Statistical Bellman-Ford Algorithm With An Application to Retiming

Mongkol Ekpanyapong, Thaisiri Watewai* and Sung Kyu Lim

School of Electrical and Computer Engineering
Georgia Institute of Technology

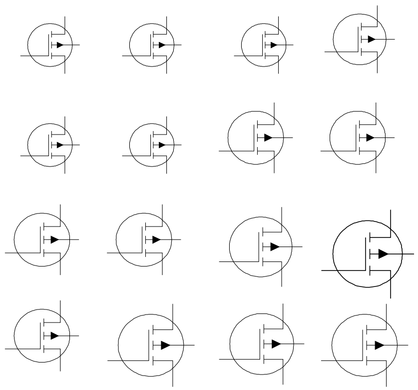
Department of Industrial Engineering and Operations Research
University of California, Berkeley*

Outline

- Introduction and Motivation
- Related Work
- Methodology
- Experimental Results
- Conclusions

Deep Submicron Design

- Make transistor on the same die having different delay

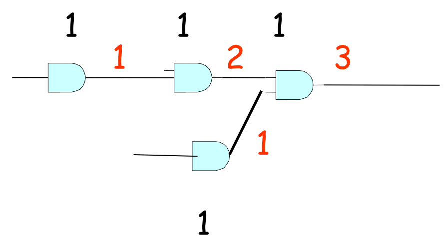


Deep Submicron Design Issues

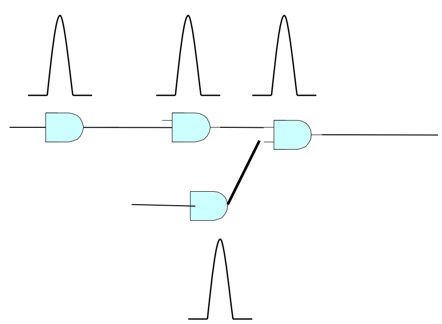
- Harder to get exact desired transistor size even on the same die
- Transistor size has directed impact on transistor delay
- Timing calculation becomes more complex
- Not only transistor size, wire size can also suffer from this variations

Static vs. Statistical Timing Analysis

- Static Timing Analysis



- Statistical Timing Analysis (Assume Gaussian distribution)



Outline

- Introduction and Motivation
- Related Work
- Methodology
- Experimental Results
- Conclusions

Related Work

- Chang, H. and Sapatnekar, S, *Statistical timing analysis considering spatial correlations using a single pert-like traversal*, ICCAD03
- Visweswariah, C., Ravindran, K., Kalafala, K., Walker, S., and Narayan, S., *First-order incremental block-based statistical timing analysis*, DAC 04
- Chen, R. and Zhou, H., *Clock schedule verication under process variations*, ICCAD04
- Cong, J. and Lim, S. K., *Retiming-based timing analysis with an application to mincut-based global placement*, TCAD04

Outline

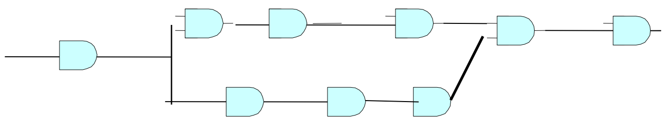
- Introduction and Motivation
- Related Work
- Methodology
- Experimental Results
- Conclusions

What Make SSTA Difficult?

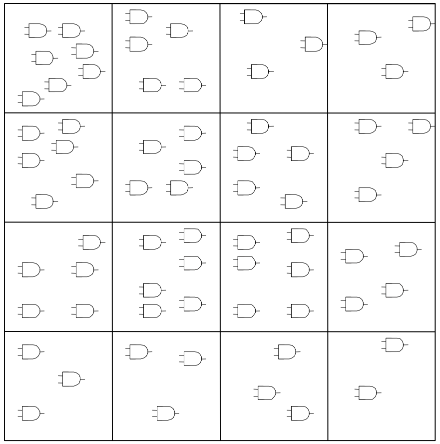
- Correlation
- Maximum/Minimum function’s approximation

