

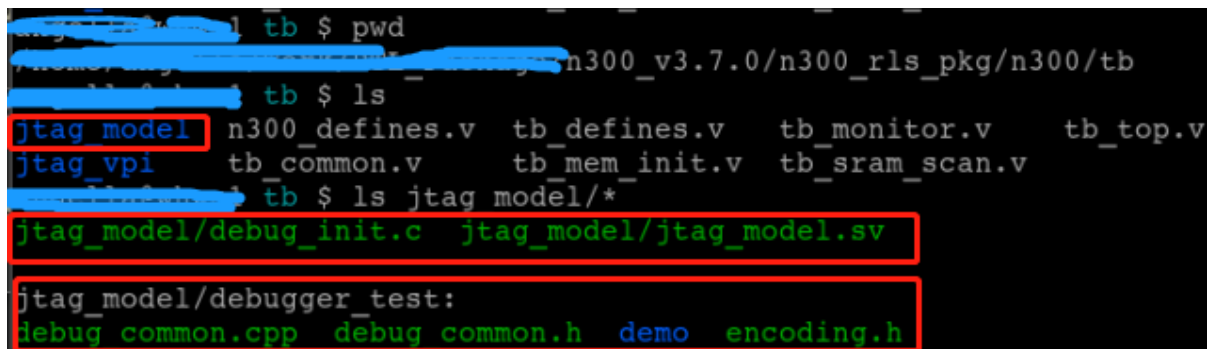
# Nuclei Core Deliver Package 如何集成JTAG Model Task

Nuclei Core deliver package 已经有了JTAG VPI的task，它是基于Verilog的PLI-VPI机制，用C来模拟JTAG的驱动机制，然后提供OpenOCD的连接server。这样就可以依靠OpenOCD + GDB来实现RISC-V Debug功能，也就可以在RTL仿真环境来检测JTAG Debug功能是否正常。关于JTAG\_VPI的TASK如何使用，我们的《Nuclei

Core Integration Guide》相应章节有step by step的描述。

后面一些客户反馈，JTAG\_VPI的task，虽然可以在仿真阶段验证JTAG Debug功能，但需要依赖OpenOCD和GDB，同时因为要开两个terminal窗口，手动操作还好，自动化操作就不顺畅了，然后只支持JTAG接口不支持cJTAG接口。基于客户这个反馈，我们Nuclei开发了新的JTAG Model的task，它是基于SystemVerilog的DPI机制，然后它直接模拟了openocd和gdb，用户可以直接在这边用C去调用jtag\_model提供的C API去读写memory，读写GPR，CSR。所以更集成也更适合仿真环境下的自动化测试。然后即支持JTAG也支持cJTAG。JTAG Model Task的package目前在我们企业微信的微盘上（<https://drive.weixin.qq.com/s?k=ABcAKgdSAFc2MWg4n1>），目前还没有整合到Deliver Package中，本文以N300的Deliver Package为例说明如何step by step的使用它。

1. 解压附件压缩包，有一个文件夹以及一个文档，文件夹下有sv和c代码，文档说明大致步骤和一些C API的含义，其中C API说明对于用户需要做什么样的测试非常有帮助。
2. 模拟JTAG\_VPI的做法，在tb目录下简历了一个jtag\_model的文件夹，把step 1的文件夹的内容放过来，如下图：



```
tb $ pwd
n300_v3.7.0/n300_rls_pkg/n300/tb
tb $ ls
jtag_model  n300_defines.v  tb_defines.v  tb_monitor.v  tb_top.v
jtag_vpi    tb_common.v      tb_mem_init.v  tb_sram_scan.v
tb $ ls jtag_model/*
jtag_model/debug_init.c  jtag_model/jtag_model.sv
jtag_model/debugger_test:
debug_common.cpp  debug_common.h  demo  encoding.h
```

3. 然后，我们在tb这边的tb\_top.v去集成jtag\_model，具体添加的地方如下：

```

tb_top.v
37 .enable      (jtag_done),
38 .init_done   (jtag_done)
39 );
40 `endif
41
42 wire          jtag_tck      ;
43 wire          jtag_tms      ;
44 wire          jtag_tms_out   ;
45 wire          jtag_tms_out_en ;
46 wire          jtag_tdi;
47 wire          jtag_tdo;
48
49 jtag_model
50 #(.DELAY(200))
51 u_jtag_model (
52   .clk      (aon_clk),
53   .rst_n    (tb_rst_n),
54
55   .jtag_tck (jtag_tck),
56   .jtag_tms (jtag_tms),
57   .jtag_tms_in (jtag_tms_out),
58   .jtag_tms_in_en (jtag_tms_out_en),
59
60   .jtag_tdi (jtag_tdi),
61   .jtag_tdo (jtag_tdo)
62 );
63
64
65 initial begin
66   nex_clk <=1'b0;
67   sys_clk <=1'b0;
68   aon_clk <=1'b0;
69   tb_rst_n <=1'b0;
70   nex_rst_n <= 1'b0;
71   #2000 tb_rst_n <=1'b1;
72   #2000 nex_rst_n <=1'b1;
73 //   #2000 tb_rst_n <=1'b0;

