(*Project*) *Statistical* Static Timing Analysis

What you know

- **Static timing with deterministic mix/max delay numbers**

What you don't know

- **Static timing where everything is *nondeterministic***
- **…ie, all timing numbers are *probabilities***
- **Called *Statistical Static Timing Analysis*, or *Statistical-STA or SSTA***

This is the Project

- **You get to build one**
- **You get to use it inside an optimization loop**
- **You get ample opportunity to be as "cool" as you like – as usual**
- **You get to participate against the best; if you like**

Where Are We?

Project

- Statistical-STA

Logistics

- Project on website
- TAU Contest Deadline: Feb 1 initial code, Feb 15 final code
- Deadline 1: Functional code (no gate sizing/optimization) due Jan 25
- Deadline 2: Optimized code due Feb 1
  - Deadline 3: 10 min presentation on Feb 7!

Tentative Schedule

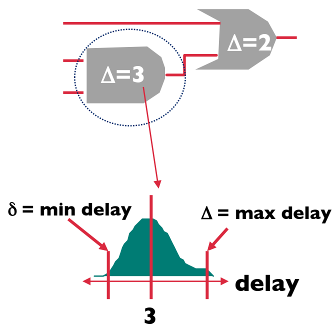
Week #	Topic	Assignment
1	Static Timing Analysis (STA)	Paper Review #1
2	Statistical Timing Analysis (SSTA)	Project
3	Technology Mapping	
4	Electrical Timing	
5	Project Presentations	
6	Placement – Simulated Annealing	
7	Placement – Partitioning	Paper Review #2
8	Placement – Direct	Homework #1
9	Routing I	
10	Routing II	
11	**Final Examination**	

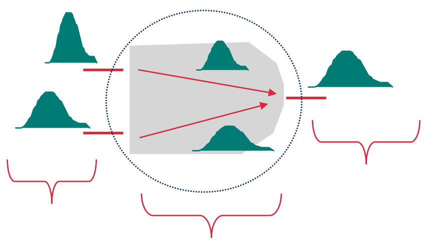
Functional Code

Circuit Optimization

What's the Central Problem?

- Delays are really *not* scalars; delay is a *distribution*
  - **All timing numbers are real *probability distributions* that give you a precise probability of the signal arriving with a given delay…**
  - **…and this distribution can still be a function of ALL usual factors: waveform slope, output loading, different delay per pin, etc.**
  - **Messy! Complicated!**





Why Does This Happen?

- CMOS scaling
  - **Everything is getting atomically small as feature sizes shrink**

- So…?
  - **So, as a % of the "nominal" value, the inevitable random manufacturing fluctuations are getting *bigger***

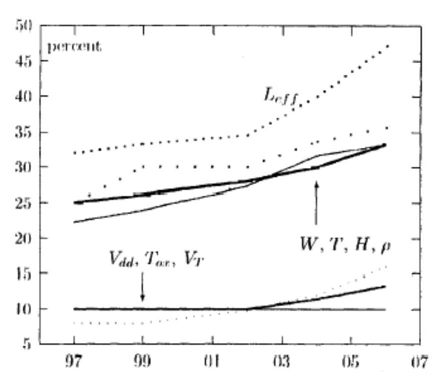
Scaling

Year	L_{eff} nm	T_{ox} nm	V_{DD} V	V_T V	W μ	H μ	ρ $\frac{m\Omega}{\square}$	L_{max}
1997	250	5	2.5	0.5	0.8	1.2	45	*2123*
1999	180	4.5	1.8	0.45	0.65	1.0	50	*1920*
2002	130	4	1.5	0.4	0.5	0.9	55	*1670*
2005	100	3.5	1.2	0.35	0.4	0.8	60	*1526*
2006	70	3	0.9	0.3	0.3	0.7	75	*1303*