Correlation

- Reconvergence Path Correlation



- Spatial Correlation



Principal Component Analysis

- Delay estimation with variation

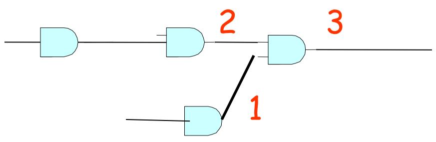
$$d = d_0 + \sum_{i\in\Gamma_g}[\frac{\partial d}{\partial L_g^i}]\triangle L_g^i + \sum_{i\in\Gamma_g}[\frac{\partial d}{\partial W_g^i}]\triangle W_g^i + \sum_{i\in\Gamma_{int}}[\frac{\partial d}{\partial T_{int}^i}]\triangle T_{int}^i$$

- Principal component analysis classifies each coefficient into orthogonal terms

$$a_0 + \Sigma^n_{i=1} a_i \,\Delta x_i + a_{n+1} \,\Delta R_a$$

$$b_0 + \Sigma^n_{i=1} b_i \,\Delta x_i + b_{n+1} \,\Delta R_b$$

Maximum/Minimum Function Approximation



$$\phi(x) \equiv \frac{1}{\sqrt{2\pi}} exp(-\frac{x^2}{2}) \quad (1)$$

$$\Phi(y) \equiv \int_{-infty}^{y} \phi(x)dx \quad (2)$$

$$\theta \equiv (\sigma_A^2 + \sigma_B^2 - 2\rho\sigma_A\sigma_B)^{1/2} \quad (3)$$

$$T_A = \int_{-\infty}^{\infty} \frac{1}{\sigma_A} \phi\left(\frac{x - a_0}{\sigma_A}\right) \Phi\left(\frac{\left(\frac{x-b_0}{\sigma_B}\right) - \rho\left(\frac{x-a_0}{\sigma_A}\right)}{\sqrt{1-p^2}}\right) dx \quad (4)$$

$$E[max(A, B)] = a_0T_A + b_0(1 - T_A) + \theta\phi[\frac{a_0 - b_0}{\theta}] \quad (5)$$

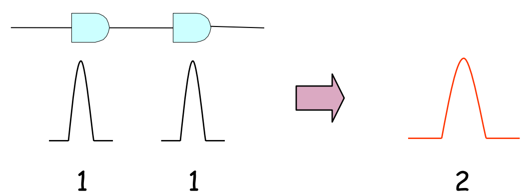
$$var[max(A, B)] = (\sigma_A^2 + a_0^2)T_A + (\sigma_B^2 + b_0^2)(1 - T_A) + (a_0 + b_0)\theta\phi(\frac{a_0 - b_0}{\theta}) - \{E[max(A, B)]\}^2 \quad (6)$$

Distribution Approximation

- Assume Gaussian distribution (can be represented by using mean and standard variation)

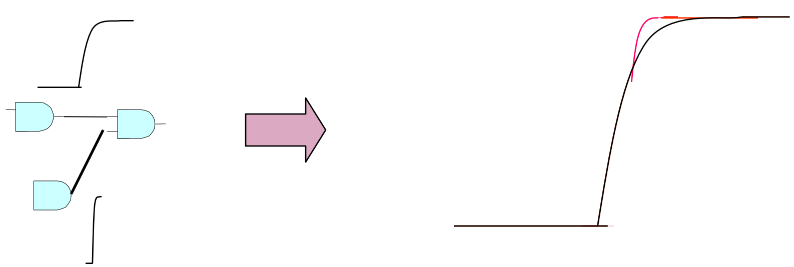
- Addition/Subtraction function results in Gaussian distribution as a solution

Probability density function
of Gaussian distribution



- Maximum/Minimum function approximation results in a new kind of distribution

Cumulative distribution function
of Gaussian distribution

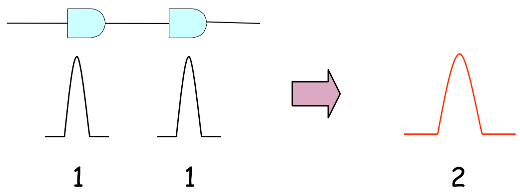


Distribution Approximation

- Assume Gaussian distribution (can be represented by using mean and standard variation)

