

LabC-Betweenness Report

T8 GIEE r11943043 潘奕亘

T8 GIEE r11943012 曾維雋

T8 GIEE b07901073 吳秉軒

A. Introduction

In the lab, the topic is to implement the algorithm “Betweenness Centrality” on HDL. We are going to modify the C++ code provided by CCC contest to a synthesizable and efficient code. The HW platform we use is Xilinx U50 Accelerate Card.

There are some requirements we are supposed to meet. First is the HW programming running time cannot over 1 hour. Second is that there are two test patterns we should pass, the one is large dataset and the other is small dataset. The score standard is shown below:

Advanced Topic - Betweenness



- Device: U50 (xrt) → (hw FPGA execution time)
 - Optimization: host & kernel

mark=0.6*[(cntbig^4)/time_big]+0.4*[(cntsmall^4)/time_small]

4. 评分规则

将所有的点按照选手计算的betweenness的值从大到小排序

将所有的点按照参考代码计算的betweenness的值从小到大排序

比较上述两个序列从第几个值开始不一样，这个值记为cnt

同时记录程序运行时间 time

小测试用例占40%分数，大测试用例占60%分数，最终得分按小测试用例得分与大测试用例得分加权相加而得

最终得分 mark=[0.6*(cntbig^4)+0.4*(cntsmall^4)]*time，mark越大对应排名得分越高，若程序运行时间超过1个小时则不计分

```
INFO: Finish kernel execution
INFO: Finish E2E execution
-----
INFO: Data transfer from host to device: 353 us
INFO: Kernel1 Data transfer from device to host: 312 us
-----
INFO: Kernel1 execution: 338.482 us
INFO: kernel total execution: 338 us
-----
INFO: FPGA execution time:3919282 us
-----
Warning: using default memory size (100000*sizeof(unsigned))
graph loading for betweenness golden...
f936ed83 Top 8 vertices are the same with golden.
```

B. Observation and acquirement

1. Algorithm analysis

By reading the code provided by CCC, there are some C data structure in it. For example, owing to that we are going to implement BFS(Breadth First Search), queue and stack are selected to store the nodes in graph. These C data structure cannot be synthesized in Xilinx HLS, so we should change the data storage method but keep the original algorithm and computing result.

2. Data structure implementation

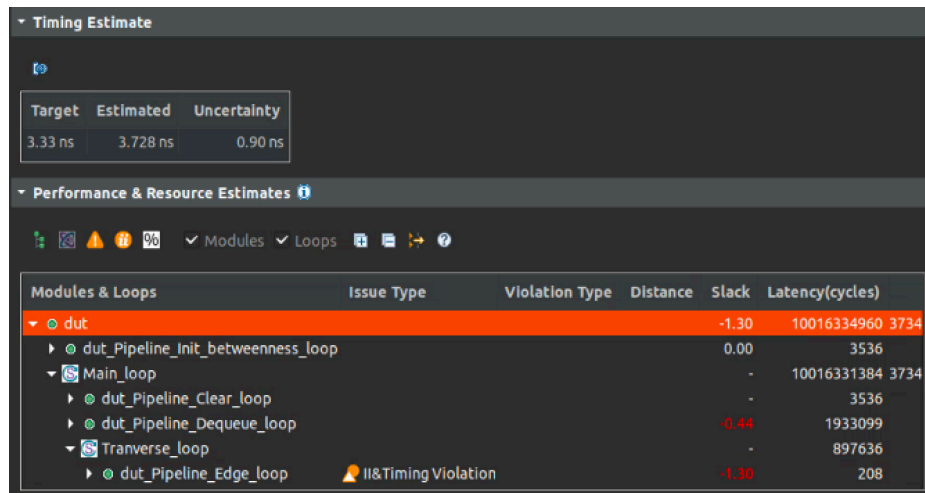
To fulfill queue and stack in HLS, we use a long enough array to store the value which may be stored into queue. As the algorithm, the queue depth is at most same with the number of vertices. Hence, the array size is same as the number of large pattern vertices.

Besides, to traverse the all element in this array, we use a long enough for-loop to go through the array. In the non-using position of array, we place -1 on it. By this, we can easily know the traverse work is whether done.

3. Synthesis

In the first version of design, there are roughly 9 array. For each array, we are likely to do multiple access in one cycle. This results in a higher number of latency. The II value is also bigger.

