复杂分布式电子学系统的调试诊断及慢控制接口的探索与实践

Design of Debug and Slow Control Interface for Complex and Distributed Electronic System

薛涛、文敬君、姜林、杨昊彦、韦亮军、潘秋彤、梁博

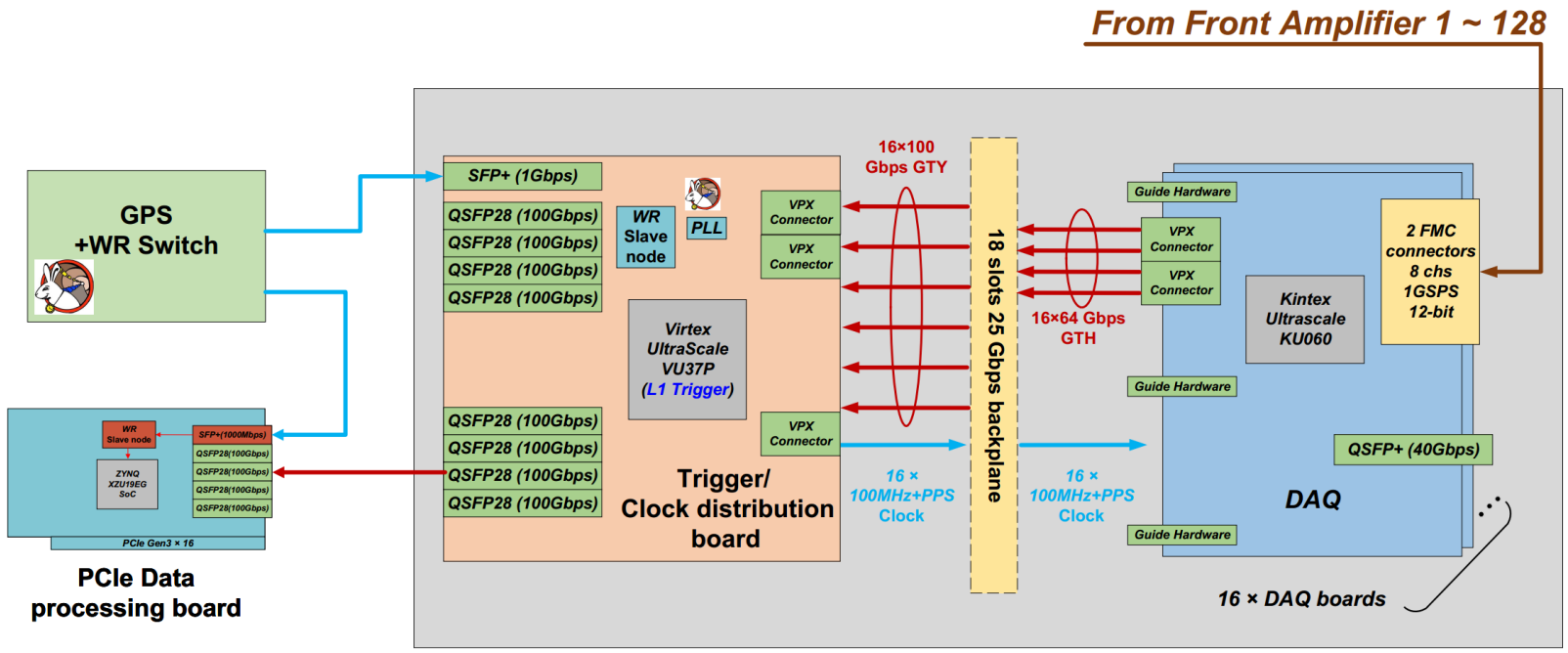
李荐民、刘以农

2024 MicroTCA/ATCA for Large Scientific Facility Control International Workshop

2024.09.20

目录

- Motivation，复杂、大型分布式电子学系统面临的困难
- 从调试接口说起
- 电子学板卡的板载调试接口的实现
- 电子学系统远程调试接口的实现
- 电子学板卡中多个FPGA的互联和慢控制接口的实现
- 实践、总结和展望

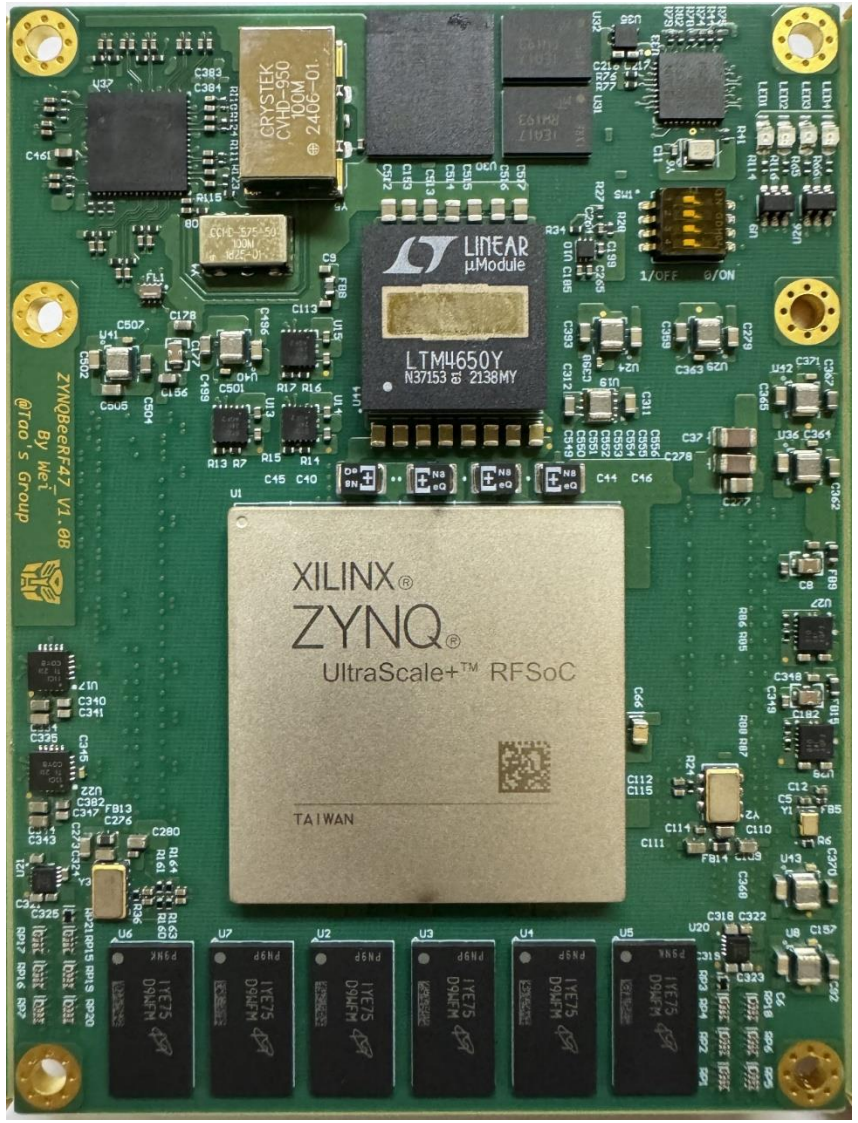


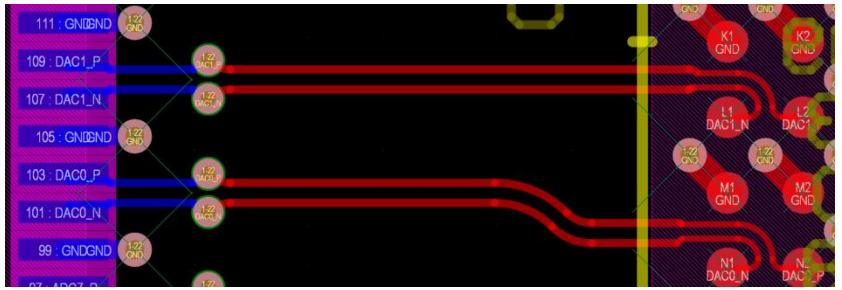
Motivation，复杂、大型分布式电子学系统面临的困难

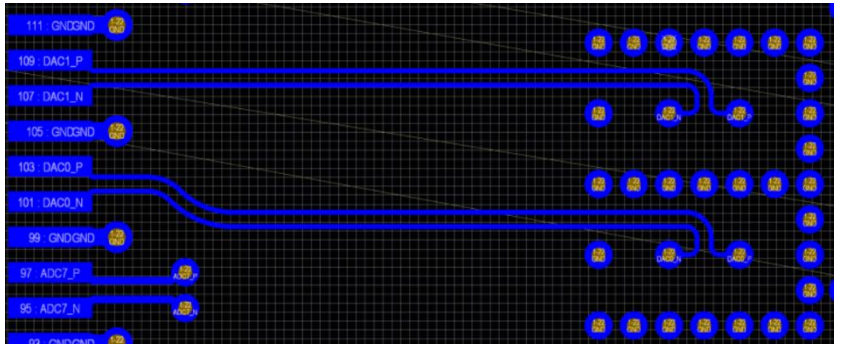
- 面向前沿物理实验的大规模电子学系统的挑战
  - 大规模、复杂、分布式电子学系统调试的痛点和需求
    - 信号完整性与电源完整性（硬件）
    - 复杂的FPGA逻辑实现与在线、在系统调试方法（固件）
    - 面向物理的复杂触发算法的FPGA实现（固件）
    - 100Gbps规模高带宽读出和数据传输（硬件与固件）
    - 100A量级的核心电源轨电流（硬件）
    - FPGA与CPU、GPU的协调工作（硬件、固件与软件）
  - 从单块板卡，到几百、几千、上万块板子组成系统的调试方法
    - 从单板到复杂系统，从简单逻辑到多板块协调调试
    - 从单一逻辑调试到与Embedded Linux的协同调试
    - 大型电子学系统的慢控制需求（高压、偏置、校准、在线测试等）
    - 多系统正常运行时的诊断