```

```

155 .io_pads_jtag_tdo_drv_o_oval (),
156 .io_pads_jtag_TMS_out_o_oval (jtag_tms_out),
157 .io_pads_jtag_DRV_TMS_o_oval (jtag_tms_out_en),
158 .io_pads_jtag_bk_o_oval (),
159
160 `ifdef N300_JTAGVPI
161 .io_pads_jtag_TCK_i_ival (jtag_TCK),
162 .io_pads_jtag_TMS_i_ival (jtag_TMS),
163 .io_pads_jtag_TDI_i_ival (jtag_TDI),
164 .io_pads_jtag_TDO_o_oval (jtag_TDO),
165 `else
166 .io_pads_jtag_TCK_i_ival (jtag_tck),
167 .io_pads_jtag_TMS_i_ival (jtag_tms),
168 .io_pads_jtag_TDI_i_ival (jtag_tdi),
169 .io_pads_jtag_TDO_o_oval (jtag_tdo),
170 `endif
171

```

让jtag\_module和tb里的core jtag 连接, 也注意, jtag\_module 不要和JTAG\_VPI 同时使用, 所以signal 连接可以写在这里

4. 集成完后, 就需要让300的tb 这边仿真编译时要加上jtag\_model.sv和debug\_init.c, 这个需要改动vsim/run下的makefile, 如下:

```
Makefile (~/.Work/RTL_Package/n300_v3.7.0/n300_rls_pkg/n300/vsim/run) - GVIM9
File Edit Tools Syntax Buffers Window Help
Makefile
32 RTL_V_FILES += $(wildcard ${VSRG_DIR}/*.v)
33 RTL_V_FILES += $(wildcard ${VSRG_DIR}/**/*.v)
34 TB_V_FILES := $(wildcard ${VTB_DIR}/*.v)
35 ifeq ($(JTAGVPI), 1)
36 TB_V_FILES += $(wildcard ${VTB_DIR}/jtag_vpi/*.v)
37 endif
38
39 TB_V_FILES += $(wildcard ${VTB_DIR}/jtag_model/*.sv)
40
41 VDB_FILES := $(wildcard ${RUN_DIR}/rv32*/simv.vdb)
42
43 # The following portion is depending on the EDA tools you are using, Please add them by yourself according
  to your EDA vendors
44 SIM_TOOL := vcs
45
46 SIM_OPTIONS := +v2k -sverilog -notice -q +lint=all,noSVA-NSVU,noPCTIO-L,noVCDE,noUI,noSVA-CE,noSVA-DIU,
  noPORTFRC,noSVA-ICP,noNS -debug_access=all -full64 -timescale=1ns/10ps
47 SIM_OPTIONS += +incdir+${VSRG_DIR}/core+${VSRG_DIR}/soc+${VTB_DIR}
48 SIM_OPTIONS += +define+DISABLE_SV_ASSERTION
49 SIM_OPTIONS += -l compile.log
```

```
64
65 ifeq ($(JTAGVPI),1)
66 SIM_OPTIONS += +define+${CORE_NAME}_JTAGVPI
67 SIM_OPTIONS += -P ${VTB_DIR}/jtag_vpi/jtag_vpi.tab -CC "-DVCS_VPI" ${VTB_DIR}/jtag_vpi/jtag_vpi.c
68 SIMV_FLAGS += +jtagvpi
69 SIMV_FLAGS += +jtag_port=${JTAGPORT}
70 endif
71
72 ifeq ($(TESTNAME),rv_sram_scan)
73 SIMV_FLAGS += +sram_check
74 endif
75
76 SIM_OPTIONS += -CC "-DVCS_VPI" ${VTB_DIR}/jtag_model/debug_init.c
77
78
79
80 all: run
```

