使用OpenOCD 做调试配置文件如何写

不论是我们的FPGA 开发板还是客户自己做的FPGA（或者实际芯片） 开发板，都可以使用我们的蜂鸟调试器来做调试。 蜂鸟调试器的用户手册和驱动以及购买链接，都可以从芯来科技官网这边获取：RISC-V开发板_专业RISC-V处理器IP和芯片解决方案公司 (nucleisys.com)

然后蜂鸟调试器就是用OpenOCD 来控制， 搭配我们demosoc FPGA开发板的OpenOCD 的配置文件从我们的Nuclei SDK /Nuclei Studio IDE 中都可以获取， 这里也把内容贴出来：

```
adapter_khz     1000

interface ftdi
ftdi_vid_pid 0x0403 0x6010
ftdi_oscan1_mode off

## bindto 0.0.0.0 can be used to cover all available interfaces.
## Uncomment bindto line to enable remote machine debug
# bindto 0.0.0.0

## If ftdi_device_desc not specified, the device description is ignored during device selection.
## So if you want to specify a dedicated FTDI device, you can select following device description:
## "Dual RS232-HS" is for Nuclei HummingBird Debugger V1
## "USB <-> JTAG-DEBUGGER" is for Nuclei HummingBird Debugger V2
## Uncomment one which match your device description
# ftdi_device_desc "Dual RS232-HS"
# ftdi_device_desc "USB <-> JTAG-DEBUGGER"

transport select jtag

ftdi_layout_init 0x0008 0x001b
ftdi_layout_signal nSRST -oe 0x0020 -data 0x0020
ftdi_layout_signal TCK -data 0x0001
ftdi_layout_signal TDI -data 0x0002
ftdi_layout_signal TDO -input 0x0004
ftdi_layout_signal TMS -data 0x0008
ftdi_layout_signal JTAG_SEL -data 0x0100 -oe 0x0100

set _CHIPNAME riscv
jtag newtap $_CHIPNAME cpu -irlen 5

set _TARGETNAME $_CHIPNAME.cpu
target create $_TARGETNAME riscv -chain-position $_TARGETNAME
$_TARGETNAME configure -work-area-phys 0x80000000 -work-area-size 10000 -work-area-backup 1

set _FLASHNAME $_CHIPNAME.flash
flash bank $_FLASHNAME fespi 0x20000000 0 0 0 $_TARGETNAME
# Set the ILM space also as flash, to make sure it can be add breakpoint with hardware trigger
#flash bank onboard_ilm fespi 0x80000000 0 0 0 $_TARGETNAME

# Expose Nuclei self-defined CSRS
# See https://github.com/riscv/riscv-gnu-toolchain/issues/319#issuecomment-358397306
# Then user can view the csr register value in gdb using: info reg csr775 for CSR MTVT(0x307)
riscv expose_csrs 416-496,770-800,835-850,1227-1231,1483-1486,1984-2040,2064-2070,2370-2380,2490-2500,4032-4040

init

if {[ info exists pulse_srst]} {
  ftdi_set_signal nSRST 0
  ftdi_set_signal nSRST z
}
halt
# We must turn on this because otherwise the IDE version debug cannot download the program into flash
flash protect 0 0 last off
```

这里也注意几点：

1. **我们的蜂鸟调试器即支持JTAG， 也支持cJTAG, 如果是cJTAG， 这里只需要更改openocd 的配置文件就好，具体就是这一行：**

```
ftdi_oscan1_mode on
```

2. 如果需要openocd 支持flash 的烧写，以这个demo 的cfg 文件来说，设计到的就是如下几行：

```
set _FLASHNAME $_CHIPNAME.flash
flash bank $_FLASHNAME fespi 0x20000000 0 0 0 $_TARGETNAME

....

# We must turn on this because otherwise the IDE version debug cannot download the program into flash
flash protect 0 0 last off
```

A. 目前我们Nuclei 做好的openocd 只支持demosoc 里的spi flash 烧写（代号： fespi）和我们Nuclei 子系统里的SPI （代号：nuspi），如果是第三方flash 烧写，需要自行修改openocd 源代码加入自己的spi 驱动， 并对应修改openocd cfg 这里的代号；

B，flash bank 的命令从OpenOCD 的用户手册上看，总结如下：

flash bank name driver base size chip-width bus-width target [driver-options]	• name：flash的名称 • driver：flash的驱动 • base：flash的地址 • size：flash大小，bytes • chip-width：flash的宽度，bytes • bus-width：数据总线宽度，bytes • target：目标板名称 • [driver-options]：其他命令或参数

