

果壳 FPGA 平台

陈煦豪

测试软件包括 Vivado 和 minicom。

Vivado 软件负责与目标机的 JTAG 接口交互，顺序完成三部分工作：

- 向目标机的 JTAG 接口传输 bitstream，完成 FPGA 上逻辑的烧写
- 将 linux.bin 通过 JTAG 接口传输给 DDR4 SODIMM 存储
- 通过 JTAG 接口配置 vio 软复位，解除 FPGA 上核(CPU)的复位状态

minicom 软件负责与目标机的 UART 接口交互，其功能为，在核执行目标代码时，输出串口打印的信息。

工程资源：https://github.com/ssdfghhghh/NutShell_U250.git

环境：32 服务器

一：生成 Bitstream

1:需要：verilog 代码，位于/NutShell_U250/build。在文件夹中替换 Topmain 后需要在 vivado 中更新文件。

配置：修改 Makefile 文件

```
DATAWIDTH ?= 64
BOARD ?= pynq # sim pynq axu3cg
CORE ?= inorder # inorder ooo embedded
```

修改\src\main\scala\top\Settings.scala

```
object PynqSettings {
  def apply() = Map(
    "FPGAPlatform" -> true,
    "NrExtIntr" -> 3,
    "ResetVector" -> 0x80000000L,
    "MemMapBase" -> 0x0000000080000000L,
    "MemMapRegionBits" -> 28,
    "MMIOBase" -> 0x0000000040600000L,
    "MMIOSize" -> 0x0000000001000000L
  )
}
```

2:Generate Bitstream

3:最终需要两个文件：位于

path/NutShell_U250/fpga/board/pynq/build/cxhU250_pynq/cxhU250_pynq.runs/impl_1/

(1) System_top_wrapper.bit

(2) System_top_wrapper.ltx

4:使用 vivado 内 hardware manager 打开一块 U250 板卡，右键 xcu250 选择 program device，烧写上述两个文件。注：避免使用 21330409X021A 板卡，这张卡有用户在使用。

二：编译程序

1：目标机上运行的目标代码为*.bin。编译*.bin 需要使用果壳可用的 Riscv 工具链。

- 1: 命令行 `sudo minicom` 打开串口观察工具, 调整至 fpga 板卡对应接口 (21330409X021A 对应 `ttyUSB2`), 波特率 115200

