

实验 1:Zynq 平台上实现 PS+PL 系统协同设计 实验报告

15061075 张安澜

一、实验目的

通过使用专用硬件加速 AES 加密解密算法，掌握：

1. 使用 Vivado/SDK 软件进行软硬件联合设计的基本流程；
2. 使用 AXI 通信总线协议进行 PL、PS 之间通信的方法；
3. 在 Zynq PL 部分进行专用硬件架构的设计和验证方法；

二、实验要求

1. 完成要求的文档的阅读并按照配套教程完成实验；
2. 实验以个人为单位完成；
3. 实验内容:PS 端以 Verilog 代码实现加密解密模块,通过 AXI 接口 IP 连接到 PS 端,生成硬件并导入 SDK 环境中,经过仿真、测试、验证之后,下载到开发板上进行实测。(注意:本实验提供加密解密模块的 Verilog 源代码,如果有兴趣的话也可以自行采用 HLS 的方式将 C/C++描述生成硬件描述并封装成 IP 核,对此部分的实现形式不做要求。);
4. 实验完成后撰写实验报告,要求包括项目架构设计介绍、关键实验步骤说明、核心设计部分的源代码；

三、实验环境

1. Vivado HLx 套装及 Vivado SDK；
2. AX7020 开发板；
3. Mini SD 卡读卡器及 USB 转接线；

四、设计方案

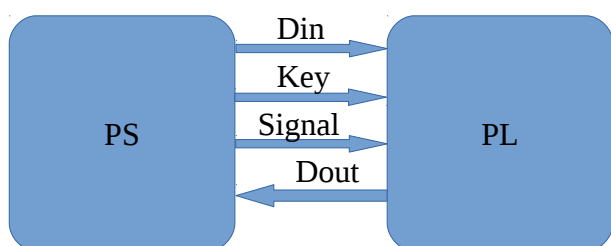
1. 选择经过 FPGA/ASIC 验证过的 AES 内核

(<http://www.aoki.ecei.tohoku.ac.jp/crypto/web/cores.html> 选择 AES 中的第一个 AES.v 文件)；

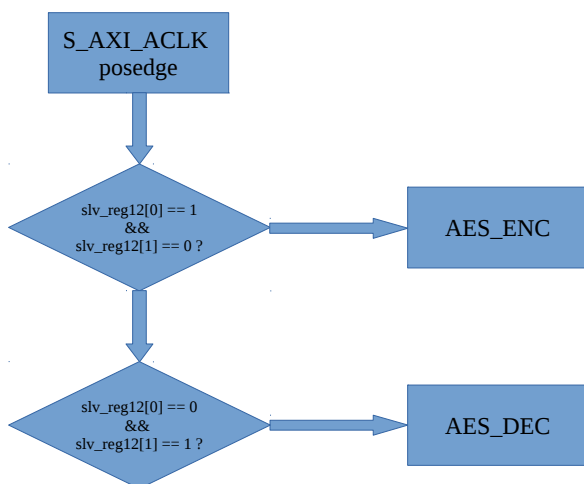
2. 创建一个 AXI Lite 接口的 IP，该 AXI 接口包括 13 个 32 位的寄存器 slv_reg0~slv_reg12。其中，slv_reg0 到 slv_reg3 这四个寄存器用于传输需要加密（或解密）的明文（或密文），slv_reg0 用于传输数据的第 127 位到第 96 位，slv_reg1 用于传输数据的第 95 位到第 64 位，slv_reg2 用于传输数据的第 63 位到第 32 位，slv_reg3 用于传输数据的第 31 位到第 0 位；slv_reg4 到 slv_reg7 这四个寄存器用于传输加密（或解密）密钥，同样，slv_reg4 用于传输密钥的第 127 位到第 96 位，slv_reg5 用于传输密钥的第 95 位到第 64 位，slv_reg6 用于传输密钥的第 63 位到第 32 位，slv_reg7 用于传输密钥的第 31 位到第 0 位；slv_reg8 到 slv_reg11 这四个寄存器用于传输加密（或解密）以后的密文（或明文），同样，slv_reg8 用于传输数据的第 127 位到第 96 位，slv_reg9 用于传输数据的第 95 位到第 64 位，slv_reg10 用于传输数据的第 63 位到第 32 位，slv_reg11 用于传输数据的第 31 位到第 0 位；最后一个寄存器 slv_reg12 用于传输控制信号，只用到了第 0 位和第 1 位，若第 0 位为 1 则需要进行加密，若第 1 位为 1 则需要进行解密，且这两位不能同时为 1；