3σ variations

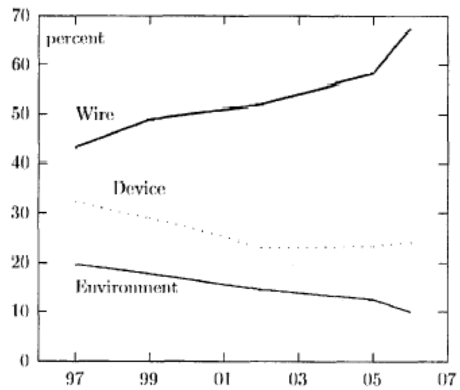
Year	L_{eff} nm	T_{ox} nm	V_T mV	W μ	H μ	ρ $\frac{m\Omega}{\square}$
1997	80	0.4	50	0.2	0.3	10
1999	60	0.36	45	0.17	0.3	12
2002	45	0.39	40	0.14	0.27	15
2005	40	0.42	40	0.12	0.27	19
2006	33	0.48	40	0.1	0.25	25

From [S. Nassif, "Modeling and Analysis of Manufacturing Variations", CICC 2001]



Things *Suck* All Over…

- Relative importance (in %) of contributions to variation from the Wires, from the Devices, and from the Environment



From [S. Nassif, "Modeling and Analysis of Manufacturing Variations", CICC 2001]

How Do We Deal With This?

- Stop pretending that our critical parameters are deterministic
  - **They're not**
- Try to model the statistical variations in some simple forms
  - **We need to fit these from measurements on fab lines**
- Trick is to get mathematical forms that are easy to manipulate, but not horribly inaccurate
  - **Not so easy – hot topic right now**
- Big idea from Nassif's paper
  - **Let's try to model everything as a sort of "first order Taylor series" expansion around the independent statistical variations**

$$Param\ P = P_0 + \frac{\partial p}{\partial x}\Delta x + \frac{\partial p}{\partial y}\Delta y + \ldots$$

nominal sensitivity variation

  - **Nassif writes it like this, in terms of independent environment, device, and wire**

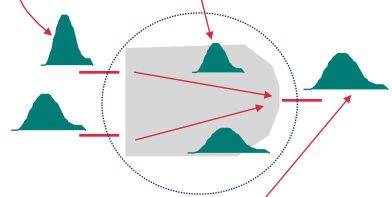
$$T = T_0 + \sum_{i=1}^{N_E} a_i E_i + \sum_{i=1}^{N_D} b_i D_i + \sum_{i=1}^{N_W} c_i W_i$$

So What Do We Want To Do? *Statistical-STA*

- Recast the core algorithms of STA
  - But now, all arrival times at the inputs of a gate…
  - … and all cell arc delays inside the gate…
  - …and the output delay of the final gate output…
  - …are statistical quantities in this sort of "Taylor series" form
- New problem
  - What's the mathematics of pushing these "Taylor series" representations thru the machinery of static timing?
  - What assumptions do we need?

$$arrival\ T = T_0 + \frac{\partial T}{\partial x}\Delta x + \frac{\partial T}{\partial y}\Delta y + \ldots$$

$$gate\ delay\ \ D = D_0 + \frac{\partial D}{\partial x}\Delta x + \frac{\partial D}{\partial y}\Delta y + \ldots$$



$$gate\ output\ \ T = T_0 + \frac{\partial T}{\partial x}\Delta x + \frac{\partial T}{\partial y}\Delta y + \ldots$$

Canonical Delay Model

- There are several formulations that all do more or less the same thing.
- We're picking the easiest one to understand
  - [C. Visweswariah, "First Order Parameterized Block Based Statistical Timing Analysis," TAU'04]
  - Uses the same idea as the Nassif formulation
- Canonical delay model

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

Canonical Delay Model: Example

- Suppose we have 2 delays, and n=2 in this example

$$A = \boxed{a_0} + \boxed{a_1\Delta x_1 + a_2\Delta x_2} + a_3 R_a$$
$$B = \boxed{b_0} + \boxed{b_1\Delta x_1 + b_2\Delta x_2} + b_3 R_b$$

$$\Delta x_1, \Delta x_2, R_a, R_b =$$

Important Facts

- A, B are statistically correlated
  - **That is – they are not independent statistical quantities**
  - **They *can't* be – they share the ΔX_i unit normal variations**
  - **This is the source of all kinds of grief in computing with these things**

$$A = a_0 + \sum_{i=1}^{n} a_i \Delta x_i + a_{n+1} R_a$$

- Reminders
  - **Mean is**
  - **Variance is**

Important Facts: Covariance

- Covariance measures correlation between random variables
  - **Two independent random vars A, B have a 0 covariance = cov(A,B)**
  - **Cov(A,B) = E[(A – E[A])(B - E[B])]**
  - **Cov(A,A) = var(A) = σ^2**

Share $\Delta x_1, \Delta x_2, \ldots, \Delta x_n$

- Cov(A, B) = ?

