Fast Buffer Insertion for Yield Optimization Under Process Variations

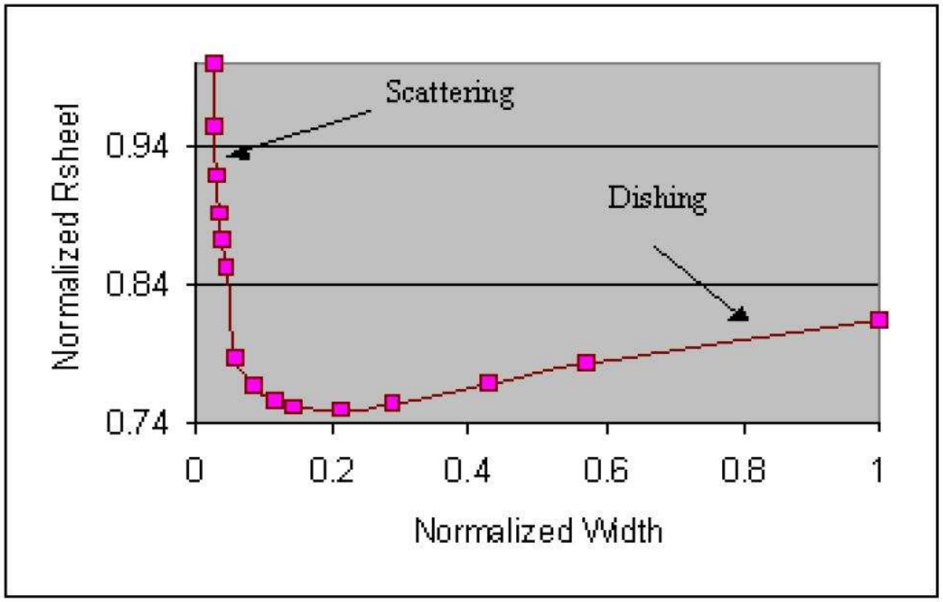
Ruiming Chen and Hai Zhou
Electrical Engineering and Computer Science
Northwestern University
Evanston, IL 60208

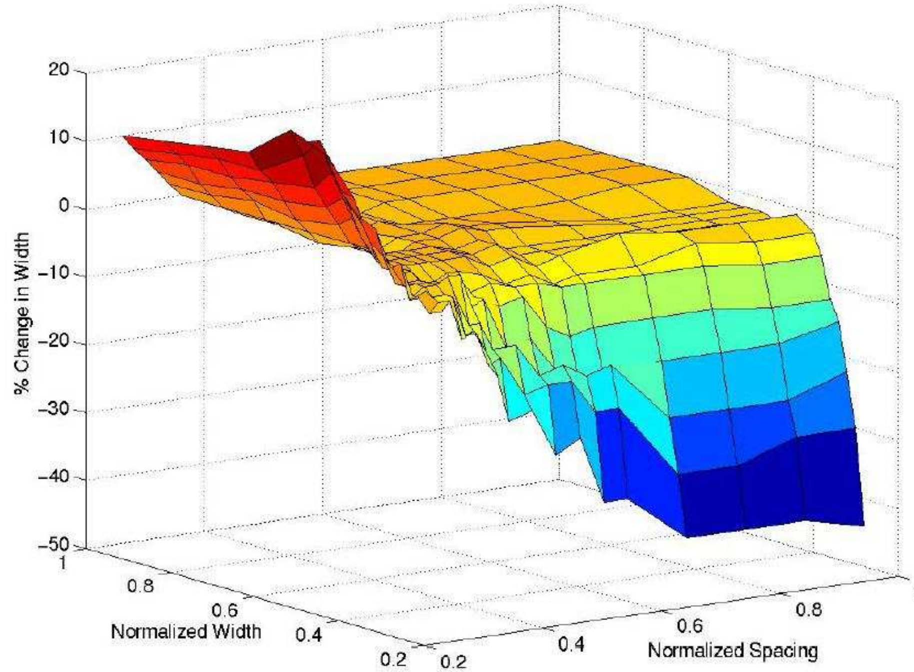
01/24/2007

Interconnect optimization by buffering

- Buffering (or buffer insertion) is the most effective method for interconnect optimization.
- Van Ginneken style dynamic programming algorithm:
  - Bottom-up merging and generation of candidate solutions
  - Top-down selection of the best solutions for delay minimization

Process variations on wires





[Nagaraj et al. DAC05]

- What impact will process variations have on buffering?
- How should we buffer under process variations?