3. 在 SDK 中以 Standalone 模式创建两个 C/C++项目，其中一个项目为 AES_ENC，用于在板上实测 IP 核的 AES 加密功能，另一个项目为 AES_DEC，用于在板上实测 IP 核的 AES 解密功能。主要的代码逻辑为：传输明文（密文）→ 传输密钥 → 传输控制信号 → 接收密文（明文）。在板上实测时，使用 JTAG 模式启动开发板，分别运行上述两个项目；

5. 架构的图示如下：



6.IP 核内部用户图示如下：



五、具体实现

1.实验步骤

- 对选择的 AES_ENC 和 AES_DEC 模块进行仿真测试，**根据 AES_TB.v 理解这两个模块加密解密的过程**；
 - 创建一个新的 IP 核，**根据设计修改自动生成的代码（这里要注意需要将除用户逻辑以外对 slv_reg8~slv_reg11 这四个寄存器的写入操作全都注释或删除去，否则会出错），添加用户逻辑（参考 AES_TB.v）**；
 - 添加完成后对用户逻辑进行仿真测试，测试正确以后依次进行 Synthesis 和 Implement**，若无报错则可进行 IP 核的封装；
 - 封装完成后创建 Design，添加 ZYNQ7000 的 IP 核(PS 部分 IP 核)并完成配置，然后添加自定义的 IP 核，进行端口连接；
 - 依次进行 Synthesis、Implement、Create Bitstream 并导出相关文件进入 SDK；
 - 在 SDK 中编写 PS 端的编程**；
 - 以 JTAG 模式启动开发板，运行上述程序并将结果与预期对比。
- 关键步骤已用粗体标出；

2.核心代码

a.IP 核中的用户逻辑

```
AES_ENC AES_ENC(
    .Din(Din),
    .Key(Key),
    .Dout(Dout_E),
    .Drdy(Drdy),
    .Krdy(Krdy),
    .RSTn(RSTn),
    .EN(EN_E),
    .CLK(S_AXI_ACLK),
    .BSY(BSY_E),
    .Dvld(Dvld_E)
);
AES_DEC AES_DEC(
    .Din(Din),
    .Key(Key),
    .Dout(Dout_D),
    .Drdy(Drdy),
    .Krdy(Krdy),
```

```

.RSTn(RSTn),
.EN(EN_D),
.CLK(S_AXI_ACLK),
.BSY(BSY_D),
.Dvld(Dvld_D)
);
integer count = 0;
always@(posedge S_AXI_ACLK)
begin
    if(slv_reg12[0] == 1'b1 && slv_reg12[1] == 1'b0)
    begin
        if(count == 0)
        begin
            EN_E <= 0;
            RSTn <= 0;
            Krdy <= 0;
            Drdy <= 0;
        end
        else if(count == 1)
        begin
            RSTn <= 1;
            EN_E <= 1;
            //Key <= 128'h000102030405060708090a0b0c0d0e0f;
            Key[127:96] <= slv_reg4;
            Key[95:64] <= slv_reg5;
            Key[63:32] <= slv_reg6;
            Key[31:0] <= slv_reg7;
            Krdy <= 1;
        end
        else if(count == 2)
        begin
            Krdy <= 0;
            //Din <= 128'h00112233445566778899aabbccddeeff;
            Din[127:96] <= slv_reg0;
            Din[95:64] <= slv_reg1;
            Din[63:32] <= slv_reg2;
            Din[31:0] <= slv_reg3;
            Drdy <= 1;
        end
        else if(count == 3)
        begin
            Drdy <= 0;
        end
        else if(count == 14)
        begin
            slv_reg8 = Dout_E[127:96];
            slv_reg9 = Dout_E[95:64];
            slv_reg10 = Dout_E[63:32];
            slv_reg11 = Dout_E[31:0];
        end
        count <= count + 1;
    end
    else if(slv_reg12[1] == 1'b1 && slv_reg12[0] == 1'b0)
    begin
        if(count == 0)
        begin
            EN_D <= 0;
            RSTn <= 0;
            Krdy <= 0;
            Drdy <= 0;