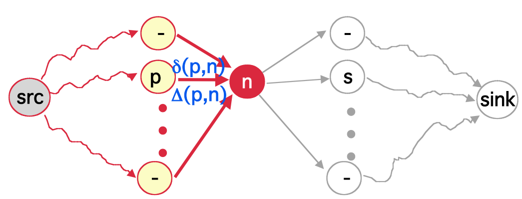
$$A = a_0 + \boxed{\sum_{i=1}^{n} a_i \Delta x_i} + a_{n+1} R_a \qquad B = b_0 + \boxed{\sum_{i=1}^{n} b_i \Delta x_i} + b_{n+1} R_b$$

=2x2 matrix

$$\begin{pmatrix} \mathrm{Cov(A,A)} & \mathrm{Cov(A,B)} \\ \mathrm{Cov(B,A)} & \mathrm{Cov(B,B)} \end{pmatrix} = \begin{bmatrix} \sum_{i=1}^{n+1} a_i^2 & \sum_{i=1}^{n} a_i b_i \\ \sum_{i=1}^{n} a_i b_i & \sum_{i=1}^{n+1} b_i^2 \end{bmatrix} = \begin{bmatrix} \sigma_a^2 & \rho\sigma_a\sigma_b \\ \rho\sigma_a\sigma_b & \sigma_b^2 \end{bmatrix}$$

What Do We Need For Statistical STA?

- ADD A+B, SUBTRACT A-B, MIN(A,B), MAX(A,B)
  - **For any A,B in this canonical statistical form**



$$AT_E(n) = \text{min delay to n} = \begin{cases} 0 \text{ if n == src} \\ \underset{p\,=\,pred(n)}{\mathrm{Min}} \{AT_E(p) + d(p,n)\} \end{cases}$$

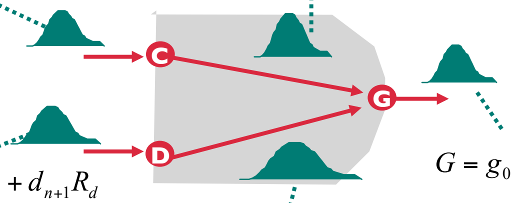
$$AT_L(n) = \text{max delay to n} = \begin{cases} 0 \text{ if n == src} \\ \underset{p\,=\,pred(n)}{\mathrm{Max}} \{AT_L(p) + \Delta(p,n)\} \end{cases}$$

Detailed Example – Just 2 Cell Arcs

- We know C, D, E, F … we want to compute G
- If scalar delays, easy; eg, for late mode: G = max[C+E, D+F]

$$C = c_0 + \sum_{i=1}^{n} c_i \Delta x_i + c_{n+1} R_c$$

$$E = e_0 + \sum_{i=1}^{n} e_i \Delta x_i + e_{n+1} R_e$$



$$D = d_0 + \sum_{i=1}^{n} d_i \Delta x_i + d_{n+1} R_d$$

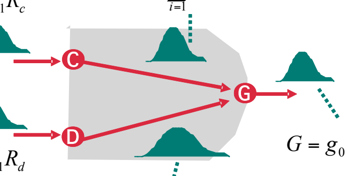
$$G = g_0 + \sum_{i=1}^{n} g_i \Delta x_i + g_{n+1} R_g$$

$$F = f_0 + \sum_{i=1}^{n} f_i \Delta x_i + f_{n+1} R_f$$

This Is What We Want

$$C = c_0 + \sum_{i=1}^{n} c_i \Delta x_i + c_{n+1} R_c$$

$$E = e_0 + \sum_{i=1}^{n} e_i \Delta x_i + e_{n+1} R_e$$



$$D = d_0 + \sum_{i=1}^{n} d_i \Delta x_i + d_{n+1} R_d$$

$$G = g_0 + \sum_{i=1}^{n} g_i \Delta x_i + g_{n+1} R_g$$

$$F = f_0 + \sum_{i=1}^{n} f_i \Delta x_i + f_{n+1} R_f$$

- Issues: how to ADD, how to MAX these canonical forms?

