODS
STATIC
LEAD FREE ROHS COMP
MSL 1
DIGILENT, INC
XF
Artix™-7 Power, Priced for Students.
Package includes Nexys 4 DDR board and USB cable
www.digilentinc.com
XILINX UNIVERSITY PROGRAM
Engineered for VIVADO Development Environment from Xilinx

Last year I had open source instruction set RISC-V running Linux emulated in qemu. However to really get into the architecture, and restore my very rusty FPGA skills, wouldn't it be fun to have RISC-V working in real hardware.

The world of RISC-V is pretty confusing for outsiders. There are a bunch of affiliated companies, researchers who are producing actual silicon (nothing you can buy of course), and the affiliated(?) lowRISC project which is trying to produce a fully open source chip. I'm starting with lowRISC since they have three iterations of a design that you can install on reasonably cheap FPGA development boards like the one above. (I'm going to try to install "Untether 0.2" which is the second iteration of their FPGA design.)

There are two FPGA development kits supported by lowRISC. They are the Xilinx Artix-7-based Nexys 4 DDR, pictured above, which I bought from Digi-Key for £261.54 (that price included tax and next day delivery from the US).

There is also the KC705, but that board is over £1,300.

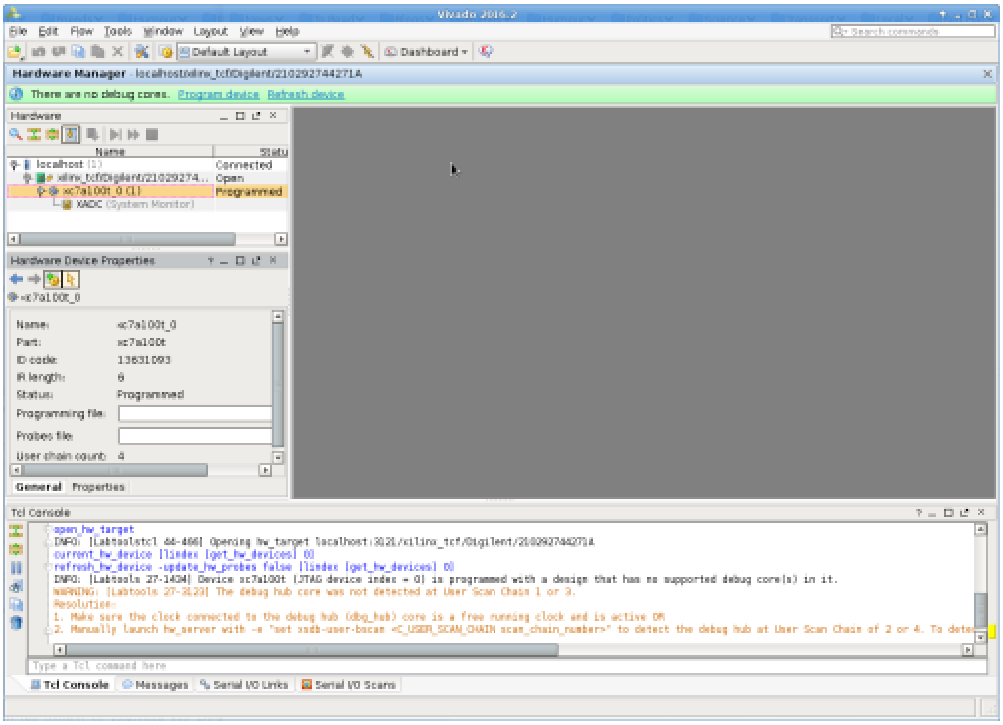
The main differences are speed and available RAM. The Nexys has 128MB of RAM only, which is pretty tight to run Linux. The KC705 has 1GB of RAM.

I'm also going to look at the dev kits recommended by SiFive, which start at US$150 (also based on the Xilinx Artix-7).

RISC-V on an FPGA, pt. 2

The first step is to install the enormous, proprietary Xilinx Vivado software. (Yes, all FPGA stuff is proprietary and strange). You can follow the general instructions here. The install took a total of 41GB of disk space (no, that is not a mistake), and took a few hours, but is otherwise straightforward.