```

```

end
else if(count == 1)
begin
    RSTn <= 1;
    EN_D <= 1;
    //Key <= 128'h13111d7fe3944a17f307a78b4d2b30c5;
    Key[127:96] <= slv_reg4;
    Key[95:64] <= slv_reg5;
    Key[63:32] <= slv_reg6;
    Key[31:0] <= slv_reg7;
    Krdy <= 1;

end
else if(count == 2)
begin
    Krdy <= 0;
    //Din <= 128'h69c4e0d86a7b0430d8cdb78070b4c55a;
    Din[127:96] <= slv_reg0;
    Din[95:64] <= slv_reg1;
    Din[63:32] <= slv_reg2;
    Din[31:0] <= slv_reg3;
    Drdy <= 1;

end
else if(count == 3)
begin
    Drdy <= 0;

end
else if(count == 14)
begin
    slv_reg8 = Dout_D[127:96];
    slv_reg9 = Dout_D[95:64];
    slv_reg10 = Dout_D[63:32];
    slv_reg11 = Dout_D[31:0];

end
count <= count + 1;
end
end

```

b.SDK 中的程序（以 AES_ENC 项目为例）

```

#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include "aes.h"
#include "xparameters.h"
#include "xil_io.h"

int main()
{
    init_platform();

    print("AES IP TEST...\n\r");
    print("AES ENC...\n\r");
    // Plain Text
    AES_mWriteReg(XPAR_AES_0_S00_AXI_BASEADDR, 0, 0x00112233);
    AES_mWriteReg(XPAR_AES_0_S00_AXI_BASEADDR, 4, 0x44556677);
    AES_mWriteReg(XPAR_AES_0_S00_AXI_BASEADDR, 8, 0x8899aabb);
    AES_mWriteReg(XPAR_AES_0_S00_AXI_BASEADDR, 12, 0xccddeeff);

    // Key
    AES_mWriteReg(XPAR_AES_0_S00_AXI_BASEADDR, 16, 0x00010203);

```

```

AES_mWriteReg(XPAR_AES_0_S00_AXI_BASEADDR, 20, 0x04050607);
AES_mWriteReg(XPAR_AES_0_S00_AXI_BASEADDR, 24, 0x08090a0b);
AES_mWriteReg(XPAR_AES_0_S00_AXI_BASEADDR, 28, 0x0c0d0e0f);

// Signal
AES_mWriteReg(XPAR_AES_0_S00_AXI_BASEADDR, 48, 0x00000001);

// Cipher Text
u32 r0,r1,r2,r3;
r0 = AES_mReadReg(XPAR_AES_0_S00_AXI_BASEADDR, 32);
r1 = AES_mReadReg(XPAR_AES_0_S00_AXI_BASEADDR, 36);
r2 = AES_mReadReg(XPAR_AES_0_S00_AXI_BASEADDR, 40);
r3 = AES_mReadReg(XPAR_AES_0_S00_AXI_BASEADDR, 44);
xil_printf("r0 = %08x, r1 = %08x, r2 = %08x, r3 = %08x \n\r", r0, r1, r2, r3);

cleanup_platform();
return 0;
}