这里特别需要注意base size 还有[Driver-options] 这两个，前者是flash 的地址，后者需要注意，如果自己的Spi Control 的地址和我们Demo SoC 的配置不同，则这里要加上Spi Control 的地址，如下写法就表示，用的是Nuclei SPI IP，Flash 地址是从0x20000000 开始，Nuclei SPI Control 的地址从0x100c0000 开始。

```
flash bank $_FLASHNAME0 nuspi 0x20000000 0 0 0 $_CHIPNAME0.cpu 0x100c0000
```

C. 如果不需要openocd 做flash 烧写，对应cfg 文件也就拿掉flash 相关的描述。

3. 另外就是cfg 里的work area 的设置，具体参考我们FAQ 的这篇文章：Nuclei Studio 或者Nuclei SDK在load elf 文件时hang住? - 芯来科技支持系统 FAQ - WIKI (nucleisys.com)。

4. 然后openocd 默认是通过TCP 端口 3333 和GDB 做Socket 通信，如果3333 端口因为某些原因被占用，也可以在cfg 里指定别的端口，语法如下，这里改了后，GDB 那边也要用对应的端口来连接。

```
gdb_port 3334
```

5. 另外考虑到OpenOCD 和目标端通过jtag 来通信也是有延迟的，在某些情况下，默认的延迟不够，需要加大这个延迟，语法如下：

```
riscv set_command_timeout_sec 3000
```

6. 再就是RISC-V Debug 这边有一个特性叫做SBA (System Bus Access)， 这个特性是在Debug 访问外部地址内容时可以通过Debug Module 来直接访问，而不是通过CPU 来访问（Bypass CPU， 这样CPU 可以free run 的同时做debug， Segger Jlink 支持RISC-V 的调试，它有一个功能叫做RTT RTT - SEGGER Wiki， Real Time Transfer 就是通过SBA 来实现，当然Segger RTT 另一个功能是通过debgugger 来捕捉printf，这样不需要串口）， RISC-V OpenOCD 这边default 是不用SBA 特性， 如果需要用到SBA 特性，在cfg 里加入：

```
riscv set_prefer_sba on
```

7. 另外如果openocd 执行时报错，且log 不是很明显哪里出错时，可以在cfg 让OpenOCD 打出更多的log 出来，帮助大家来分析，具体做法就是在“init” 加入 debug_level 3：

```
...
debug_level 3
init
...
```

8. 如果是SMP 多核， openocd 里已经有“-rtos hwthread” 的支持，如下是一个我们Nuclei 双核900 的一个cfg 范例：

```
adapter_khz     1000

interface ftdi
ftdi_vid_pid 0x0403 0x6010
ftdi_oscan1_mode off

## bindto 0.0.0.0 can be used to cover all available interfaces.
## Uncomment bindto line to enable remote machine debug
# bindto 0.0.0.0

## If ftdi_device_desc not specified, the device description is ignored during device selection.
## So if you want to specify a dedicated FTDI device, you can select following device description:
## "Dual RS232-HS" is for Nuclei HummingBird Debugger V1
## "USB <-> JTAG-DEBUGGER" is for Nuclei HummingBird Debugger V2
## Uncomment one which match your device description
# ftdi_device_desc "Dual RS232-HS"
# ftdi_device_desc "USB <-> JTAG-DEBUGGER"

transport select jtag

ftdi_layout_init 0x0008 0x001b
ftdi_layout_signal nSRST -oe 0x0020 -data 0x0020
ftdi_layout_signal TCK -data 0x0001
ftdi_layout_signal TDI -data 0x0002
ftdi_layout_signal TDO -input 0x0004
ftdi_layout_signal TMS -data 0x0008
ftdi_layout_signal JTAG_SEL -data 0x0100 -oe 0x0100

set _CHIPNAME riscv
jtag newtap $_CHIPNAME cpu -irlen 5

set _TARGETNAME_0 $_CHIPNAME.cpu0
set _TARGETNAME_1 $_CHIPNAME.cpu1
target create $_TARGETNAME_0 riscv -chain-position $_CHIPNAME.cpu -coreid 0 -rtos hwthread
target create $_TARGETNAME_1 riscv -chain-position $_CHIPNAME.cpu -coreid 1

target smp $_TARGETNAME_0 $_TARGETNAME_1

$_TARGETNAME configure -work-area-phys 0x80000000 -work-area-size 10000 -work-area-backup 1

set _FLASHNAME $_CHIPNAME.flash
flash bank $_FLASHNAME fespi 0x20000000 0 0 0 $_TARGETNAME
# Set the ILM space also as flash, to make sure it can be add breakpoint with hardware trigger
#flash bank onboard_ilm fespi 0x80000000 0 0 0 $_TARGETNAME

# Expose Nuclei self-defined CSRS
# See https://github.com/riscv/riscv-gnu-toolchain/issues/319#issuecomment-358397306
# Then user can view the csr register value in gdb using: info reg csr775 for CSR MTVT(0x307)
riscv expose_csrs 416-496,770-800,835-850,1227-1231,1483-1486,1984-2040,2064-2070,2370-2380,2490-2500,4032-4040

init

if {[ info exists pulse_srst]} {
  ftdi_set_signal nSRST 0
  ftdi_set_signal nSRST z
}
halt
# We must turn on this because otherwise the IDE version debug cannot download the program into flash
flash protect 0 0 last off
```

