lab4_fsic-fpga Group 4 Report

1. Refer to lab-fsim-sim, Integrate FIR into PRJ1 (axilite, axi-stream in/out)

As in the `lab-fsim-sim`, we first imported the reference FIR project into the `user_prj1` folder `(/fsic_fpga/vivado/vvd_srcs/caravel_soc/rtl/user/user_subsys/user_prj/user_prj1/rtl/)` and correctly connected the user project wrapper and the FIR module.

As the lab-fsim-sim, we first import the reference FIR project to the `user_prj1` folder `(/fsic_fpga/vivado/vvd_srcs/caravel_soc/rtl/user/user_subsys/user_prj/user_prj1/rtl/)`, and connect the user project wrapper and the FIR module correctly.

We also updated all file lists and includes in the TCL file necessary for simulation and synthesis. Update list:

```
/fsic_fpga/vivado/vvd_srcs/caravel_soc/rtl/user/user_subsys/user_prj/user_prj1/rtl/rtl.f
/fsic_fpga/vivado/vvd_caravel_fpga_fsic_sim.tcl
/fsic_fpga/vivado/vvd_caravel_fpga_fsic.tcl
```

2. Refer to hls_userdma, design a dma for FIR

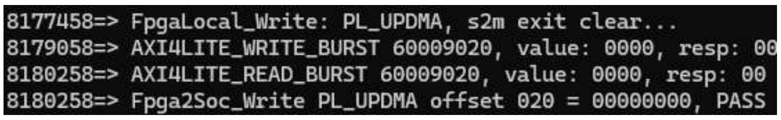
We referred to the user DMA template and, due to an infinite while loop issue, we set `BUF_LEN` =64. Then, we modified script.tcl to export the RTL of the new version of DMA.

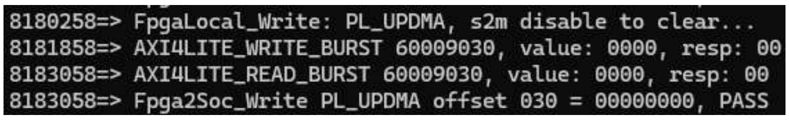
3. Vivado simulation

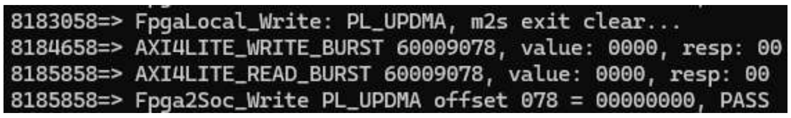
We disabled all tests in the original testbench `fsic.tb.v`, such as `Fpga2Soc_CfgRead()`, `Fpga2Soc_CfgWrite()`, `FpgaLocal_CfgRead()`, `SocLocal_MbWrite()`, `FpgaLocal_MbWrite()`, `SocLa2DmaPath(), and SocUp2DmaPath()`. We then designed suitable tasks `SocUp2DmaPath_FIR()` and `CheckuserDMADone_FIR()` for testing our DMA and FIR IP, based on the tasks `SocUp2DmaPath()` and `CheckuserDMADone()` in original testbench.

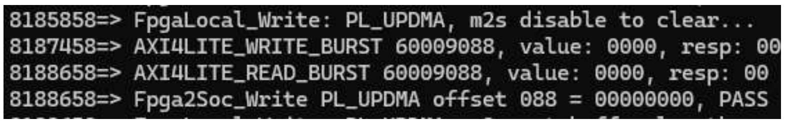
`SocUp2DmaPath_FIR()` performed the following tasks:

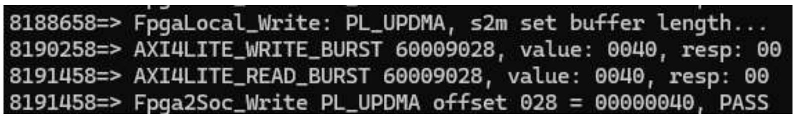
Target Behavior	Simulation Result
DMA load input	

