答疑

2021年5月27日　20:02

已有修复

- 已修复在初级题中include ap_int库，平台会报错的bug
- 由于发现部分选手直接把正确答案赋值给输出的数组，目前题目的testbench已更新
  - 平台将在下一周对目前选手的所有初级题提交重新评分
  - 在重新评分完成之后，如果发现平台测试得到反馈的性能与本地测试的性能有很大出入，可发邮件至xup_china@xilinx.com申诉

若发现再有试图通过多次提交等手段来获取testbench信息的行为，组委会有权取消该题成绩

2　　XILINX

常见问题

- 竞赛规定的软件和硬件平台是什么？可以使用Vivado HLS软件吗？
  - 竞赛评分系统采用Vitis HLS软件，如果使用Vivado HLS则有可能不完全兼容，无法通过竞赛评分系统的仿真，影响竞赛成绩。
  - 竞赛的目标器件硬件平台是Alveo U50平台， 但比赛中不需要生成最终bit文件，仅考察算法实现功能和综合报告的性能，能完成co-sim即可，因此不需要参赛队伍配备硬件平台。
- 为什么提交成功后，积分没有变？
  - 目前排名10分钟更新一次，烦请等待至更新时间查看积分和排名变化。

注意事项

- 提交代码时无需提交tcl文件和test.cpp文件，如果提交了服务器的tcl文件也会将用户提交的tcl覆盖，所以请不要将设计写在test.cpp中
- Clock运行时所有选手均配置为10ns，但性能公式中有fmax一项，因此性能更好的代码还是能获得更高的性能分
- 由于平台目前仅兼容至多一层子文件夹，若有两层及以上的子文件夹，请将文件结构精简

注意事项

- 初级题Sobel filter中gx，gy以及总gradient均需保证是0~255之间的整数（如果小于0，gradient取0，如果大于255，gradient取255），总gradient将gx和gy直接相加即可

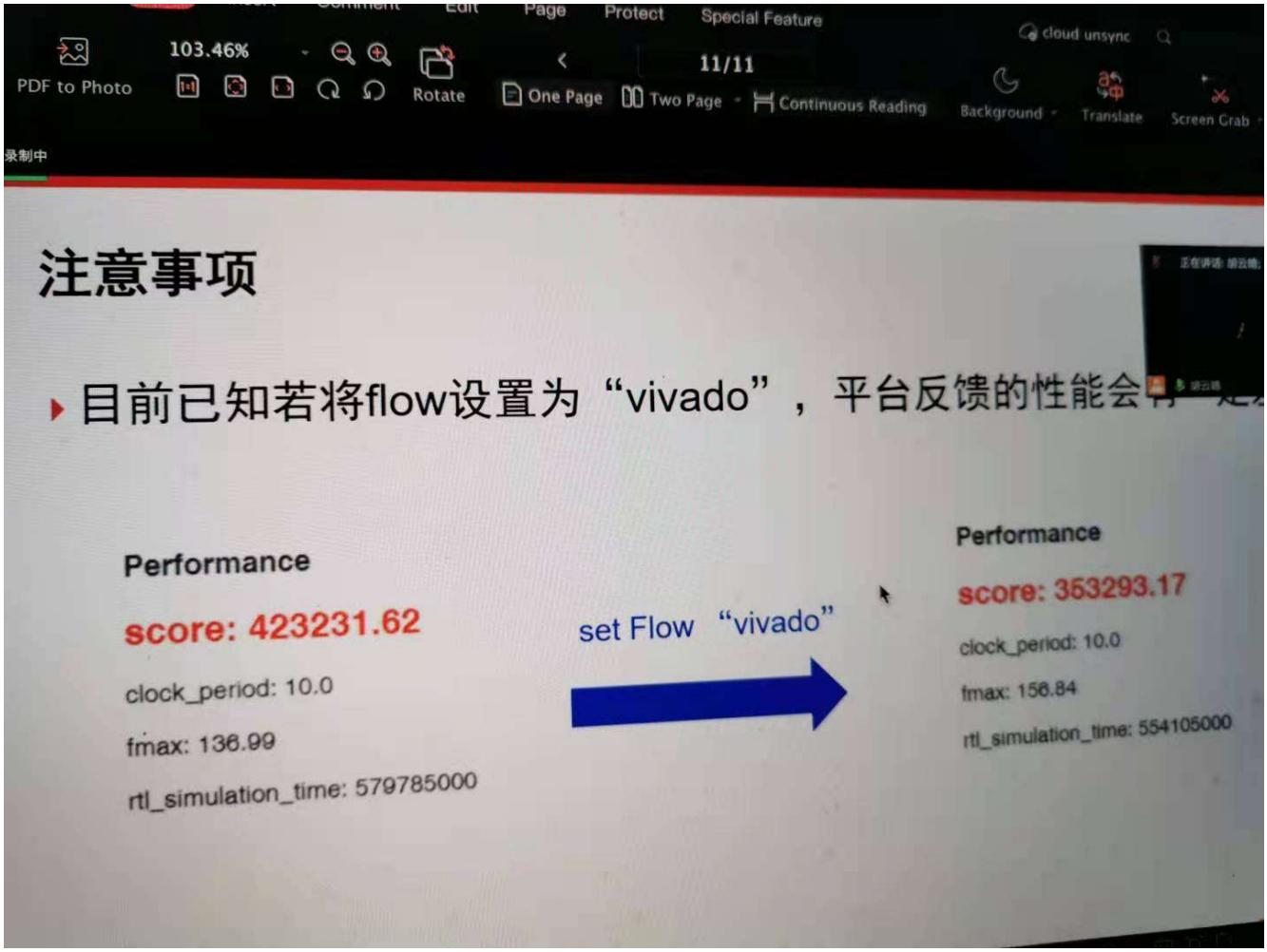
- 如果使用的是Vitis图形界面，在选择板卡时可能搜索不到竞赛指定的板卡xcu50-fsvh2104-2-e，因此建议直接运行官方提供的tcl脚本进行仿真测试