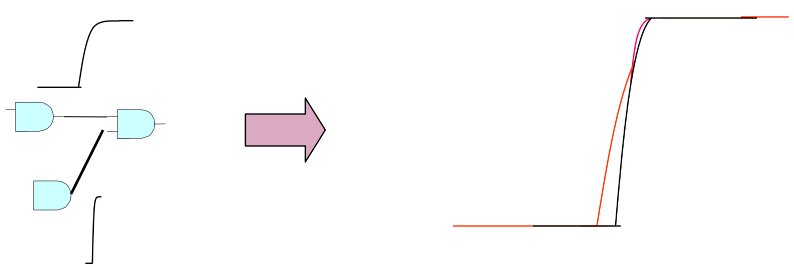
- Addition/Subtraction function can be computed using convolution

Probability density function of Gaussian distribution



- Maximum/Minimum function approximation results in a new kind of distribution

Cumulative distribution function of Gaussian distribution



Retiming Algorithm

- Retiming Algorithm
  - Can handle sequential circuits
- Bellman-Ford Algorithm
  - Can be modified to handle Statistical Timing Analysis
  - Faster comparing with other approaches

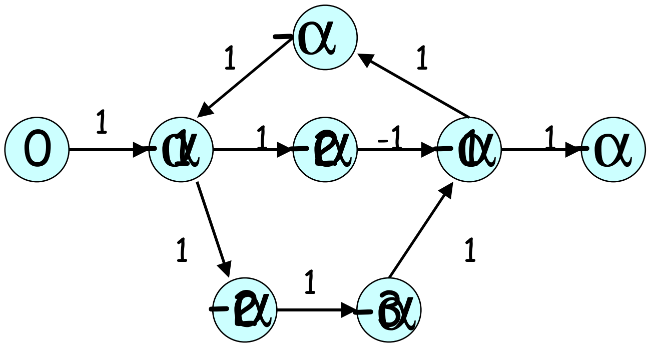
Longest Path Bellman-Ford Algorithm

Bellman-Ford Algorithm

input: directed graph (G, w, s)

output: longest path lengths from s

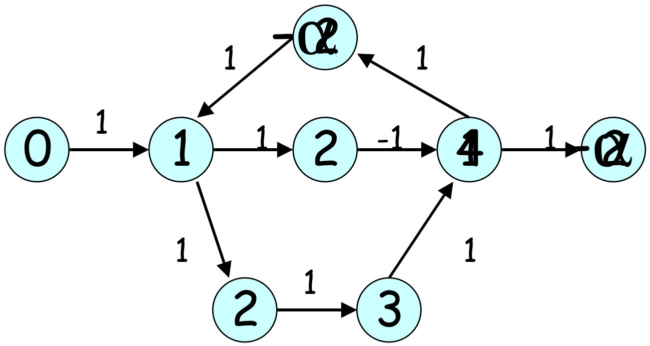
```
for (each v ∈ V)
    a[v] ← −∞;
a[s] ← 0;
iter ← 1;
while (STOP = FALSE and iter < |V|)
    iter ← iter + 1;
    STOP = TRUE;
    for (each edge (u, v) ∈ E)
        if (a[v] < a[u] + w(u, v))
            a[v] ← a[u] + w(u, v);
            STOP = FALSE;
for (each (u, v) ∈ E)
    if (a[v] < a[u] + w(u, v))
        return (FALSE);
return (TRUE);
```



Longest Path Bellman-Ford Algorithm

Bellman-Ford Algorithm
input: directed graph (G, w, s)
output: longest path lengths from s