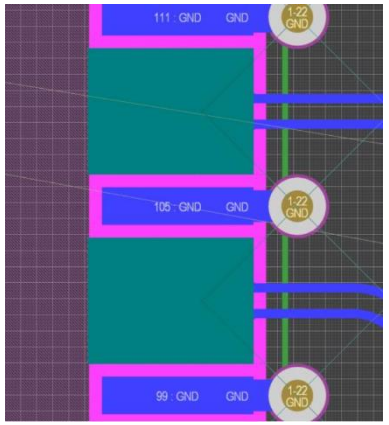
Motivation，复杂、大型分布式电子学系统面临的困难

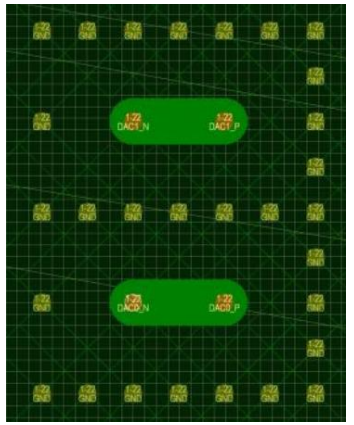
- 信号完整性（Zynq Ultrascale+ RFSoC举例）

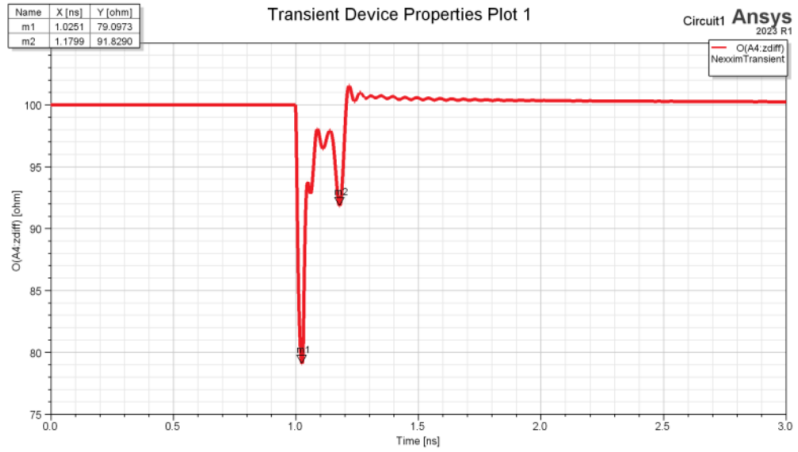






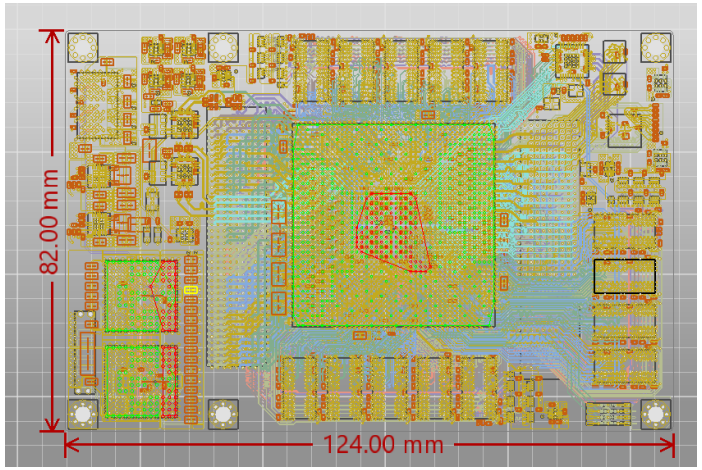


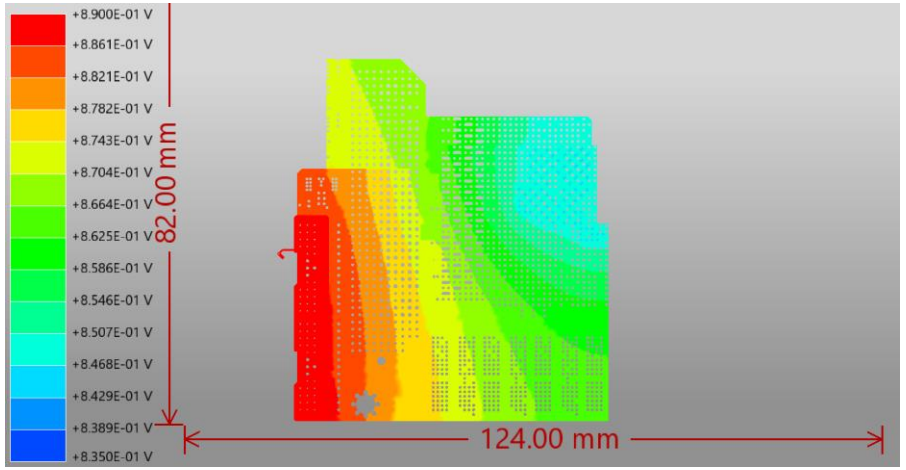


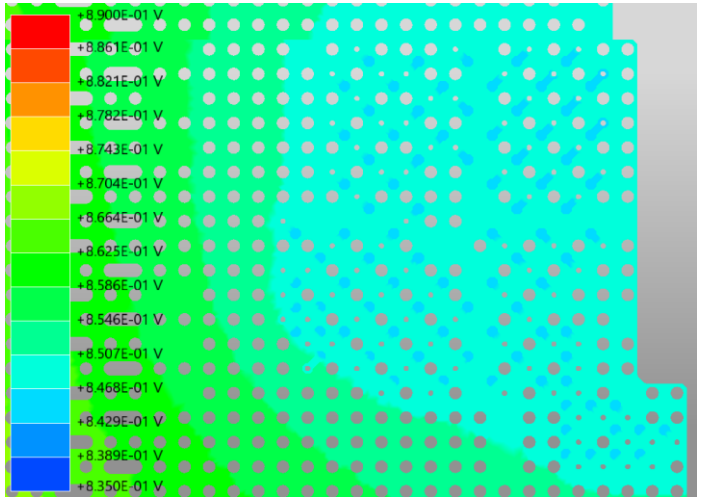


Motivation，复杂、大型分布式电子学系统面临的困难

- 电源完整性（Zynq Ultrascale+ ZU19举例）







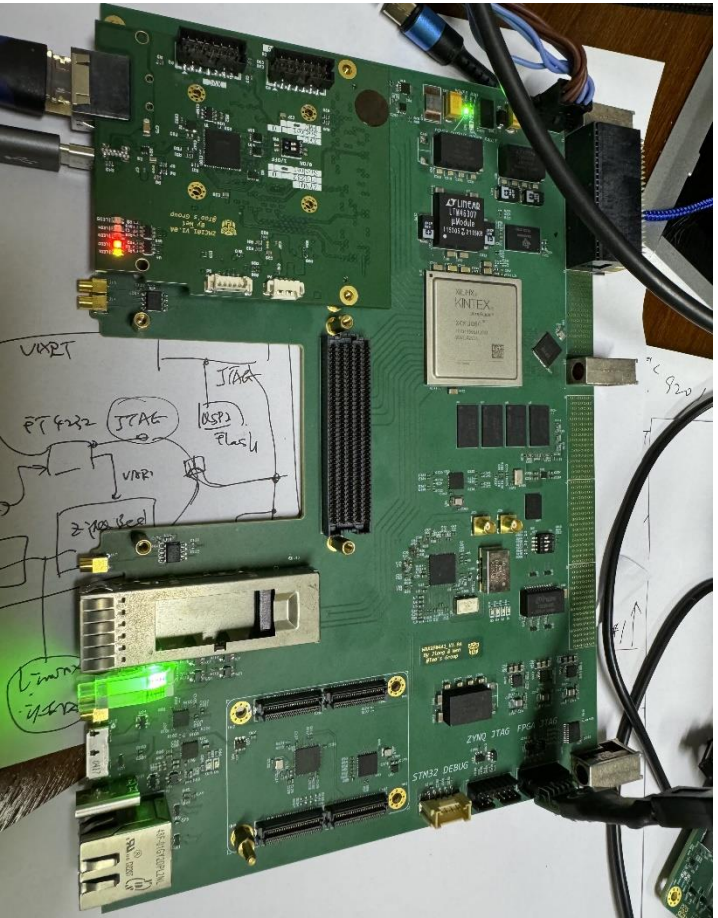
- 大电流平面DC-IR Drop仿真
  - 0.89V/100A电流下的直流压降，ZYNQ Ultrascale+ 端~0.843V
  - 电源平面均为1oz L10 28um,L11 30um,L13 28um

从调试接口说起

- FPGA调试接口的历史、现状和思路
  - 最常用的USB-JTAG紧凑型调试器及其局限性

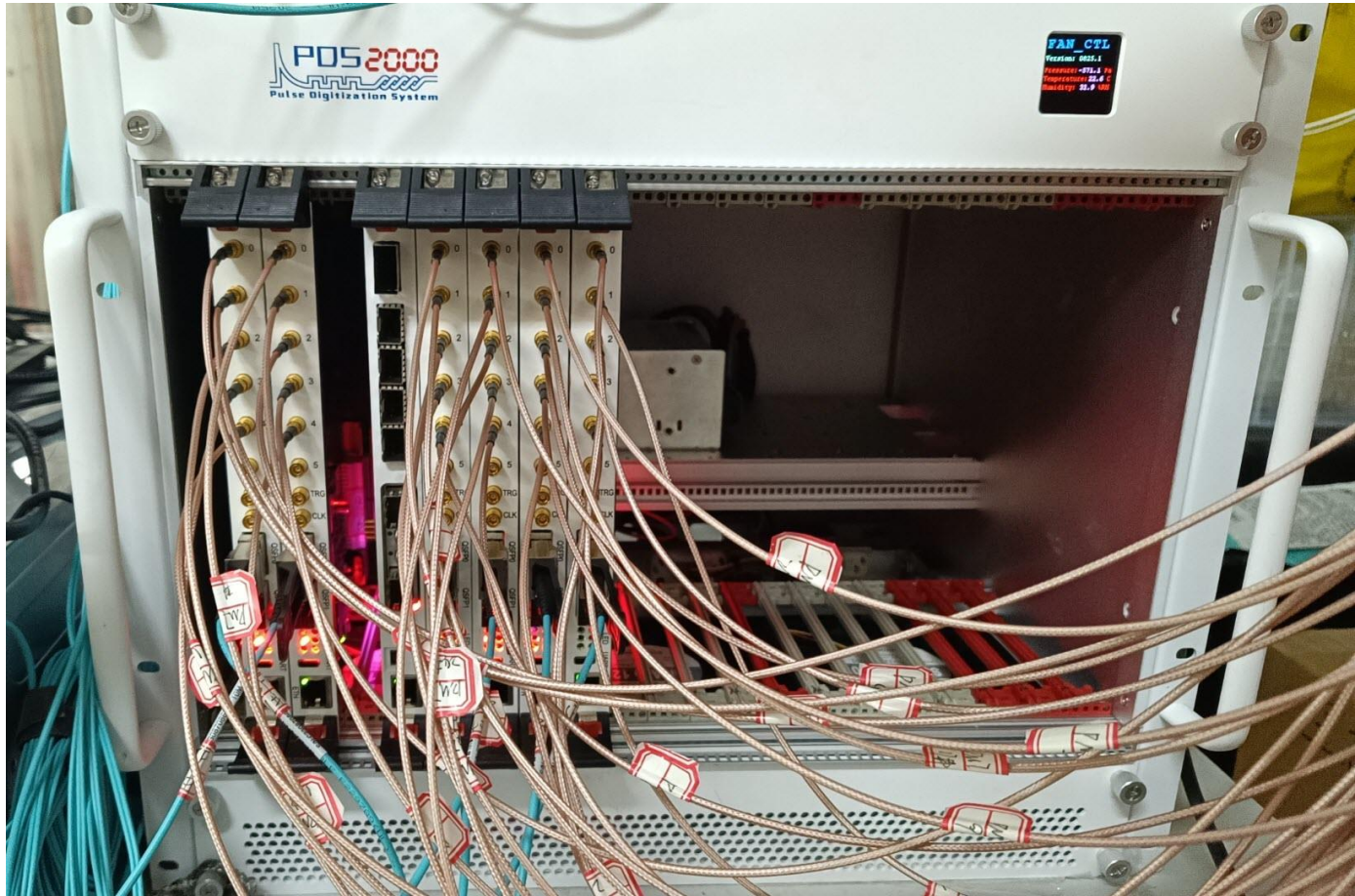
Xilinx Parallel Cable IV

Xilinx Platform Cable USB II



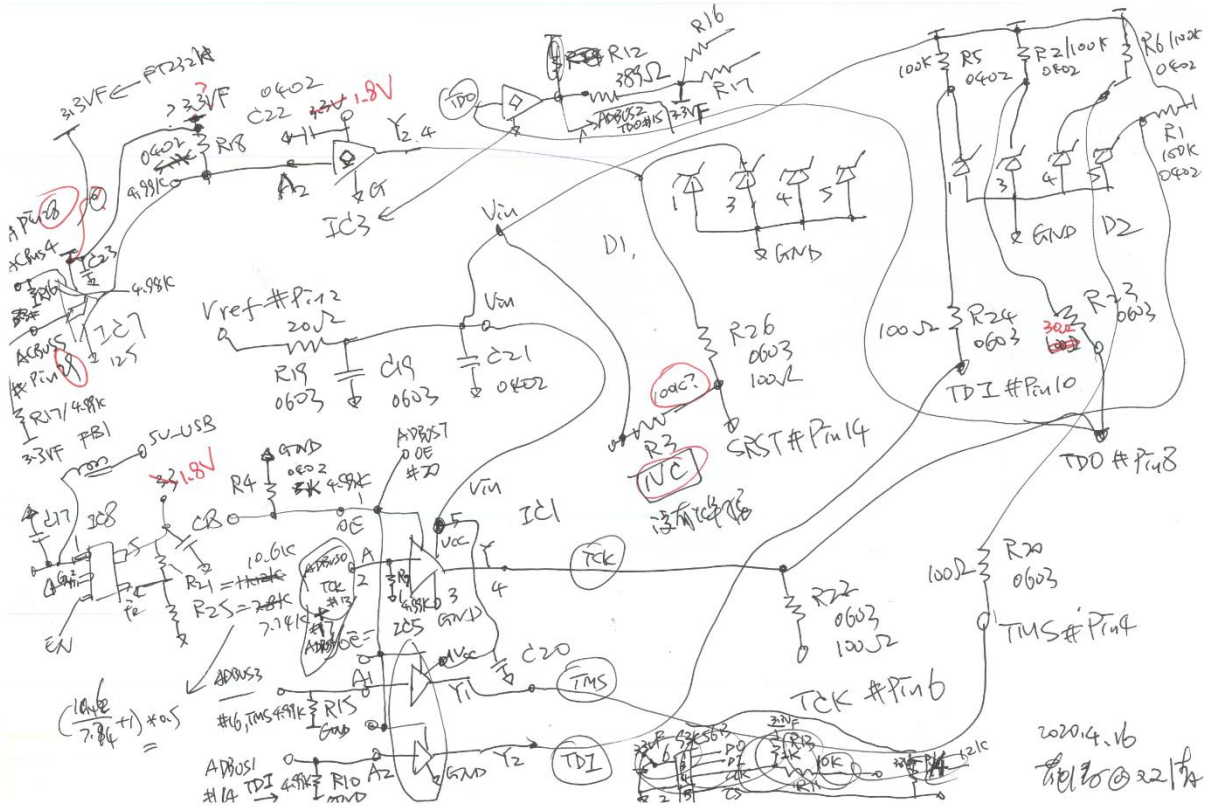
Digilent JTAG-SMT2

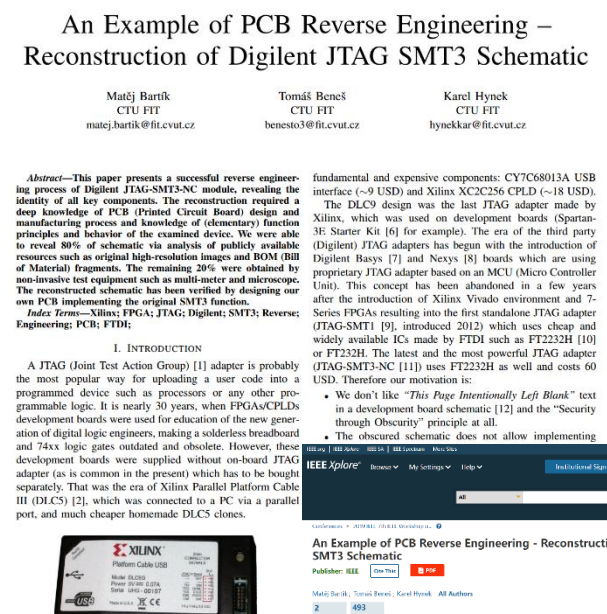
Digilent JTAG-HS3



电子学板卡的板载调试接口的实现

- USB-JTAG/UART调试接口的实现思路
  - 使用FTDI公司的FT2232HQ和FT4232HQ
    - FT2232HQ包含2个Channel，可以实现JTAG+UART
    - FT4232HQ包含4个Channel，可以实现JTAG+3×UART
    - 推荐使用TMUX1574做为JTAG二选一MUX
    - 推荐使用SN74AXC4T774做为JTAG电平转换
    - https://ieeexplore.ieee.org/abstract/document/8977100





An Example of PCB Reverse Engineering – Reconstruction of Digilent JTAG SMT3 Schematic

Matěj Bartík
CTU FIT
matej.bartik@fit.cvut.cz

Tomáš Beneš
CTU FIT
benesto3@fit.cvut.cz

Karel Hynek
CTU FIT
hynekkar@fit.cvut.cz

***Abstract*—This paper presents a successful reverse engineering process of Digilent JTAG-SMT3-NC module, revealing the identity of all key components. The reconstruction required a deep knowledge of PCB (Printed Circuit Board) design and manufacturing process and knowledge of (elementary) function principles and behavior of the examined device. We were able to reveal 80% of schematic via analysis of publicly available resources such as original high-resolution images and BOM (Bill of Material) fragments. The remaining 20% were obtained by non-invasive test equipment such as multi-meter and microscope. The reconstructed schematic has been verified by designing our own PCB implementing the original SMT3 function.**

***Index Terms*—Xilinx; FPGA; JTAG; Digilent; SMT3; Reverse; Engineering; PCB; FTDI;**

I. Introduction

A JTAG (Joint Test Action Group) [1] adapter is probably the most popular way for uploading a user code into a programmed device such as processors or any other programmable logic. It is nearly 30 years, when FPGAs/CPLDs development boards were used for education of the new generation of digital logic engineers, making a solderless breadboard and 74xx logic gates outdated and obsolete. However, these development boards were supplied without on-board JTAG adapter (as is common in the present) which has to be bought separately. That was the era of Xilinx Parallel Platform Cable III (DLC5) [2], which was connected to a PC via a parallel port, and much cheaper homemade DLC5 clones.

Fig. 1. Xilinx Platform Cable USB (DLC9)

The scenario became the same for DLC5 successors such as Xilinx Platform Cable USB I [3] (see Fig. 1) and II [4] (DLC9 and DLC10 respectively). The schematic and other resources of the DLC9 leaked nearly ten years ago [5] which allows to produce significantly cheaper clones (approximately 30 USD) compared to genuine DLC10 (225 USD). The price of a DLC9 clone [5] cannot be lowered further regarding two fundamental and expensive components: CY7C68013A USB interface (~9 USD) and Xilinx XC2C256 CPLD (~18 USD).

