Weekly Report for Yu Hu's work in week5

February 13, 2005

I continue working on the implementation of Weiping Shi's work this week. My works include the follows,

1. I've implemented the framework of the work to handle single-Vdd power. The outline of the algorithm is borrowed from John Lillis's ICCCAD'95 work, in which all candidates are organized as subsets indexed by E in strictly increased order, such as $\{E_1, T_1\}$, $\{E_2, T_2\}$, ..., $\{E_n, T_n\}$, where $E_i < E_j$ if $i < j$, and T_i is a candidate tree whose nodes are (Q, C) pairs.

2. Tested by six benchmarks generated by King Ho (s1-s6), I obtained a statistic of candidates number in the same power value. Table1 shows the results from experiments. In this table, column "pnum" shows the number of distinct power values, and column "max_subset_num" shows the maximum candidates number with the same power value, i.e. maximum candidates number in T_n in some subset $\{E_n, T_n\}$.

Table 1: Statistic of the number of candidates and power value

net	pnum	max_subset_num
s6	63	1830
s5	48	1080
s4	42	672
s3	18	160
s2	16	88
s1	15	80

From these observations, we found that the number of distinct power values [1] isn't as many as we expected, and the number of candidates with a same power value is much more than we expected. So I believe we can perform Shi's Dac'03 pruning technologies in each subset $\{E_i, T_i\}$ to get a substantial speedup.

3. I've implemented a new pruning criterion, predictive pruning (in Weiping Shi's DAC'03), in each subset $\{E_i, T_i\}$. By using predictive pruning, we can prune the much more candidates and get some speedup. Tested by s1-s6, table2 shows the experimental results. In this table, column "non-tri-left", "non-tri-pruned" and "non-time" show the number of the candidates left after pruning, number of candidates pruned, and running time (cpu-pm1.4GHz, mem-128M) using traditional criterion. Similarly, we can get the denotation of column prefixed by "p-". Note that the fraction in square brackets denotes p-tri-left/non-tri-left.

Table 2: Predictive pruning vs. traditional pruning

net	non-tri-left	non-tri-pruned	non-time	p-tri-left	p-tri-pruned	p-time
s6	290451	119894	35.120s	78639 [1/4]	42143	30.386s
s5	169238	51935	0.984s	53026 [1/3]	21246	0.832s
s4	64714	26914	0.365s	20193 [1/3]	10775	0.331s
s3	24496	4409	0.213s	14139 [1/2]	3480	0.167s
s2	10961	1296	0.109s	6060 [1/2]	976	0.056s
s1	9830	1039	0.100s	5313 [1/2]	915	0.059s

From the experimental results, we can see that the predictive pruning can really prune much more redundant candidates to narrow the search space. At present, there still exist some problems in my codes, in which the power-slack tradeoff curves don't match in some points before and after performing predictive pruning. I'm trying to find why this problem exists.

[1] In my implementation for single-Vdd, power values are presented with sum of capacitances.