

Lab4 矩阵乘法器（自选）

【该实验与张展鹏同学共同完成】

一、实验目标

- 1、使用 Vitis HLS 构建一个项目
- 2、在 Vitis HLS 中生成矩阵乘法加速的 IP 核，进行仿真、综合与 IP 导出。
- 3、使用 Vivado 对 HLS 导出的 IP 进行集成。
- 4、在 Jupyter Notebook 上完成 IP 的调用。

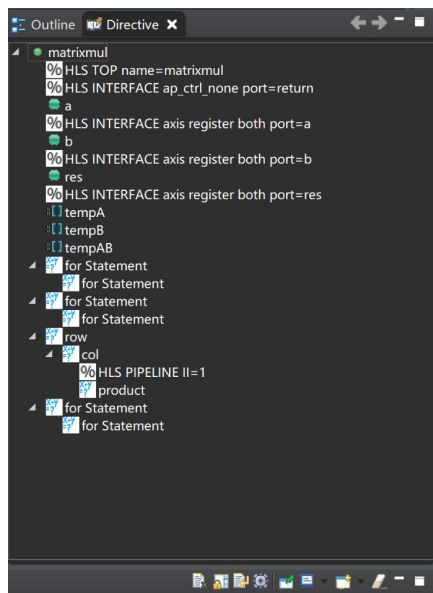
二、实验环境

- 1、PYNQ-Z2 远程实验室服务或物理板卡
- 2、Vitis HLS
- 3、Vivado

三、实验过程遇到的问题

【实验步骤】

- 1、依照 Lab1、Lab2 方式进行项目创建。
 - 2、mul.h 文件中定义输入的 a,b 矩阵维度（32x32）和输出矩阵的维度(32x32)。
 - 3、mul.cpp 文件中写明矩阵乘法的完整逻辑。
- 注：相关代码文件详见 Github。
- 4、给 mul.cpp 文件添加接口约束和循环流水如下：



5、C simulation 结果

```
Vitis HLS Console
INFO: [SIM 211-2] ***** CSIM start *****
INFO: [SIM 211-4] CSIM will launch GCC as the compiler.
make: 'csim.exe' is up to date.
Test passed.
INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] ***** CSIM finish *****
INFO: [HLS 200-111] Finished Command csim_design CPU user time: 0 seconds. CPU system time: 0 seconds. Elapsed time: 0.947 seconds; current allocated memory: 1.256 GB.
INFO: [HLS 200-112] Total CPU user time: 4 seconds. Total CPU system time: 2 seconds. Total elapsed time: 4.451 seconds; peak allocated memory: 1.256 GB.
Finished C simulation.
```

6、C synthesis 结果

The screenshot shows the Synthesis Summary Report for 'matrixmul' in Vitis HLS 2021.2. The report includes the following sections:

- General Information:** Date: Mon Jul 17 18:00:42 2023, Version: 2021.2 (Build 3367213 on Tue Oct 19 02:48:09 MDT), Project: mux1, Solution: solution1 (Vivado IP Flow Target), Product: zynq, Target device: xc7z020-clg484-1.
- Timing Estimate:** Target: 10.00 ns, Estimated: 7.245 ns, Uncertainty: 2.70 ns.
- Performance & Resource Estimates:** A table showing modules and loops with their respective metrics.

| Modules & Loops | Issue Type | Violation Type | Distance | Slack | Latency(cycles) | Latency(ns) | Iteration Latency | Interval | Trip Count | Pipelined | B |
|--|--------------|----------------|----------|-------|-----------------|-------------|-------------------|----------|------------|-----------|----|
| matrixmul | - | - | - | 5145 | 5.14564 | - | 5146 | - | 5146 | - | no |
| matrixmul_Pipeline_VITIS_LOOP_12_1_VITIS_LOOP_13_2 | - | - | - | 1026 | 1.02664 | - | 1026 | - | 1026 | - | no |
| matrixmul_Pipeline_VITIS_LOOP_17_3_VITIS_LOOP_18_4 | - | - | - | 1026 | 1.02664 | - | 1026 | - | 1026 | - | no |
| matrixmul_Pipeline_row_col | II Violation | - | - | 2056 | 2.05664 | - | 2056 | - | 2056 | - | no |
| matrixmul_Pipeline_VITIS_LOOP_40_5_VITIS_LOOP_41_6 | - | - | - | 1027 | 1.02764 | - | 1027 | - | 1027 | - | no |