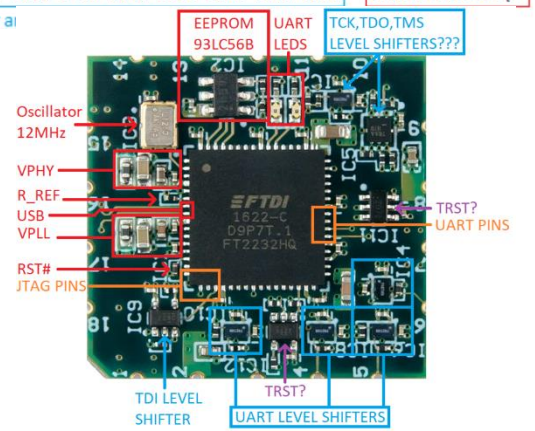
The DLC9 design was the last JTAG adapter made by Xilinx, which was used on development boards (Spartan-3E Starter Kit [6] for example). The era of the third party (Digilent) JTAG adapters has begun with the introduction of Digilent Basys [7] and Nexys [8] boards which are using proprietary JTAG adapter based on an MCU (Micro Controller Unit). This concept has been abandoned in a few years after the introduction of Xilinx Vivado environment and 7-Series FPGAs resulting into the first standalone JTAG adapter (JTAG-SMT1 [9], introduced 2012) which uses cheap and widely available ICs made by FTDI such as FT2232H [10] or FT232H. The latest and the most powerful JTAG adapter (JTAG-SMT3-NC [11]) uses FT2232H as well and costs 60 USD. Therefore our motivation is:

- We don't like *"This Page Intentionally Left Blank"* text in a development board schematic [12] and the "Security through Obscurity" principle at all.
- The obscured schematic does not allow implementing

978-1-7281-5709-2/19/$31.00 ©2019 IEEE

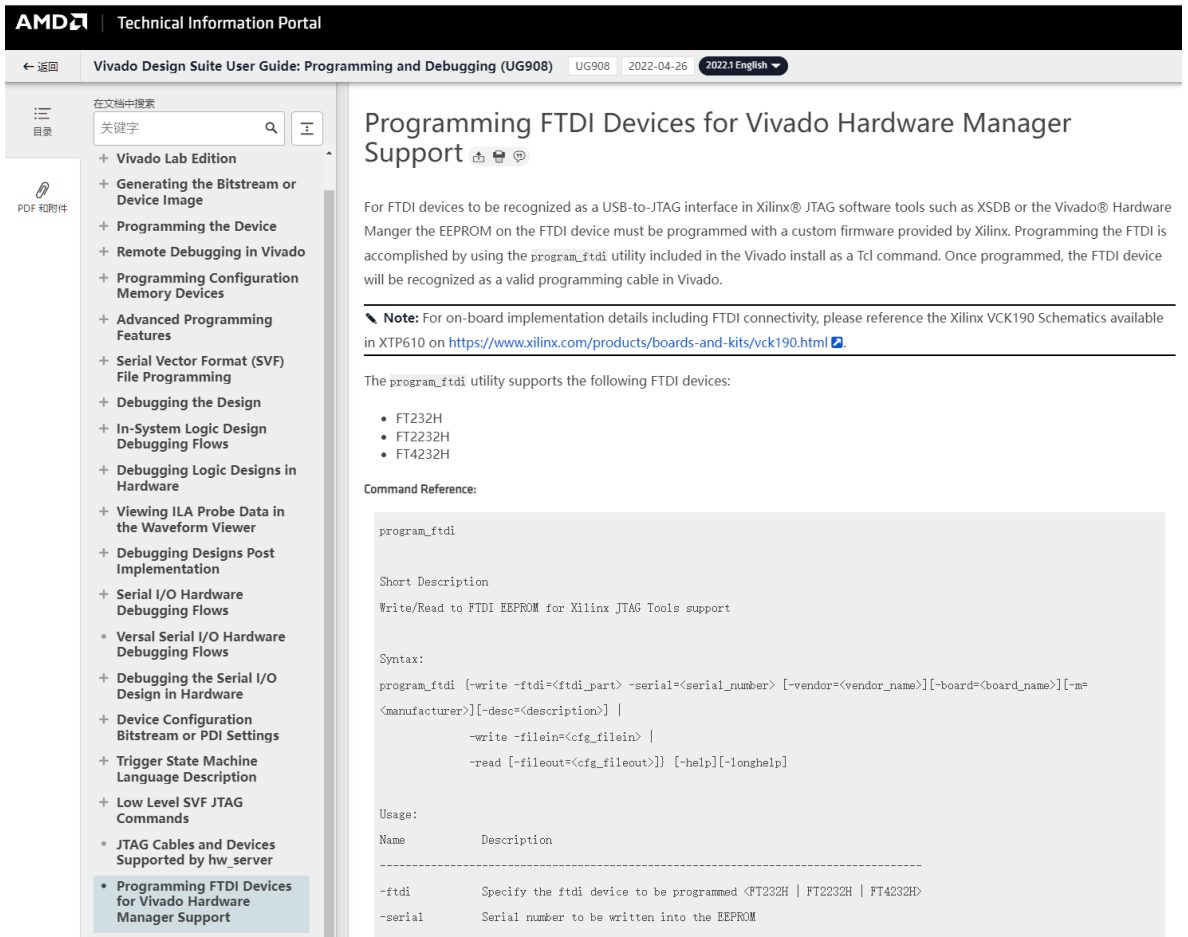
Device Output

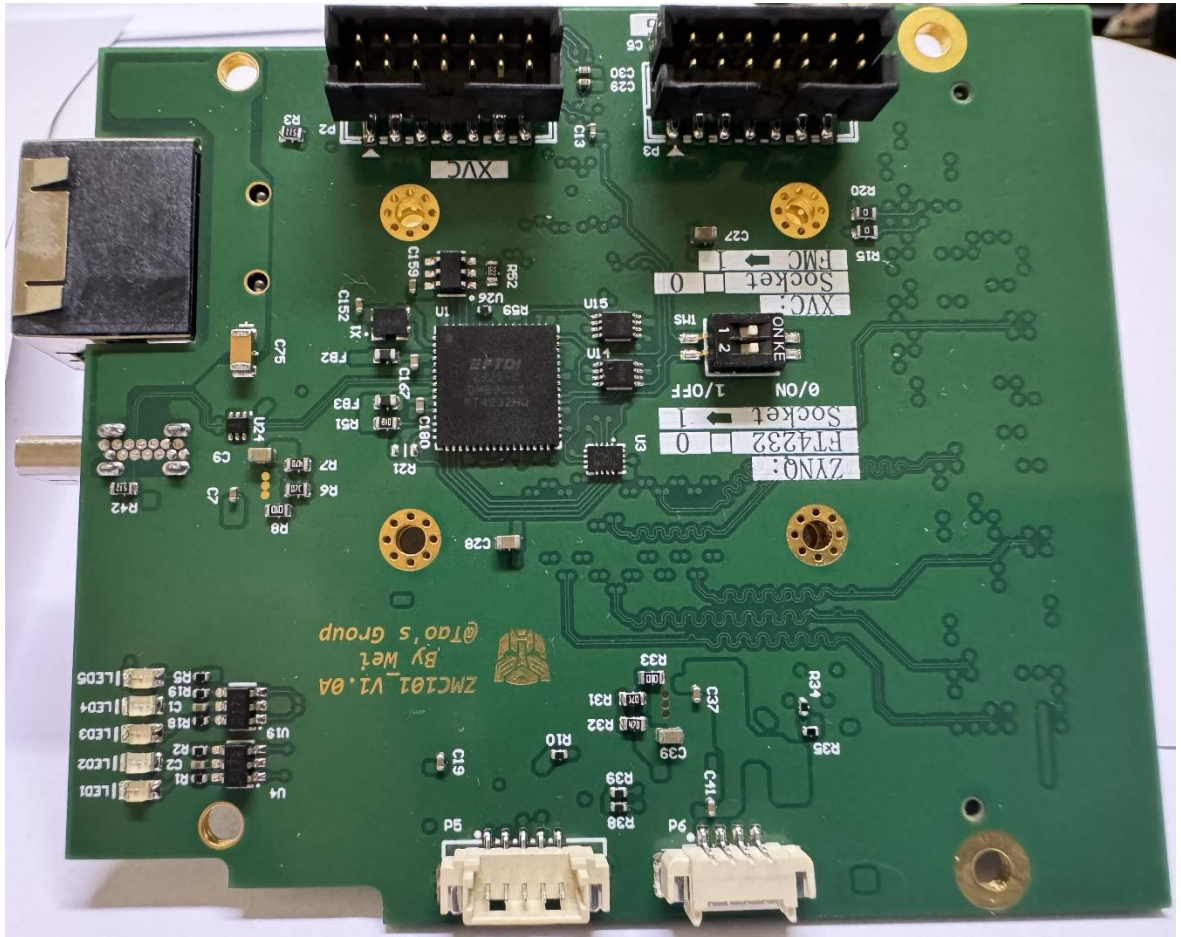
```
Word  MSB
0000: 8801 0403 6010 0700 00C0 0008 0000 129A  ....`...........
0008: 34AC 1AE0 0000 0000 0056 0001 92C7 356A  4........V.....5j
0010: 0157 3120 744A 6761 6D53 3374 0000 0000  .W1 tJgamS3t....
0018: 0000 0000 4400 6769 6C69 6E65 2074 544A  ....D.giline tTJ
0020: 4741 532D 544D 0033 0000 0000 0000 0000  GAS-TM.3........
0028: 0000 0001 0000 0000 0000 0000 0000 0000  ................
0030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0038: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0040: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0048: 0000 0000 0000 0000 0000 0312 0044 0069  .............D.i
0050: 0067 0069 006C 0065 006E 0074 0334 0044  .g.i.l.e.n.t.4.D
0058: 0069 0067 0069 006C 0065 006E 0074 0020  .i.g.i.l.e.n.t.
0060: 0041 0064 0065 0070 0074 0020 0055 0053  .A.d.e.p.t. .U.S
0068: 0042 0020 0044 0065 0076 0069 0063 0065  .B. .D.e.v.i.c.e
0070: 031A 0032 0031 0030 0033 0035 0037 0041  ...2.1.0.3.5.7.A
0078: 0037 0044 0030 0030 0045 0302 0000 7B82  .7.D.0.0.E....{.
```



电子学板卡的板载调试接口的实现

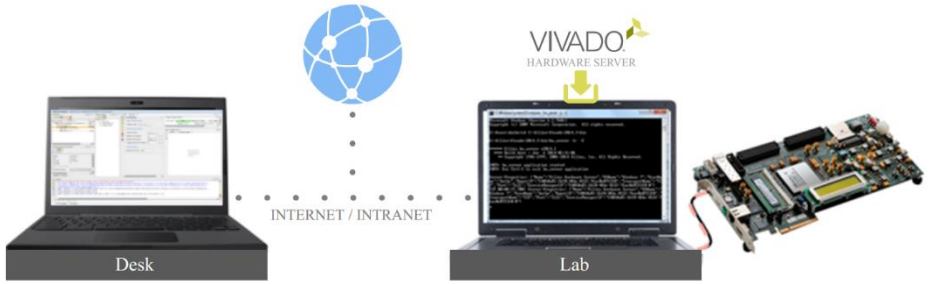
- Vivado的支持：将FT232HQ/FT2232HQ/FT4232HQ配置为USB-JTAG/UART调试器
  - Vivado从2022.1版本开始支持 program_ftdi 将FTDI公司的FT2232HQ和FT4232HQ变成USB-JTAG调试器
    - FT2232HQ包含2个Channel，可以实现JTAG+UART。Vivado将其 Channel A 编程为JTAG接口，Channel B 默认为UART接口。
    - FT4232HQ包含4个Channel，可以实现JTAG+3×UART。Vivado将其 Channel A 编程为JTAG接口，Channel B/C/D 默认为UART接口。可以修订EEPROM将 Channel B 配置为SWD/JTAG接口用于板载底层控制器/STM32的编程接口。



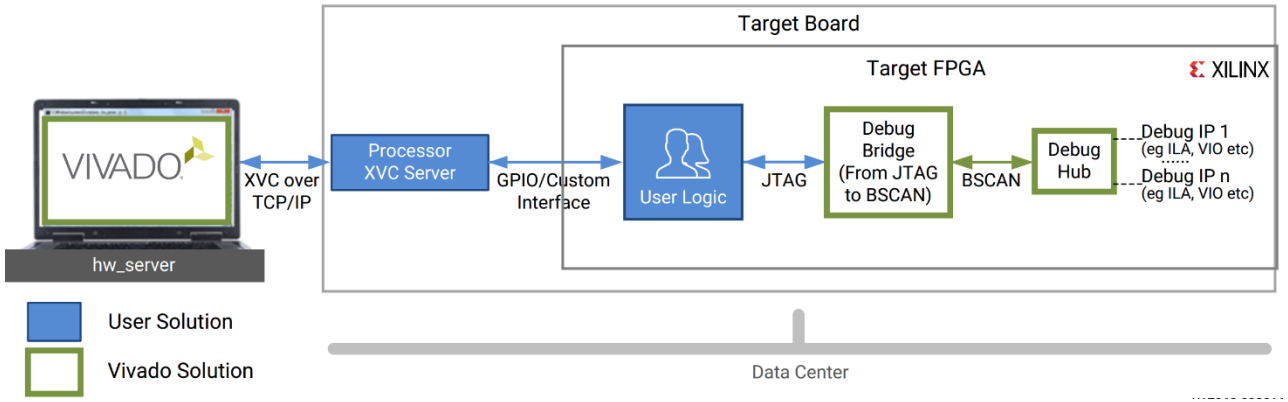


电子学系统远程调试接口的实现

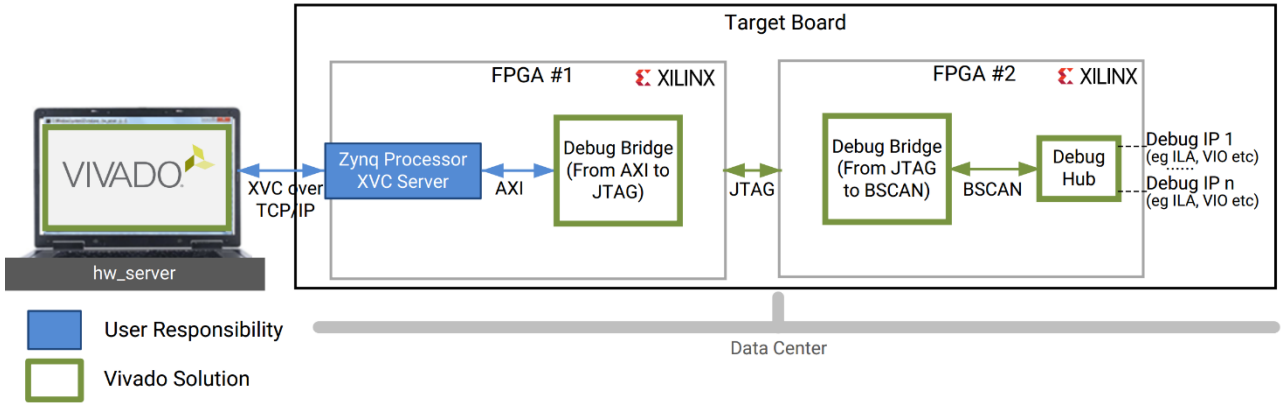
- XVC（Xilinx Virtual Cable）通过以太网进行远程调试的JTAG接口
  - 使用Debug Bridge IP核及运行与Embedded Linux上的 xvcServer
  - https://github.com/Xilinx/XilinxVirtualCable
  - https://support.xilinx.com/s/article/974879?language=en_US