Outline

- Previous work
- Delay model of interconnects under process variations
- Problems in the pruning of inferior solutions
- Fast greedy selection of statistical solutions
- Experimental results
- Conclusions

Previous work

- Deng and Wong [ICCAD05] considered buffering a path under process variations; found buffering based on nominal values is sufficient.

- Khandelwal et al. [ICCAD03] and Davoodi et al. [ICCD05] applied statistical pruning for tree buffering: expensive two-stage (solution generation then pruning) approach.

- Xiong et al. [ISPD06] proposed "transitive closure" based pruning: but the ordering property ($P_r(B \geq A) \geq \eta$ or $P_r(A \geq B) \geq \eta$) is not ture unless $\eta = 0.5$.

Our objective:

An efficient algorithm for buffering big trees with good results.

Delay model of interconnects under process variations

- Elmore delay model
- R, C: Gaussian distributions of canonical form:

$$
\begin{aligned}
R &= R_0 + \sum_{i=1}^{n} R_i \epsilon_i + r_{n+1} \\
C &= C_0 + \sum_{i=1}^{n} C_i \epsilon_i + c_{n+1}
\end{aligned}
$$

Operations in buffer insertion

- Add a wire (R_w, C_w): $D' = D + R_w(C + C_w/2)$ and $C' = C + C_w$.
- Insert a buffer (r_b, c_b): $D' = D + d_b + r_b C$ and $C' = c_b$.
- Merge two branches (D_1, C_1) and (D_2, C_2): $D' = \max(D_1, D_2)$ and $C' = C_1 + C_2$.
- With process variations: R_w, C_w, d_b, r_b, c_b, etc. all become random variables.
- Both max and multiplication of random variables are needed.
- They are approximated back to the canonical forms by moment matching.

- $D = \mathsf{max}(D1, D2) = D_0 + \sum_{i=1}^{n} D_i \epsilon_i + d_{n+1}$.
- $D = RC = D_0 + \sum_{i=1}^{n} D_i \epsilon_i + d_{n+1}$.

Prune inferior solutions

- (D_1, C_1) and (D_2, C_2) are two solutions at the same node
- Without process variations, if $D_1 \leq D_2$ and $C_1 \leq C_2$, (D_2, C_2) is inferior, and can be deleted.
- With process variations, if

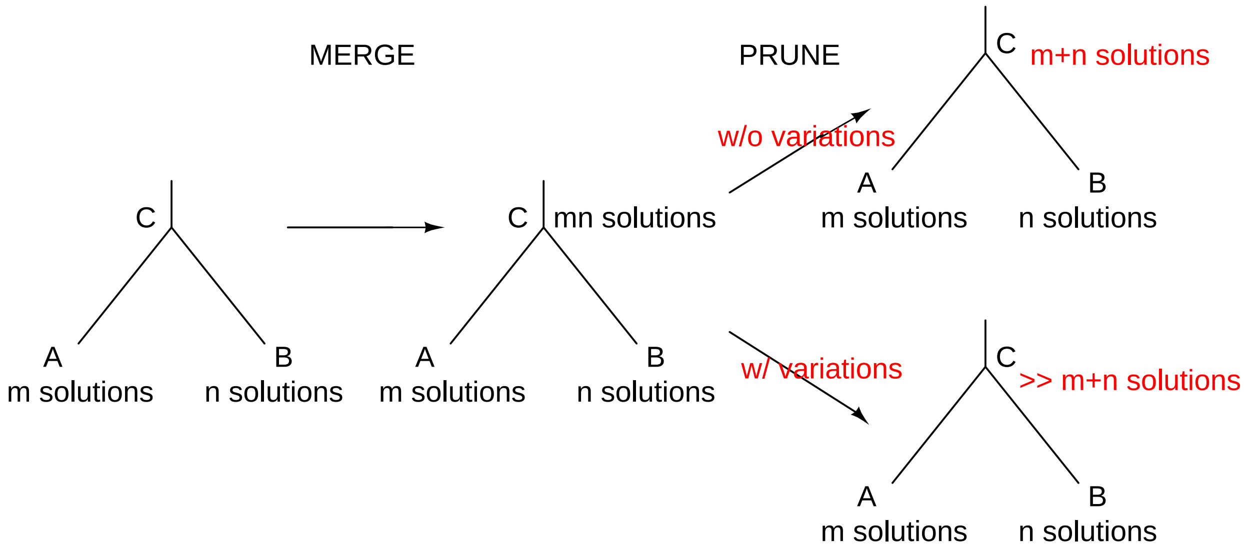
$$Pr(D_1 \leq D_2, C_1 \leq C_2) = 100\%,$$