Furthermore, owing to that there are some computation operated in one cycle, in synthesis report, the slack time is also negative. The HLS result is shown below.



Target	Estimated	Uncertainty
3.33 ns	3.728 ns	0.90 ns

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)
dut				-1.30	10016334960 3734
dut_Pipeline_Init_betweenness_loop				0.00	3536
Main_loop				-	10016331384 3734
dut_Pipeline_Clear_loop				-	3536
dut_Pipeline_Dequeue_loop				-0.44	1933099
Traverse_loop				-	897636
dut_Pipeline_Edge_loop	II&Timing Violation			-1.30	208

4. Negative slack time

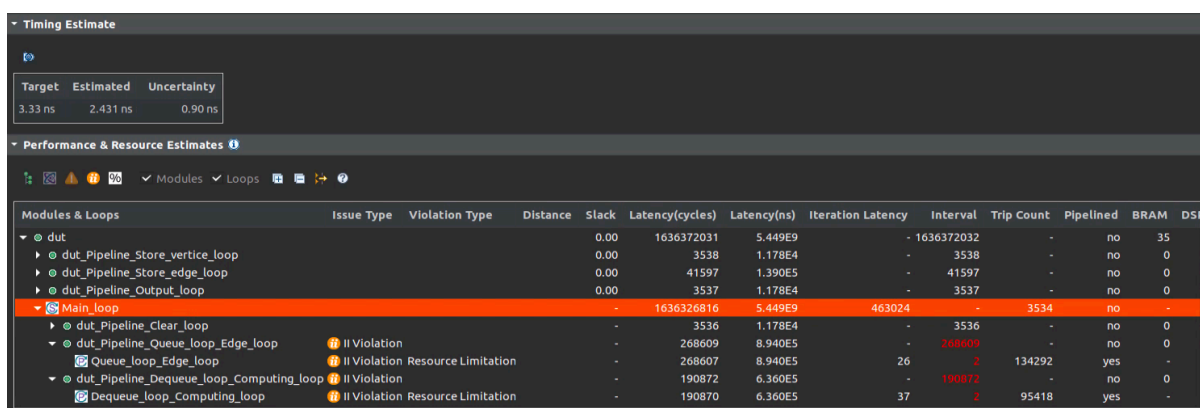
To address the negative slack time problem, we use more static scalar to store some middle computing value. In the dequeue loop, there are multiplication and division operations. We cannot put this two operation in one cycle because these operations would result long critical path.

Also, multiple access can result in long latency. We should try to split the read and write operation into different timing, so the slack time can be non-negative. To solve it, we divide the read and write operation to different array, which means we would use more arrays to store some related information in order to reduce the loading on one array.

5. II violation

However, the main cause of latency is data dependency. With the data dependency, we cannot achieve low II value. Hence, we observe the array data dependency in the primary II violation loop and use `#pragma HLS dependenc type=inter false` to tell the compiler that it don't care the dependency of these array.

Besides, to reduce the for-loop latency, we should do pipelining. By doing this above, the latency is going to non-negative and II value is down to 2.



Target	Estimated	Uncertainty
3.33 ns	2.431 ns	0.90 ns

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSF
dut				0.00	1636372031	5.449E9	-	1636372032	-	no	35	
dut_Pipeline_Store_vertice_loop				0.00	3538	1.178E4	-	3538	-	no	0	
dut_Pipeline_Store_edge_loop				0.00	41597	1.390E5	-	41597	-	no	0	
dut_Pipeline_Output_loop				0.00	3537	1.178E4	-	3537	-	no	0	
Main_loop				-	1636326816	5.449E9	463024	-	3534	no	-	
dut_Pipeline_Clear_loop				-	3536	1.178E4	-	3536	-	no	0	
dut_Pipeline_Queue_loop_Edge_loop	II Violation			-	268609	8.940E5	-	268609	-	no	0	
Queue_loop_Edge_loop	II Violation	Resource Limitation		-	268607	8.940E5	26	2	134292	yes	-	
dut_Pipeline_Dequeue_loop_Computing_loop	II Violation			-	190872	6.360E5	-	190872	-	no	0	
Dequeue_loop_Computing_loop	II Violation	Resource Limitation		-	190870	6.360E5	37	2	95418	yes	-	

6. Wrong computing result

After solve timing violation and reduce II value, we put the code on U50 Accelerator Card to test. The code can run completely in 45 minutes.

However, in this version, the computing result does not achieve the highest value. The problem is probably the data dependency.