The console output at the bottom shows a warning: "The II Violation in module 'matrixmul' Pipeline row_col (loop 'row_col'): Unable to schedule 'load' operation ('tempA' load 20, mul.cpp:24)".

7、R/RTL Cosimulation

8、Export RTL 结果

The screenshot shows the Synthesis Summary Report for 'matrixmul' in Vitis HLS 2021.2, specifically the Export RTL section. The report includes the following sections:

- General Information:** Date: Mon Jul 17 18:00:42 2023, Version: 2021.2 (Build 3367213 on Tue Oct 19 02:48:09 MDT), Project: mux1, Solution: solution1 (Vivado IP Flow Target), Product: zynq, Target device: xc7z020-clg484-1.
- Timing Estimate:** Target: 10.00 ns, Estimated: 7.245 ns, Uncertainty: 2.70 ns.
- Performance & Resource Estimates:** A table showing modules and loops with their respective metrics.
- Export RTL:** The console output shows the command 'source run_ipack.tcl -notrace' and the generation of the output file 'mux1/solution1/impl/export.zip'.

| Modules & Loops | Issue Type | Violation Type | Distance | Slack | Latency(cycles) | Latency(ns) | Iteration Latency | Interval | Trip Count | Pipelined | B |
|--|--------------|----------------|----------|-------|-----------------|-------------|-------------------|----------|------------|-----------|----|
| matrixmul | - | - | - | 5145 | 5.14564 | - | 5146 | - | 5146 | - | no |
| matrixmul_Pipeline_VITIS_LOOP_12_1_VITIS_LOOP_13_2 | - | - | - | 1026 | 1.02664 | - | 1026 | - | 1026 | - | no |
| matrixmul_Pipeline_VITIS_LOOP_17_3_VITIS_LOOP_18_4 | - | - | - | 1026 | 1.02664 | - | 1026 | - | 1026 | - | no |
| matrixmul_Pipeline_row_col | II Violation | - | - | 2056 | 2.05664 | - | 2056 | - | 2056 | - | no |
| matrixmul_Pipeline_VITIS_LOOP_40_5_VITIS_LOOP_41_6 | - | - | - | 1027 | 1.02764 | - | 1027 | - | 1027 | - | no |

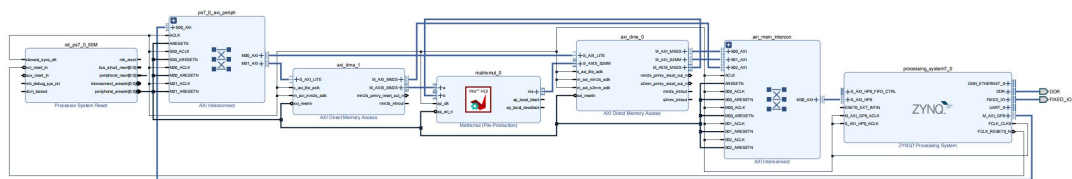
The console output at the bottom shows the command 'source run_ipack.tcl -notrace' and the generation of the output file 'mux1/solution1/impl/export.zip'.