  (D_2, C_2) is inferior, and can be deleted.

  – Too few can be deleted using the 100% probability.

  – Relax: $Pr(D_1 \leq D_2,\ C_1 \leq C_2) \geq \eta$, where $\eta \in (0.5, 1)$.

Still too many solutions



- Still $O(mn)$ statistical solutions after pruning, e.g. "r1", 34 deterministic solutions and 595 statistical solutions at a node close to the sinks.

- Even pruning down to $O(m+n)$ solutions will take $O(m^2n^2)$ running time, if pairs of merged solutions need to be compared (Khandelwal et al. ICCAD03). There is no guarantee that final solution will not be worse than deterministic approaches.

Our idea

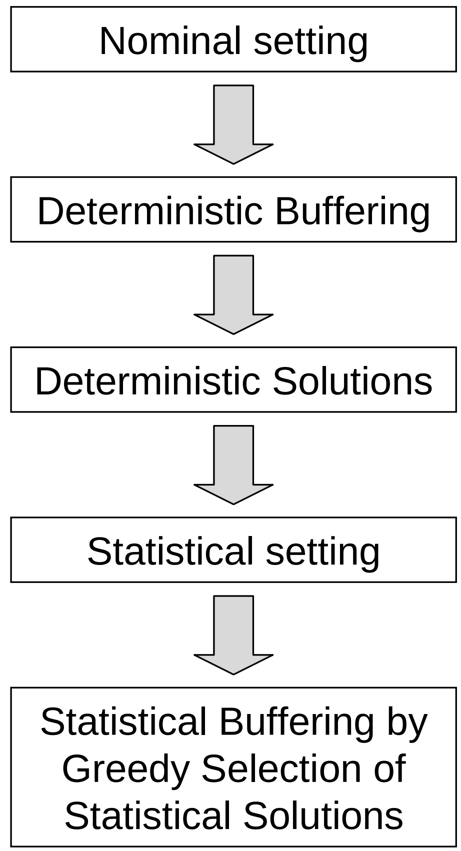
- Not comparing every pair...
- but selecting the “best” over each subset given by a deterministic approach.

Which one: nominal or worst?

- Nominal case: all the R and C are fixed at their nominal values
- Worst case: all the R and C are fixed at their $\mu + 3\sigma$ values
- Buffering the nominal case has higher yield in general:
  - Gaussian distribution has highest probability density at mean value.
  - For a wire, larger R will correspond to smaller C: the impact on Elmore delay may be canceled out.

	p1	p2	r1	r2	r3	r4	r5
Wor Yield	63.31	73.37	49.00	62.14	69.20	75.62	71.20
Nom Yield	62.88	79.21	62.60	67.03	81.30	78.56	70.65

General flow of our approach



What kind of solutions can improve the delay?

deterministic solutions: $D_1 > D_2 > D_3 > D_4 > \ldots > D_n$

corresponding statistical solutions: $D_1' > D_2' > D_3' > D_4' > \ldots > D_n'$ (mean)