5. 在vsim下创建helloworld的C Case（步骤可以参考《Nuclei Integration Guide》），然后run这个case，第一次run，可能会看到console上的这个打印，原因是如下图右侧所以，它需要step1的debugger.so这个动态库，且是在helloworld sim时的path下去找这一个：

```
ILM 0x03: 00000000
ILM 0x04: 800005ae
ILM 0x05: 800005ae
ILM 0x06: 800005ae
ILM 0x07: 00000000
ILM 0x16: 800005ae
ILM 0x20: 800005ae
SEED = 20220610145446
*Verdi* Loading libscore_vcs202012.so
FSDB Dumper for VCS, Release Verdi_R-2020.12-1, Linux x86_64/64bit, 01/17/2021
(C) 1996 - 2021 by Synopsys, Inc.
*Verdi* : Create FSDB file 'tb_top.fsdh'
*Verdi* : Begin traversing the scope (tb_top), layer (0).
*Verdi* : Enable +mda dumping.
*Verdi* : End of traversing.
*****
TESTCASE=
tag/n300_v3.7.0/n300_rls_pkg/n300/vsim/run/../../../../riscv-tests/isa/generated/helloworld
jtag_model use 4wire
None ext debug case
Nuclei SDK Build Time: Jun 10 2022, 14:05:33
Download Mode: ILM
CPU Frequency 1051852 Hz
Hart 0, MISA: 0x4014112d
MISA: RV32IMACFDSU
0: Hello World From Nuclei RISC-V Processor!
1: Hello World From Nuclei RISC-V Processor!
2: Hello World From Nuclei RISC-V Processor!
3: Hello World From Nuclei RISC-V Processor!

debug_init.c (~/.Work/RTL_Package/n300_v3.7.0/n300_rls_pkg/n300/tb/jtag_model) - GVIM7
File Edit Tools Syntax Buffers Window Help
debug_init.c
1
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <unistd.h>
5 #include <dlfcn.h>
6
7 // call debug init
8 void debug_init() {
9     if (!access("../debugger.so", F_OK)) {
10         printf("ext debug case\n");
11         void *handle = dlopen("../debugger.so", RTLD_LAZY);
12         void (*test_init)();
13         test_init = dlsym(handle, "dbg_test_init");
14         test_init();
15     }
16     else {
17         printf("none ext debug case\n");
18     }
19 }
```

6. 回答step1，编译debugger.so，然后copy到helloworld case 仿真路径：

```

angella@whws1 demo $ pwd
/home/angella/rtl_package/n300 v3.7.0/n300 rls pkg/n300/tb/jtag model/debugger test/demo
angella@whws1 demo $ make clean; make
rm *.elf *.diss *.so -rf
g++ -I../.. /debug_common.cpp -I/home/share/tools/edatools/synopsys/vcs/R-2020.12-1/include -shared -fPIC -g db
g test_code.cpp -o debugger.so
angella@whws1 demo $ cp debugger.so ../
debug_common.cpp debug_common.h demo/ encoding.h
angella@whws1 demo $ cp debugger.so ../../
debugger_test/ debug_init.c jtag_model.sv
angella@whws1 demo $ cp debugger.so ../../../vsim/
jtag_model/ n300_defines.v tb_defines.v tb_monitor.v tb_top.v
jtag_vpi/ tb_common.v tb_mem_init.v tb_sram_scan.v
angella@whws1 demo $ cp debugger.so ../../../vsim/run/helloworld/. -rf
bin/ helloworld/ install/ Makefile n300_defines.h run/
angella@whws1 demo $ cp debugger.so ../../../vsim/run/helloworld/. -rf
cp: overwrite './../../vsim/run/helloworld/.debugger.so'? y
angella@whws1 demo $