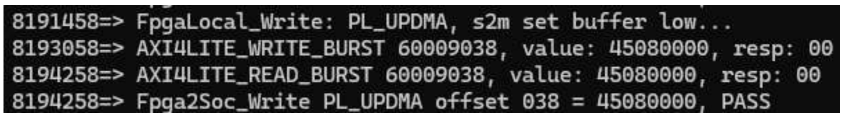
s2m exit clear	 `8177458=> FpgaLocal_Write: PL_UPDMA, s2m exit clear...`
	`8179058=> AXI4LITE_WRITE_BURST 60009020, value: 0000, resp: 00`
	`8180258=> AXI4LITE_READ_BURST 60009020, value: 0000, resp: 00`
	`8180258=> Fpga2Soc_Write PL_UPDMA offset 020 = 00000000, PASS`

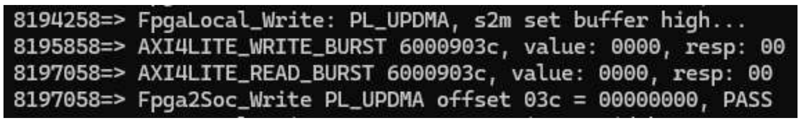
s2m disable to clear	 `8180258=> FpgaLocal_Write: PL_UPDMA, s2m disable to clear...`
	`8181858=> AXI4LITE_WRITE_BURST 60009030, value: 0000, resp: 00`
	`8183058=> AXI4LITE_READ_BURST 60009030, value: 0000, resp: 00`
	`8183058=> Fpga2Soc_Write PL_UPDMA offset 030 = 00000000, PASS`

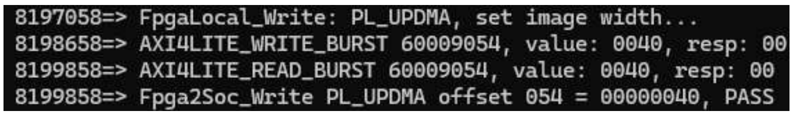
m2s exit clear	 `8183058=> FpgaLocal_Write: PL_UPDMA, m2s exit clear...`
	`8184658=> AXI4LITE_WRITE_BURST 60009078, value: 0000, resp: 00`
	`8185858=> AXI4LITE_READ_BURST 60009078, value: 0000, resp: 00`
	`8185858=> Fpga2Soc_Write PL_UPDMA offset 078 = 00000000, PASS`

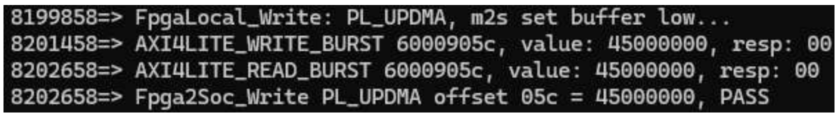
m2s disable to clear	 `8185858=> FpgaLocal_Write: PL_UPDMA, m2s disable to clear...`
	`8187458=> AXI4LITE_WRITE_BURST 60009088, value: 0000, resp: 00`
	`8188658=> AXI4LITE_READ_BURST 60009088, value: 0000, resp: 00`
	`8188658=> Fpga2Soc_Write PL_UPDMA offset 088 = 00000000, PASS`

s2m set buffer length	 `8188658=> FpgaLocal_Write: PL_UPDMA, s2m set buffer length...`
	`8190258=> AXI4LITE_WRITE_BURST 60009028, value: 0040, resp: 00`
	`8191458=> AXI4LITE_READ_BURST 60009028, value: 0040, resp: 00`
	`8191458=> Fpga2Soc_Write PL_UPDMA offset 028 = 00000040, PASS`

s2m set buffer low	 `8191458=> FpgaLocal_Write: PL_UPDMA, s2m set buffer low...`
	`8193058=> AXI4LITE_WRITE_BURST 60009038, value: 45080000, resp: 00`
	`8194258=> AXI4LITE_READ_BURST 60009038, value: 45080000, resp: 00`
	`8194258=> Fpga2Soc_Write PL_UPDMA offset 038 = 45080000, PASS`