$$\max\Big[\; c_0 + \sum_{i=1}^{n} c_i \Delta x_i + c_{n+1} R_c \;+\; e_0 + \sum_{i=1}^{n} e_i \Delta x_i + e_{n+1} R_e,$$

$$d_0 + \sum_{i=1}^{n} d_i \Delta x_i + d_{n+1} R_d \;+\; f_0 + \sum_{i=1}^{n} f_i \Delta x_i + f_{n+1} R_f \;\Big]$$

Arrival Times   Cell Arc Delays

ADD (and SUB) Turn Out To Be Easy

- *Almost* just term by term ADD (or SUB)…
- … except last "local" var

$$\max[\quad (c_0 + d_0) + \sum_{i=1}^{n} (c_i + d_i)\Delta x_i + \sqrt{c_{n+1}^2 + d_{n+1}^2}\, R_a \quad ,$$
$$(e_0 + f_0) + \sum_{i=1}^{n} (e_i + f_i)\Delta x_i + \sqrt{e_{n+1}^2 + f_{n+1}^2}\, R_b \quad]$$

$$= \max[\quad a_0 + \sum_{i=1}^{n} a_i \Delta x_i + a_{n+1} R_a \quad ,$$
$$b_0 + \sum_{i=1}^{n} b_i \Delta x_i + b_{n+1} R_b \quad]$$

$$= g_0 + \sum_{i=1}^{n} g_i \Delta x_i + g_{n+1} R_g$$

MAX (or MIN) Are A Pain

- Lots of "deep" stats to make this work; for us, just formulas

$$\begin{bmatrix} \sum_{i=1}^{n+1} a_i^2 & \sum_{i=1}^{n} a_i b_i \\ \sum_{i=1}^{n} a_i b_i & \sum_{i=1}^{n+1} b_i^2 \end{bmatrix} = \begin{bmatrix} \sigma_a^2 & \rho\sigma_a\sigma_b \\ \rho\sigma_a\sigma_b & \sigma_b^2 \end{bmatrix}$$

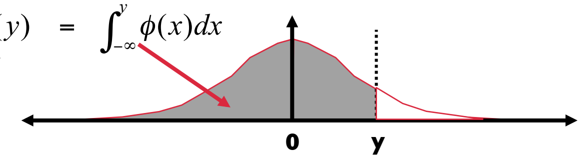
$$\phi(x) = \tfrac{1}{\sqrt{2\pi}} \exp(-x^2/2)$$
$$\Phi(y) = \int_{-\infty}^{y} \phi(x)\,dx$$
$$\theta = \sqrt{(\sigma_a^2 + \sigma_b^2 - 2\rho\sigma_a\sigma_b)}$$

$$T_A = \Phi\left(\frac{a_0 - b_0}{\theta}\right)$$

Aside: "erf" Gotchas

- Want this: $\Phi(y) \;=\; \int_{-\infty}^{y} \phi(x)dx$

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{x^2/2}$$

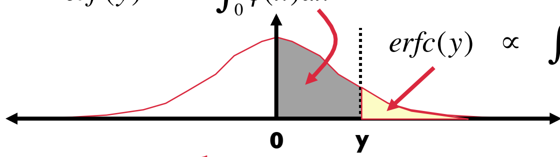


- Often get "erf(y)" and "erfc(y)" in math software

$$erf(y) \;\propto\; \int_0^y \phi(x)dx$$

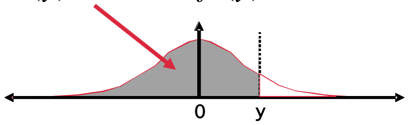
$$erf(y) = \frac{2}{\pi}\int_0^y e^{-t^2}dt$$

$$erfc(y) \;\propto\; \int_y^{\infty} \phi(x)dx$$

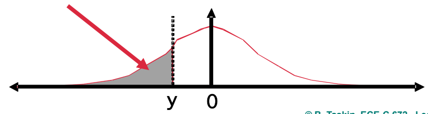


- Need to twiddle some to get what you want

$$\Phi(y) \;\propto\; 1 - erfc(y)$$



$$\Phi(y) \;\propto\; erfc(-y)$$



So, MAX(A,B) = …?

