Introduction to High Performance Computing
PA1
Report

计 86　任一　2018011423

2020 年 3 月 22 日

实验环境	
操作系统:	Windows10 家庭版 18362.72 Windows Subsystem for Linux
mpicc 版本:	gcc version 7.5.0

1 Ex3.1.13

1.1 解题思路

本题主要任务是解决 MPI_Scatter 和 MPI_Gather 在数组长度无法被进程数整除时不能使用的问题，因此在本题中，我使用了 MPI_Scatterv 和 MPI_Gatherv 进行数组的分发和收集。

在 MPI_Scatterv 和 MPI_Gatherv 这两个函数中，与 MPI_Scatter 和 MPI_Gather 较为不同的两个参数是 displacement 和 datacount，这两个参数都是整型数组类型的。displacement 指各进程接收到的数据在数组中的开始位置，datacount 则是每个进程所接收的数据量，通过这两个参数的指定，MPI_Scatterv 和 MPI_Gatherv 即可实现每个进程分配指定数量的数据，因而更为灵活一些。

由于第一次书面作业的 Exercise1.1 中涉及到了此题的理论基础，即在数组长度不能被进程数整除时，各进程的元素分配计算，因此只需参考第一次作业 Exercise1.1 题的结果，即可完成此题中 displacement 和 datacount 参数的计算。不过还有一种特殊情况是，数组长度小于进程数。我对于这种情况的处理是，优先进程编号小的进程获得数据，即进程编号小于数组长度的进程分配一个元素，其他进程不分配元素也不参与运算。但是涉及到其他无元素进程的内存分配问题还值得商榷，例如出现 malloc(0) 这样的情况，这种情况的可靠性还有待商榷，不过在本实验的测试中，这种方法并没有问题。

1.2 测试

本并行程序的测试通过与串行程序的对拍进行。在本机运行 make check 命令即可进行对拍 (用到了 mpiexec 命令). 经过本地测试，对拍 1000 组数据没有产生错误。下面是部分对拍测试截图。

图 1: 并行程序与对拍程序部分结果图

图 2: 并行程序与对拍程序部分结果图

2 Exercise 3.11

2.1 解题思路

本题使用 MPI_Scan 函数，对 p 个进程，每个进程有 10 个元素的数组求了前缀和。我的思路是，首先对每个进程求局部和，然后利用 MPI_Scan 函数，使得每个进程 i 都有进程 0 至 i 这 i+1 个进程的元素总和，接着在每个进程里，通过用 MPI_Scan 得到的总和，从后向前累计减去当前进程中的元素，即可所求的全局前缀和。最后将这些全局前缀和通过 MPI_Gather 函数，汇总到 0 号进程统一输出。

2.2 测试

为了测试，我编写了串行求前缀和的程序，通过运行 make check 命令 (里面用到了 mpiexec 命令)，对并行程序和串行程序进行对拍。通过本机测试

得到，对拍 1000 组长度为 40 的前缀求和，没有发生错误。

2.3 运行方式

为了能够满足题目中" 每个进程随机生成的 10 个数" 以及与串行程序对拍的需求，我参考了 ch3 中 mpi_odd_even.c 中实现的命令行参数控制数据输入方式。

具体来说，运行 mpiexec -n <p> parallel -<g|i>, 其中 p 代表进程数，-g 代表使用自动随机生成的数据，-i 代表使用用户从命令行输入的 10p 个数据。通过-i 的命令设置，我能够从文件中重定向输入，从而完成与串行程序的对拍。

2.3.1 DigitalLife_Online 仿真结果

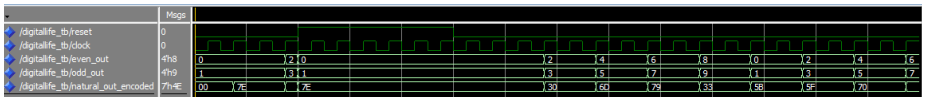

图 3: testbench 对 DigitalLife_Online 文件夹下的文件进行的仿真结果波形图 (放大后清晰可见)

在上图中，可以看到随着 clock 信号每 2 次上升沿的出现 (每 2 个时钟周期进行一次数列值的更新)，各个数列 (奇数列、偶数列) 都发生一次变化。在 reset 信号从 0 变为 1 时，各信号回到了初始状态，reset 信号从 1 变为 0 时，各信号继续随时钟变化而变化。其中最后一个信号为编码后的自然数数列，对应没有译码的数码管使用，在本次仿真中其数值没有具体含义。[1]