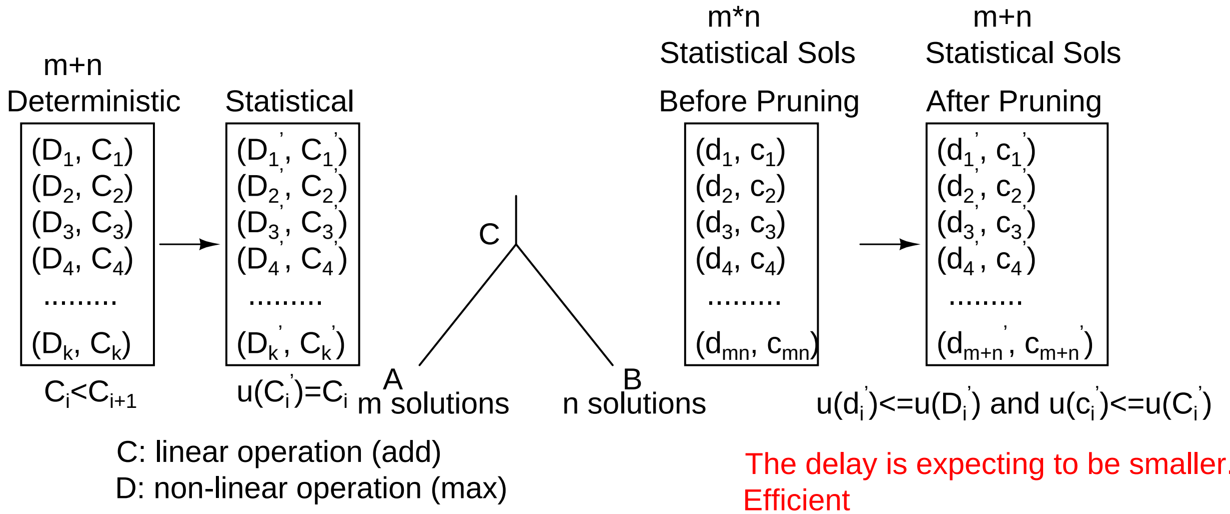
deterministic position inferior, pruned

Actual position

$C_1 < C_2 < C_3 < C_4 < \ldots < C_n$ (mean)

- C involves only linear operations, while D involves non-linear multiplication and max operations.
- E.g., a solution (D, C) is inferior to (D_2, C_2) in deterministic situation, but is not inferior in statistical situation where (D_3, C_3) becomes inferior.
- We need to keep those statistical solutions that are inferior in deterministic approach but not inferior in statistical approach. These solutions have high probability to improve the delay.

Greedy selection of statistical solutions



- For each non-inferior deterministic solution (D_i, C_i), we select a statistical solution (d, c) satisfying

$$\mu(c) \leq C_i$$

and $\mu(d)$ is minimal.