```

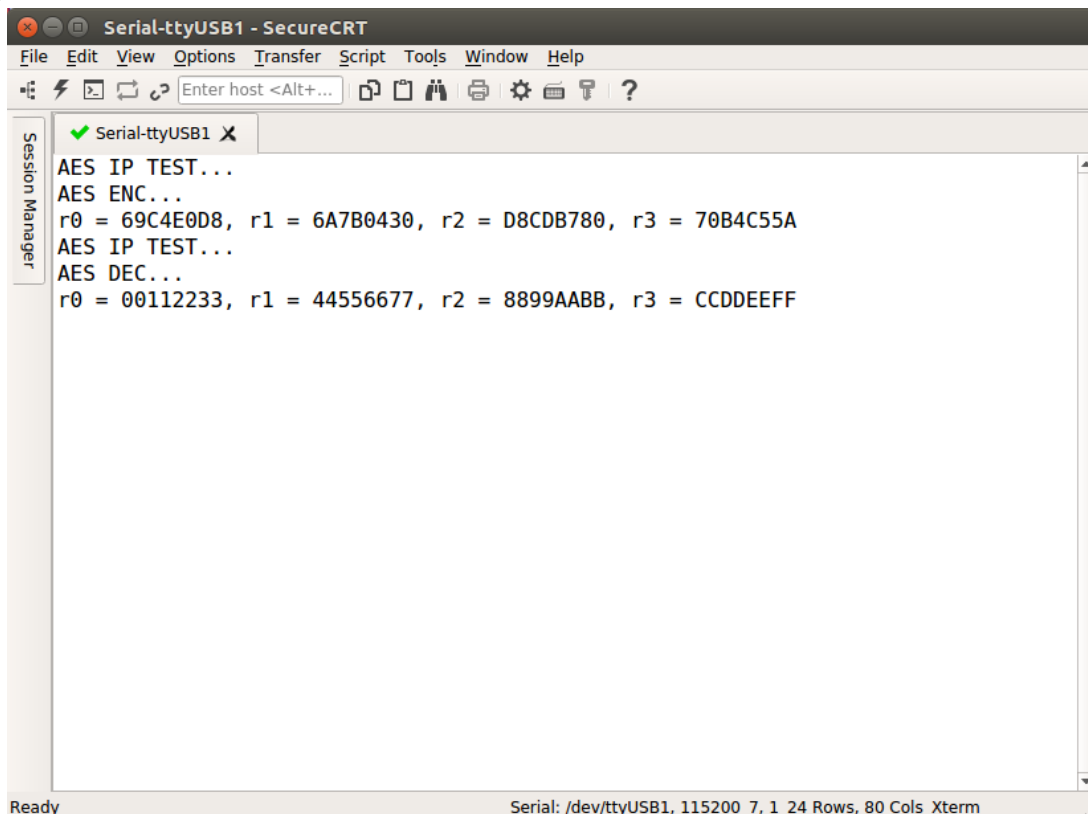
3.遇到的困难与解决方案

主要问题：对使用 Vivado/SDK 软件进行软硬件联合设计的基本流程比较陌生，对需要实现的东西感觉非常模糊，遇到问题不知道如何调试。

解决方案：阅读文档《ALINX 黑金 Zynq7000 开发平台配套教程 V1.02》中的 9、10、13 章实验；在网上查阅了一些资料，其中 <http://blog.csdn.net/shushm/article/details/49536845> 这个博客实现了一个 16 位加法器，并且对 IP 核中自动生成的代码做了一定的讲解；阅读文档《非常重要——SDK 调试技巧》；

六、实验结果

分别运行 AES_ENC 和 AES_DEC 两个项目（注意设置 Run Configuration），得到如下图所示结果：



```

Serial-ttyUSB1 - SecureCRT
File Edit View Options Transfer Script Tools Window Help
[Icons] Enter host <Alt+... [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons]
Session Manager
Serial-ttyUSB1 X
AES IP TEST...
AES ENC...
r0 = 69C4E0D8, r1 = 6A7B0430, r2 = D8CDB780, r3 = 70B4C55A
AES IP TEST...
AES DEC...
r0 = 00112233, r1 = 44556677, r2 = 8899AABB, r3 = CCDDEEFF

Ready
Serial: /dev/ttyUSB1, 115200 7, 1 24 Rows, 80 Cols Xterm

```

七、分析与讨论

在完成了以后感觉这个实验其实并不是很难，很多问题都是由于第一次使用这些开发环境以及对自己需要实现的东西没有概念而产生的：比如一开始执行 Implement 的时候没有创建 HDL Wrapper 导致一直报错；在一开始上板实测的时候由于没有复位系统导致输出一直为 0 等。

个人的收获主要是对 Vivado/SDK 软件进行软硬件联合设计的基本流程有了更熟练的掌握并且对 Zynq 平台上 PS+PL 系统协同设计有了一定的认识。