$$= \max[\;\; a_0 + \sum_{i=1}^{n} a_i \Delta x_i + a_{n+1}R_a,\quad b_0 + \sum_{i=1}^{n} b_i \Delta x_i + b_{n+1}R_b\;]$$

$$= g_0 + \sum_{i=1}^{n} g_i \Delta x_i + g_{n+1}R_g$$

Canonical form

New local random var

$$g_0 = E[\max(A,B)] = a_0 T_A + b_0(1-T_A) + \theta\phi\left(\frac{a_0 - b_0}{\theta}\right)$$

= mean

$$g_i = T_A a_i + (1-T_A)b_i,\quad for\; i = 1,2,\ldots n$$

= shared variations
= sensitivity terms

- ..but we're not done yet, there's one more term…

…But What About g_{i+1}?

$$= \max[\quad a_0 + \sum_{i=1}^{n} a_i \Delta x_i + a_{n+1} R_a, \quad b_0 + \sum_{i=1}^{n} b_i \Delta x_i + b_{n+1} R_b \,]$$

$$= g_0 + \sum_{i=1}^{n} g_i \Delta x_i + g_{n+1} R_g$$

$$\mathrm{var}[\max(A,B)] \quad = \sum_{i=1}^{n+1} g_i^2 \qquad = (\sigma_a^2 + a_0^2) T_A + (\sigma_b^2 + b_0^2)(1 - T_A)$$

$$+ (a_0 + b_0)\theta\phi\left(\frac{a_0 - b_0}{\theta}\right) - E[\max(A,B)]^2$$

$$\mathrm{var}[\max(A,B)] \quad = \sum_{i=1}^{n} g_i^2 + g_{n+1}^2 \quad \Rightarrow \boxed{g_{n+1} = \sqrt{\mathrm{var}(\max[A,B]) - \sum_{i=1}^{n} g_i^2}}$$

- **Idea: we "match" the right variance of the final max(A,B) result by tweaking spread on this final "local" var term R_g**

Aside: MIN(A,B) = …?

$$= \min[\quad a_0 + \sum_{i=1}^{n} a_i \Delta x_i + a_{n+1} R_a, \quad b_0 + \sum_{i=1}^{n} b_i \Delta x_i + b_{n+1} R_b \,]$$

$$= g_0 + \sum_{i=1}^{n} g_i \Delta x_i + g_{n+1} R_g$$

subtract

$$g_0 = E[\min(A,B)] = a_0 \boxed{(1 - T_A)} + b_0 \boxed{T_A} - \theta\phi\left(\frac{a_0 - b_0}{\theta}\right)$$

$$g_i = \boxed{(1 - T_A)} a_i + \boxed{T_A} b_i, \quad for\; i = 1,2,\ldots n$$

$$\mathrm{var}[\min(A,B)] \quad = \sum_{i=1}^{n+1} g_i^2 \qquad = (\sigma_a^2 + a_0^2)\boxed{(1 - T_A)} + (\sigma_b^2 + b_0^2)\boxed{T_A}$$

subtract

$$- (a_0 + b_0)\theta\phi\left(\frac{a_0 - b_0}{\theta}\right) - E[\boxed{\min}(A,B)]^2$$

$$\mathrm{var}[\min(A,B)] \quad = \sum_{i=1}^{n} g_i^2 + g_{n+1}^2 \quad \Rightarrow g_{n+1} = \sqrt{\mathrm{var}(\boxed{\min}[A,B]) - \sum_{i=1}^{n} g_i^2}$$

More Gotchas: MAX(A,B,C…?)