9、在 Vivado 中导入矩阵乘法 IP 核

10、进行 block design，添加 ZYNQ、HLS、DMA 的 IP。因在实验中设有两个两个 AXI 流数据的入口，一个 AXI 流数据的输出，所以要对 DMA 的读写端口数量进行修改，配置两个 DMA 的读端口，1 个 DMA 的写端口。

配置结果：DMA0：读端口、写端口；DMA1：读端口

11、最终完成的 block design 如下:



12、生成顶层调用文件、bit 文件。

13、在 jupyter notebook 中新建 ipynb 文件，Python 调用相关 IP 文件。设置 a、b、res 三个矩阵。其中 a 矩阵元素均为 6，b 矩阵元素均为 8。得到 res 矩阵结果如下：

注：完整代码参见 jupyter notebook。

```
In [2]: dma0.sendchannel.transfer(a)
dma1.sendchannel.transfer(b)
print("[a]=", a)
print("[b]=", b)
```

```
[a] = [[6 6 6 ... 6 6 6]
[6 6 6 ... 6 6 6]
[6 6 6 ... 6 6 6]
...
[6 6 6 ... 6 6 6]
[6 6 6 ... 6 6 6]
[6 6 6 ... 6 6 6]]
[b] = [[8 8 8 ... 8 8 8]
[8 8 8 ... 8 8 8]
[8 8 8 ... 8 8 8]
...
[8 8 8 ... 8 8 8]
[8 8 8 ... 8 8 8]
[8 8 8 ... 8 8 8]]
```

```
In [3]: dma0.recvchannel.transfer(res)

print("[res]=", res)
```

```
[res]= [[1536 1536 1536 ... 1536 1536 1536]
[1536 1536 1536 ... 1536 1536 1536]
[1536 1536 1536 ... 1536 1536 1536]
...
[1536 1536 1536 ... 1536 1536 1536]
[1536 1536 1536 ... 1536 1536 1536]
[1536 1536 1536 ... 1536 1536 1536]]
```

【程序报错】

1、在分配内存的过程中，关于 `allocate()` 的使用有误。

错例：

```
a = allocate(shape=(32,), dtype='u4')
b = allocate(shape=(32,), dtype='u4')
res = allocate(shape=(32,), dtype='u4')
```

错例输出:

```
print(res)
```

[0 0]

查询 `allocate()` 相关文档如下:

Create a contiguous array of 5 32-bit unsigned integers

```
from pynq import allocate
input_buffer = allocate(shape=(5,), dtype='u4')
```

`device_address` property of the buffer

```
input_buffer.device_address
```

Writing data to the buffer:

```
input_buffer[:] = range(5)
```

Flushing the buffer to ensure the updated data is visible to the programmable logic:

```
input_buffer.flush()
```

随后修正如下：

```
a=allocate (shape=(32,32) ,dtype='u4')
```

```
b=allocate (shape=(32,32) ,dtype='u4')
```

```
res=allocate (shape=(32,32) ,dtype='u4')
```

2、在调试数据数量过程中，**res 矩阵总是仅输出 0**。经单独调试如下代码后：

```
dma0.sendchannel.transfer(a)
```

```
dma1.sendchannel.transfer(b)
```

```
dma0.recvchannel.transfer(res)
```

确定是数据的传输出现问题。首先考虑到是否未能成功调用 IP 核。利用 `mmol.ip_dict.keys()` 语句查找后，的确未找到矩阵乘法 IP 核。经继续查阅相关资料，得知调用过程中对矩阵大小的定义必须与 HLS 中 `mul.h` 的矩阵行列定义一致才能成功调用。调用失败的情况均是因为输入数据量过小（矩阵为 `4x4`）。经修改矩阵定义，并以循环嵌套结构进行正确赋值后，该问题解决。

四、实验收获与总结

- 1、对 Vitis HLS、Vivado 和 jupyter notebook 的使用有了更深入的了解。
- 2、在实验过程中完成了管脚约束，手动连线等操作，更好地掌握了新内容。
- 3、本实验是第一次接触 Python 编程，在遇到不懂的语法时，掌握了正确的查阅学习方法，高效地解决了遇到的问题。并锻炼了自己排查 bug 的能力。
- 4、与同学合作完成实验，在实验过程中相互学习和帮助，做到学以致用。