The difficult bit was getting the Vivado software to actually see the hardware. It turns out that Xilinx use a proprietary USB driver, and, well, long story short you have to run the `install_drivers` script buried deep in the Vivado directory tree. All it does is put a couple of files under `/etc/udev/rules.d`, but it didn't do `udevadm control --reload-rules` so you have to do that as well, replug the cable, and it should be detectable in Vivado:


Vivado 2016.2
File Edit Flow Tools Window Layout View Help
Default Layout
Dashboard
Hardware Manager - localhost/xilinx_tcf/Digilent/210292744271A
There are no debug cores. Program device Refresh device
Hardware
Hardware Device Properties
General Properties
Tcl Console
Type a Tcl command here
Tcl Console Messages Serial I/O Links Serial I/O Scans

Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/21029274...	Open
xc7a100t_0 (1)	Programmed
XADC (System Monitor)	

Name:	xc7a100t_0
Part:	xc7a100t
ID code:	13631093
IR length:	6
Status:	Programmed
Programming file:	
Probes file:	
User chain count:	4

```
open_hw_target
INFO: [Labtoolstcl 44-466] Opening hw_target localhost:3121/xilinx_tcf/Digilent/210292744271A
current_hw_device [lindex [get_hw_devices] 0]
refresh_hw_device -update_hw_probes false [lindex [get_hw_devices] 0]
INFO: [Labtools 27-1434] Device xc7a100t (JTAG device index = 0) is programmed with a design that has no supported debug core(s) in it.
WARNING: [Labtools 27-3123] The debug hub core was not detected at User Scan Chain 1 or 3.
Resolution:
1. Make sure the clock connected to the debug hub (dbg_hub) core is a free running clock and is active OR
2. Manually launch hw_server with -e "set xsdb-user-bscan <C_USER_SCAN_CHAIN scan_chain_number>" to detect the debug hub at User Scan Chain of 2 or 4. To dete
```

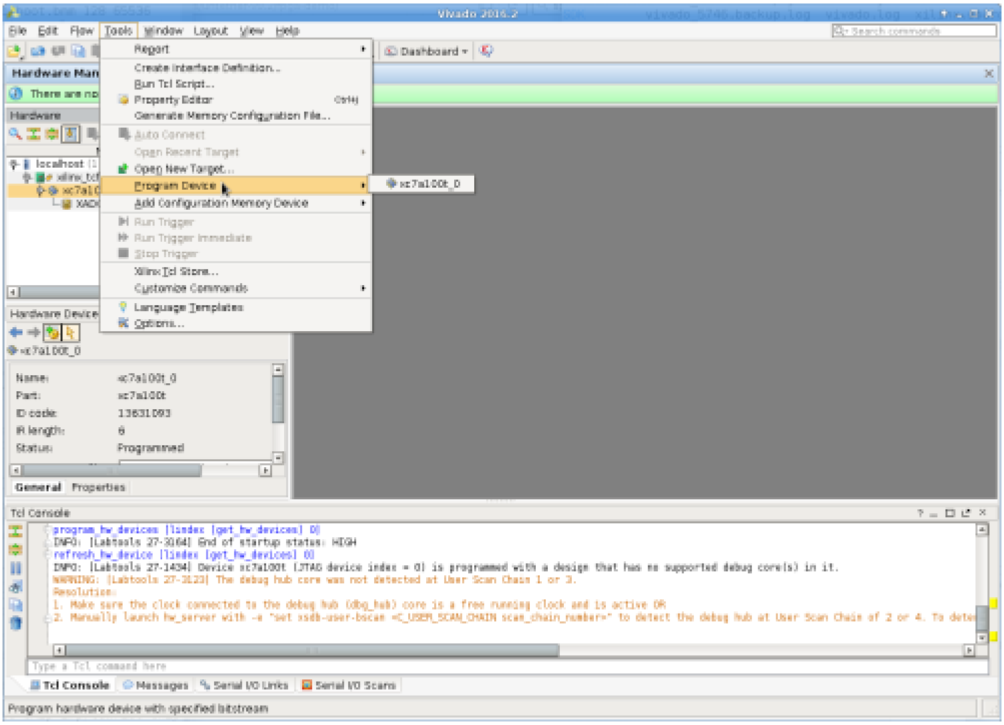
RISC-V on an FPGA, pt. 3

Compiling the lowRISC software, full GCC, Linux and busybox was straightforward. You have to follow the documentation here very carefully. The only small problem I had was their environment variable script uses `$TOP` which is also used by something else (bash? – not sure, anyway you just have to modify their script so it doesn't use that name).