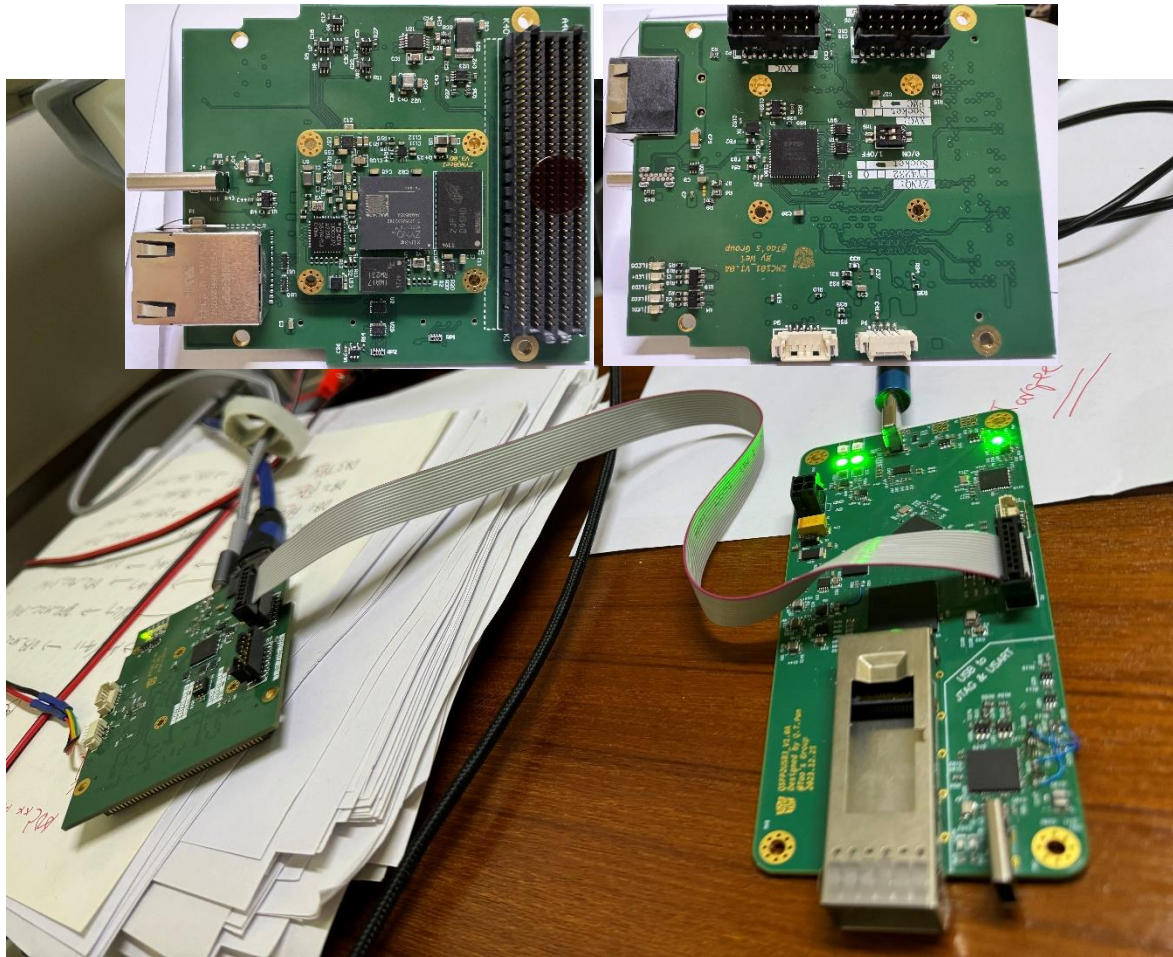


From AXI to BSCAN



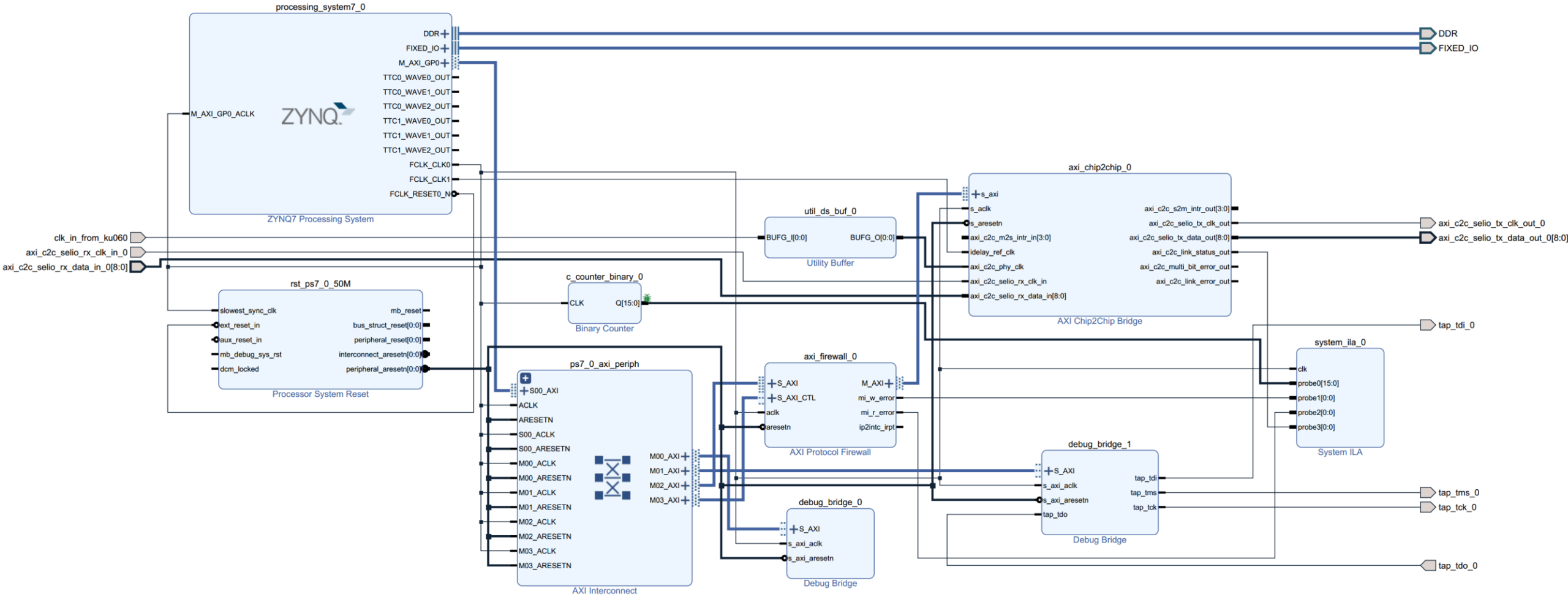
AXI to JTAG Debug Bridge





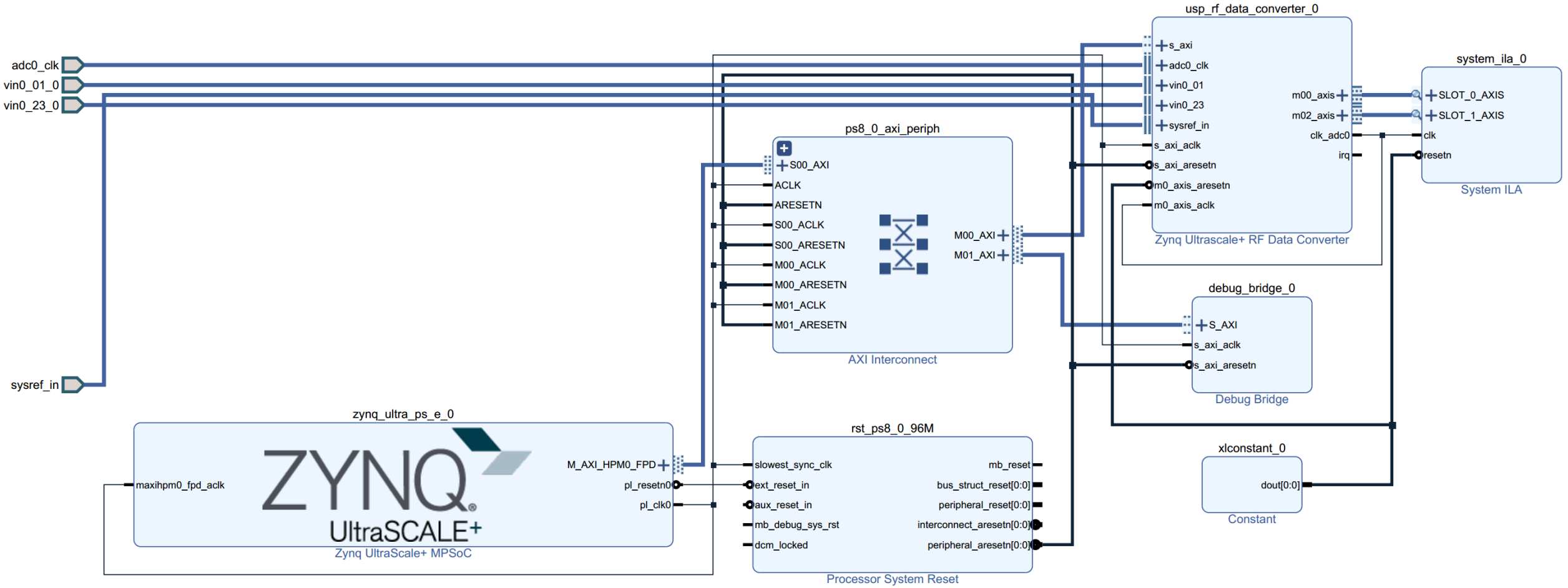
电子学系统远程调试接口的实现：Simple Example

- 使用XVC/Debug Bridge实现ZYNQ/ZYNQMP上复杂PL端固件的远程调试
  - 使用Debug Bridge IP核（AXI to BSCAN）及运行与Embedded Linux上的 xvcServer
  - 配合ILA/VIO实现ZYNQ/ZYNQMP正常运行Embedded Linux并调试PL端固件



电子学系统远程调试接口的实现： RFSoC Example

- 使用XVC/Debug Bridge实现ZYNQ Ultrascale+ RFSoC上RF Data Converter 端固件的远程调试
  - 使用Debug Bridge IP核（AXI to BSCAN）及运行与Embedded Linux上的 xvcServer
  - 配合ILA/VIO实现ZYNQ Ultrascale+ RFSoC正常运行Embedded Linux并调试RF Data Converter端固件