7. Data dependency

We try to delete some dependency pragma, and see the computing result whether becoming correct. However, the result is not changed.

Besides, considering that our inner loop may have some storing latency longer than II value, we try to increase the inner loop boundary. Finally, the computing result are all correct.

8. Exit condition

There are som exit condition in our code (i.e break, continue) because we use for-loop to substitute the original queue and stack behavior. If we want to use pragma dataflow, then we should remove these exit condition. However, there are some loopback dependency variable in the design, so we decided not to use dataflow pragma.

By remove the exit condition in the code, we find that for large pattern it would consume less latency time, but for small case the latency time is longer. The two cases results are shown below (upper is with break, lower is without break)

```
----- Betweenness -----
Found Platform
Platform Name: Xilinx
Info: Context created
Info: Command queue created
Found Device=xilinx_u50_gen3x16_xdma_base_5
INFO: Importing ./build_dir.hw.u50/dut.xclbin
Loading: './build_dir.hw.u50/dut.xclbin'
Info: Program created
Info: Program created
kernel has been created
graph loading ...
kernel start-----
    vertex number=1015 edge number=11003
Finish set kernel
Finish set kernel1
Finish set kernel2
Finish set kernel3
Finish set kernel4
INFO: Finish kernel execution
INFO: Finish E2E execution
-----
INFO: Data transfer from host to device: 348 us
-----
INFO: Kernel1 Data transfer from device to host: 224 us
-----
INFO: Kernel1 execution: 1.00829e+06 us
-----
INFO: kernel total execution: 1008290 us
-----
INFO: FPGA execution time:1188219 us
-----
Warning: using default memory size (100000*sizeof(unsigned)) for tmp0, tm
p1, tmp2, tmp3. Define INTERFACE_MEMSIZE in the top.hpp & top.cpp for cus
tomize memory size.
graph loading for betweenness golden...
f936ed83 Top 1015 vertices are the same with golden.
```

```
----- Betweenness -----
Found Platform
Platform Name: Xilinx
Info: Context created
Info: Command queue created
Found Device=xilinx_u50_gen3x16_xdma_base_5
INFO: Importing build_dir.hw.u50/dut.xclbin
Loading: 'build_dir.hw.u50/dut.xclbin'
Info: Program created
Info: Program created
kernel has been created
graph loading ...
kernel start-----
    vertex number=3534 edge number=41594
Finish set kernel
Finish set kernel1
Finish set kernel2
Finish set kernel3
Finish set kernel4
INFO: Finish kernel execution
INFO: Finish E2E execution
-----
INFO: Data transfer from host to device: 307 us
-----
INFO: Kernel1 Data transfer from device to host: 344 us
-----
INFO: Kernel1 execution: 5.20494e+06 us
-----
INFO: kernel total execution: 5204939 us
-----
INFO: FPGA execution time:9614188 us
-----
Warning: using default memory size (100000*sizeof(unsigned)) for tmp0, tm
p1, tmp2, tmp3. Define INTERFACE_MEMSIZE in the top.hpp & top.cpp for cus
tomize memory size.
graph loading for betweenness golden...
f936ed83 Top 4 vertices are the same with golden.
```

```
----- Betweenness -----
Found Platform
Platform Name: Xilinx
Info: Context created
Info: Command queue created
Found Device=xilinx_u50_gen3x16_xdma_base_5
INFO: Importing ./build_dir.hw.u50/dut.xclbin
Loading: './build_dir.hw.u50/dut.xclbin'
Info: Program created
Info: Program created
kernel has been created
graph loading ...
kernel start-----
    vertex number=3534 edge number=41594
Finish set kernel
Finish set kernel1
Finish set kernel2
Finish set kernel3
Finish set kernel4
INFO: Finish kernel execution
INFO: Finish E2E execution
-----
INFO: Data transfer from host to device: 293 us
-----
INFO: Kernel1 Data transfer from device to host: 294 us
-----
INFO: Kernel1 execution: 6.53684e+06 us
-----
INFO: kernel total execution: 6536842 us
-----
INFO: FPGA execution time:6659443 us
-----
Warning: using default memory size (100000*sizeof(unsigned)) for tmp0, tm
p1, tmp2, tmp3. Define INTERFACE_MEMSIZE in the top.hpp & top.cpp for cus
tomize memory size.
graph loading for betweenness golden...
f936ed83 Top 3534 vertices are the same with golden.
```

```

----- Betweenness -----
Found Platform
Platform Name: Xilinx
Info: Context created
Info: Command queue created
Found Device=xilinx_u50_gen3x16_xdma_base_5
INFO: Importing ./build_dir.hw.u50/dut.xclbin
Loading: './build_dir.hw.u50/dut.xclbin'
Info: Program created
Info: Program created
kernel has been created
graph loading ...
kernel start-----
vertex number=1015 edge number=11003
Finish set kernel
Finish set kernel1
Finish set kernel2
Finish set kernel3
Finish set kernel4
INFO: Finish kernel execution
INFO: Finish E2E execution

-----
INFO: Data transfer from host to device: 327 us
-----
INFO: Kernel1 Data transfer from device to host: 413 us
-----
INFO: Kernel1 execution: 1.5668e+06 us
-----
INFO: kernel total execution: 1566802 us
-----
INFO: FPGA execution time:1674985 us
-----
Warning: using default memory size (100000*sizeof(unsigned)) for tmp0, tm
p1, tmp2, tmp3. Define INTERFACE_MEMSIZE in the top.hpp & top.cpp for cus
tomize memory size.
graph loading for betweenness golden...
f936ed83 Top 1015 vertices are the same with golden.