The above gets you enough to boot Linux on Spike, the RISC-V functional emulator. That's not interesting to me, let's see it running on my FPGA instead.

LowRISC again provide detailed instructions for compiling the FPGA which you have to follow carefully. I had to remove the `-version` flags in a script, but otherwise it went fine.

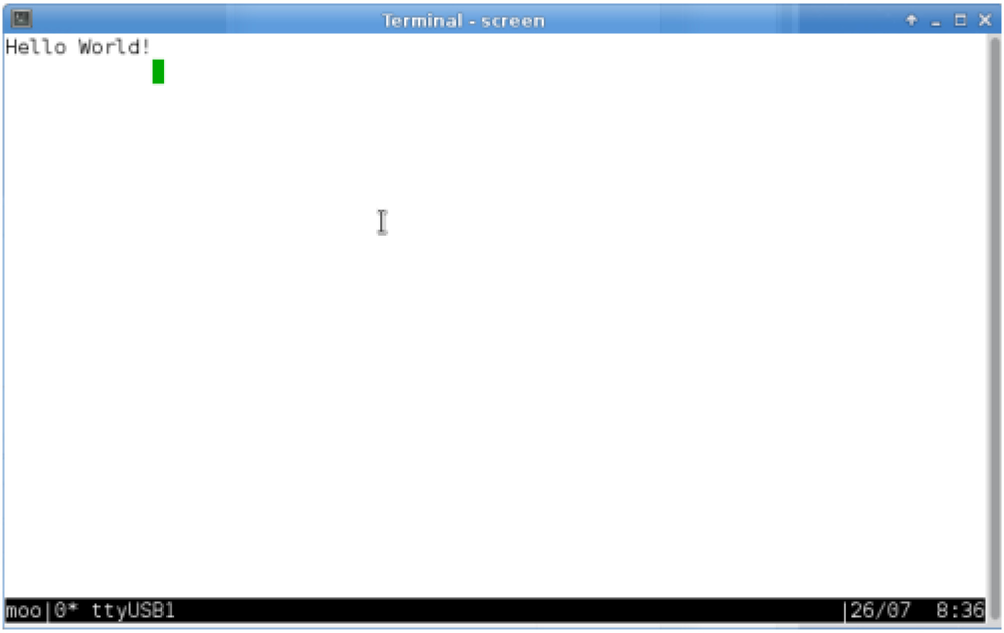
It wasn't clear to me what you're supposed to do with the final bitstream (`./lowrisc-chip-imp/lowrisc-chip-imp.runs/impl_1/chip_top.new.bit`) file, but in fact you use it in Vivado to program the device:



Name:	xc7a100t_0
Part:	xc7a100t
ID code:	13631093
IR length:	6
Status:	Programmed

```
program_hw_devices [lindex [get_hw_devices] 0]
INFO: [Labtools 27-3164] End of startup status: HIGH
refresh_hw_device [lindex [get_hw_devices] 0]
INFO: [Labtools 27-1434] Device xc7a100t (JTAG device index = 0) is programmed with a design that has no supported debug core(s) in it.
WARNING: [Labtools 27-3123] The debug hub core was not detected at User Scan Chain 1 or 3.
Resolution:
1. Make sure the clock connected to the debug hub (dbg_hub) core is a free running clock and is active OR
2. Manually launch hw_server with -e "set xsdb-user-bscan <C_USER_SCAN_CHAIN scan_chain_number>" to detect the debug hub at User Scan Chain of 2 or 4. To dete
```