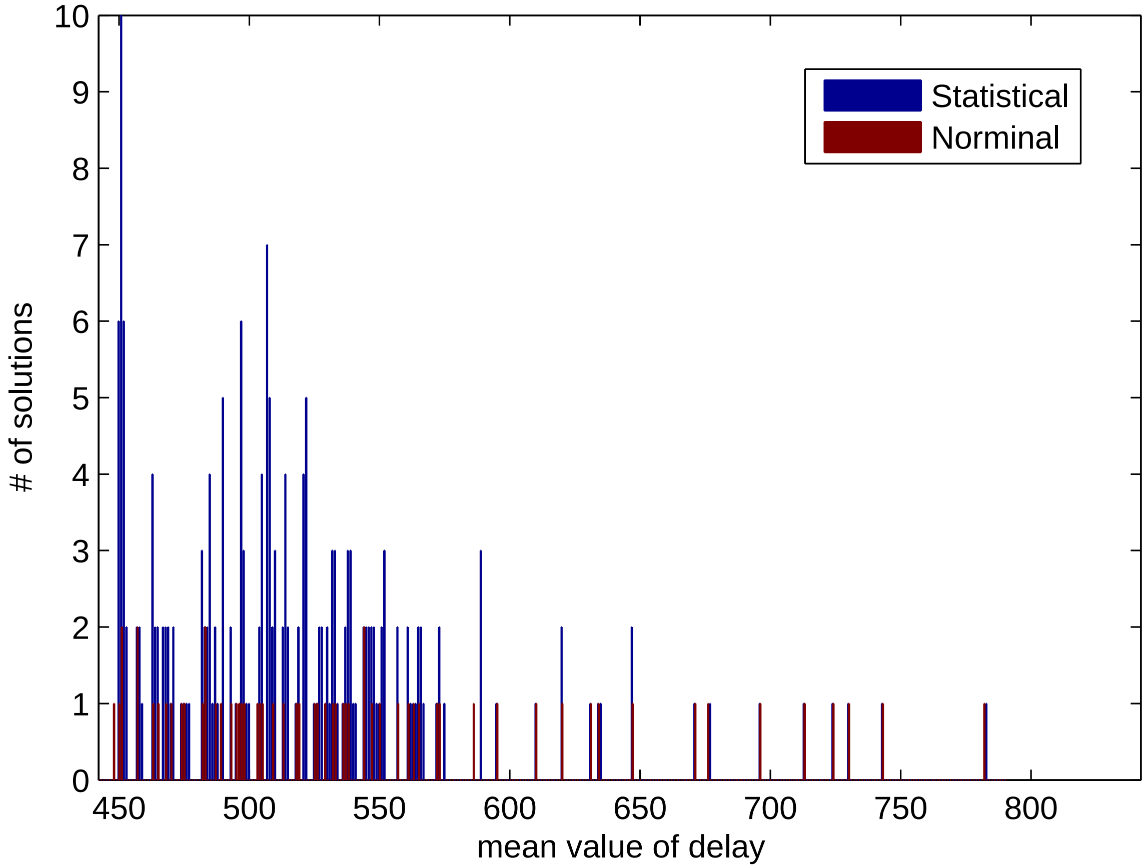
Greedy selection can improve delay

- Suppose (D_2, C_2) is the statistical representation of a non-inferior deterministic solution.
- E.g., attach a wire (R_w, C_w): $D'_i = D_i + R_w(C_i + C_w/2)$ and $C'_i = C_i + C_w$.
- Suppose $\mu(C_1) < \mu(C_2)$ and $\mu(D_1) < \mu(D_2)$
- $\mu(C'_1) < \mu(C'_2)$: the order of C does not change because of the linear operation.

Greedy selection can improve delay

- $Pr(\mu(D_1') \leq \mu(D_2'))$ is high when C_1 is close to C_2. Often this case: the deterministic solutions are tightly surrounded by statistical solutions.

- Thus, $Pr(D_1' \leq D_2')$ and $Pr(C_1' \leq C_2')$ are high. If we replace (D_2, C_2) by (D_1, C_1), the probability that the delay gets improved is high. This is the reason why the greedy selection can impove the delay.



Statistical
Norminal
# of solutions
mean value of delay
0
1
2
3
4
5
6
7
8
9
10
450
500
550
600
650
700
750
800

Characteristics of testcases

Big trees

name	# sinks	# nodes	# buffer locations
p1	269	537	268
p2	603	1205	602
r1	267	533	266
r2	598	1195	597
r3	862	1723	861
r4	1903	3805	1902
r5	3101	6201	3100

Experimental results

Nets	V-G(Nom)		Khandelwal		Xiong	Ours			Gain
	# B	Yield (%)	T (s)	Yield (%)	Yield (%)	# B	T (s)	Yield (%)	(%)
p1	162	62.88	N/A	N/A	63.88	158	10.22	75.28	12.40
p2	268	79.21	N/A	N/A	73.60	268	27.92	94.37	15.16
r1	166	62.60	198.75	77.81	59.10	169	5.26	79.38	16.78
r2	358	67.03	N/A	N/A	62.90	363	17.53	79.49	12.46
r3	517	81.30	N/A	N/A	79.30	523	14.20	92.48	11.18
r4	1187	78.56	N/A	N/A	79.26	1192	52.67	87.26	8.70
r5	1893	70.65	N/A	N/A	71.03	1918	76.63	80.36	9.71
avg									12.34