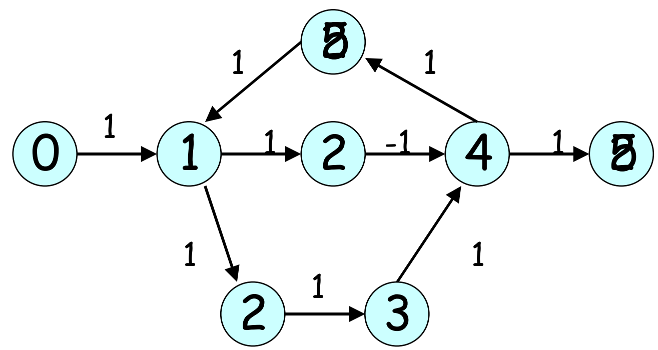
```
for (each v ∈ V)
    a[v] ← −∞;
a[s] ← 0;
iter ← 1;
while (STOP = FALSE and iter < |V|)
    iter ← iter + 1;
    STOP = TRUE;
    for (each edge (u, v) ∈ E)
        if (a[v] < a[u] + w(u, v))
            a[v] ← a[u] + w(u, v);
            STOP = FALSE;
for (each (u, v) ∈ E)
    if (a[v] < a[u] + w(u, v))
        return (FALSE);
return (TRUE);
```



Longest Path Bellman-Ford Algorithm

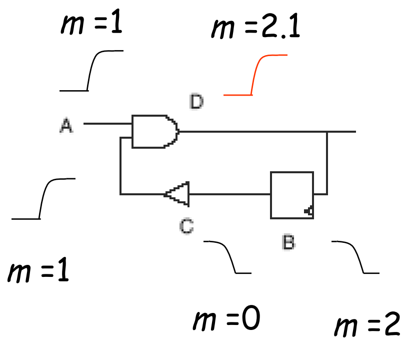
Bellman-Ford Algorithm
input: directed graph (G, w, s)
output: longest path lengths from s

```
for (each v ∈ V)
    a[v] ← −∞;
a[s] ← 0;
iter ← 1;
while (STOP = FALSE and iter < |V|)
    iter ← iter + 1;
    STOP = TRUE;
    for (each edge (u, v) ∈ E)
        if (a[v] < a[u] + w(u, v))
            a[v] ← a[u] + w(u, v);
            STOP = FALSE;
for (each (u, v) ∈ E)
    if (a[v] < a[u] + w(u, v))
        return (FALSE);
return (TRUE);
```