```

6. cp 过去后，我们只需要在vsim 目录下，在run\_test helloworld 就好，不过因为目前的makefile，每run 一次 case，run 目录下对应这个case 的目录都要被删除掉然后新建，这个会导致step 5 copy 过去的动态库被删掉，所以我们暂时先改makefile，不需要删掉文件夹。然后重新执行：make run\_test TESTNAME= helloworld. 运行结果如下：

```

Makefile (~/Work/RTL_Package n300 v3.7.0/n300 rls pkg/n300/vsim/run) - GVIM9
File Edit Tools Syntax Buffers Window Help
Makefile
98 endif
99
100 verilog:
101     ${WAV_TOOL} ${WAV_OPTIONS} +incdir+"${VSRC_DIR}/core"+"${VSRC_DIR}/soc"+"${VTB_DIR}" ${RTL_V_FILES}
102     ${TB_V_FILES} -logdir ${RUN_DIR}/verilog -nologo &
103
104 verilog_rtlonly:
105     ${WAV_TOOL} ${WAV_OPTIONS} +incdir+"${VSRC_DIR}/core"+"${VSRC_DIR}/soc"+"${VTB_DIR}" -f ${RUN_DIR}/
106     rtlonly_flist &
107
108 compile_rtlonly:
109     ${SIM_TOOL} ${SIM_OPTIONS} +incdir+"${VSRC_DIR}/core"+"${VSRC_DIR}/soc"+"${VTB_DIR}" -f ${RUN_DIR}/
110     rtlonly_flist
111
112 verilog_core:
113     ${WAV_TOOL} ${WAV_OPTIONS} +incdir+"${VSRC_DIR}/core"+"${VSRC_DIR}/soc"+"${VTB_DIR}" -f ${RUN_DIR}/
114     core_flist &
115
116 compile_core:
117     ${SIM_TOOL} ${SIM_OPTIONS} +incdir+"${VSRC_DIR}/core"+"${VSRC_DIR}/soc"+"${VTB_DIR}" -f ${RUN_DIR}/
118     core_flist
119
120 run: compile
121     # rm -rf ${TEST_RUNDIR}
122     mkdir -p ${TEST_RUNDIR}
123     cd ${TEST_RUNDIR}; ${SIM_EXEC} ${SIMV_FLAGS} +WFI_FORCE_IRQ=${WFI_FORCE_IRQ} +
124     FORCE_DELAY=${FORCE_DELAY} +FORCE_IRQ=${FORCE_IRQ} +FORCE_ECC=${FORCE_ECC} +
125     FORCE_RESP_ERR=${FORCE_RESP_ERR} +DUMPWAVE=${DUMPWAVE} +WAVEFSDB=${WAVEFSDB} +TESTCASE=${TESTCASE} +
126     SEED=${SEED} |& tee ${TESTNAME}.log; cd ${RUN_DIR};

```

这一次run case，不要删除对应目录

```
Verilog End of Elaborating.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
TESTCASE=
/home/angella/Work/RTL_Pack
age/n300_v3.7.0/n300_rls_pkg/n300/vsim/run/../../riscv-tests/isa/generated/helloworld
jtag_model use 4wire
ext debug case
enter dbg_test_init
dmi enable finish
riscv xlen: 32
enter dbg_test code
xpr:5 = 99999
freq:5 = 99999
$finish called from file "/home/angella/Work/RTL_Package/n300_v3.7.0/n300_rls_pkg/n300/vsim/run/../../install/tb/jt
ag_model/jtag_model.sv", line 555.
$finish at simulation time 1332800000
V C S S i m u l a t i o n R e p o r t
Time: 1332800000 ps
```

可以这次仿真就成功的执行了jtag module 这一个task，且task 这边的打印都在console 上：

```

debug_init.c
1
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <unistd.h>
5 #include <dlfcn.h>
6
7 // call debug init
8 void debug_init() {
9     if (!access("./debugger.so", F_OK)) {
10         printf("ext debug case\n");
11         void *handle = dlopen("./debugger.so", RTLD_LAZY);
12         void (*test_init)();
13         test_init = dlsym(handle, "dbg_test_init");
14         test_init();
15     }
16     else
17         printf("none ext debug case\n");
18 }
19
--
--
--
--
<g_model/debug_init.c CWD: /home/angella/Work/RTL_Package/n300_v3.7.0/n300_rls_pkg/n300/tb/jtag_vpi Line: 1
- /Work/RTL_Package/n300_v3.7.0/n300_rls_pkg/n300/tb/jtag_model/debug_init.c" 19L, 418C

dbg_test_code.cpp
9
10 // halt cpu
11 halt_cpu();
12
13 // write memory
14 error = write_memory(0x90000000, 0x5a5a, MEM_SIZE_16);
15 error = read_memory(0x90000000, &value, MEM_SIZE_16);
16
17 // access xpr
18 error = write_xpr(5, 0x99999);
19 error = read_xpr(5, &value);
20 printf("xpr:%d = %x\n", 5, value);
21
22 // access freg
23 error = write_freg(5, 0x99999);
24 error = read_freg(5, &value);
25 printf("freg:%d = %x\n", 5, value);
26
27 // cpu exit debug mode
28 resume_cpu();
29
30 return 0;
31 };

```

到这里就成功执行了，如果用户是准备把这个jtag model task用到自己的SoC环境里，也可以依照本文说的步骤，一步一步来实现。