Final Project
BDDs Re-ordering

Cheng-Hsiang Chiu
Department of Electrical and Computer Engineering
University of Utah, Salt Lake City, Utah

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(#iterations_each_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_ratio
}
```

Generate proposed ordering

- 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

- Virtual machine, Ubuntu 20.04
- i7-9700k CPU @ 3.6GHz
- 4GB memory size
- C/C++17

Experimental Results – 1

- temperature = 1000, #iterations_each_temp = 1000, FROZEN_TEMPERATURE = 0.1, degrade_ratio = 0.9

	Time (secs)	My_ node_ size
test_L14	81	5
testXOR	77	4
adder8	117	82
mult6		
c499		
c880		

	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, #iterations_each_temp = 100, FROZEN_TEMPERATURE = 0.1, degrade_ratio = 0.9

	Time (secs)	My_ node_ size
test_L14		
testXOR		
adder8		
mult6		
c499		
c880		

	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, #iterations_each_temp = 10, FROZEN_TEMPERATURE = 0.1, degrade_ratio = 0.9

	Time (secs)	My_ node_ size
test_L14		
testXOR		
adder8		
mult6		
c499		
c880		

	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, #iterations_each_temp = ?, FROZEN_TEMPERATURE = 0.1, degrade_ratio = 0.9

	1000	100	10
test_L14	5 / 81		
testXOR	4 / 77		
adder8	82 / 117		
mult6	1103 /		
c499	(,)		
c880			

Future works

- Refine current strategies of generating proposed ordering.

- SA parameters, such as starting temperature, degrade ratio, 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.

Thank You

https://github.com/cheng-hsiang-chiu/ECE6740-CAD