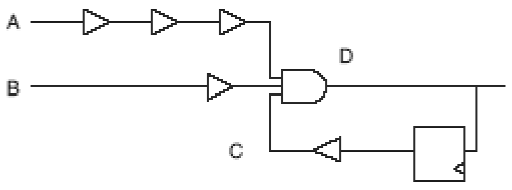


Problems of Bellman-Ford Update

- The approximation error can cause infinite update

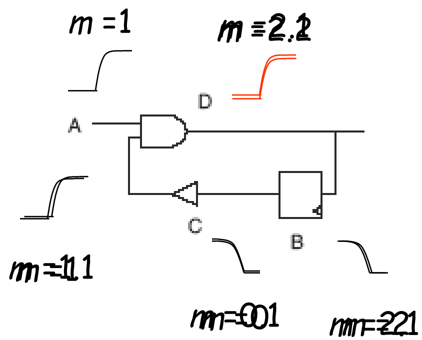


- If error bound technique is used, some paths can be ignored



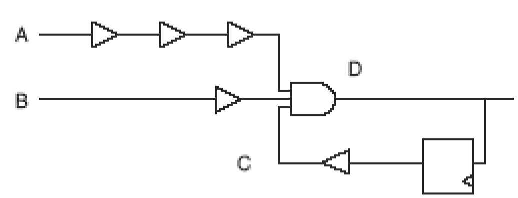
Problems of Bellman-Ford Update

- The approximation error can cause infinite update



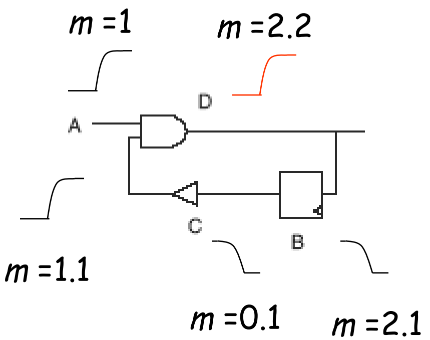
Error bound = 0.2

- If error bound technique is used, some paths can be ignored



Problems of Bellman-Ford Update

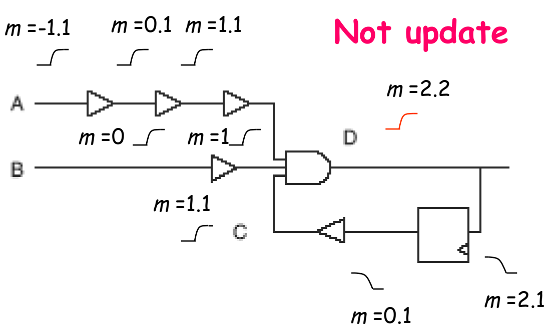
- The approximation error can cause infinite update



Error bound = 0.2

- If error bound technique is used, some paths can be ignored

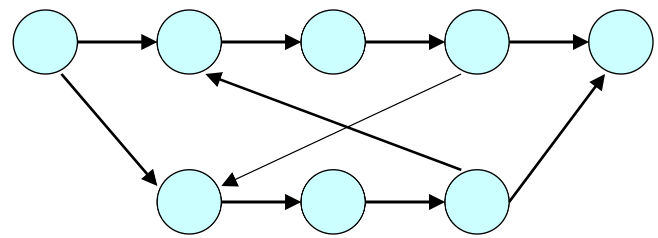
Error bound = 0.2



K-SBF Algorithm

K-bounded Statistical Bellman-Ford Algorithm
input: directed graph G with node/edge delay pdf
output: longest path length distribution from source

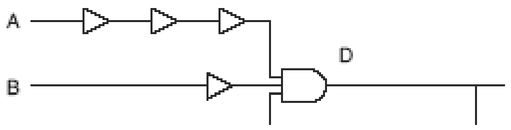
DFS(s) to find all backward edges;
for (each $v \in V$)
  if (backward edge connected to v)
    $BN \leftarrow v$;
$K = 0$;
for (each $v \in BN$)
  perform DFS(v);
  $k = |$backward nodes connected to $v|$;
  if ($K < k$)
    $K = k$;
for (each $v \in V$)
  $a[v] \leftarrow -\infty$;
$a[s] \leftarrow 0$;
for ($itr = 1$ to $k + 1$)
  for (each $v \in V$)
    $a[v] \leftarrow max_{u \in FI(v)}(a[u] + d(v) + d(u, v))$;
$\mathbf{P}(cycle) \leftarrow check\_pos\_cycle()$;
if ($\mathbf{P}(cycle) \leq \mathbf{P}a$)
  return (FALSE);
return (TRUE);