注意事项

- 评分系统对每一题都设置了运行时间上限，如果提交设计的运行时间过长，将返回错误：“context deadline exceeded”

- 如果当初注册了两个以上的账户，且均提交过，请联系我们注销，否则账户的提交将影响选手的排名

- 如果有修改队伍信息的需求，请发邮件至xup_china@xilinx.com，我们后台会进行修改。



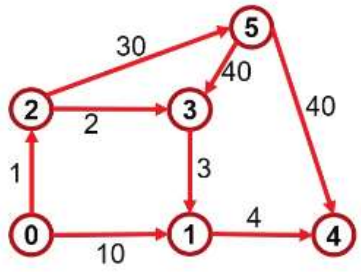
Set vivado flow和set vitis flow会有性能差别

高级题说明

- 高级题题目的具体细节已上传至github，可在 https://github.com/xupsh/ccc2021/tree/main/problems下载源码
- 框架代码是算法的C-model，您需要自行设计可综合co-sim的设计
- 如有任何问题，请发邮件至xup_china@xilinx.com反馈

Graph

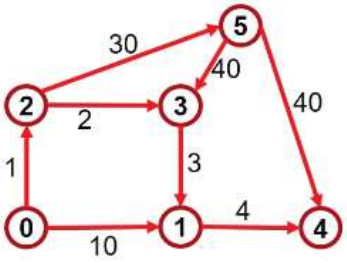
> Mathematical structures used to model pairwise relations between objects



From \ To	0	1	2	3	4	5
0	0	10	1	0	0	0
1	0	0	0	0	4	0
2	0	0	0	2	0	30
3	0	3	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	40	40	0

Sparse matrix

> Coordinate list (COO)



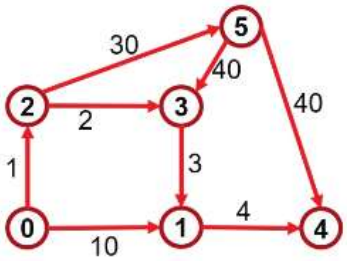
	0	1	2	3	4	5	To
0	0	10	1	0	0	0	
1	0	0	0	0	4	0	
2	0	0	0	2	0	30	
3	0	3	0	0	0	0	
4	0	0	0	0	0	0	
5	0	0	0	40	40	0	

From

Row (From)	Column (To)	Value (Weight)
0	1	10
0	2	1
2	3	2
3	1	3
5	3	40
5	4	40
2	5	30
1	4	4

Sparse matrix

> Compressed Sparse Row (CSR)



	0	1	2	3	4	5	To
0	0	10	1	0	0	0	
1	0	0	0	0	4	0	
2	0	0	0	2	0	30	
3	0	3	0	0	0	0	
4	0	0	0	0	0	0	
5	0	0	0	40	40	0	

From

	offset	Column (To)	Value (Weight)
0	0	1	10
1	2	2	1
2	3	4	4
3	5	3	2
4	6	5	30
5	6	1	3
6	8	3	40
7		4	40

> Traverse all the neighbor vertices of vertex 2

```
unsigned u = 2;
for(i=offset[u]; i<offset[u+1]; i++) {
        v=column[i];
}
```

Overall

```
void dut(unsigned numVert,
         unsigned numEdge,
         unsigned* offset,
         unsigned* column,
         float* weight,
         float* max_dist,
         unsigned* src,
         unsigned* des,

         unsigned* tmp0,
         unsigned* tmp1,
         unsigned* tmp2,
         unsigned* tmp3) {
```