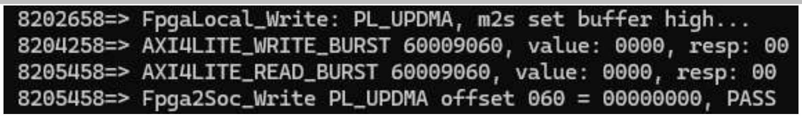
s2m set buffer high	 `8194258=> FpgaLocal_Write: PL_UPDMA, s2m set buffer high...`
	`8195858=> AXI4LITE_WRITE_BURST 6000903c, value: 0000, resp: 00`
	`8197058=> AXI4LITE_READ_BURST 6000903c, value: 0000, resp: 00`
	`8197058=> Fpga2Soc_Write PL_UPDMA offset 03c = 00000000, PASS`

set image width	 `8197058=> FpgaLocal_Write: PL_UPDMA, set image width...`
	`8198658=> AXI4LITE_WRITE_BURST 60009054, value: 0040, resp: 00`
	`8199858=> AXI4LITE_READ_BURST 60009054, value: 0040, resp: 00`
	`8199858=> Fpga2Soc_Write PL_UPDMA offset 054 = 00000040, PASS`

m2s set buffer low	 `8199858=> FpgaLocal_Write: PL_UPDMA, m2s set buffer low...`
	`8201458=> AXI4LITE_WRITE_BURST 6000905c, value: 45000000, resp: 00`
	`8202658=> AXI4LITE_READ_BURST 6000905c, value: 45000000, resp: 00`
	`8202658=> Fpga2Soc_Write PL_UPDMA offset 05c = 45000000, PASS`

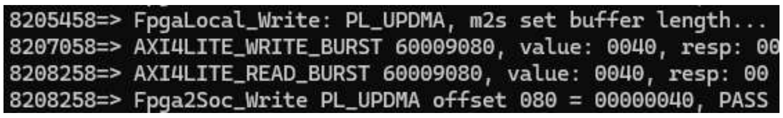
Target Behavior | **Simulation Result**

m2s set buffer high



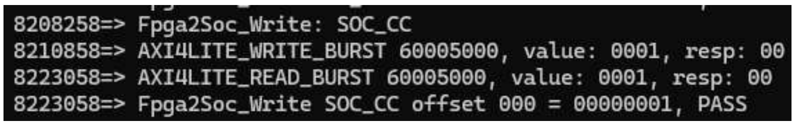
```
8202658=> FpgaLocal_Write: PL_UPDMA, m2s set buffer high...
8204258=> AXI4LITE_WRITE_BURST 60009060, value: 0000, resp: 00
8205458=> AXI4LITE_READ_BURST 60009060, value: 0000, resp: 00
8205458=> Fpga2Soc_Write PL_UPDMA offset 060 = 00000000, PASS
```

m2s set buffer length



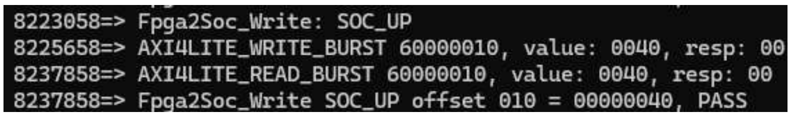
```
8205458=> FpgaLocal_Write: PL_UPDMA, m2s set buffer length...
8207058=> AXI4LITE_WRITE_BURST 60009080, value: 0040, resp: 00
8208258=> AXI4LITE_READ_BURST 60009080, value: 0040, resp: 00
8208258=> Fpga2Soc_Write PL_UPDMA offset 080 = 00000040, PASS
```

