Problem C: GPU Accelerated Logic Rewriting

Ghasem Pasandi, Sreedhar Pratty, David Brown
NVIDIA Corp., Santa Clara, CA

Q&A

Q1. In the evaluation part, the script are scored based on the **run-time** and **GPU utility**. Any requirement about **QoR** of the output circuit of our implementation? Are we allowed to change the methodology of rewrite (e.g. change the traversal order, cut enumeration algorithms, ...), which will probably result in a different QoR?
A1. AIG node count is added to scoring function as QoR, and in the new revision of the problem, changes like traversal order, cut enumeration algorithms are allowed.

Q2. It it possible to implement the "standard" rewriting algorithm as the one in ABC and innovate with respect to the algorithmic design? Is the rewriting quality (in terms of circuit size reduction) evaluated? Currently, it is not involved in the scoring function, which means one could implement something fast but useless and win the competition.
A2. Please see the revised problem contest. Algorithmic changes are allowed and now there is a QoR metric in the scoring function.

Q3. Is there any space for optimization by using GPU acceleration? how many nodes does the Biggest case has in the benchmarks? I have tested the EFPL benchmarks on the competition website. And I choose the AIG case with the largest area, which has over 200 thousand "AND nodes". But it only costs about three seconds to run "drw" in abc. I'd like to know if there is any space for optimization by using GPU acceleration? And, by the way, may I know how many nodes does the biggest case has in the benchmarks? I've tried to rewrite some functions in a "CUDA" way, but it seems that the effect is not good, and the running time is much longer than before. I think it's because the cuda kernel function has been executed too many times as the case hasmany "AND nodes".
A3. There are larger benchmarks that take much more time than 3 seconds. Take a look at MtM benchmarks:https://www.epfl.ch/labs/lsi/page-102566-en-html/benchmarks/
For sixteen benchmark circuit above, it took ~425 seconds for drw to finish on my system. A good implementation of drw on GPU should be able to decrease this large run-time.

Q4. Please help to advise if we need to support the option paremeter "-C -N -l -f -z -r -v -w -h" in origin "drw" command.
A4. In the final submission, they should support these options but for alpha submission, it is not needed to support them all and a default setting for options should be enough.

Q5. Please help advise if we need to support AIG files with latches.
A5. No, we will test their codes using only combinational circuits.

Q6. Q1. Could we ignore ABC and develop something new that can do the logic rewriting (which could be more GPU-friendly)?

A6. Yes it is okay to bypass ABC and develop a new rewriting function that is GPU-friendly, but the code should support the commands that are required and mentioned in the contest problem such as reading/writing AIGs, etc.

Q7. Could you please help to advise any alternative links of MtM benchmarks? because we cannot download the MtM benchmarks at https://www.epfl.ch/labs/lsi/page-102566-en-html/benchmarks/. We tried to use different networks but they didn't work.

A7. Try this link: https://zenodo.org/record/2572934#.XGxRiS3MzuM

Q8. As mentioned before, the parameters "-C -N -l -f -z -r -v -w -h" should be supported in the final submission. But if we use a different logic rewriting algorithm on GPUs, some parameters (which are specific to the rewriting algorithm in ABC) become meaningless. Do we still have to support them?

A8. In case of implementing a different rewriting algorithm, they won't need to support all switches of ABC's drw that are specific to this function. However, they should add similar switches with providing similar flexibility/functionality.

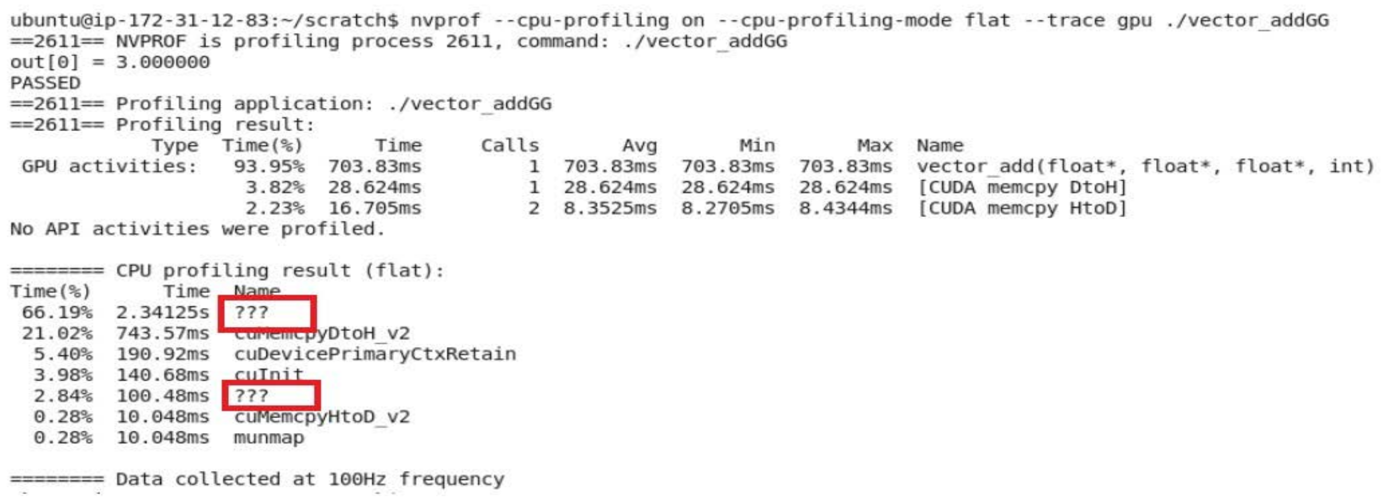
Q9. Can we ignore the literal names in the output AIG? (i.e. using cec -n instead of cec for checking).

A9. It is preferred that you do not modify names of CIs/COs in your code, so, no please keep using "cec" command for equivalence checking not "ece -n".

Q10. The evaluation of Problem C includes four parameters K1, K2, K3 and D. May I ask what K1, K2, K3 and D are in the alpha test? Thank you!

A10. K1=K2=K3=4, and d=2

Q11. When I use the nvprof command to profile the program, there are some names of the function as "???" Could you help advise what it means?



```
ubuntu@ip-172-31-12-83:~/scratch$ nvprof --cpu-profiling on --cpu-profiling-mode flat --trace gpu ./vector_addGG
==2611== NVPROF is profiling process 2611, command: ./vector_addGG
out[0] = 3.000000
PASSED
==2611== Profiling application: ./vector_addGG
==2611== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   93.95%  703.83ms         1  703.83ms  703.83ms  703.83ms  vector_add(float*, float*, float*, int)
                    3.82%  28.624ms         1  28.624ms  28.624ms  28.624ms  [CUDA memcpy DtoH]
                    2.23%  16.705ms         2  8.3525ms  8.2705ms  8.4344ms  [CUDA memcpy HtoD]
No API activities were profiled.

======== CPU profiling result (flat):
Time(%)      Time  Name
 66.19%  2.34125s  ???
 21.02%  743.57ms  cuMemcpyDtoH_v2
  5.40%  190.92ms  cuDevicePrimaryCtxRetain
  3.98%  140.68ms  cuInit
  2.84%  100.48ms  ???
  0.28%  10.048ms  cuMemcpyHtoD_v2
  0.28%  10.048ms  munmap

======== Data collected at 100Hz frequency
```

A11. We recommend you to take a look at profiling user doc for debugging and learning purposes. Available here: https://docs.nvidia.com/cuda/pdf/CUDA_Profiler_Users_Guide.pdf