2.4 JieLabs 测试结果

见录屏文件 JieLabsTest.mp4。在视频中展示了十六进制自然数数列、偶数列和奇数列。可看到大约每秒各数列都会发生变化。按下 reset 后可以看到各数列回到初始状态，不按 reset 时各信号随时钟变化而变化。

[1] 为什么图中第一次遇到 2 个上升沿时，波形没有变化？详见 3.1 中的说明。

3 思考与总结

3.1 遇到的问题与解决方法

在我第一次仿真时，我遇到了奇怪的问题，即我在代码中设置为在 clock 信号上升沿时，更新数列和输出的值，但是得到的波形图却是在每次 clk 信号下降沿时发生变化，这令我十分不解。在参考了 StackOverflow 的解答并且与吕志远助教交流后，我学习到一个 process 中 signal 赋值的过程，只会在 process 结束时一起执行，被赋予的值是所有 signal 在 process 进行前的值，即所有信号在本轮 process 中，值仍为上一轮的值。

因此，在刚才发现的下降沿信号发生变化的问题中，出现上升沿时，该程序会进行数列临时变量的更新和输出信号的更新，但在上升沿结束时输出信号仍然保持为上一轮的结果。在下降沿时，不会有数列临时变量的更新，有输出信号的更新，这时输出信号更新为了上升沿本意要更新的数值，因此出现了在下降沿时信号变化而上升沿中信号不便的问题。出现该问题的代码和波形图如下，注意第 27 行的代码和上面的注释。

```
1  entity DigitalLife is
2  port (
3      reset:in std_logic := '0';
4      clock:in std_logic := '0';
5      natural_out_hex:
6          out std_logic_vector(3 downto 0) := "0000"
7  );
8  end entity DigitalLife;
9  architecture bhv of DigitalLife is
10     signal natural_seq_hex :
11         std_logic_vector(3 downto 0) := "0000";
12 begin
13     process (clock, reset) begin
14         if (reset = '1') then
15             natural_seq_hex <= "0000";
16             natural_out_hex <= "0000";
17         elsif (rising_edge(clock)) then
18             if (natural_seq_hex = 15) then
```

```
19                  natural_seq_hex <= (others => '0');
20              else
21                  natural_seq_hex <= natural_seq_hex +1;
22              end if;
23          end if;
24      -- When falling edge, this also be activated and
25      -- the value of the last rising edge
26      -- is updated at falling edge.
27      natural_out_hex <= natural_seq_hex;
28      end process;
29  end architecture bhv;
```

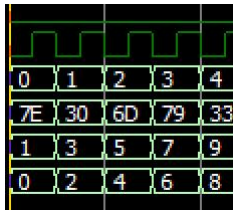

图 4: 下降沿时数据变化而上升沿时数据不变

这个问题我通过将对输出信号的赋值加到了对于上升沿的判断分支中解决了，这样做不会在下降沿时触发赋值，只会在下一次上升沿时对输出变量为上一次上升沿时的值。不过这样做也有一定的小问题，即在第一次遇到上升沿时，输出信号并不会变化，只会在第二次上升沿及其之后发生规律的变化。这也解释了 2.1.1 和 2.1.2 中脚注提出的两个小疑问。

3.2 一些感想与建议

非常感谢老师和助教在本次实验中给予我的帮助，无论是在群聊的答疑还是助教细心地与我私聊解决问题，都让我感受到了老师和助教的用心！

不过在 VHDL 入门、testbench 入门时，我感觉还是遇到了很大的困难。可能一方面我在面对生疏知识时接受没有那么迅速，另一方面可能也是关于 VHDL 的网络资源不是特别丰富。有可能的话，建议老师和助教为新入门的同学准备更多的参考资料或者学习路线，以帮助同学们更好地掌握这门课的知识，享受数字逻辑实验的乐趣。

4 参考资料

1. RISING EDGE AND FALLING EDGE PROBLEM
https://stackoverflow.com/questions/50461404
2. MODELSIM SIMULATION
https://blog.csdn.net/u013273161/article/details/82454134
3. TESTBENCH TUTORIAL
https://vhdlguide.readthedocs.io/en/latest/vhdl/testbench.html#
4. TESTBENCH CREATION
https://www.doulos.com/knowhow/perl/testbench_creation/
5. HOW SIGNAL ASSIGNMENT WORK IN PROCESS
https://stackoverflow.com/questions/5060635/how-does-signal-assignment-work-in-a-proce