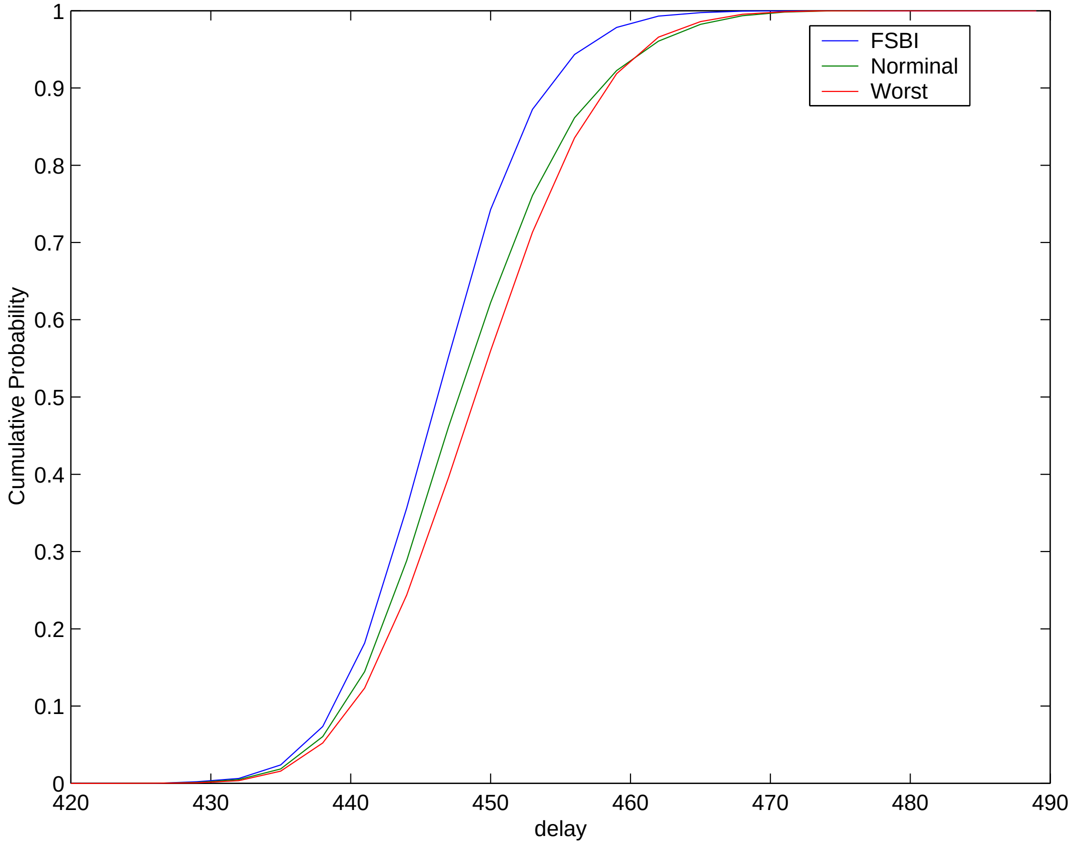
"N/A" means that the testcase cannot be finished because of the memory limit (2GB) or time limit (3 hours).

Experimental results

Nets	V-G(Nom)		Khandelwal		Xiong	Ours			Gain
	# B	Yield (%)	T (s)	Yield (%)	Yield (%)	# B	T (s)	Yield (%)	(%)
p1	162	62.88	N/A	N/A	63.88	158	10.22	75.28	12.40
p2	268	79.21	N/A	N/A	73.60	268	27.92	94.37	15.16
r1	166	62.60	198.75	77.81	59.10	169	5.26	79.38	16.78
r2	358	67.03	N/A	N/A	62.90	363	17.53	79.49	12.46
r3	517	81.30	N/A	N/A	79.30	523	14.20	92.48	11.18
r4	1187	78.56	N/A	N/A	79.26	1192	52.67	87.26	8.70
r5	1893	70.65	N/A	N/A	71.03	1918	76.63	80.36	9.71
avg									12.34

No loss on the quality of solutions for “r1”

CDFs for “r2”



Our approach has higher yield in the whole range.

Conclusions

- Process variations have impacts on buffer insertion.
- Our experiments show that buffering by nominal case get relatively good results.
- Proposed a statistical optimization methodology that utilizes a good deterministic approach as a guidance for efficient statistical solution selection.
- Designed an efficient and effective buffering technique considering process variations
  - Greedy selection of statistical solutions.
  - Local refinement based on deterministic solutions.

Thank you