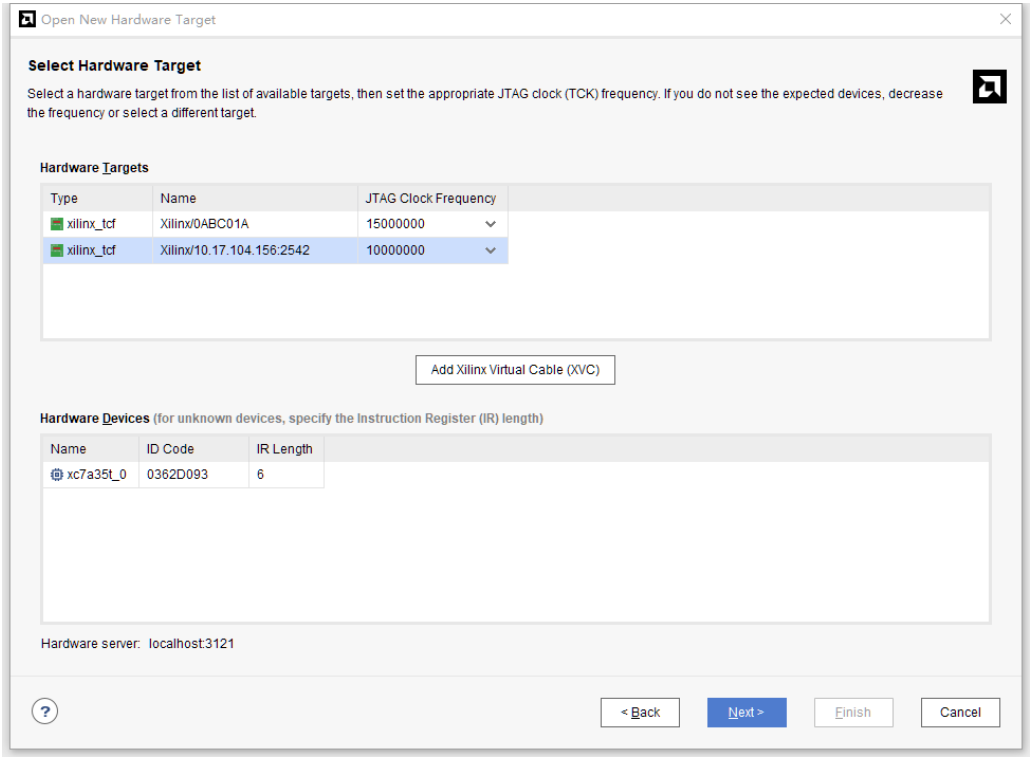


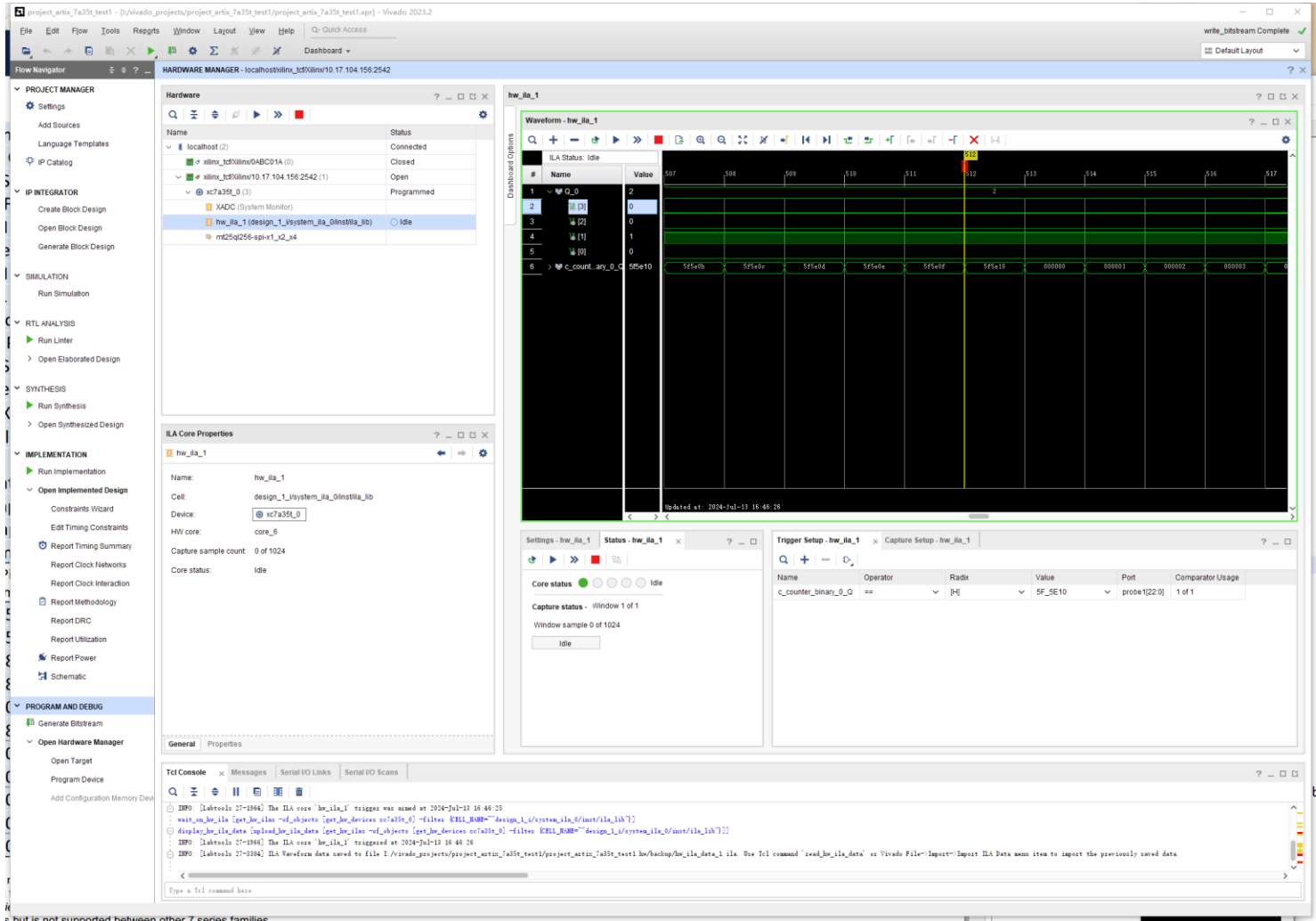
电子学系统远程调试接口的实现

- 使用XVC/Debug Bridge实现ZYNQ/ZYNQMP上复杂PL端固件的远程调试
  - 使用Debug Bridge IP核（AXI to BSCAN）及运行与Embedded Linux上的 xvcServer
  - 配合ILA/VIO实现ZYNQ/ZYNQMP正常运行Embedded Linux并调试PL端固件
- xvcServer的编译和运行
  - 交叉编译器：
    - For ZYNQ-7000
      - https://releases.linaro.org/components/toolchain/binaries/latest-7/arm-linux-gnueabihf/
      - tar -xvf gcc-linaro-7.5.0-2019.12-x86_64_arm-linux-gnueabihf.tar
      - PATH=$PATH:/tools/Cross-Compiler/gcc-linaro-7.5.0-2019.12-x86_64_arm-linux-gnueabihf/bin
      - export CROSS_COMPILE=arm-linux-gnueabihf-
      - export ARCH=arm
    - For ZYNQMP
      - https://releases.linaro.org/components/toolchain/binaries/latest-7/aarch64-linux-gnu/
      - tar -xvf gcc-linaro-7.5.0-2019.12-x86_64_aarch64-linux-gnu.tar
      - PATH=$PATH:/tools/Cross-Compiler/gcc-linaro-7.5.0-2019.12-x86_64_aarch64-linux-gnu/bin
      - export CROSS_COMPILE=aarch64-linux-gnu-
      - export ARCH=aarch64
  - 交叉编译：
    - arm-linux-gnueabihf-gcc -o xvcServer xvcServer.c
  - 运行：
    - xvcServer -d uio0 -p 2542 &

电子学系统远程调试接口的实现

- 使用XVC/Debug Bridge实现ZYNQ/ZYNQMP上复杂PL端固件的远程调试
  - 使用Debug Bridge IP核（AXI to BSCAN）及运行与Embedded Linux上的 xvcServer
  - 配合ILA/VIO实现ZYNQ/ZYNQMP正常运行Embedded Linux并调试PL端固件





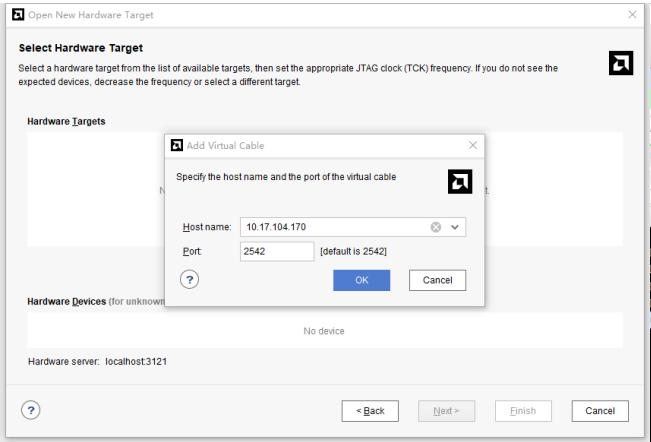
BTW，当然也可以在其他Embedded Linux系统中实现XVC

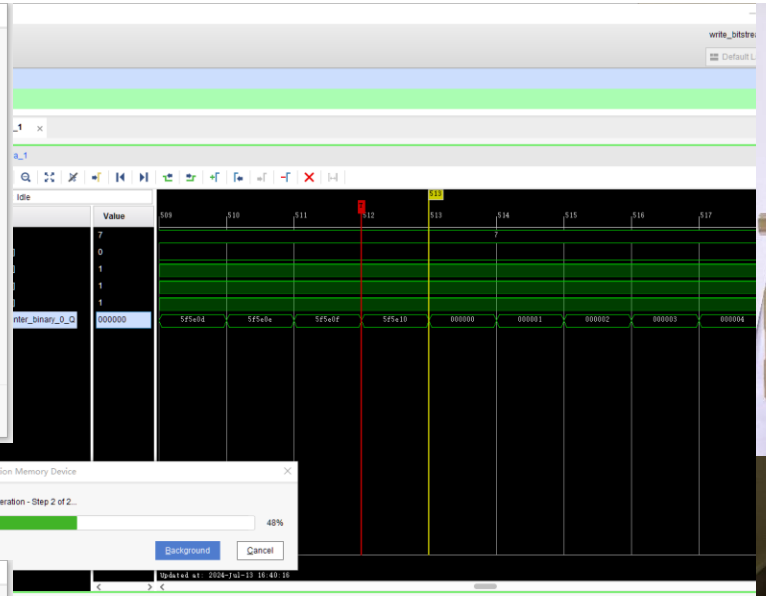
比如在各种Pi开发板中实现XVC

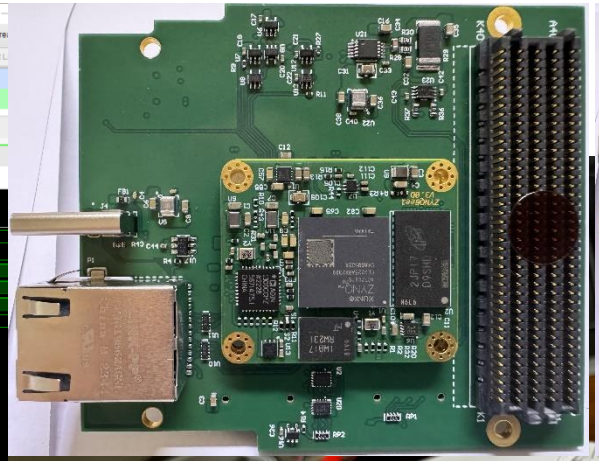
- https://bitbucket.org/Mylium/xvcpi/src/master/
- https://github.com/kholia/xvcpi

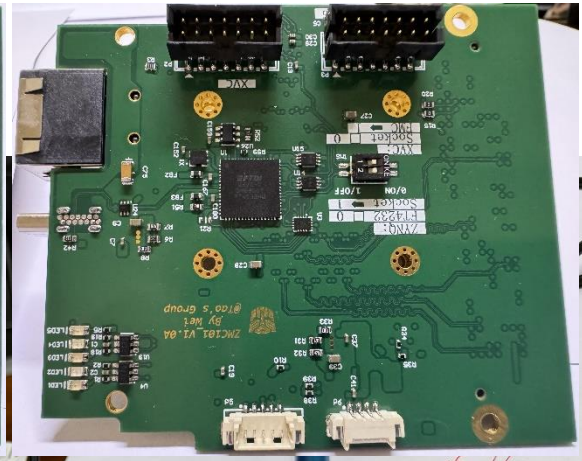
电子学系统远程调试接口的实现

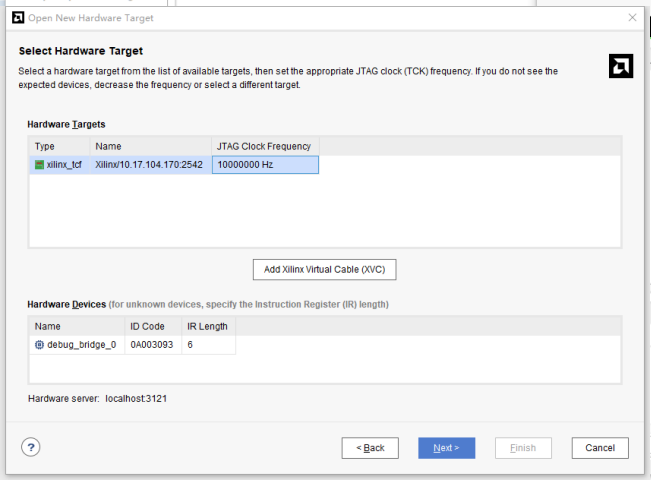
- 使用XVC/Debug Bridge实现ZYNQ/ZYNQMP上复杂PL端固件的远程调试
  - 使用Debug Bridge IP核（AXI to JTAG）及运行与Embedded Linux上的 xvcServer
  - 配合ILA/VIO实现正常使用Vivado远程调试连接到ZYNQ/ZYNQMP XVC接口上其他FPGA