select FIR IP user project



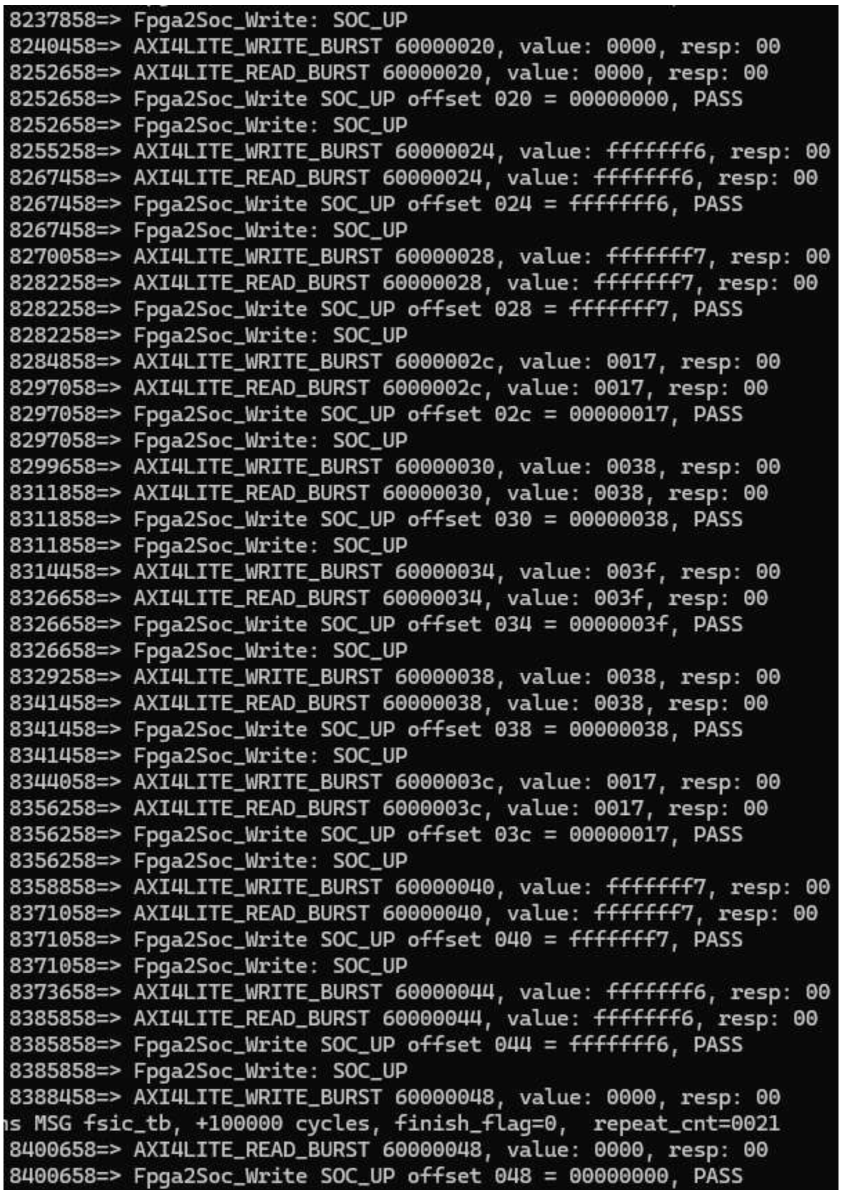
```
8208258=> Fpga2Soc_Write: SOC_CC
8210858=> AXI4LITE_WRITE_BURST 60005000, value: 0001, resp: 00
8223058=> AXI4LITE_READ_BURST 60005000, value: 0001, resp: 00
8223058=> Fpga2Soc_Write SOC_CC offset 000 = 00000001, PASS
```