The simple hello world example was successful (output shown below is from `/dev/ttyUSB1` connected to the dev board):



```
Hello World!

moo|0* ttyUSB1                                                          |26/07   8:36
```

RISC-V on an FPGA, pt. 4

It boots!

```
lowRISC boot program
=====================================
Load boot into memory
[    0.000000] Linux version 3.14.41-g9a25e8d (rjones@moo
.home.annexia.org) (gcc version 5.2.0 (GCC) ) #1 Mon Jul
25 19:07:50 BST 2016
[    0.000000] Available physical memory: 126MB
[    0.000000] Zone ranges:
[    0.000000]   Normal   [mem 0x00200000-0x07ffffff]
[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
[    0.000000]   node   0: [mem 0x00200000-0x07ffffff]
[    0.000000] Built 1 zonelists in Zone order, mobility
grouping on.  Total pages: 31815
[    0.000000] Kernel command line: root=/dev/htifblk0
[    0.000000] PID hash table entries: 512 (order: 0, 409
6 bytes)
[    0.000000] Dentry cache hash table entries: 16384 (or
der: 5, 131072 bytes)
[    0.000000] Inode-cache hash table entries: 8192 (orde
r: 4, 65536 bytes)
[    0.000000] Sorting __ex_table...
[    0.000000] Memory: 124488K/129024K available (1725K k
ernel code, 120K rwdata, 356K rodata, 68K init, 211K bss,
 4536K reserved)
[    0.000000] SLUB: HWalign=64, Order=0-3, MinObjects=0,
 CPUs=1, Nodes=1
[    0.000000] NR_IRQS:2
[    0.150000] Calibrating delay using timer specific rou
tine.. 20.01 BogoMIPS (lpj=100097)
[    0.150000] pid_max: default: 32768 minimum: 301
[    0.150000] Mount-cache hash table entries: 512 (order
: 0, 4096 bytes)
[    0.150000] Mountpoint-cache hash table entries: 512 (
order: 0, 4096 bytes)
[    0.150000] devtmpfs: initialized
[    0.150000] NET: Registered protocol family 16
[    0.150000] bio: create slab  at 0
[    0.150000] Switched to clocksource riscv_clocksource
[    0.150000] NET: Registered protocol family 2
[    0.150000] TCP established hash table entries: 1024 (
order: 1, 8192 bytes)
[    0.150000] TCP bind hash table entries: 1024 (order:
1, 8192 bytes)
[    0.150000] TCP: Hash tables configured (established 1
024 bind 1024)
[    0.150000] TCP: reno registered
[    0.150000] UDP hash table entries: 256 (order: 1, 819
2 bytes)
[    0.150000] UDP-Lite hash table entries: 256 (order: 1
, 8192 bytes)
[    0.150000] NET: Registered protocol family 1
[    0.150000] futex hash table entries: 256 (order: 0, 6
144 bytes)
[    0.150000] io scheduler noop registered
[    0.150000] io scheduler cfq registered (default)
[    0.180000] htifcon htif0: detected console
[    0.190000] console [htifcon0] enabled
[    0.190000] htifblk htif1: detected disk
[    0.190000] htifblk htif1: added htifblk0
[    0.190000] TCP: cubic registered
[    0.190000] VFS: Mounted root (ext2 filesystem) readon
ly on device 254:0.
[    0.190000] devtmpfs: mounted
[    0.190000] Freeing unused kernel memory: 68K (ffffff
f80000000 - ffffffff80011000)
# uname -a
Linux ucbvax 3.14.41-g9a25e8d #1 Mon Jul 25 19:07:50 BST
2016 riscv GNU/Linux
```

RISC-V on an FPGA, pt. 5

I've learned a few things about this process. These are just my random thoughts in no particular order:

- You need the heavily proprietary Xilinx Vivado to compile the Verilog source code to a bitstream. However you can use free tools to write the bitstream to the FPGA (eg. xc3sprog). **Edit:** There is another option: You can write the bitstream onto the SD-card and switch the jumper JP1 to SD-card. On boot the FPGA will load the bitstream from the SD-card. This is actually much nicer because it means you don't need to manually reprogram the FPGA every time you power it up.
- The core lowRISC CPU is free, but that's not everything that goes into the bitstream. Also in the bitstream are some peripherals, such as the UART, and those are Xilinx proprietary IP. So the bitstream isn't fully free and more importantly isn't redistributable. (Note, according to this talk there is a plan to fix this).
- This is a nice summary of the v0.2 architecture, esp. page 4