- To do MAX (or MIN) or more than 2 things… how?

- Simple solution: just do computation in repeated pairs
  - **MAX(A,B,C) = MAX(MAX(A,B), C)**
  - **MIN(A,B,C) = MIN(MIN(A,B), C)**
  - **Etc for more than 3 inputs to these**

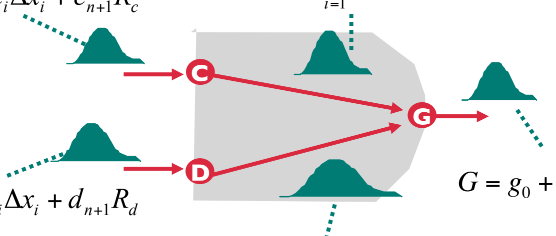
- Does the order matter? Experiments say “yes”
  - **Pick the pair with the smallest 2 means (eg, a_0, b_0 values) and do the MAX( ) on this pair first**
  - **Then pick the next smallest remaining mean, and do the next MAX( )**
  - **…etc. Same deal for MIN( )**

So – We Can Really *Do* This

- Given C, D, E, F … we can compute G, in this standard form
  - **Can compute any early mode or late mode timing distrib you want**
  - **Only need ADD, SUB, MAX, MIN ops on these statistical quantities**

$$C = c_0 + \sum_{i=1}^{n} c_i \Delta x_i + c_{n+1} R_c$$

$$E = e_0 + \sum_{i=1}^{n} e_i \Delta x_i + e_{n+1} R_e$$



$$D = d_0 + \sum_{i=1}^{n} d_i \Delta x_i + d_{n+1} R_d$$

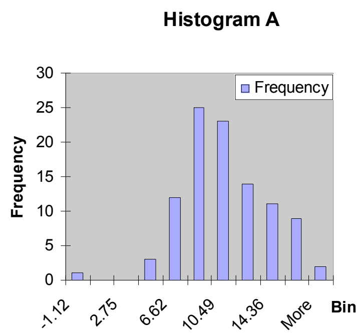
$$G = g_0 + \sum_{i=1}^{n} g_i \Delta x_i + g_{n+1} R_g$$

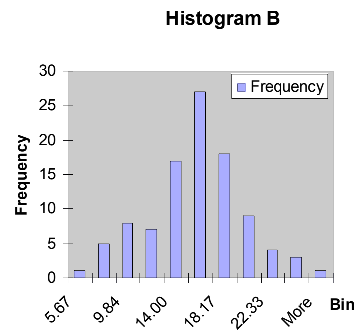
$$F = f_0 + \sum_{i=1}^{n} f_i \Delta x_i + f_{n+1} R_f$$

Trivial Example (I'll Post The .xl Spreadsheet)

$$A = a0 + a_1\Delta x_1 + a_2\Delta x_2 + a_3 R_a$$
$$= 10 + 1\Delta x_1 + 2\Delta x_2 + 3R_a$$

$$B = b_0 + b_1\Delta x_1 + b_2\Delta x_2 + b_3 R_b$$
$$= 15 + 2\Delta x_1 + 3\Delta x_2 + 2R_b$$





Based on 100 independent zero-mean unit-variance normal N(0,1) random numbers generated for each of Δx_1, Δx_2, R_a, R_b

Trivial Example

Actual distrib of max(A,B) computed over the 100 random A, B numbers we computed

Mean 15.10188267

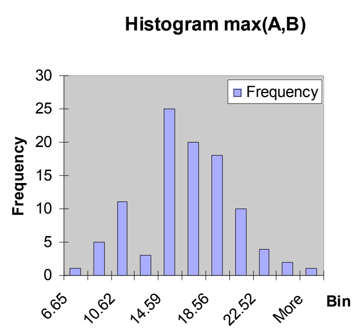
Var 15.11166587

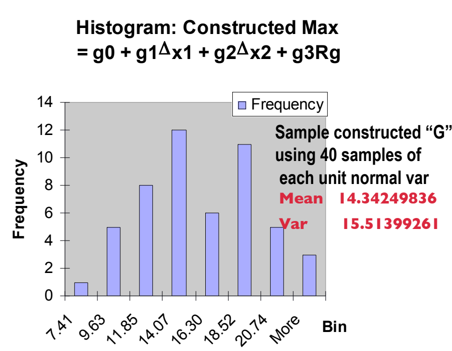
Compute max(A,B) in std form

$$= g_0 + g_1\Delta x_1 + g_2\Delta x_2 + g_3 R_g$$
$$= 15.33 + 1.93\Delta x_1 + 2.93\Delta x_2 + 1.64 R_g$$