configure data length in FIR IP



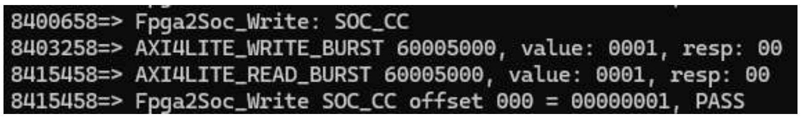
```
8223058=> Fpga2Soc_Write: SOC_UP
8225658=> AXI4LITE_WRITE_BURST 60000010, value: 0040, resp: 00
8237858=> AXI4LITE_READ_BURST 60000010, value: 0040, resp: 00
8237858=> Fpga2Soc_Write SOC_UP offset 010 = 00000040, PASS
```

configure taps in FIR IP



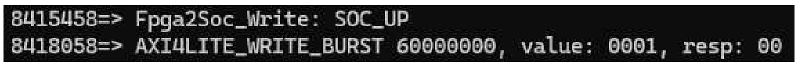
```
8237858=> Fpga2Soc_Write: SOC_UP
8240458=> AXI4LITE_WRITE_BURST 60000020, value: 0000, resp: 00
8252658=> AXI4LITE_READ_BURST 60000020, value: 0000, resp: 00
8252658=> Fpga2Soc_Write SOC_UP offset 020 = 00000000, PASS
8252658=> Fpga2Soc_Write: SOC_UP
8255258=> AXI4LITE_WRITE_BURST 60000024, value: fffffff6, resp: 00
8267458=> AXI4LITE_READ_BURST 60000024, value: fffffff6, resp: 00
8267458=> Fpga2Soc_Write SOC_UP offset 024 = fffffff6, PASS
8267458=> Fpga2Soc_Write: SOC_UP
8270058=> AXI4LITE_WRITE_BURST 60000028, value: fffffff7, resp: 00
8282258=> AXI4LITE_READ_BURST 60000028, value: fffffff7, resp: 00
8282258=> Fpga2Soc_Write SOC_UP offset 028 = fffffff7, PASS
8282258=> Fpga2Soc_Write: SOC_UP
8284858=> AXI4LITE_WRITE_BURST 6000002c, value: 0017, resp: 00
8297058=> AXI4LITE_READ_BURST 6000002c, value: 0017, resp: 00
8297058=> Fpga2Soc_Write SOC_UP offset 02c = 00000017, PASS
8297058=> Fpga2Soc_Write: SOC_UP
8299658=> AXI4LITE_WRITE_BURST 60000030, value: 0038, resp: 00
8311858=> AXI4LITE_READ_BURST 60000030, value: 0038, resp: 00
8311858=> Fpga2Soc_Write SOC_UP offset 030 = 00000038, PASS
8311858=> Fpga2Soc_Write: SOC_UP
8314458=> AXI4LITE_WRITE_BURST 60000034, value: 003f, resp: 00
8326658=> AXI4LITE_READ_BURST 60000034, value: 003f, resp: 00
8326658=> Fpga2Soc_Write SOC_UP offset 034 = 0000003f, PASS
8326658=> Fpga2Soc_Write: SOC_UP
8329258=> AXI4LITE_WRITE_BURST 60000038, value: 0038, resp: 00
8341458=> AXI4LITE_READ_BURST 60000038, value: 0038, resp: 00
8341458=> Fpga2Soc_Write SOC_UP offset 038 = 00000038, PASS
8341458=> Fpga2Soc_Write: SOC_UP
8344058=> AXI4LITE_WRITE_BURST 6000003c, value: 0017, resp: 00
8356258=> AXI4LITE_READ_BURST 6000003c, value: 0017, resp: 00
8356258=> Fpga2Soc_Write SOC_UP offset 03c = 00000017, PASS
8356258=> Fpga2Soc_Write: SOC_UP
8358858=> AXI4LITE_WRITE_BURST 60000040, value: fffffff7, resp: 00
8371058=> AXI4LITE_READ_BURST 60000040, value: fffffff7, resp: 00
8371058=> Fpga2Soc_Write SOC_UP offset 040 = fffffff7, PASS
8371058=> Fpga2Soc_Write: SOC_UP
8373658=> AXI4LITE_WRITE_BURST 60000044, value: fffffff6, resp: 00
8385858=> AXI4LITE_READ_BURST 60000044, value: fffffff6, resp: 00
8385858=> Fpga2Soc_Write SOC_UP offset 044 = fffffff6, PASS
8385858=> Fpga2Soc_Write: SOC_UP
8388458=> AXI4LITE_WRITE_BURST 60000048, value: 0000, resp: 00
s MSG fsic_tb, +100000 cycles, finish_flag=0,  repeat_cnt=0021
8400658=> AXI4LITE_READ_BURST 60000048, value: 0000, resp: 00
8400658=> Fpga2Soc_Write SOC_UP offset 048 = 00000000, PASS
```