RISC-V on an FPGA, pt. 6

(Click for larger image)

The board on the left is the Digilent Nexys 4 DDR which I was using yesterday to boot Linux. It costs £261 (about $341) including tax and next day delivery. The board on the right is the cheaper Digilent Arty Board, which cost me $148 (two day delivery from the US) + £20 tax.

There are clear differences in the number of connectors, LEDs and buttons. The Nexys has VGA, ethernet, USB, 8 digit LED, many lights, a temperature sensor, and lots of buttons. The Arty has just the bare minimum as you'd expect given the price difference. Both boards use the same Xilinx Artix-7 family of FPGA, **but they are not exactly the same.** The expensive board on the left uses the XC7A100T which has 100K+ logic cells, the cheap board on the right uses the

XC7A35T with 33K cells. What this means practically is that you are more limited in the size of designs which can be programmed on the smaller chip. Does this matter for RISC-V? Yes and no. The untether-v0.2 design I was using yesterday takes about 50K cells so it won't fit on the smaller board. However SiFive apparently (I have not checked) have a reduced design that will fit. (Note that none of this affects the operating system or available RAM — those are separate issues. However a smaller design will have to cut a few corners, leave out parts of the chip that implement optimizations, etc and so will generally run slower).

Oddly the Arty Board (on the right) has *more* DDR3 RAM — 256MB vs 128MB. That is an improvement, since doing any real Linux work in 128MB of RAM is tough, but still not massive. The other significant difference is the Arty Board does not have a microSD-card socket. It has (just) 16MB of on-board flash instead. I'm not clear how you get Linux on there, but that's what I'll be exploring.

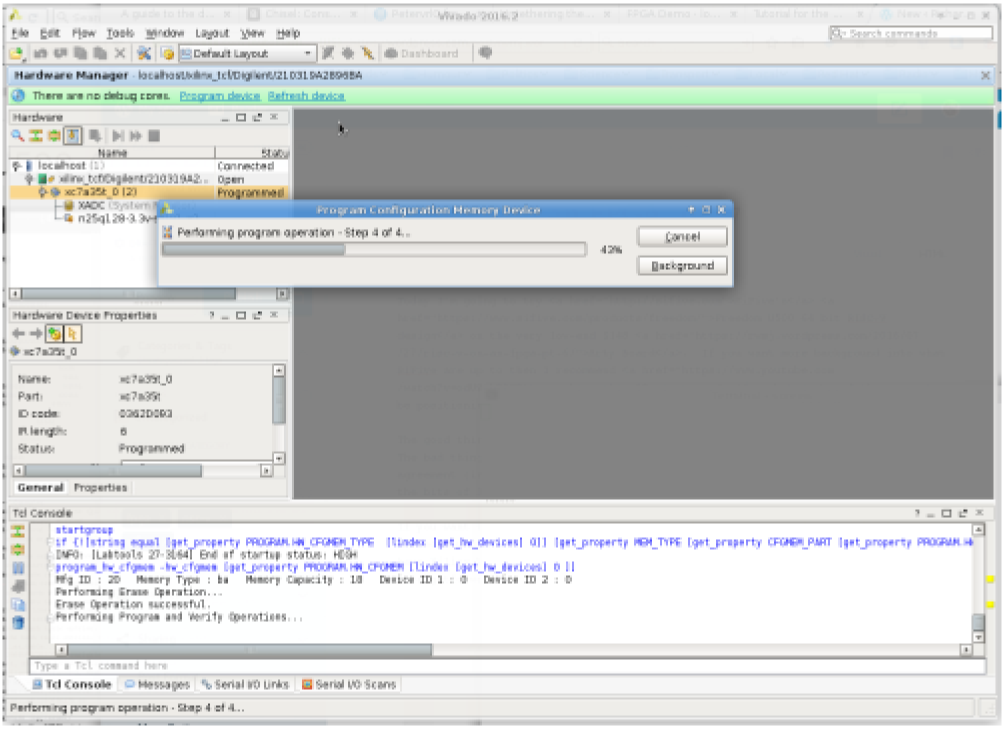
Finally it's worth saying that both boards are incomplete, although in very minor ways. The Nexys 4 comes with an OTG USB cable, which is all you need to power the board, program it and use the serial port. However it omits a microSD-card which you will need to store Linux / other RISC-V software that you want to run. The Arty comes without any cables and thus requires that you supply an OTG USB cable. As mentioned above there seems to be no microSD-card option at all.