Backward node

Retiming Delay Computation

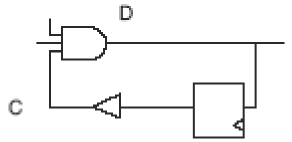
- Arrival time computation



$$\phi \geq a[t] = \max_{i=1,\ldots,K} \{\psi_i - \kappa_i \phi\}$$

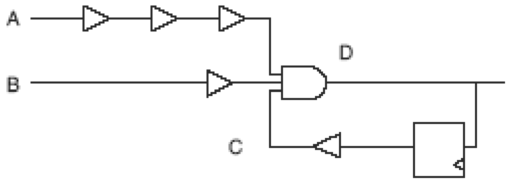
- Maximum cycle computation

$$\zeta_j = \xi_j - \sigma_j \phi \leq 0 \qquad j = 1\ldots, C.$$



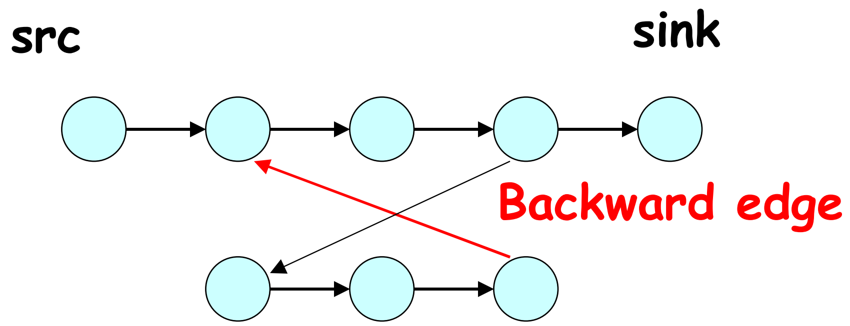
- Retiming delay computation

$$\phi = \max\left[\max_{i=1\ldots,K}\left\{\frac{\psi_i}{\kappa_i+1}\right\}, \max_{j=1\ldots,C}\left\{\frac{\xi_j}{\sigma_j}\right\}\right]$$



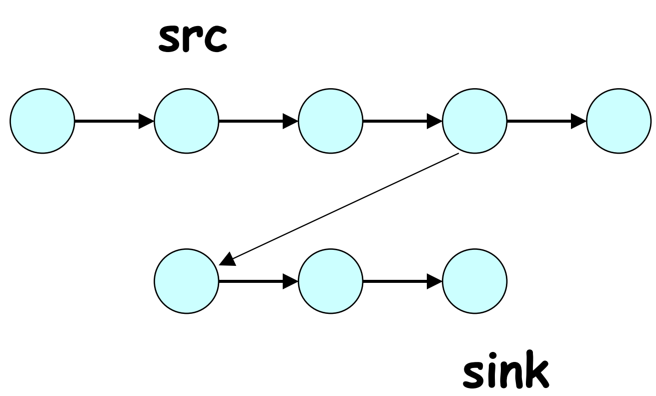
Algorithm for Maximum Cycle Distribution/Positive Cycle Detection

- Based on modified algorithm by Chen et al, ICCAD04

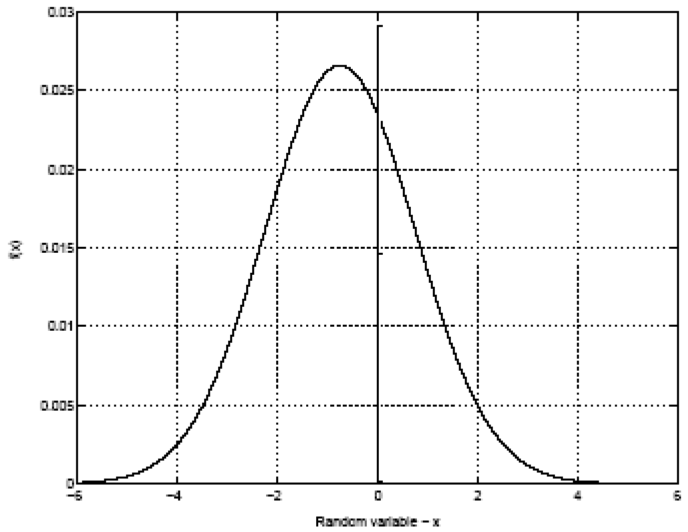


Algorithm for Maximum Cycle Distribution/Positive Cycle Detection

- Removing backward edge and compute cycle distribution from new src to new sink of the new graph



Positive Cycle Detection



mean (μ) + 3 · std. variation (σ) ≤ 0

Bound on Retiming Delay

- Definition

ϕ^l : the value of clock period when gate and interconnect are replaced by best case delay value

ϕ^m : the value of clock period when gate and interconnect are replaced by mean case delay value

ϕ^u: the value of clock period when gate and interconnect are replaced by worst case delay value

- $\phi^l \leq \phi \leq \phi^u$

- $\phi^m \leq E[\phi]$

Outline

- Introduction and Motivation
- Related Work
- Methodology
- Experimental Results
- Conclusions