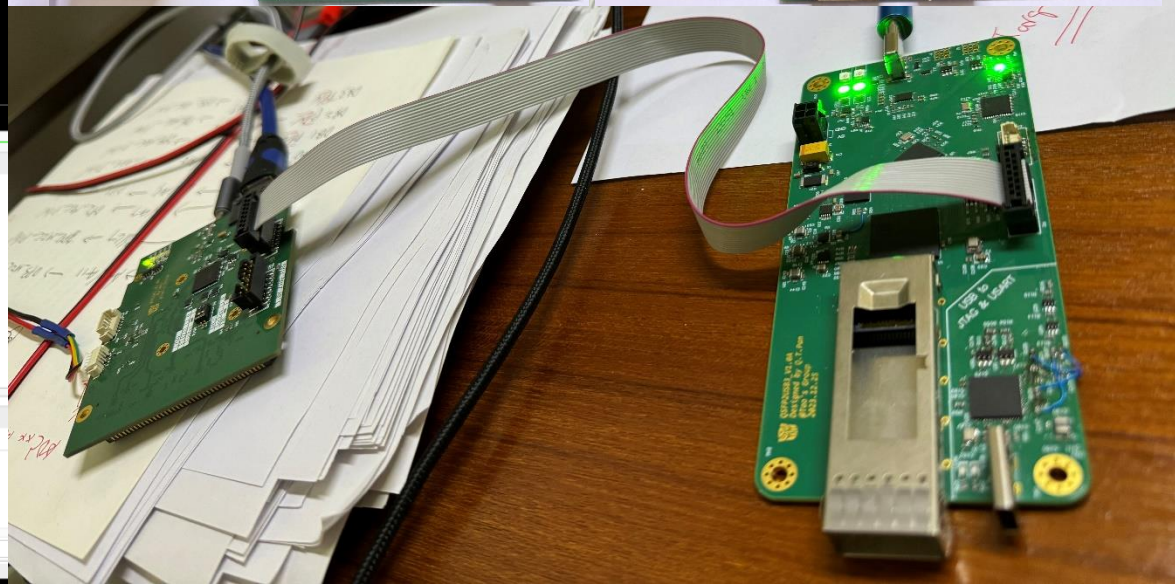






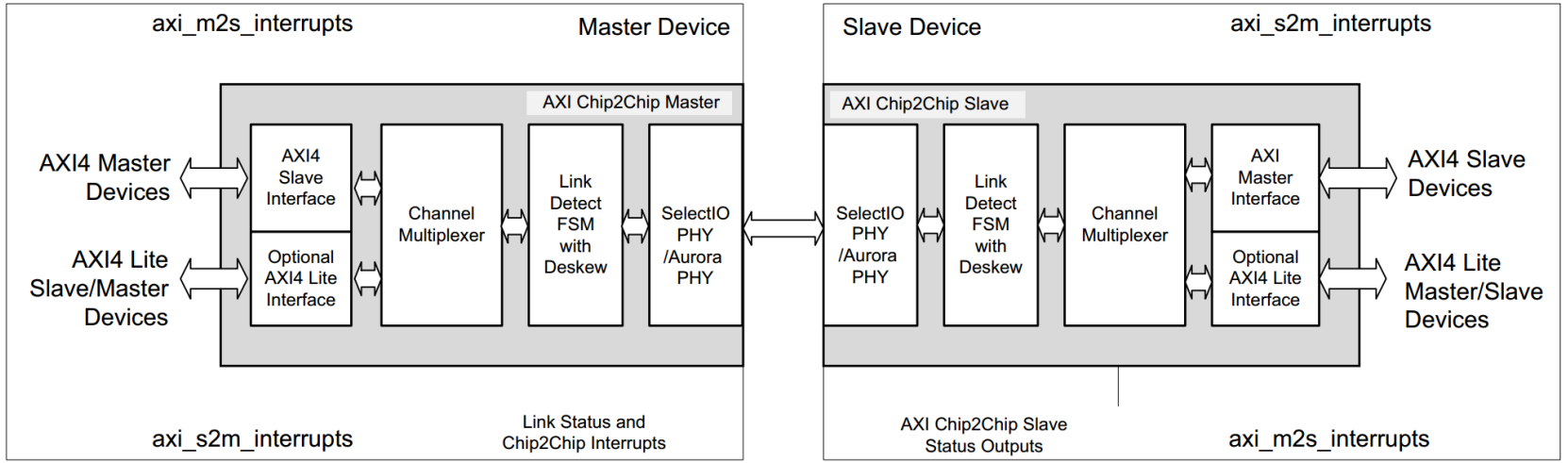


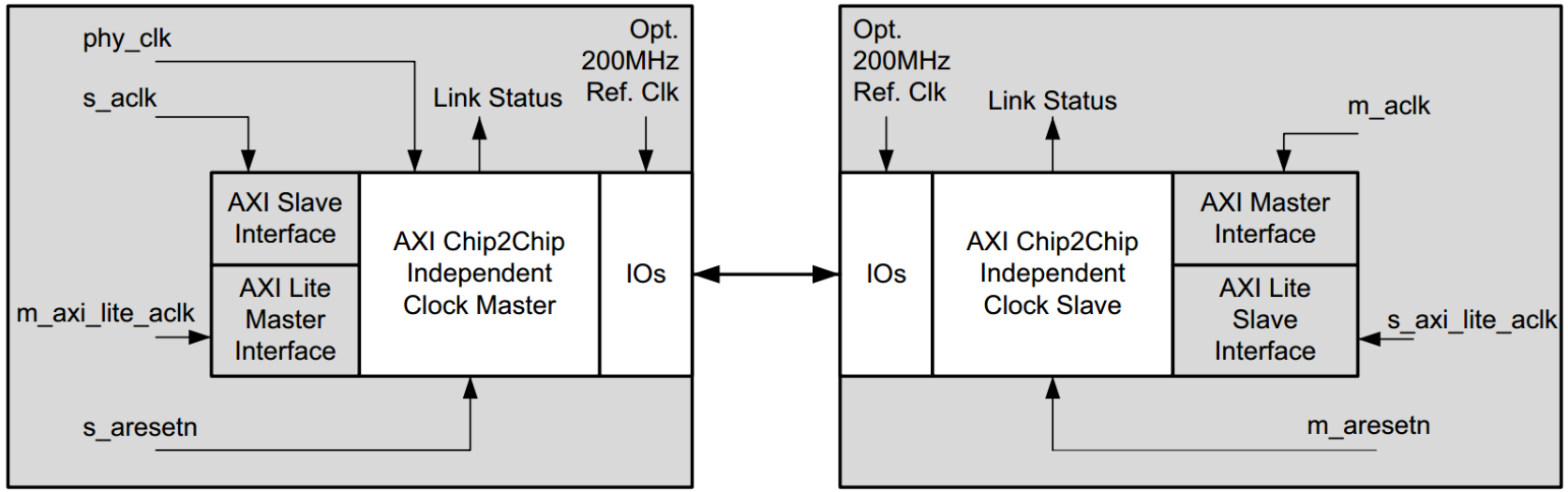




电子学板卡中多个FPGA的互联和慢控制接口的实现

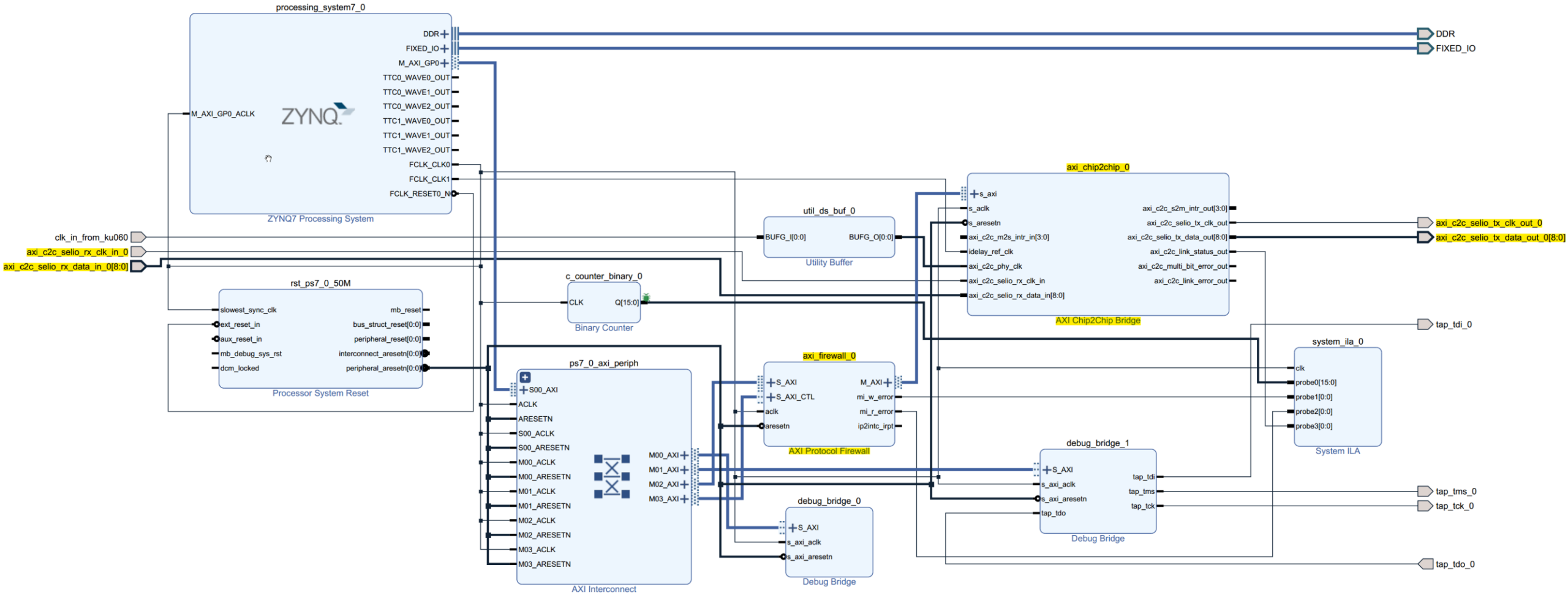
- 使用Chip2Chip（SelectIO/Aurora）实现ZYNQ/ZYNQMP与大规模FPGA的互联和慢控制
  - 使用c2c，经过SelectIO（Single Ended/LVDS）或者Aurora（GT）将两个或者多个FPGA高速互联
  - 至少需要20跟/对IO或者LVDS，或者1对GT（RX+TX）





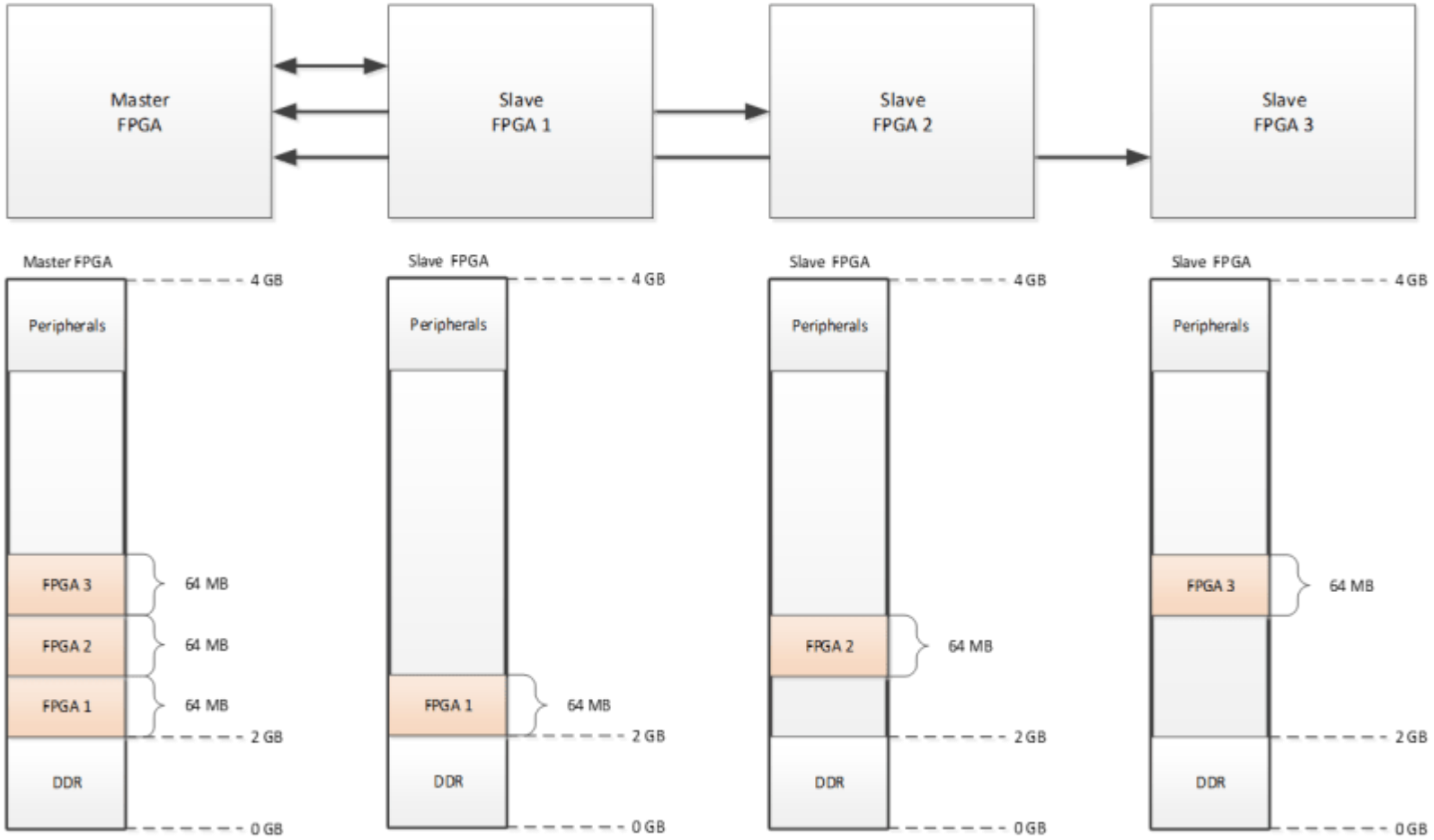
电子学板卡中多个FPGA的互联和慢控制接口的实现

- 使用Chip2Chip（SelectIO/Aurora）实现ZYNQ/ZYNQMP与大规模FPGA的互联和慢控制
  - 可以使用 AXI Protocol Firewall 有效隔离c2c和系统AXI BUS，并保证调试阶段不被 hang



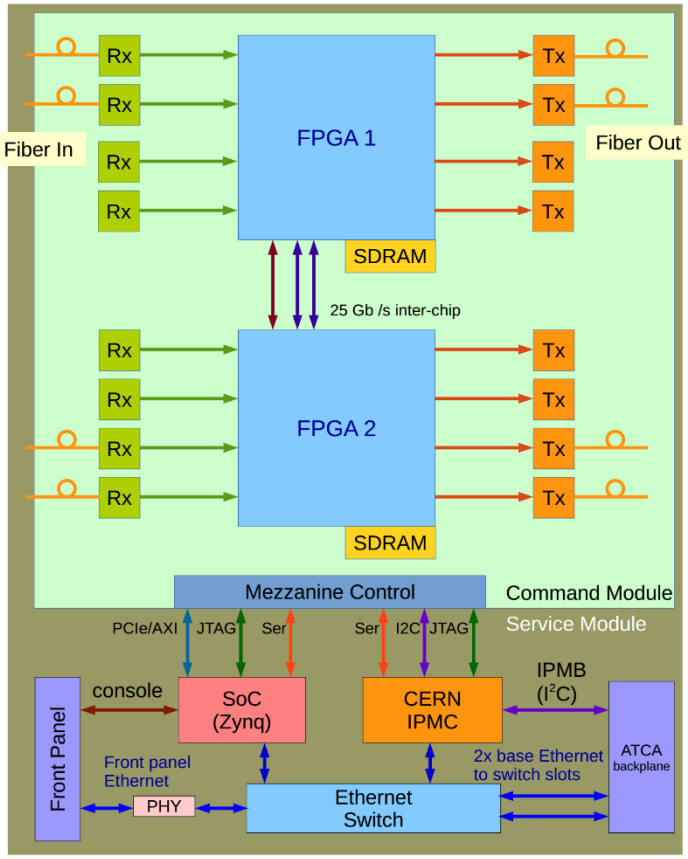
电子学板卡中多个FPGA的互联和慢控制接口的实现

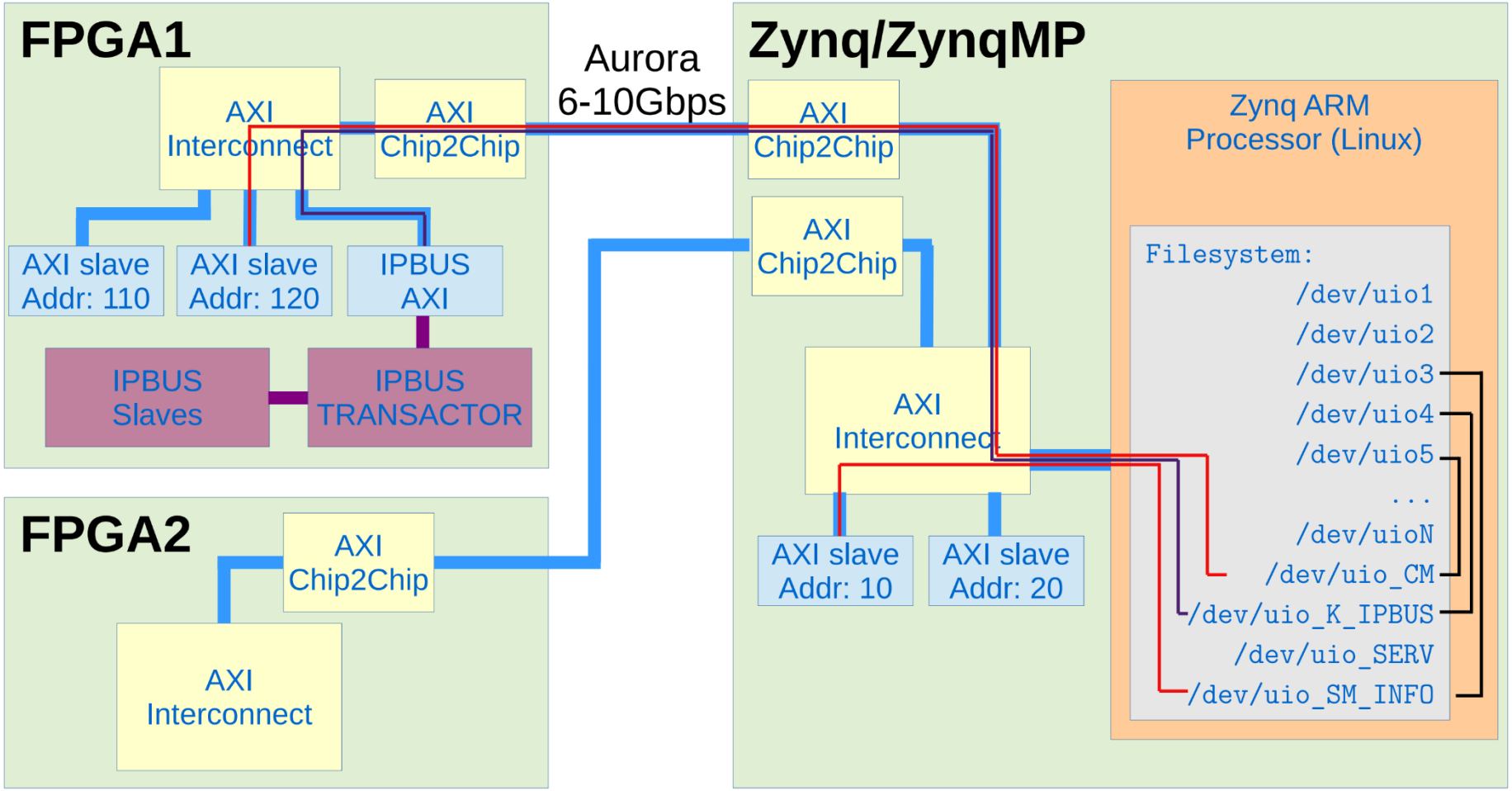
- 使用Chip2Chip（SelectIO/Aurora）实现ZYNQ/ZYNQMP与大规模FPGA的互联和慢控制
  - 实现c2c之后，其他FPGA就像内存映射一样，可以将其一段Memory映射到主FPGA（ZYNQ/ZYNQMP）的地址空间