RISC-V on an FPGA, pt. 7

Today I'm going to try SiFive's Freedom U500 64 bit RISC-V design on the very low-end $148 Arty Board. If you want more background into what SiFive are up to then I recommend watching this 15 minute video, but in brief they seem to be positioning themselves as a distributor and integrator of RISC-V.

The good thing is they compile everything you need including the Xilinx bitstream. The bad thing is that these are behind a registration wall with a poisonous license agreement (in particular, non-commercial use only). I hope this is only because of the bits of 3rd party proprietary IP they are using for ethernet, flash, UART, etc.

If you want to do this yourself, read the getting started guide here. Assuming you have Xilinx Vivado installed, following the instructions is completely straightforward except for one point: You need to do "Boot from Configuration Memory device" after programming. Anyway you will have a booting Linux/RISC-V in a few minutes.


Vivado 2016.2
File Edit Flow Tools Window Layout View Help
Default Layout
Dashboard
Hardware Manager - localhost/xilinx_tcf/Digilent/210319A28958A
There are no debug cores. Program device Refresh device
Hardware
Name
Status
localhost (1)
Connected
xilinx_tcf/Digilent/21031942...
Open
xc7a35t_0 (2)
Programmed
XADC (System Monitor)
n25q128-3.3v
Program Configuration Memory Device
Performing program operation - Step 4 of 4...
42%
Cancel
Background
Hardware Device Properties
xc7a35t_0
General
Properties
Tcl Console
Type a Tcl command here
Tcl Console
Messages
Serial I/O Links
Serial I/O Scans
Performing program operation - Step 4 of 4...

Name:	xc7a35t_0
Part:	xc7a35t
ID code:	0362D093
IR length:	6
Status:	Programmed

```
startgroup
if {![string equal [get_property PROGRAM.HW_CFGMEM_TYPE  [lindex [get_hw_devices] 0]] [get_property MEM_TYPE [get_property CFGMEM_PART [get_property PROGRAM.HW
INFO: [Labtools 27-3164] End of startup status: HIGH
program_hw_cfgmem -hw_cfgmem [get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices] 0 ]]
Mfg ID : 20   Memory Type : ba   Memory Capacity : 18   Device ID 1 : 0   Device ID 2 : 0
Performing Erase Operation...
Erase Operation successful.
Performing Program and Verify Operations...
```