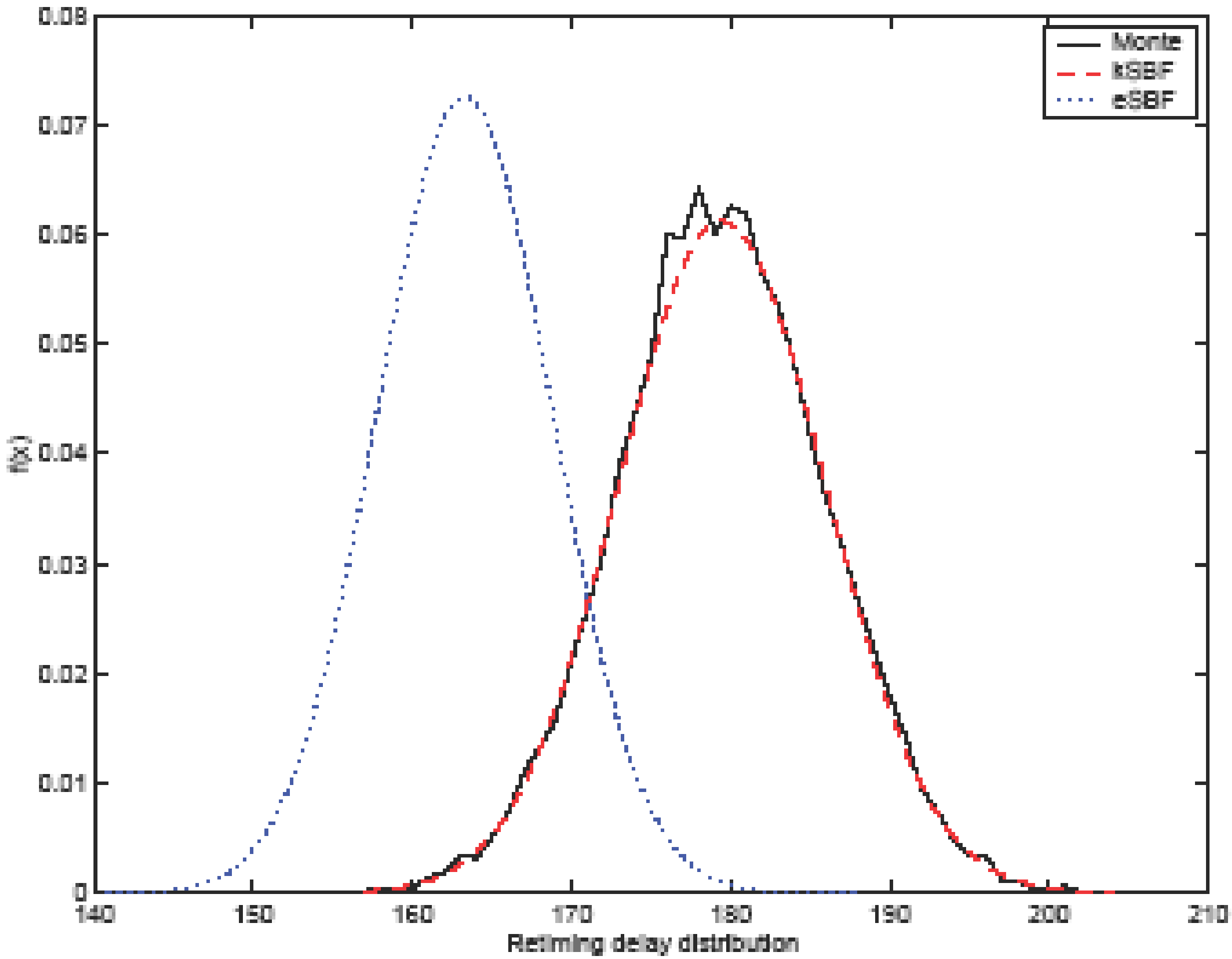
Sensitivity Parameters for SSTA

L	W	Tox	W	H	T
0.099	-0.099	-0.095	0.019	-0.133	-0.12

Benchmark Characteristics

ckt	gate	PI	PO	FF	$K+1$	b-node
s5378	2828	36	49	163	76	95
s9234	5597	36	39	211	239	354
s13207	8027	31	121	669	510	637
s15850	9786	14	87	597	495	699
s38417	22397	28	106	1636	1444	1660
s38584	19407	12	278	1452	1860	2054
b14o	5401	32	299	245	451	616
b15o	7092	37	519	449	988	1408
b20o	11979	32	512	490	1486	2197
b21o	12156	32	512	490	1511	2209
b22o	17351	32	725	703	1870	2770

