Final Project
BDDs Re-ordering

Cheng-Hsiang Chiu
Department of Electrical and Computer Engineering
University of Utah, Salt Lake City, Utah
https://github.com/cheng-hsiang-chiu/ECE6740-CAD

Problem

- Minimize the size of the BDDs by re-ordering the variables in the BDDs.

Steps

- Read in an BLIF file
- Convert BLIF format into BDD
- Pick the re-ordering variable that results in a minimum number of nodes in the BDD by simulated annealing algorithm.

Algorithm – Simulated Annealing (SA)

```
while(TEMPERATURE < FROZEN_TEMPERATURE) {
        iterate(MAX_ITERATIONS_PER_TEMPERATURE) {
            generate_prop_ordering()
            delta_cost = prop_node_size – curr_node_size
            if(delta_cost < 0) {
                curr_ordering = prop_ordering
                if(prop_node_size < best_node_size) {
                    best_ordering = prop_ordering
                }
            }
            else {
                if(exp(-delta_cost/TEMPERATURE) > rand()) {
                    curr_ordering = prop_ordering
                }
            }
        }
        TEMPERATURE *= DEGRADE
}
```

Generate proposed ordering

- Randomly pick a strategy
  - Strategy 1: shuffle variables in a random range.
  - Strategy 2: swap two random variables.
  - Strategy 3: reverse the current ordering.
  - Strategy 4: swap first half and second half.

Experimental Environments

- Ubuntu 20.04
- Intel(R) Xeon(R) Gold 6138 CPU @ 2.00GHz
- L1d cache: 1.3 MiB
- L1i cache: 1.3 MiB
- L2 cache: 40 MiB
- L3 cache: 55 MiB
- CPU(s): 80
- Memory: 256GiB
- C/C++17

Experimental Results – 1

- TEMPERATURE = 1000, MAX_ITERATIONS_PER_TEMPERATURE = 1000, FROZEN_TEMPERATURE = 0.1, DEGRADE = 0.9

	Time (secs)	My_ node_ size	Memor y
test_L14	18	5	42 MB
testXOR	18	4	42 MB
adder8	29	82	44 MB
mult6	111	1116	43 MB
c499	152122	45922	2.1 GB
c880	54187	51863	1.2 GB

	Default_ node_ size	ABC_ node_ size
test_L14	6	6
testXOR	4	4
adder8	1267	104
mult6	1158	444
c499	45922	800
c880	346660	654

Experimental Results – 2

- TEMPERATURE = 1000, MAX_ITERATIONS_PER_TEMPERATURE = 100, FROZEN_TEMPERATURE = 0.1, DEGRADE = 0.9

	Time (secs)	My_ node_ size
test_L14	2	5
testXOR	2	4
adder8	3	95
mult6	11	1116
c499	14077	45922
c880	5309	85242

	Default_ node_ size	ABC_ node_ size
test_L14	6	6
testXOR	4	4
adder8	1267	104
mult6	1158	444
c499	45922	800
c880	346660	654

Experimental Results – 3

- TEMPERATURE = 1000, MAX_ITERATIONS_PER_TEMPERATURE = 10, FROZEN_TEMPERATURE = 0.1, DEGRADE = 0.9

	Time (secs)	My_ node_ size
test_L14	0.2	5
testXOR	0.2	4
adder8	0.3	174
mult6	2.1	1158
c499	1533	45922
c880	531	92524

	Default_ node_ size	ABC_ node_ size
test_L14	6	6
testXOR	4	4
adder8	1267	104
mult6	1158	444
c499	45922	800
c880	346660	654

Experimental Results – Performance vs Execution time

- TEMPERATURE = 1000, MAX_ITERATIONS_PER_TEMPERATURE = ?, FROZEN_TEMPERATURE = 0.1, DEGRADE = 0.9

	1000	100	10
test_L14	5 / 81 sec	5 / 2sec	5 / 0.2 sec
testXOR	4 / 77 sec	4 / 2 sec	4 / 0.2 sec
adder8	82 / 117 sec	95 / 3 sec	174 / 0.3 sec
mult6	1116 / 111 sec	1116 / 11 sec	1158 / 2.1 sec
c499	45922 / 152122 sec	45922 / 14077 sec	45922 / 1533 sec
c880	51863 / 54187 sec	85242 / 5309 sec	92524 / 531 sec

Experimental Results – Performance Improvement

	Default	ABC	Mine
test_L14	0%	0 %	16.67%
testXOR	0%	0%	0%
adder8	0%	91.79%	93.53%
mult6	0%	61.66%	3.76%
c499	0%	98.26%	0 %
c880	0%	99.81%	84.37%

Conclusions

- Refine current strategies of generating proposed ordering.
- SA parameters, such as TEMPERATURE, DEGRADE, and FROZEN_TEMPERATURE, could be learned by more test cases.
- To examine more solution space, the work could be implemented in a parallel manner with, such as OpenMP, CUDA, and SYCL.

Contribution

- Contribute lines 56 – 470.

- Some lines in functions initialize() and calculate_node_size() are copied from original github project.

Thank You