实践： CERN

- CERN，Apollo Blade Platform Team build APOLLO Platform for CMS and ATLAS





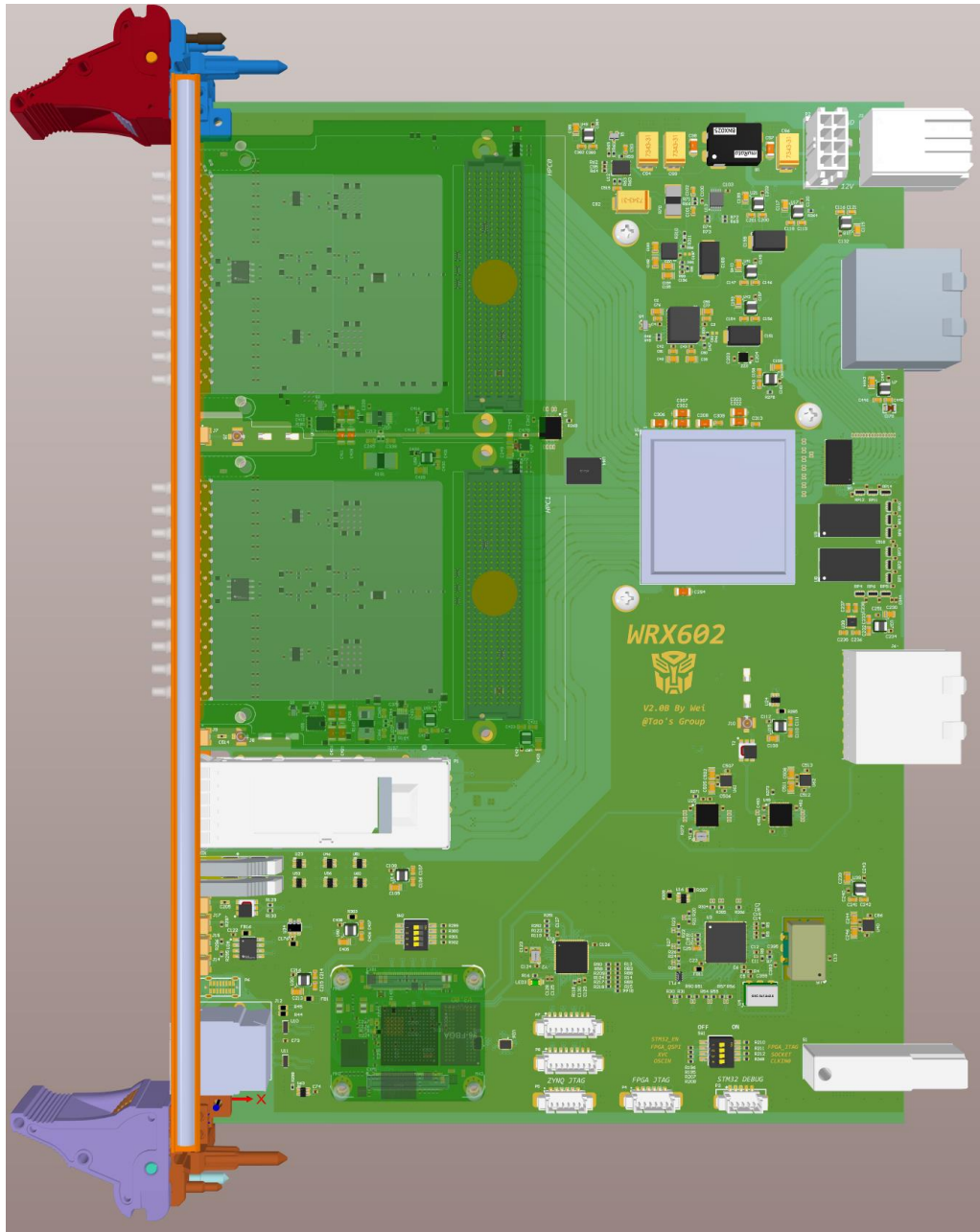
非常棒的设计。不但有硬件设计思路，还包含Embedded Linux上的软件设计思路。

https://indico.cern.ch/event/799275/contributions/3413769/attachments/1861634/3059771/DGastler_SoCWS_2019-06-13.pdf

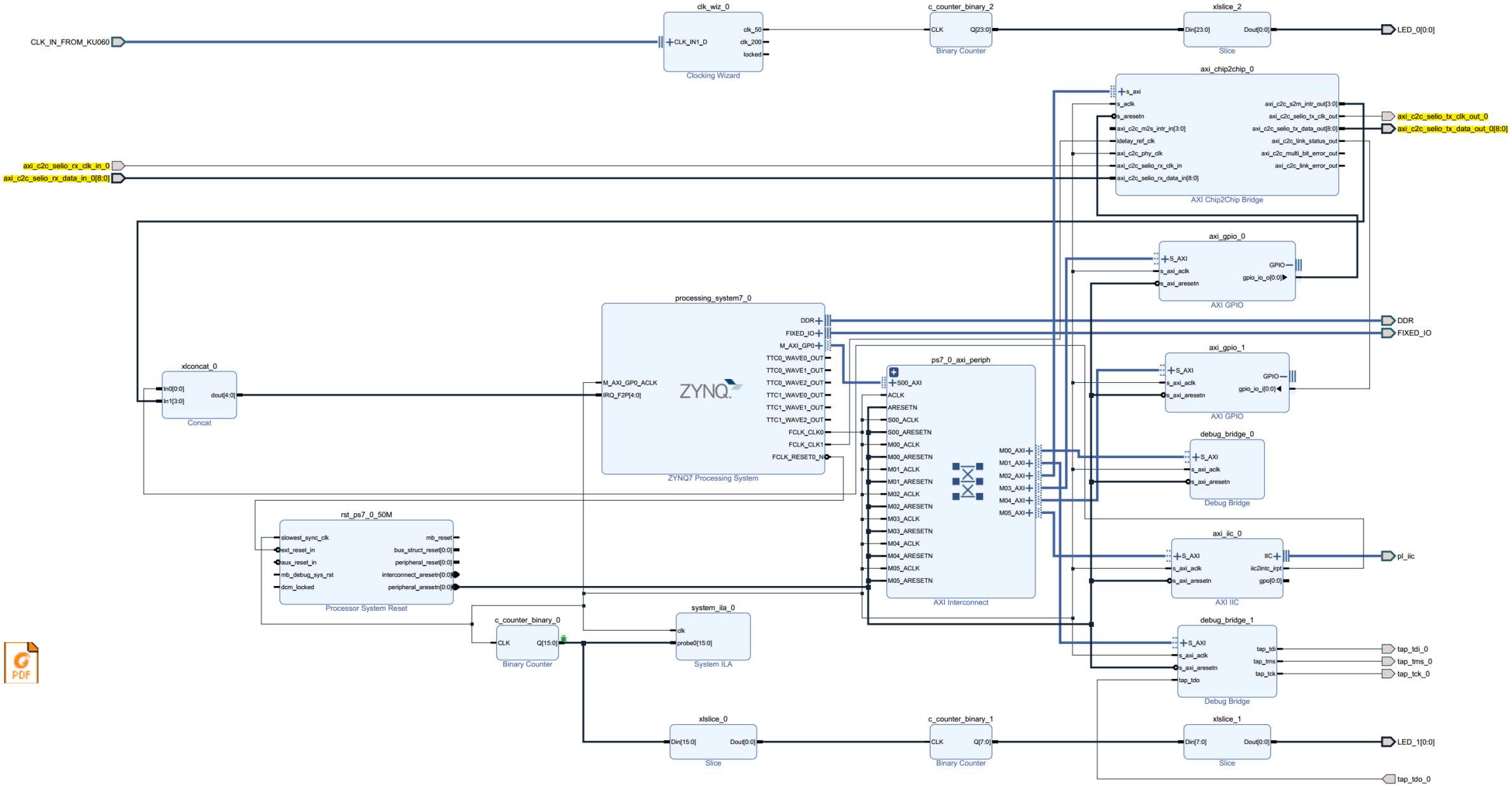
https://indico.cern.ch/event/996093/contributions/4376632/attachments/2260716/3837066/Dgastler-2021-06-09-ApolloUpdate.pdf

实践：我们-锦屏暗物质探测实验和中微子实验读出电子学

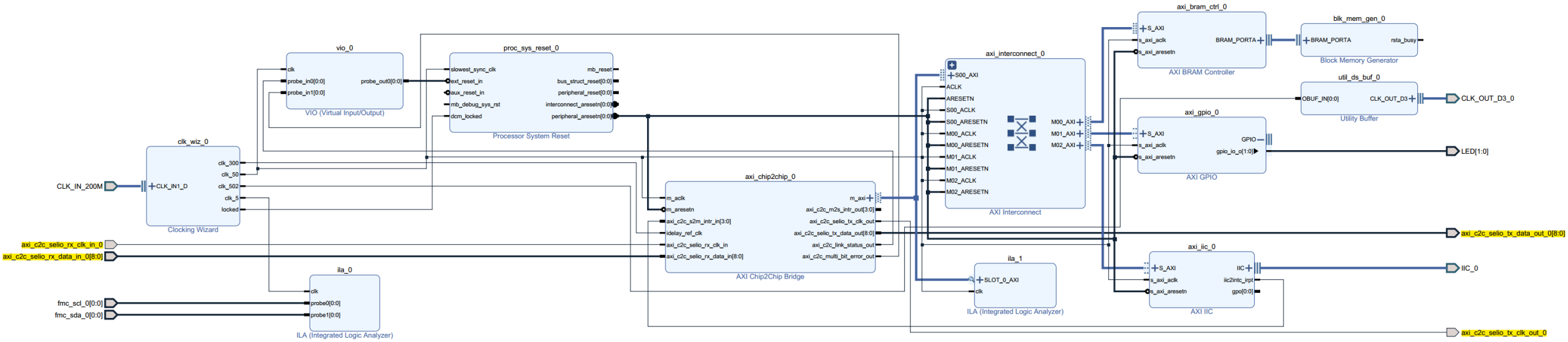
- 在锦屏暗物质探测实验和锦屏中微子观测实验中，读出电子学系统的核心，2×FMC载板WRX602使用了自主研制的ZYNQBee1 V3.0D模块通过XVC和Chip2Chip，实现ZYNQ与大规模FPGA（XCKU060）的互联和慢控制
  - FT4232HQ用于单板调试接口
    - 前面板使用TypeC接口，Channel A实现JTAG，经过电平转换和MUX，可选直接连接到ZYNQ或者KU060
    - FT4232HQ的Channel B实现SWD，连接到底层控制的STM32
    - FT4232HQ的Channel C/D实现UART，分别连接到ZYNQ和STM32
  - ZYNQBee1 V3.0D
    - 512MB DDR3 SDRAM
    - 64MB QSPI Flash
    - 16GB eMMC
    - Ethernet PHY
    - 尺寸为 30mm×35mm
    - 运行Embedded Linux 和 xvcServer，支持SSH远程
  - 2×FMC接口，均支持LA/HA及8 Lanes，板载时钟 LMK04828
  - 前面板QSFP光纤接口和背板8 Lanes/16 Gbps Serdes接口