Distribution Comparison on S5378



Monte = Monte-Carlo Simulation
eSBF = SBF with Error Bound

Distribution Comparison

ckt	Monte-Carlo		eSBF		kSBF	
	mean	std.dev.	mean	std.dev.	mean	std.dev.
s5378	179.65	6.27	163.29	5.49	179.47	6.51
s9234	229.24	16.18	164.58	7.14	225.53	10.51
s13207	304.93	13.27	279.47	8.72	305.76	13.05
s15850	363.16	15.83	317.32	7.57	363.37	15.72
s38417	187.11	4.53	184.95	5.28	189.00	8.41
s38584	437.02	19.41	399.62	10.45	436.75	19.33
b14o	161.19	10.70	117.42	5.37	160.43	5.31
b15o	247.00	10.00	212.72	18.45	247.88	10.23
b20o	259.68	9.03	229.16	6.12	267.13	9.33
b21o	267.57	6.18	240.01	4.08	267.98	9.24
b22o	286.63	18.92	300.49	25.17	320.62	15.26
runtime	**21 days**		**2.7 hours**		**3.7 hours**	

Worst Case Delay Comparison

ckt	Normal dly		Retiming dly	
	deter	stat	deter	stat
s5378	66.96	57.25	58	49.19
s9234	110.2	94.98	65	53.62
s13207	117.26	105.47	101	82.34
s15850	147.05	124.62	112	86.47
s38417	83.38	72.49	54	53.83
s38584	116.95	103.95	102	83.51
b14o	90.53	79.27	61	56.6
b15o	99.11	88.28	73	58.69
b20o	136.12	114.78	83	70.75
b21o	163.54	139.15	90	77.91
b22o	145.45	121.96	85	72.05
Avg.	**1**	**0.87**	**0.7**	**0.6**

Algorithm Summary

	STA	SSTA	RTA	SRTA
goal	computation of deterministic timing slack values in combinational circuits	computation of statistical timing slack distribution in combinational circuits	computation of deterministic timing slack values after retiming in sequential circuits	computation of statistical timing slack distribution after retiming in sequential circuits
delay values	deterministic	statistical distribution	deterministic	statistical distribution
retiming	no	no	yes	yes
circuit graph	directed acyclic graph, FF removed	directed acyclic graph, FF removed	cyclic retiming graph, FF becomes edge weight	cyclic retiming graph, FF becomes edge weight
basic algorithm	topological sort	topological sort, statistical min/max and arithmetic operation	Bellman-Ford	Bellman-Ford, statistical min/max and arithmetic operation
approach	visit nodes in forward (backward) topological order to compute arrival (require) time.	visit nodes in forward (backward) topological order to compute and propagate statistical arrival (require) time distribution.	compute longest path for cyclic graph with negative edge weights to compute timing slack after retiming.	compute "statistical longest path distribution" and "slack distribution after retiming" with statistical Bellman-Ford algorithm
complexity	$O(n)$	$O(n)$	$O(n^2)$, $O(k \cdot n)$ in practice	$O(n^2)$, $O(k \cdot n)$ in practice
advantage	simple and fastest	handle statistical analysis	model FFs and predict delay after retiming	perform statistical retiming and report delay distribution after retiming
disadvantage	can't handle retiming nor statistical variations	slow and no retiming consideration	slow and can't handle statistical variations	slow

Outline

- Introduction and Motivation
- Related Work
- Methodology
- Experimental Results
- Conclusions

Conclusions

- In deep submicron design, it is harder to control process parameters.
- Process variation will make timing analysis more difficult
- Here, we propose Statistical Bellman-Ford algorithm that can be used to compute longest path with cycle
- We show that our result is accurate comparing with Monte-Carlo simulation

QUESTIONS?