图 3: minicom 界面

```
File Edit View Search Terminal Help
#####
ffffffffffffff vvvvvvvvvvvvvvvvvvvv
ffffffffffffff vvvvvvvvvvvvvvvvvvvv
ffffffffffffff vvvvvvvvvvvvvvvvvvvv
ffffffffffffff vvvvvvvvvvvvvvvvvvvv
ffffffffffffff vvvvvvvvvvvvvvvvvvvv
ffffffffffffff vvvvvvvvvvvvvvvvvvvv
ff          vvvvvvvvvvvvvvvvvvvv ff
        vvvvvvvvvvvvvvvvvvvvvvvvvv ffff
fffff vvvvvvvvvvvvvvvvvvvvvvvvvv fffff
fffffff vvvvvvvvvvvvvvvvvvvvvvvv ffffffff
fffffffff vvvvvvvvvvvvvvvvvvvvvvvv ffffffff
fffffffffff vvvvvvvvvvvvvvvvvvvvvvvv ffffffff
fffffffffffff vvvvvvvvvvvvvvvvvvvvvvvv ffffffff
fffffffffffff vv          ffffffff
fffffffffffff          ffffffff
fffffffffffff          ffffffff
fffffffffffff          ffffffff
#####
INSTRUCTION SETS WANT TO BE FREE
[ 0.000000] OF: fdt: Ignoring memory range 0x80000000 - 0x80200000
[ 0.000000] Linux version 4.18.0-00046-g2ba394515c09-dirty (wkf@xiangshan-051
[ 0.000000] bootconsole [early0] enabled
[ 0.000000] Initial ramdisk at: 0x(____ptrval____) (23552 bytes)
[ 0.000000] Zone ranges:
[ 0.000000]   DMA32    empty
[ 0.000000]   Normal  [mem 0x000000000080200000-0x00000000081fffffff]
[ 0.000000] Movable zone start for each node
[ 0.000000] Early memory node ranges
[ 0.000000]   node 0: [mem 0x000000000080200000-0x00000000081fffffff]
[ 0.000000] Initmem setup node 0 [mem 0x000000000080200000-0x00000000081fffffff]
[ 0.000000] Cannot allocate SWIOTLB buffer
[ 0.000000] elf hwcap is 0x112d
[ 0.000000] Built 1 zonelists, mobility grouping on. Total pages: 7575
[ 0.000000] Kernel command line: root=/dev/mmcblk0 rootfstype=ext4 ro rootwan
[ 0.000000] Dentry cache hash table entries: 4096 (order: 3, 32768 bytes)
[ 0.000000] Inode-cache hash table entries: 2048 (order: 2, 16384 bytes)
[ 0.000000] Sorting __ex table...
[ 0.000000] ...available (666K kernel code, 80K data, 1)
```

```
File Edit View Search Terminal Help
[ 0.000000] Zone ranges:
[ 0.000000] DMA32 empty
[ 0.000000] Normal [mem 0x0000000000000000-0x0000000000000000]
[ 0.000000] Movable zone start for each node
[ 0.000000] Early memory node ranges
[ 0.000000] node 0: [mem 0x0000000000000000-0x0000000000000000]
[ 0.000000] Initmem setup node 0 [mem 0x0000000000000000-0x0000000000000000]
[ 0.000000] Cannot allocate SMITLB buffer
[ 0.000000] elf hwcap is 0x112d
[ 0.000000] Built 1 zonelists, mobility grouping on. Total pages: 7575
[ 0.000000] Kernel command line: root=/dev/mmcblk0 rootfstype=ext4 ro rootwan
[ 0.000000] Dentry cache hash table entries: 4096 (order: 3, 32768 bytes)
[ 0.000000] Inode-cache hash table entries: 2048 (order: 2, 16384 bytes)
[ 0.000000] Sorting _ex table...
[ 0.000000] Memory: 29140K/30720K available (666K kernel code, 80K rdata, 1)
[ 0.000000] SLUB: Hwalign=64, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
[ 0.000000] NR_IRQS: 0, nr_irqs: 0, preallocated irq: 0
[ 0.000000] clocksource: riscv_clocksource: mask: 0xffffffffffffffff max_cycles
[ 0.000000] console [hvc0] enabled
[ 0.000000] console [hvc0] enabled
[ 0.000000] bootconsole [early0] disabled
[ 0.000000] bootconsole [early0] disabled
[ 0.000000] Calibrating delay loop (skipped), value calculated using timer f)
[ 0.000000] pid_max: default: 4096 minimum: 301
[ 0.000000] Mount-cache hash table entries: 512 (order: 0, 4096 bytes)
[ 0.000000] Mountpoint-cache hash table entries: 512 (order: 0, 4096 bytes)
[ 0.010000] clocksource: jiffies: mask: 0xffffffff max_cycles: 0xffffffff, ms
[ 0.010000] clocksource: Switched to clocksource riscv_clocksource
[ 0.010000] Unpacking initramfs...
[ 0.020000] worksets: timestamp bits=62 max_order=13 bucket_order=0
[ 0.020000] random: get_random_bytes called from 0xffffffff0001998a with crn0
[ 0.020000] Freeing unused kernel memory: 88K
[ 0.020000] This architecture does not have kernel memory protection.
Hello, RISC-V World!
[ 0.020000] init[1]: unhandled signal 4 code 0x1 at 0x0000000000000000 in he]
[ 0.020000] CPU: 0 PID: 1 Comm: init Not tainted 4.18.0-00046-g2ba394515c09-3
[ 0.020000] sepc: 0000000000000000 ra : 0000000000000000 sp : 0000003fffdcb0
[ 0.020000] gp : 0000000000000000 tp : 0000000000000000 t0 : 0000000000000000
[ 0.030000] t1 : 0000000000000000 t2 : 0000000000000000 s0 : 0000000000000000
[ 0.030000] t3 : 0000000000000000 s1 : 0000000000000000
```

图 5、6：启动 linux 系统