Weekly Report for Yu Hu's work in week6

February 20, 2005

1. **Read King Ho's code in BIC package.**

   I spent a couple of days to read BIC code carefully. Since the main goal of my first step of this project to integrate Weiping's work into BIC to handle single-Vdd buffer insertion problems (without tree construction), I read code in two sub-directories "buf_determ" and "share" and finally grasped whole framework of King Ho's code.

   As King Ho's explanation, "buf_determ" code can be run with "-B n" parameter, which means sampling n elements in each subset. If this parameter isn't added, "buf_determ" will run in John Lillis's alg., otherwise, it'll run in King Ho's DAC'05 alg..

2. **Make some experiments based on BIC package.**

   After read through BIC code, I made the following experiments:

   Firstly, I organized all candidates as $\{C_1, pairs_1\}$, $\{C_2, pairs_2\}$, $\ldots$, $\{C_n, pairs_n\}$, where C_i is the capacitance value of candidates$\in$subset $\{C_1, pairs_1\}$, $C_i < C_j$ if $i < j$, and $pairs_i$ is a sub-subset whose elements are (P, RAT) pairs, where P is power dissipation of the current candidate. The executable produced as this organization is called "buf_determ_pq".

   Then, I organized all candidates as $\{P_1, pairs_1\}$, $\{P_2, pairs_2\}$, $\ldots$, $\{P_n, pairs_n\}$, where P_i is the power dissipation value of candidates$\in$subset $\{P_1, pairs_1\}$, $P_i < P_j$ if $i < j$, and $pairs_i$ is a sub-subset whose elements are (C, RAT) pairs, where C is capacitance of the current candidate. The executable produced as this organization is called "buf_determ_cq". We should note that, in both "buf_determ_pq" and "buf_determ_cq", elements in $pairs_i$ is organized as a Red-Black tree in King Ho's implementation.

   After that, I ran both "buf_determ_pq" and "buf_determ_cq" with s1-s4, and there're two observations obtained based on the comparison:

   (a) The different power values of "buf_determ_pq" is about twice less than "buf_determ_cq", and the average number of elements in $pairs_i$ of "buf_determ_pq" is twice larger. This means the pruning tree strategy in King Ho's DAC'05 paper will perform better under "buf_determ_pq". Furthermore, more candidates can be pruned under "buf_determ_pq". The experimental results show that "buf_determ_pq" is over 3 times faster than "buf_determ_cq" without sampling.

   (b) In both organization modes, the candidates number in $pairs_i$ is not large enough. There're over 50% of $pairs_i$ have less than 10 candidates. Obviously, this will worsen the efficiency of pruning tree strategy. In another words, the efficiency of pruning tree will be shown if the tree has large elements, otherwise, the overhead of maintaining the pruning tree (such as rotation and check nodes' color) will overwhelm the pruning efficiency, which makes the running time longer.

   Based on observation(b) shown above, I rewrote the pruning strategy without using pruning tree, instead, I used a simple vector to organize elements in $pairs_i$. Tested with s1-s4, the experimental results show that I got about 20% speedup upon Red-Black tree pruning in both "buf_determ_pq" and "buf_determ_cq". Surely, this verified the statements in observation(b).

   Based on observation(a) and (b), I used vector instead of tree data structure to organize candidates in each $pairs_i$, and I used "buf_determ_pq" mode to organize whole candidates in my implementation.

3. **Combine aggressive predictive pruning and sampling into BIC and get some speedup.**

   (a) Firstly, I added predictive pruning into BIC, which means I used the minimal-output-resistance buffer in buffer library to perform predictive pruning. But I found that little speedup can be obtained.

(b) Then I used maximal-output-resistance buffer to perform predictive pruning (aggressive predictive pruning in Weiping Shi's ASPDAC'05 paper), and obtained substantial speedup comparing with John Lillis's alg.. Also I found that the results produced after aggressive pruning are almost as the same as optimal (Lillis's alg.).

(c) After that, I combined aggressive pruning and sampling together. Using the same settings (sampling parameter is set as 20) and testing with s1-s6 with single-Vdd buffers, I compared my new approach with King Ho's DAC'05 approach (SVB). I obtained over 3 times speedup upon King Ho's approach, meanwhile the solution produced by my new approach even has a little lower power dissipation with the same RAT. The running time is shown as Tab.1. Note that the running time of John Lillis's alg. isn't given here since it's much slower than King Ho's alo..

Table 1: Running time (Yu's vs. King Ho's)

net	Yu's	King Ho's
s1	1	3
s2	2	5
s3	3	13
s4	14	63
s5	37	154
s6	55	271