- Tmp0, tmp1, tmp2, tmp3 for external memory
  - Leave unused if not required

```
const unsigned MEMSIZE=INTERFACE_MEMSIZE;
#pragma HLS INTERFACE m_axi offset = slave latency = 32 num_write_outstanding = 1 num_read_outstanding = \
    16 max_write_burst_length = 2 max_read_burst_length = 256 bundle = gmem0 port = offset depth = MEMSIZE

#pragma HLS INTERFACE m_axi offset = slave latency = 32 num_write_outstanding = 1 num_read_outstanding = \
    16 max_write_burst_length = 2 max_read_burst_length = 256 bundle = gmem1 port = column depth = MEMSIZE

#pragma HLS INTERFACE m_axi offset = slave latency = 32 num_write_outstanding = 1 num_read_outstanding = \
    16 max_write_burst_length = 2 max_read_burst_length = 256 bundle = gmem2 port = weight depth = MEMSIZE

#pragma HLS INTERFACE m_axi offset = slave latency = 32 num_write_outstanding = 1 num_read_outstanding = \
    16 max_write_burst_length = 2 max_read_burst_length = 256 bundle = gmem3 port = max_dist depth = MEMSIZE

#pragma HLS INTERFACE m_axi offset = slave latency = 32 num_write_outstanding = 1 num_read_outstanding = \
    16 max_write_burst_length = 2 max_read_burst_length = 256 bundle = gmem4 port = src depth = MEMSIZE

#pragma HLS INTERFACE m_axi offset = slave latency = 32 num_write_outstanding = 1 num_read_outstanding = \
    16 max_write_burst_length = 2 max_read_burst_length = 256 bundle = gmem5 port = des depth = MEMSIZE

#pragma HLS INTERFACE m_axi offset = slave latency = 32 num_write_outstanding = 1 num_read_outstanding = \
    16 max_write_burst_length = 2 max_read_burst_length = 256 bundle = gmem6 port = tmp0 depth = MEMSIZE

#pragma HLS INTERFACE m_axi offset = slave latency = 32 num_write_outstanding = 1 num_read_outstanding = \
    16 max_write_burst_length = 2 max_read_burst_length = 256 bundle = gmem7 port = tmp1 depth = MEMSIZE

#pragma HLS INTERFACE m_axi offset = slave latency = 32 num_write_outstanding = 1 num_read_outstanding = \
    16 max_write_burst_length = 2 max_read_burst_length = 256 bundle = gmem8 port = tmp2 depth = MEMSIZE

#pragma HLS INTERFACE m_axi offset = slave latency = 32 num_write_outstanding = 1 num_read_outstanding = \
    16 max_write_burst_length = 2 max_read_burst_length = 256 bundle = gmem9 port = tmp3 depth = MEMSIZE
```

- INTERFACE_MEMSIZE
  - Define in top.hpp
  - Change the memory size for external memory
  - Fill it in the depth in the interface pragma

留了四个临时用的指针，可以对外面存临时数据用

所有顶层的指针要写interface pragma，然后depth要设置为interface_memsize

Minimum Spanning Tree

```
void dut(unsigned int numVert,
         unsigned int numEdge,
         unsigned int* offset,
         unsigned int* column,
         float* weight,
         unsigned* mst,
         unsigned* tmp0,
         unsigned* tmp1,
         unsigned* tmp2,
         unsigned* tmp3);
```

- numVert: input, number of vertices
- numEdge: input, number of edges
- Offset, column, weight: input, CSR graph
- Mst: output
  - a vector with size of number of vertices
  - The parent vertex of each vertex in the generated tree
    - If a vertex is the root of a tree, the parent vertex is itself
  - Find the parent of vertex 2
    - Parent = mst[2];
  - Find the grandparent of vertex 2
    - Grandparent = mst[mst[2]];
- Correctness
  - If all vertices are included in the tree
- The total weight in the tree should be small and performance is high
  - (Total weight)^2 * (simulate time), the smaller the better

Coloring

```
void dut(unsigned int numVert,
         unsigned int numEdge,
         unsigned int* offset,
         unsigned int* column,
         unsigned* color,
         unsigned* tmp0,
         unsigned* tmp1,
         unsigned* tmp2,
         unsigned* tmp3);
```

- numVert: input, number of vertices
- numEdge: input, number of edges
- Offset, column, weight: input, CSR graph
- Color: output
  - a vector with size of number of vertices
  - The color ID of each vertex
    - Color ID of vertex 2: colorID = color[2];
- Correctness
  - If all neighbor vertices have different colors
- Use smaller number of colors and performance is high
  - (number of colors)^2 * (simulate time), the smaller the better

Diameter

```
void dut(unsigned numVert,
         unsigned numEdge,
         unsigned* offset,
         unsigned* column,
         float* weight,
         float* max_dist,
         unsigned* src,
         unsigned* des,

         unsigned* tmp0,
         unsigned* tmp1,
         unsigned* tmp2,
         unsigned* tmp3);
```

- numVert: input, number of vertices
- numEdge: input, number of edges
- Offset, column, weight: input, CSR graph
- Max_dist: output
  - A vector with a single element, showing the max diameter
- Src, des: output,
  - A vector with a single element, showing the shortest distance between the source vertex and destination vertex of the calculated diameter
- Correctness
  - If the max diameter equals the shortest distance of the reported source vertex and destination vertex
- If the max diameter is high and performance is high
  - (1/(max_diameter))^2 * (simulate time), the smaller the better

PYNQ HACK 暑期学校

- 所有初赛顺利完成的同学都将优先获得线下暑期学校参加的机会
- xupsh.github.io/camp
  - 7月15日 南京线下活动
  - 需提前报名注册

XILINX

XILINX

Thank You

0203... 那个diameter的估计，他

0203... 不应该是个误差吗