recheck user project sel



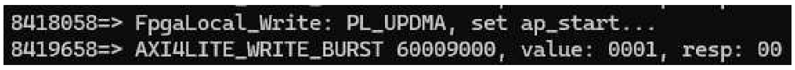
```
8400658=> Fpga2Soc_Write: SOC_CC
8403258=> AXI4LITE_WRITE_BURST 60005000, value: 0001, resp: 00
8415458=> AXI4LITE_READ_BURST 60005000, value: 0001, resp: 00
8415458=> Fpga2Soc_Write SOC_CC offset 000 = 00000001, PASS
```

configure ap_start in FIR IP



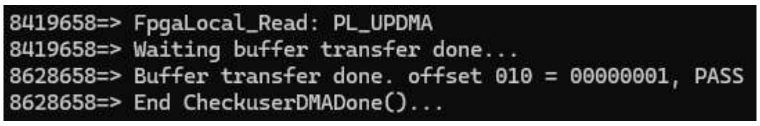
```
8415458=> Fpga2Soc_Write: SOC_UP
8418058=> AXI4LITE_WRITE_BURST 60000000, value: 0001, resp: 00
```

set ap_start



```
8418058=> FpgaLocal_Write: PL_UPDMA, set ap_start...
8419658=> AXI4LITE_WRITE_BURST 60009000, value: 0001, resp: 00
```

`CheckuserDMADone_FIR()` performed the following tasks:

Target Behavior	Simulation Result
polling ap_done	

```
8419658=> FpgaLocal_Read: PL_UPDMA
8419658=> Waiting buffer transfer done...
8628658=> Buffer transfer done. offset 010 = 00000001, PASS
8628658=> End CheckuserDMADone()...
```