```

```

----- Betweenness -----
Found Platform
Platform Name: Xilinx
Info: Context created
Info: Command queue created
Found Device=xilinx_u50_gen3x16_xdma_base_5
INFO: Importing ./build_dir.hw.u50/dut.xclbin
Loading: './build_dir.hw.u50/dut.xclbin'
Info: Program created
Info: Program created
kernel has been created
graph loading ...
kernel start-----
vertex number=3534 edge number=41594
Finish set kernel
Finish set kernel1
Finish set kernel2
Finish set kernel3
Finish set kernel4
INFO: Finish kernel execution
INFO: Finish E2E execution

-----
INFO: Data transfer from host to device: 492 us
-----
INFO: Kernel1 Data transfer from device to host: 364 us
-----
INFO: Kernel1 execution: 5.45477e+06 us
-----
INFO: kernel total execution: 5454769 us
-----
INFO: FPGA execution time:5619216 us
-----
Warning: using default memory size (100000*sizeof(unsigned)) for tmp0, tm
p1, tmp2, tmp3. Define INTERFACE_MEMSIZE in the top.hpp & top.cpp for cus
tomize memory size.
graph loading for betweenness golden...
f936ed83 Top 3534 vertices are the same with golden.

```

C. Performance

To analyze and test our code is whether have problem, we first use Vitis HLS for programming. Below is our final version's synthesis summary report, and more detail are put in the GitHub.

Timing Estimate

Target	Estimated	Uncertainty
3.33 ns	2.431 ns	0.90 ns

Performance & Resource Estimates

Modules

Loops

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSF
dut				0.00	1636372031	5.449E9	-	1636372032	-	no	35	
dut_Pipeline_Store_vertice_loop				0.00	3538	1.178E4	-	3538	-	no	0	
dut_Pipeline_Store_edge_loop				0.00	41597	1.390E5	-	41597	-	no	0	
dut_Pipeline_Output_loop				0.00	3537	1.178E4	-	3537	-	no	0	
Main_loop				-	1636326816	5.449E9	463024	-	3534	no	-	
dut_Pipeline_Clear_loop				-	3536	1.178E4	-	3536	-	no	0	
dut_Pipeline_Queue_loop_Edge_loop	II Violation			-	268609	8.940E5	-	268609	-	no	0	
Queue_loop_Edge_loop	II Violation	Resource Limitation		-	268607	8.940E5	26	2	134292	yes	-	
dut_Pipeline_Dequeue_loop_Computing_loop	II Violation			-	190872	6.360E5	-	190872	-	no	0	
Dequeue_loop_Computing_loop	II Violation	Resource Limitation		-	190870	6.360E5	37	2	95418	yes	-	

Based on the B-8. Section, we are going to choose the best performance design for this contest. After the calculation, without break ones has better score. The reason is that in this score standard, the large pattern has bigger ratio, which represent it has more influence.

There is something to notice. After run HW, **please run our code with testbench for 2~3 times to get more accurate score**, thank you.

The final score is about 16908349.76

D. Code

https://github.com/PAN-YI-HSUAN/2022_HLS/tree/main/LabC-Betweenness