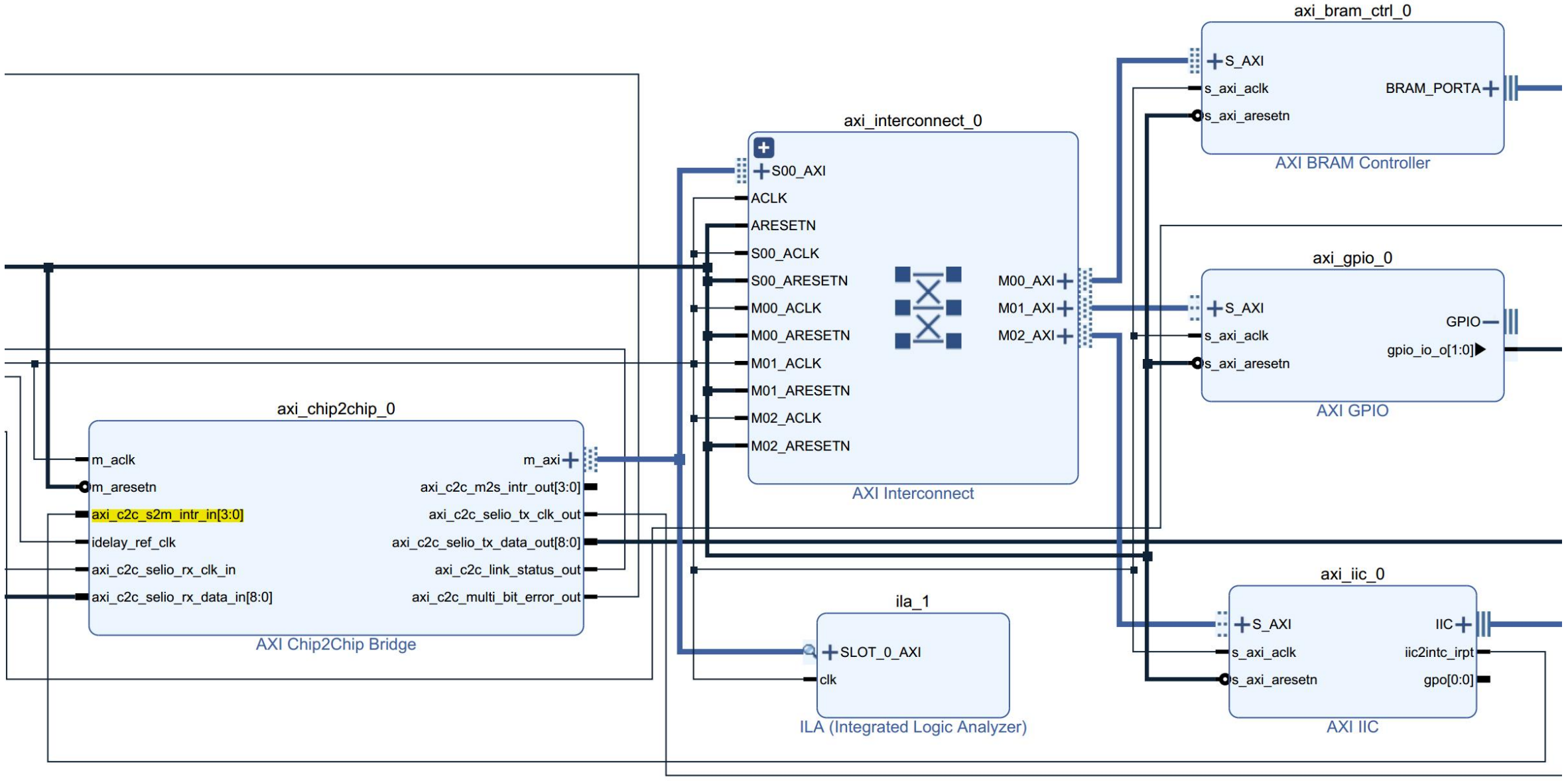
实践： Chip2Chip/ZYNQBee1



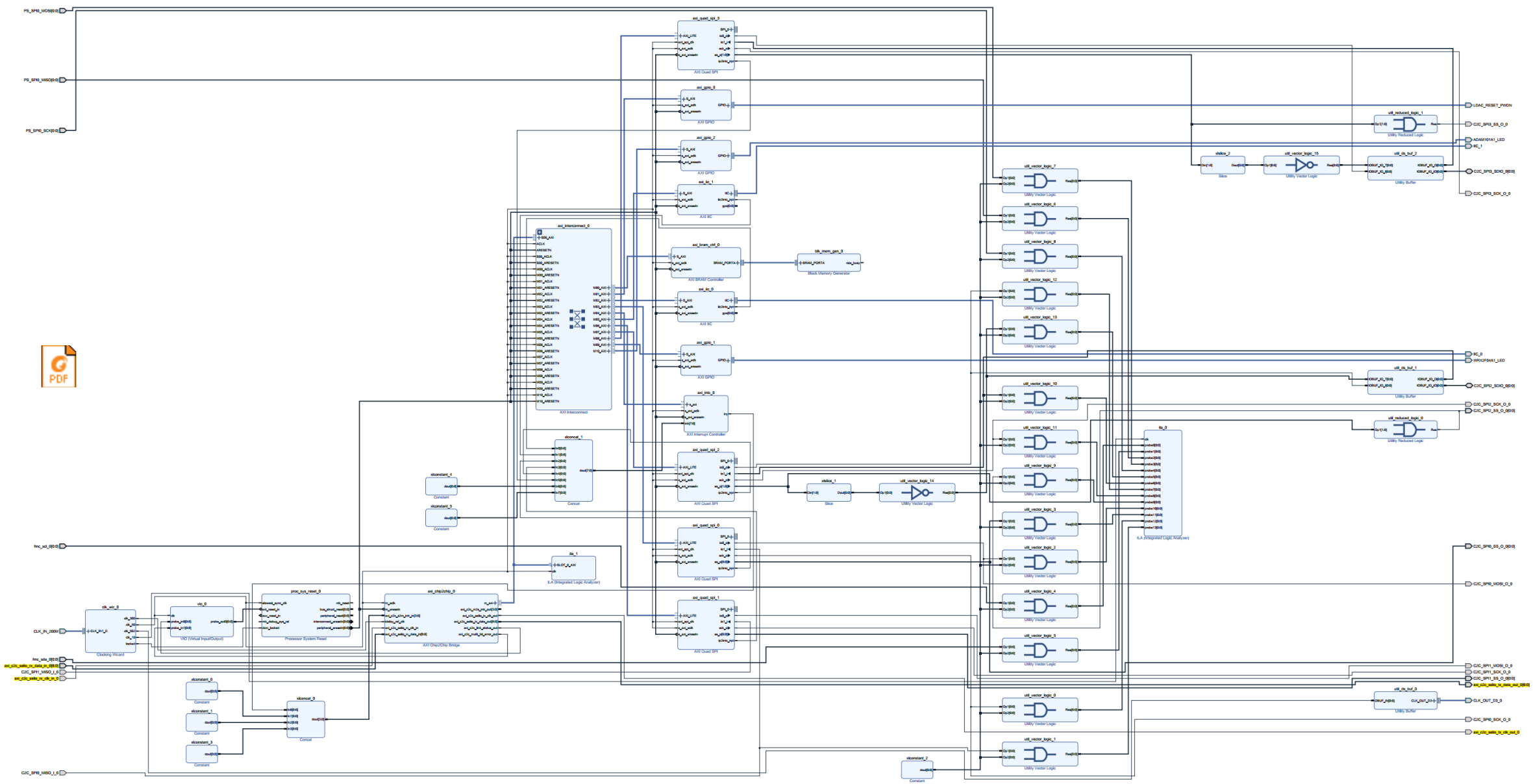
实践： Chip2Chip/KU060 Simple



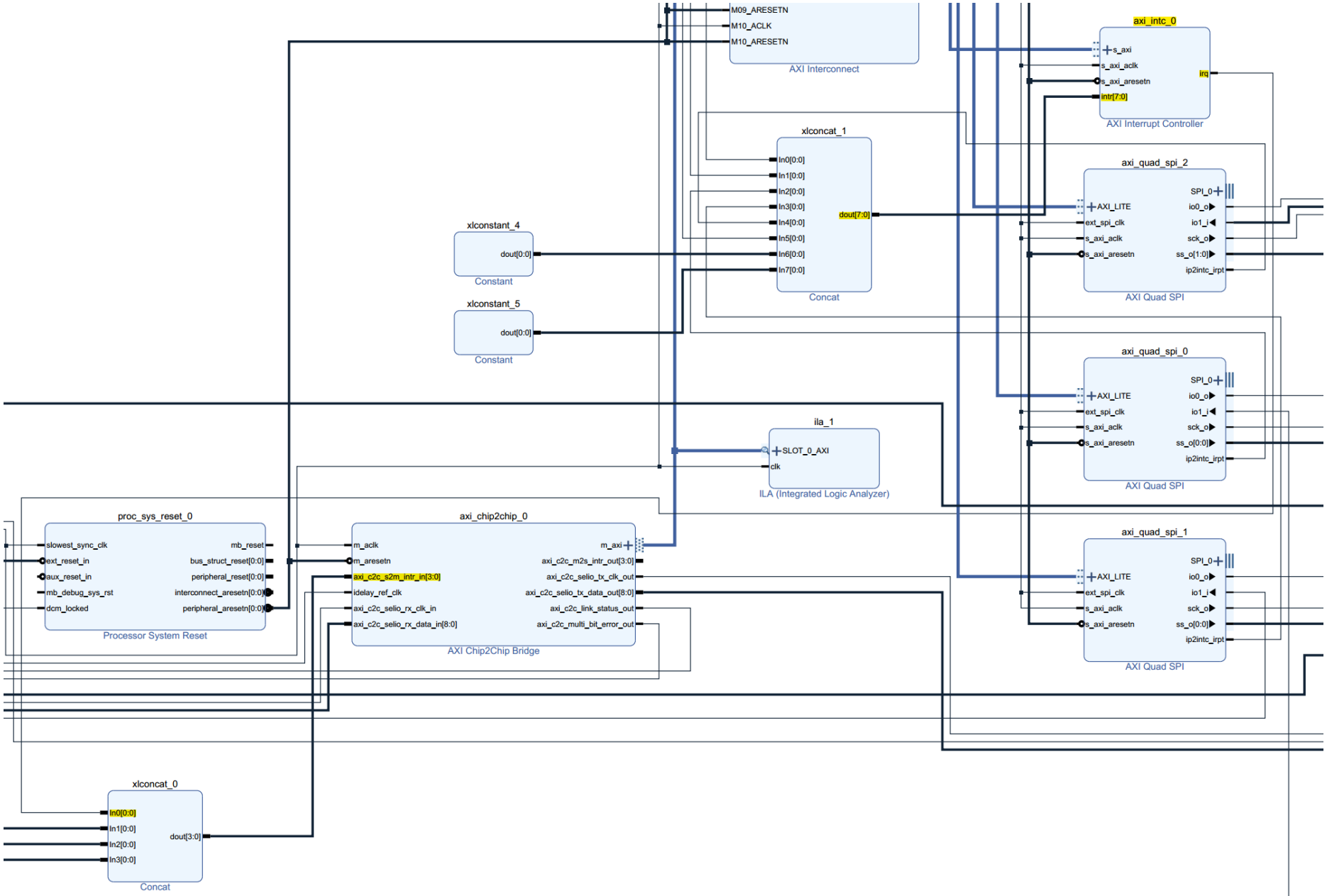
实践：Chip2Chip/KU060 Simple 实现细节与关键内容



实践： Chip2Chip/KU060 Complex



实践：Chip2Chip/KU060 Complex 实现细节与关键内容



实践：Chip2Chip/ZYNQBee1+KU060 实现细节与关键内容

```
/* C2C PL AXI_INTC  */
axi_intc_1: axi_intc@7AAC0000 {
    #interrupt-cells = <2>;
    interrupts = < 0 30 4 >;
    xlnx,sense-of-irq-edge-type = "Rising";
    xlnx,kind-of-intr = <0x0>;
    xlnx,kind-of-edge = <0xFFFFFFFF>;
    xlnx,irq-is-level = <1>;
    xlnx,has-ivr = <1>;
    compatible = "xlnx,axi-intc-4.1" , "xlnx,xps-intc-1.00.a";
    xlnx,disable-synchronizers = <0>;
    xlnx,edk-special = "INTR_CTRL";
    xlnx,kind-of-lvl = <0xFFFFFFFF>;
    xlnx,ivar-reset-value = <0x00000000 0x00000010>;
    xlnx,irq-active = <0x1>;
    interrupt-parent = <&intc>;
    xlnx,rable = <0>;
    xlnx,en-cascade-mode = <0>;
    xlnx,ip-name = "axi_intc";
    xlnx,has-ilr = <0>;
    reg = <0x7AAC0000 0x10000>;
    xlnx,addr-width = <0x20>;
    clocks = <&clkc 15>;
    xlnx,s-axi-aclk-freq-mhz = <50>;
    xlnx,num-sw-intr = <0>;
    xlnx,irq-connection = <1>;
    xlnx,num-intr-inputs = <0x8>;   /* Number of C2C PL Peripheral Interrupts */
    xlnx,has-sie = <1>;
    xlnx,has-cie = <1>;
    xlnx,enable-async = <0>;
    xlnx,num-sync-ff = <2>;
    xlnx,mb-clk-not-connected = <1>;
    xlnx,has-ipr = <1>;
    xlnx,edk-iptype = "PERIPHERAL";
    xlnx,sense-of-irq-level-type = "Active_High";
    xlnx,cascade-master = <0>;
    xlnx,processor-clk-freq-mhz = <100>;
    status = "okay";
    xlnx,is-fast = <0x0>;
    clock-names = "s_axi_aclk";
    xlnx,has-fast = <0>;
    xlnx,ivar-rst-val = <0x0000000000000010>;
    interrupt-controller;
    interrupt-names = "irq";
    xlnx,async-intr = <0xFFFFFF00>; /* 0-means level interrupt */
    xlnx,name = "axi_intc_1";
};
```

```
/* PL AXI_IIC in ZYNQ*/
axi_iic_0: axi_iic@41600000 {
    //interrupts = < 0 29 4 >;
    interrupts = < 0 2 >;
    xlnx,iic-freq-khz = <100>;
    compatible = "xlnx,axi-iic-2.1" , "xlnx,xps-iic-2.00.a";
    xlnx,scl-inertial-delay = <0>;
    interrupt-parent = <&axi_intc_0>;
    xlnx,rable = <0>;
    xlnx,ip-name = "axi_iic";
    xlnx,disable-setup-violation-check = <0>;
    reg = <0x41600000 0x10000>;
    clocks = <&clkc 15>;
    xlnx,gpo-width = <1>;
    xlnx,edk-iptype = "PERIPHERAL";
    xlnx,static-timing-reg-width = <0>;
    xlnx,sda-level = <1>;
    status = "okay";
    clock-names = "s_axi_aclk";
    xlnx,ten-bit-adr = <0>;
    xlnx,default-value = <0x00>;
    interrupt-names = "iic2intc_irpt";
    xlnx,timing-reg-width = <32>;
    xlnx,iic-board-interface = "Custom";
    xlnx,iic-freq = <100000>;
    xlnx,smbus-pmbus-host = <0>;
    xlnx,sda-inertial-delay = <0>;
    xlnx,name = "axi_iic_0";
    xlnx,axi-aclk-freq-mhz = <50>;
};
```

```
/* PL AXI_IIC through Chip2Chip in KU060  */
/* HPC1_FMC IIC  */
axi_iic_2: axi_iic@7AAA1000 {
    //interrupts = < 0 30 4 >;
    interrupts = < 1 2 >;
    xlnx,iic-freq-khz = <100>;
    compatible = "xlnx,axi-iic-2.1" , "xlnx,xps-iic-2.00.a";
    xlnx,scl-inertial-delay = <0>;
    interrupt-parent = <&axi_intc_1>;
    xlnx,rable = <0>;
    xlnx,ip-name = "axi_iic";
    xlnx,disable-setup-violation-check = <0>;
    reg = <0x7AAA1000 0x1000>;
    clocks = <&clkc 15>;
    xlnx,gpo-width = <1>;
    xlnx,edk-iptype = "PERIPHERAL";
    xlnx,static-timing-reg-width = <0>;
    xlnx,sda-level = <1>;
    status = "okay";
    clock-names = "s_axi_aclk";
    xlnx,ten-bit-adr = <0>;
    xlnx,default-value = <0x00>;
    interrupt-names = "iic2intc_irpt";
    xlnx,timing-reg-width = <32>;
    xlnx,iic-board-interface = "Custom";
    xlnx,iic-freq = <100000>;
    xlnx,smbus-pmbus-host = <0>;
    xlnx,sda-inertial-delay = <0>;
    xlnx,name = "axi_iic_2";
    xlnx,axi-aclk-freq-mhz = <50>;
    /* C2C_PL_PMBUS */
    /* Leave NULL, we will add with devicetree overlay.  */
};
```

总结和展望

- 使用FTDI公司的FT2232HQ和FT4232HQ，配合Vivado开发环境，能够有效解决单板电子学板卡的板载调试接口需求；

- 使用ZYNQBee1 运行Embedded Linux，实现Debug Bridge，能够实现远程JTAG调试、编程接口。实现大规模、复杂、分布式电子学系统的调试；

- 使用Chip2Chip，能够将其他大型FPGA的控制、数据接口映射到ZYNQBee1运行的Embedded Linux中，非常有利于大规模、复杂、分布式电子学系统控制和维护。

- 未来在锦屏暗物质探测实验和锦屏中微子观测实验中使用上述技术来满足大规模电子学系统的各项需求。

谢谢!


ANQUANHUOCHAI
安全
大紫
SAFETY MATCHES
XILINX
ZYNQ
XC7Z020
CLG484ABX2241
TAIWAN