4. Build FPGA

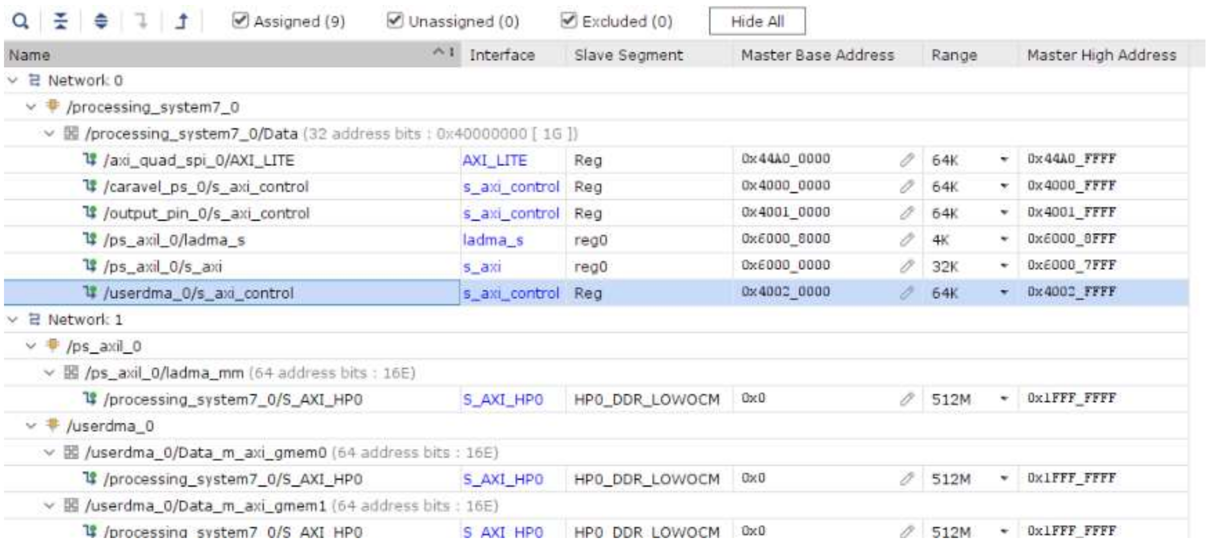
Since `hls_userdma` was re-exported, we updated `vvd_caravel_fpga_fsic.tcl` to reconnect `hls_userdma` and `ps_axil`. We then ran the TCL script to generate `caravel_fpga.bit` and `caravel_fpga.hwh` for later PYNQ validation.

5. Firmware code

We used the original firmware code.

6. Jupyter-notebook Python code

We referred to the original Jupyter Python code and skipped the LADMA verification part. However, we used the LADMA verification part as a reference to write our own UPDMA verification part, which follows the same flow as in the Vivado testbench.



Name	Interface	Slave Segment	Master Base Address	Range	Master High Address
Network 0					
/processing_system7_0					
/processing_system7_0/Data (32 address bits : 0x40000000 [1G])					
/axi_quad_spi_0/AXI_LITE	AXI_LITE	Reg	0x44A0_0000	64K	0x44A0_FFFF
/caravel_ps_0/s_axi_control	s_axi_control	Reg	0x4000_0000	64K	0x4000_FFFF
/output_pin_0/s_axi_control	s_axi_control	Reg	0x4001_0000	64K	0x4001_FFFF
/ps_axil_0/ladma_s	ladma_s	reg0	0x6000_8000	4K	0x6000_8FFF
/ps_axil_0/s_axi	s_axi	reg0	0x6000_0000	32K	0x6000_7FFF
/userdma_0/s_axi_control	s_axi_control	Reg	0x4002_0000	64K	0x4002_FFFF
Network 1					
/ps_axil_0					
/ps_axil_0/ladma_mm (64 address bits : 16E)					
/processing_system7_0/S_AXI_HP0	S_AXI_HP0	HP0_DDR_LOWOCM	0x0	512M	0x1FFF_FFFF
/userdma_0					
/userdma_0/Data_m_axi_gmem0 (64 address bits : 16E)					
/processing_system7_0/S_AXI_HP0	S_AXI_HP0	HP0_DDR_LOWOCM	0x0	512M	0x1FFF_FFFF
/userdma_0/Data_m_axi_gmem1 (64 address bits : 16E)					
/processing_system7_0/S_AXI_HP0	S_AXI_HP0	HP0_DDR_LOWOCM	0x0	512M	0x1FFF_FFFF

Using the memory map generated by Vivado, we set MMIO for UPDMA in Python: `mmio_UPDMA = MMIO(0x40020000, 0x00010000)`. We rewrote the testbench in Python to access the configuration registers with `mmio.read()`, `mmio.write()`, `mmio_UPDMA.read()`, and `mmio_UPDMA.write()`.

Issue and Solution

There were 600 test vectors for the FIR in the original file, so we initially set `BUF_LEN` =600 in `hls_userdma` to test the FIR with the whole set of vectors. However, during the simulation stage, hls_userdma reported a deadlock.

We passed the testbench by setting `BUF_LEN` to 64.