E max(A,B) 15.33205045

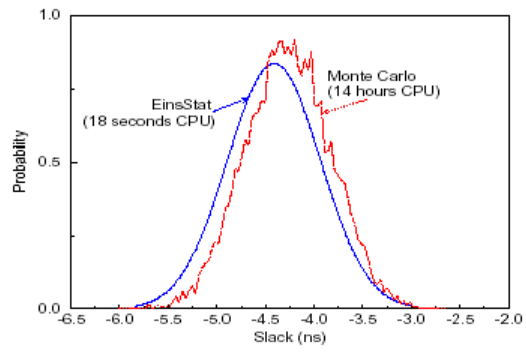
var max(A,B) 15.02582274





Works Well In Real Use: Fast, Decent Accuracy

- From [Visweswariah, TAU'04] paper
  - **Monte Carlo sampling of their conventional STA tool, EinsTimer**
  - **Versus the statistical version of the tool, EinsStat**
  - **3042 logic gates, longest path is about 60 gates (and wires) deep**



Project : Statistical STA

- Executive summary

Build One

(Let us know how it worked)

Project : Statistical STA

Helpful Explanatory Details

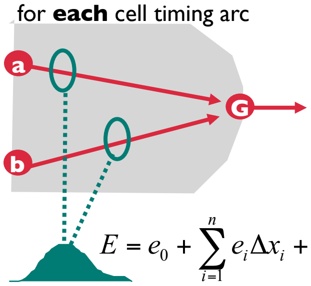
- **Input: Netlist file**
  - **A file with a gate-level netlist in a simple, standard format**
  - **File will tell you primary inputs, gates, wires, primary outputs**
- **Input: Stats file**
  - **A file in a simple format that tells you the "canonical form" for all the statistical quantities you need for your netlist**
  - **Arrival times for the primary inputs**
  - **Delay times for the cell arcs in the gates**
  - **To make life simple: assume all cell arcs in a gate are same distrib**
  - **…but, to make life interesting…**

Project : Statistical STA

Helpful Explanatory Details

- **Input: Stats file (cont)**
  - **To make life interesting…**
  - **Each gate is allowed to come in several flavors with a different statistical model, ie, with different numbers in the stats model**

Same delay distrib
for **each** cell timing arc



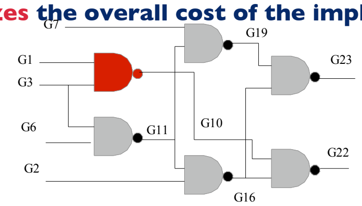
$$E = e_0 + \sum_{i=1}^{n} e_i \Delta x_i + e_{n+1} R_e$$

But we can supply several different models
for each gate, each with a unique $cost val

