Weekly Report for Yu Hu's work in week9

March 13, 2005

In this work, I've done the following works:

1. **Found and fixed the debug of my code in 3D sampling for routing.** I checked my 3D sampling code in routing, and found that I had made a stupid mistake. I fixed it and 3D sampling can work in routing now. The solution produced by 3D sampling is correct. The experimental results show that we can obtain substantial speedup by setting the number of sampling options properly.

2. **Found the reason of inconsistency of predictive min-delay between my code and ISLPED'05 code.** I think that I've found the reason of inconsistency between BIC and ISLPED data. Actually, the delay calculations in both packages are correct, and the main problem isn't caused by inconsistent settings.

   In the predictive pruning implementation of BIC package, I IGNORED the length of minimal non-buffered interconnect. Based on the formula in ISLPED'05 paper, this length is calculated as 3900um (using my BIC settings). The inconsistency happened within this length, which means, there might be no buffer inserted within a 3900nm interconnect in an optimal final buffered tree. In this situation, ISLPED'05 paper gave a much larger estimation of unit-length-min-delay. But, ISLPED'05 paper formula can work well when the interconnect length is larger than 3900nm.

   Based on this observation and analysis, the predictive minimal delay pruning shouldn't be performed within the length of minimal non-buffered interconnect (e.g. 3900nm for my current settings in BIC), and we may just predict the upstream delay lower bound for those nodes whose path lengths from source are larger than 3900nm. I've tested several nets, and the results are ok now.

3. **Trying to find the reason of inconsistency of predictive min-power between my code and ISLPED'05 code.**

   I made some experiments to test the correctness and the effectiveness of Cynthia's min-power estimation model in her ISLPED'05 paper.

   Firstly, I route a 2-pin net (with 20000um length) with BIC and calculate the upstream power (total power downstream from source minus total power downstream from the current node) for each node, also the distance from source to current node is calculated. So the unit-length power is obtained for each node. Then I calculated the upstream delay slack based on the formulae in Section 2 in her ISLPED'05 paper. At last a unit-length power vs. slack table is generated.

   After that, I generated a unit-length power vs. slack table for a 19-pin net (s1.cmp in BIC) in the same way.

   Then, I calculated the unit-length-min-power vs. slack table (prediction table) based on the formulae in Section 3 in your ISLPED'05 paper.

   Finally, I drew power-slack curves for 2-pin, s1.cmp and prediction tables, respectively. The results are shown in Fig.1.

   There are two observations based on Fig.1.

   (a) When you take a look at "pwr-slack.eps", you may find that the ISLPED'05 paper gives a much lower prediction for s1.cmp (this is reasonable, since we just consider a 2-pin net in our modeling, and branches in the path contribute a lot to the power.). The BAD thing is, the ISLPED'05 paper gives a little higher prediction for 2-pin net in some slacks, such as 0.15, and this is unreasonable as the ISLPED'05 paper gives a lower bound of upstream power.