After the cut, the boot output.

```
              SiFive RISC-V Coreplex
[    0.000000] Linux version 4.6.2-ga9474b8 (aou@i0.inter
nal.sifive.com) (gcc version 6.1.0 (GCC) ) #3 Sun Jul 10
17:43:16 PDT 2016
[    0.000000] bootconsole [early0] enabled
[    0.000000] Available physical memory: 250MB
[    0.000000] Initial ramdisk at: 0xffffffff800140c0 (21
82877 bytes)
[    0.000000] Zone ranges:
[    0.000000]   Normal   [mem 0x0000000080600000-0x00000
0008fffffff]
[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
[    0.000000]   node   0: [mem 0x0000000080600000-0x0000
00008fffffff]
```

```
[    0.000000] Initmem setup node 0 [mem 0x0000000080600000-0x000000008fffffff]
[    0.000000] Built 1 zonelists in Zone order, mobility
grouping on.  Total pages: 63125
[    0.000000] Kernel command line: earlyprintk
[    0.000000] PID hash table entries: 1024 (order: 1, 81
92 bytes)
[    0.000000] Dentry cache hash table entries: 32768 (or
der: 6, 262144 bytes)
[    0.000000] Inode-cache hash table entries: 16384 (ord
er: 5, 131072 bytes)
[    0.000000] Sorting __ex_table...
[    0.000000] Memory: 246856K/256000K available (2187K k
ernel code, 113K rwdata, 416K rodata, 2216K init, 224K bs
s, 9144K reserved, 0K cma-reserved)
[    0.000000] SLUB: HWalign=64, Order=0-3, MinObjects=0,
 CPUs=1, Nodes=1
[    0.000000] NR_IRQS:0 nr_irqs:0 0
[    0.000000] clocksource: riscv_clocksource: mask: 0xff
ffffff max_cycles: 0xffffffff, max_idle_ns: 191126044627
 ns
[    0.000000] Calibrating delay loop (skipped), value ca
lculated using timer frequency.. 2.00 BogoMIPS (lpj=10000
)
```

```
[    0.000000] pid_max: default: 32768 minimum: 301
[    0.010000] Mount-cache hash table entries: 512 (order
: 0, 4096 bytes)
[    0.010000] Mountpoint-cache hash table entries: 512 (
order: 0, 4096 bytes)
[    0.050000] devtmpfs: initialized
[    0.070000] clocksource: jiffies: mask: 0xffffffff max
_cycles: 0xffffffff, max_idle_ns: 19112604462750000 ns
[    0.090000] NET: Registered protocol family 16
[    0.180000] clocksource: Switched to clocksource riscv
_clocksource
[    0.220000] NET: Registered protocol family 2
[    0.230000] TCP established hash table entries: 2048 (
order: 2, 16384 bytes)
[    0.240000] TCP bind hash table entries: 2048 (order:
2, 16384 bytes)
[    0.240000] TCP: Hash tables configured (established 2
048 bind 2048)
[    0.250000] UDP hash table entries: 256 (order: 1, 819
2 bytes)
[    0.250000] UDP-Lite hash table entries: 256 (order: 1
, 8192 bytes)
[    0.260000] NET: Registered protocol family 1
[    3.960000] Unpacking initramfs...
[    7.820000] console [sbi_console0] enabled
[    7.820000] console [sbi_console0] enabled
[    7.820000] bootconsole [early0] disabled
[    7.820000] bootconsole [early0] disabled
[    7.840000] futex hash table entries: 256 (order: 0, 6
144 bytes)
[    7.850000] workingset: timestamp_bits=61 max_order=16
 bucket_order=0
[    8.250000] io scheduler noop registered
[    8.260000] io scheduler cfq registered (default)
[    9.780000] gpio leds: loaded 4 GPIOs
[    9.870000] Freeing unused kernel memory: 2216K (fffff
fff80000000 - ffffffff8022a000)
[    9.880000] This architecture does not have kernel mem
ory protection.
Starting logging: OK
Starting network...

Welcome to Buildroot
sifive login: root
# cat /proc/cpuinfo
hart    : 0
isa     : RV64G

# uname -a
Linux sifive 4.6.2-ga9474b8 #3 Sun Jul 10 17:43:16 PDT 20
16 riscv GNU/Linux
```

RISC-V on an FPGA, pt. 8

Some thoughts on SiFive Freedom U500 on the low-end Arty Board:

- 256MB (249MB available), vs 128MB for the Nexys 4. However memory is used for the filesystem initramfs / tmpfs (see next point) so there is a trade-off between RAM and storage. Recall the Nexys 4 has a microSD card for storage.
- The root filesystem is the initramfs, presumably loaded off the 16MB of SPI flash included with the board. So it's more like an embedded target than something you could do any development on.
- Slow – noticeably slower than lowRISC on the Nexys 4. However it's also under half the price, and this is an FPGA design, and the performance of the real hardware will be completely different.
- I was kind of expecting that ethernet would work, but in any case it doesn't appear to work for me.
- You can replace their Linux kernel and root image with your own — see section 5 in the documentation. However that step, as well as programming the board, requires the proprietary Vivado software. AFAICT there is no free software alternative.