9. 如果是AMP 多核，多个core 用jtag chain 在一起，如下是我们的双核300 的一个cfg 范例：

```
adapter_khz     1000

interface ftdi
ftdi_vid_pid 0x0403 0x6010
ftdi_oscan1_mode off

## bindto 0.0.0.0 can be used to cover all available interfaces.
## Uncomment bindto line to enable remote machine debug
# bindto 0.0.0.0

## If ftdi_device_desc not specified, the device description is ignored during device selection.
## So if you want to specify a dedicated FTDI device, you can select following device description:
## "Dual RS232-HS" is for Nuclei HummingBird Debugger V1
## "USB <-> JTAG-DEBUGGER" is for Nuclei HummingBird Debugger V2
## Uncomment one which match your device description
# ftdi_device_desc "Dual RS232-HS"
# ftdi_device_desc "USB <-> JTAG-DEBUGGER"

transport select jtag

ftdi_layout_init 0x0008 0x001b
ftdi_layout_signal nSRST -oe 0x0020 -data 0x0020
ftdi_layout_signal TCK -data 0x0001
ftdi_layout_signal TDI -data 0x0002
ftdi_layout_signal TDO -input 0x0004
ftdi_layout_signal TMS -data 0x0008
ftdi_layout_signal JTAG_SEL -data 0x0100 -oe 0x0100


set _CHIPNAME0 riscv0

jtag newtap $_CHIPNAME0 cpu -irlen 5
set _TARGETNAME0_0 $_CHIPNAME0.cpu
target create $_TARGETNAME0_0 riscv -chain-position $_CHIPNAME0.cpu -coreid 0

set _CHIPNAME1 riscv1

jtag newtap $_CHIPNAME1 cpu -irlen 5
set _TARGETNAME1_0 $_CHIPNAME1.cpu
target create $_TARGETNAME1_0 riscv -chain-position $_CHIPNAME1.cpu -coreid 0

$_TARGETNAME0_0 configure -work-area-phys 0x88100000 -work-area-size 10000 -work-area-backup 1
$_TARGETNAME1_0 configure -work-area-phys 0x88000000 -work-area-size 10000 -work-area-backup 1

  # Expose Nuclei self-defined CSRS
# See https://github.com/riscv/riscv-gnu-toolchain/issues/319#issuecomment-358397306
# Then user can view the csr register value in gdb using: info reg csr775 for CSR MTVT(0x307)
riscv expose_csrs 416-496,770-800,835-850,1227-1231,1483-1486,1984-2040,2064-2070,2370-2380,2490-2500,4032-4040

init

if {[ info exists pulse_srst]} {
  ftdi_set_signal nSRST 0
  ftdi_set_signal nSRST z
}
halt
```