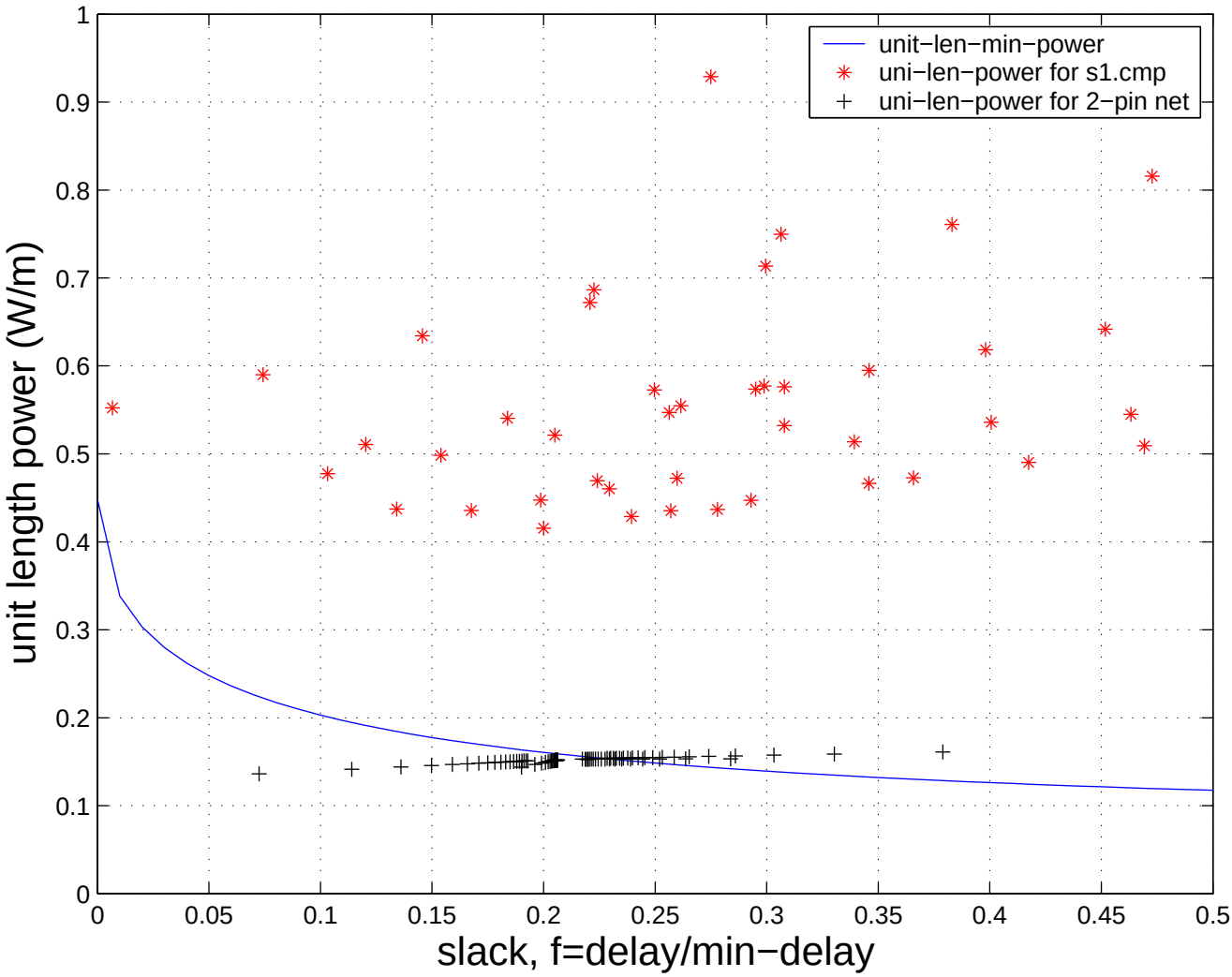

Figure 1: power-slack curves

(b) One thing Cynthia noticed in Fig.1 for the unit-length-power for 2-pin net is that the unit length power is almost the same across all delay slacks or even slightly increasing as the delay slack gets larger. Based on the formula, the unit length power would be larger with smaller delay slack and then as the delay slack increases, the unit length power would decrease. This observation seem not correspond to my other cases. This might be something to look into as well.

However, I haven't figure out the reason for these two things at present due to the time limitation.

4. **Modified my code and had some testing for some large examples on both routing and buffer insertion** King Ho sent me two large examples, grid.10 and grid.20. When I ran grid.10, I found that the memory usage is about 2Gb, which makes the memory swapping take much long time. So I checked the memory leakage in my code in routing, and made some modification to free unused memory. The main idea is to free all pruned solutions and unsampled options (those solutions aren't sampled). Note that, we must use a counter to trace the number of the current solutions are indexed by others (to get a trace from a solution in source when we find an optimal solution).

   After the modification, I ran grid.10, and get a solution in about one hour with about 500Mb memory usage (I just keep 5 solutions in each 3D sampling). However, I still can't get the result for grid.20, since it used about 2Gb memory. At present, I can't find an efficient way to conquer the memory explosion when the example goes larger.

5. **Did some testing for dual-Vdd buffer insertion** King Ho gave me some advice to match BIC and ISLPED: "To make them match, you will need to recreate the entire buffer library based on the formulas in the ISLPED paper. For example, you need to derive the Reff by Rmin / size using 4xxx / 16 for 16x buffers, instead of the SPICE simulated values that I have given you in the package."

   Based on this, I recreated buffer library which include 16X, 32X and 64X dual-Vdd buffers. And I tested my buffer insertion code by s1.cmp - s9.cmp and obtained up from 10 to 30 times speedup. Furthermore, I tested much larger examples (r1.cmp - r5.cmp, contained from 2k to 20k nodes), and found that there're something incorrect in some of my experimental results. I'm trying to find the reason.