Weekly Report for Yu Hu's work in week7

February 27, 2005

In this week, I tried to add predictive power pruning strategy into BIC package. The following work has been done.

1 Framework of Predictive Power Pruning

We can perform predictive power pruning as following step:

step.1 Pre-calculate the unit-length min-delay table indexed by buffer type and vdd. Details of this table can be found in section3.1.

step.2 Pre-calculate the unit-length min-power for a given slack (extra percentage over min-delay). This table is indexed by vdd, buffer type and slack. Details is shown in section3.2

step.3 Run min-delay program to get the maximum RAT $RAT*$ at source;

step.4 For each node in the tree, calculate the path length from source to this node;

step.5 Now enter the bottom-up solution propagation. As shown in Fig.1, we use the min-power of path from source to v1 as the upstream predictive power from node v1, which is also a lower bound of upstream power from v1. For current node v1 (see Fig.1), Suppose α1-(p1,c1,rat1) and α2-(p2,c2,rat2) are two options at

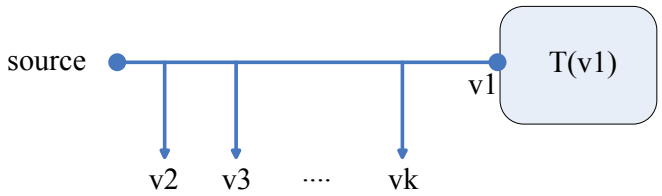

Figure 1: The general situation from the current node to source

node v1, which is the root of sub-tree T(v1). The distance from source to v1 is dv. The current voltage is high-Vdd. Then we calculate the slack of α1 as

$$f1 = \frac{rat1 - RAT*}{unitLenMinDelay \times dv} - 1 \tag{1}$$

Lookup $unitLenMinDelay$ in the unit-length min-power table with high-Vdd and f1, we can get the min-power for slack $f1$ as $pre{-}p1$. Now **we can prune** α1 **if** $p1{+}pre{-}p1 > p2$ (Predictive Power Pruning Rule).

2 The Experiments I've Done with Predictive Power Pruning

I integrate predictive power pruning strategy into BIC, and found that the results after performing predictive power pruning aren't correct, because most all non-redundant options are pruned. When I debugged the code, I found that, $p1{+}pre{-}p1 > p2$ always holds for almost any option pairs p1 and p2. This is because, in the beginning (nodes near with sinks) of bottom-up, $pre{-}p1$ (the predictive min-power of the path from current node to source) is always

much larger than p2 (often over 10 times larger), so surely $p1+pre-p1 > p2$ holds. So I don't think the this pruning can work. **Prof. He, do I mis-understand the predictive power pruning in my implementation?**

Furthermore, I tested another way to perform the predictive power pruning. By running min-delay program followed by "treestat" in BIC package, I obtain the associated power of the result buffered-tree as "power_upper_bound". Then use "power_upper_bound" to prune, which means, for each newly generated option (p,c,rat), calculate its predictive min-power $pre - p$ as mentioned in 5, it should be pruned if $p + pre > power\_upper\_bound$. However, the experimental results show that this pruning isn't so effective, since the "power_upper_bound" is much looser ($p + pre < power\_upper\_bound$ in most cases) which makes little options can be pruned by this rule. If I set a tighter upper, such as power_upper_bound*0.9, this pruning will show some effectiveness, but surely it'll lose some optimum of both delay and power. However, this pruning will be meaningful for the problem allowing user to input a power bound and get a buffered tree satisfying this bound. Obviously, if the user-input bound is much less than min-power under RAT*, we can get a substantial speedup. At present, experiments aren't enough, and I'll do some extensive experiments with this idea in the next week. **Prof. He, is this idea sound and worthwhile?**

3 Appendix

3.1 Unit Length Min-delay Table Calculation

As shown in section2.1 in King Ho's ISLPED paper, the optimal (minimum) unit-length delay is the function of r_s, c_o, c_p, r and c, where r_s, c_o, c_p are parameters for unit-size buffer, which are output resistance, input capacitance and input capacitance, respectively. **Cautions: high-Vdd and low-Vdd buffers in different technical nodes (130nm, 90nm, 65nm, ...) should be calculated respectively in this table.** In BIC package, I didn't find the corresponding setting of c_p, and King Ho told me $d_{int} = c_o \times r_o$, so I calculated $c_o = d_{int}/r_o$ by using BIC's settings. Since I didn't consider wire sizing, I used min-width wire in my calculation, which means I chose r and c of the min-width wire by using BIC's settings. All these parameters I used to calculate this table is shown in Tab.1.

Table 1: Settings used in unit-length min-delay table

parameter	value	comments
r_s	4732Ω	this value is obtained from Tab.1 in King Ho's DAC'05 dualVdd paper
c_o	$15.2fF$	$c_o = d_{int}/r_o$, where $d_{int} = 72.0ps$, $r_o = 4732\Omega$.
c_p	$0.47fF$	
r	$0.186\Omega/\mu m$	Caculated in BIC with min-width wire
c	$0.0519fF/\mu m$	$c = cx + cg = 0.0237fF/\mu m + 0.0282fF/\mu m$
minDelay	$0.09fs/\mu m$	unit length minimum delay for 65nm with high-Vdd buffer

Note that I just give the settings for calculating for high-Vdd buffer in 65nm node, since I just need the unit-length min-delay under this situation for single-Vdd buffer insertion. By the way, the units in BIC and ISLPED (matlab code) aren't consistent. In matlab code, all parameters are set as primary unit (such as m, s, f, Ω, etc.). In BIC, wire length unit is μm, capacitance unit is fF, RAT unit is ps, power per switch unit is fJ.

In this calculation, I have two questions, **King Ho, could you explain them? Thanks.**:

1. By using the settings in BIC and $c_o = d_{int}/r_o$, c_o is about 50 times of c_p. I think that it should be unreasonable.
2. I doubt with the correctness of eq.(1) in King Ho's ISLPED paper. It's said that, the delay of the driving repeater and the wire segment is $\tau = r_s(c_o + c_p) + r_s cl/ + rlsc_o + rcl^2/2$. Note that r_s **is unit driving resistance, not the unit buffer output resistance!** If the unit buffer output resistance is r_o, the correct formula should be $\tau = r_s(c_o + c_p) + r_o cl/ + rlsc_o + rcl^2/2$.

3.2 Unit Length Min-power for a Given Slack Table Calculation

By calling the function "get_power_density_breakdown" written in King Ho's matlab code, I obtain the following unit-length min-power table (Tab.2). In the calculation of Tab.2, I used the settings shown in Tab.1.

Table 2: Unit-length min-power table for high-Vdd buffer in 65nm node

slack	vdd(v)	power(W/m)
0.1000	0.9000	0.0599
0.2000	0.9000	0.0555
0.3000	0.9000	0.0532
0.4000	0.9000	0.0516
0.5000	0.9000	0.0505
0.6000	0.9000	0.0497
0.7000	0.9000	0.0490
0.8000	0.9000	0.0485
0.9000	0.9000	0.0481
1.0000	0.9000	0.0478
0.1000	1.2000	0.1035
0.2000	1.2000	0.0963
0.3000	1.2000	0.0923
0.4000	1.2000	0.0898
0.5000	1.2000	0.0879
0.6000	1.2000	0.0866
0.7000	1.2000	0.0856
0.8000	1.2000	0.0847
0.9000	1.2000	0.0840
1.0000	1.2000	0.0835