10. 如果是AMP 多核，然后AMP 多核有些core 因为一些原因对于debug 不available （比如一些core 处在stop on reset 状态，或者被power gating），那么需要让openocd 知道哪些core 可以被debug，可以被debug 的core 才需要target create， 比如如下是一个2个core 的jtag chain，但是只让openocd debug core 0的cfg：

```
adapter_khz     1000

interface ftdi
ftdi_vid_pid 0x0403 0x6010
ftdi_oscan1_mode off

## If ftdi_device_desc not specified, the device description is ignored during device selection.
## So if you want to specify a dedicated FTDI device, you can select following device description:
## "Dual RS232-HS" is for HummingBird Debugger V1
## "USB <-> JTAG-DEBUGGER" is for HummingBird Debugger V2
## Uncomment one which match your device description
# ftdi_device_desc "Dual RS232-HS"
# ftdi_device_desc "USB <-> JTAG-DEBUGGER"

transport select jtag

ftdi_layout_init 0x0008 0x001b
ftdi_layout_signal nSRST -oe 0x0020 -data 0x0020
ftdi_layout_signal TCK -data 0x0001
ftdi_layout_signal TDI -data 0x0002
ftdi_layout_signal TDO -input 0x0004
ftdi_layout_signal TMS -data 0x0008
ftdi_layout_signal JTAG_SEL -data 0x0100 -oe 0x0100

#core 0 can be debug
set _CHIPNAME0 riscv0
jtag newtap $_CHIPNAME0 cpu -irlen 5
set _TARGETNAME0 $_CHIPNAME0.cpu
target create $_TARGETNAME0 riscv -chain-position $_TARGETNAME0 -coreid 0

#core 1 just list but not be debug this time
set _CHIPNAME1 riscv1
jtag newtap $_CHIPNAME1 cpu -irlen 5 -expected-id 0x10000fff


$_TARGETNAME0 configure -work-area-phys 0x88100000 -work-area-size 10000 -work-area-backup 1

set _FLASHNAME1 $_CHIPNAME1.flash
flash bank $_FLASHNAME1 nuspi 0x90000000 0 0 0 $_TARGETNAME0 0x81050000

#jtag newtap $_CHIPNAME1 cpu -irlen 5 -expected-id 0x12030a6d

# Set the ILM space also as flash, to make sure it can be add breakpoint with hardware trigger
#flash bank onboard_ilm nuspi 0x80000000 0 0 0 $_TARGETNAME

# Expose Nuclei self-defined CSRS range 770-800,835-850,1984-2032,2064-2070
# See https:
# Then user can view the csr register value in gdb using: info reg csr775 for CSR MTVT(0x307)
#riscv expose_csrs 770-800,835-850,1984-2032,2064-2070

init
#reset
if {[ info exists pulse_srst]} {
  ftdi_set_signal nSRST 0
  ftdi_set_signal nSRST z
}

foreach t [target names] {
  targets $t
  halt
}
# We must turn on this because otherwise the IDE version debug cannot download the program into flash
flash protect 0 0 last off
```

除了我们的蜂鸟调试器，很多第三方调试器也可以直接来使用，包括众多FTDI 芯片在内的，比如Olimex 做的usb jtag 调试器（ARM-USB-TINY (olimex.com)），也比如Trace32 和Jlink。 其中Jlink 可能知道的人更多一些。如果用Jlink 来做调试，有如下注意点：

1. Jlink 是Segger 公司的商业产品，Segger 公司对外已经公开申明： 其硬件版本V9 以上支持RISC-V 调试（目前是V10 & V11），软件版本是v6.62 版本之上支持RISC-V 调试。
2. 我们芯来科技也和Segger 有官方合作，我们的所有系列的Core 型号可以在他们的软件工具里直接选择。
3. Jlink 搭配Segger 官方的软件也是同时支持JTAG 和cJTAG 两个接口的， 在软件工具界面上来选择。
4. 如果用Jlink， 我们建议使用Jlink 配套的软件来使用； 如果想使用Jlink + openocd， 也是可以，建议先看Segger 公司官方wiki：OpenOCD - SEGGER Wiki。 这里注意三点，A， 如果用OpenOCD， 那么只支持JTAG 接口； B, openocd 要控制Jlink需要替换JLink 自带的usb 驱动； C， 芯来科技的demosoc FPGA 板，对应的openocd cfg 文件如下：

```
adapter_khz     1000

# The below 2 lines configure openocd to work with jlink
interface jlink
transport select JTAG
#reset_config trst_only

set _CHIPNAME riscv
jtag newtap $_CHIPNAME cpu -irlen 5

set _TARGETNAME $_CHIPNAME.cpu
target create $_TARGETNAME riscv -chain-position $_TARGETNAME
$_TARGETNAME configure -work-area-phys 0x80000000 -work-area-size 10000 -work-area-backup 1

set _FLASHNAME $_CHIPNAME.flash
flash bank $_FLASHNAME fespi 0x20000000 0 0 0 $_TARGETNAME
# Set the ILM space also as flash, to make sure it can be add breakpoint with hardware trigger
#flash bank onboard_ilm fespi 0x80000000 0 0 0 $_TARGETNAME

# Expose Nuclei self-defined CSRS
# See https://github.com/riscv/riscv-gnu-toolchain/issues/319#issuecomment-358397306
# Then user can view the csr register value in gdb using: info reg csr775 for CSR MTVT(0x307)
riscv expose_csrs 416-496,770-800,835-850,1227-1231,1483-1486,1984-2040,2064-2070,2370-2380,2490-2500,4032-4040

init

if {[ info exists pulse_srst]} {
  ftdi_set_signal nSRST 0
  ftdi_set_signal nSRST z
}
halt
# We must turn on this because otherwise the IDE version debug cannot download the program into flash
flash protect 0 0 last off
```