Model	$$	e0	e1	e2	e3
1	$1	10	1.1	2.2	3.3
2	$1.3	12	1.4	2.8	4.1
…	$…	…	…	…	…

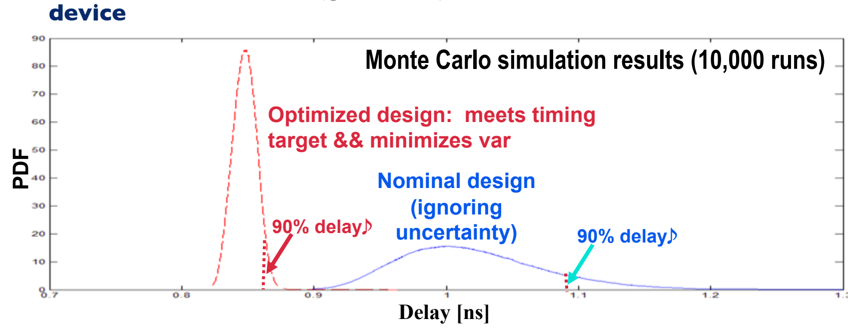
Project : Statistical STA

- What do you have to do?
- Basic requirements
  1. **Using "model #1" for each gate, compute late mode arrival time (AT_L) in the standard statistical form for the primary output**
  2. **And compute the "cost" of this implementation (trivial)**
  3. **Then optimize this implementation, so that it**
     1. **Meets a specific target for the mean of the one primary output**
        ie, we want E(output arrival) < T_{req}
     2. **Minimizes the variance of the primary output arrival**
     3. **Minimizes the overall cost of the implementation (ie, $$/gate)**



Variability Optimization: Optimization of Circuit Delay Under Uncertainty

- This is a hot research topic right now. Example from Stephen Boyd at Stanford, done at transistor rather than gate level
  - **32-bit adder: 1714 devices, 6400 paths**
  - **Design variables: 1714 device sizes (W/L)**
  - **Variation model:** $\sigma \propto (\text{gate area})^{-1/2}$ **= 10% σ(delay) for min device**



Project : Optimizing Variance & Cost??

- Option 1: Simulated annealing
- Roughly speaking

```
While (result still getting better) {
    Pick a random gate
    Flip it to a different model
    Rerun Statistical-STA
    Update a suitable cost
      function that measures
      "goodness" of this result
    If (this is worth keeping)
        Keep it
    Else ( put back old model)
}
```

- Pro: pretty easy to do overall, prob get very good results
- Con: slow as a dog, most likely

- Option 2: Probably critical paths
- Roughly speaking
  - Visweswariah discusses how to compute path "criticalities" that are probabilities
  - Get concepts similar to AT and RAT and slack, but now it's the probability that an edge is on a critical path, etc
  - Enumerate "probably critical" paths, look at most critical nodes, and heuristically try to flip them to the gate model that improves the overall result the most
  - Repeat till it doesn't get any better
- Pro: probably pretty fast
- Con: not obvious what right set of "pick/flip" ideas are for the gates

Project : Beat the World

- Option 3: TAU 2013 Contest
- See https://sites.google.com/site/taucontest2013/

TAU 2013 contest: Variation aware timing analysis

TAU 2013 invites you to participate in a software tool development contest on variation aware parametric timing analysis. With the increasing number and significance of manufacturing and environmental sources of variability in the design and use of modern VLSI designs, timing analysis considering the uncertainty due to variability is essential. At the same time, the growing complexity of modern designs require fast timing analysis, thereby requiring tools to adopt accuracy and run-time tradeoffs. The goals of this contest are the following.

- Motivate university level students to learn about modern VLSI design timing analysis in the presence of variability and encourage research in this area
- Provide insight to some challenging aspects of a "fast" parametric (e.g. statistical) static timing analysis tool, and look for novel solutions, the results of which may be interesting to both industry and academia
- Encourage use of parallel techniques for timing analysis (especially multi-threaded techniques)
- Facilitate creation of a public university level variation aware timer (would serve as a framework for contests in future)

Contestants would be asked to additionally provide a presentation (.ppt or .pdf) describing key novel contributions of their tool. The top three contestants would be invited to present their work as part of the TAU-20 talks. *Look out for details on "surprise" awards for the top contestants on the website soon!*

Important dates

- Contest announced: October 12, 2012
- Early binaries due (for verification/initial feedback): February 1, 2013
- Final version of binaries due with 2 page document describing key novel ideas: **February 15, 2013**
- Results announced: March 27, 2013 (at TAU workshop)

- Pro: some pre-written code, people are paying attention!
- Con: Slightly different format than those on these slides, synchronous circuit extension; people are paying attention!

Project : Statistical STA Grading [110pts]

- [50 pts] Writeup
  - What did you do, and why? What algorithms and heuristic choice? What data structures?
  - What tricks – if any – for performance, efficiency, etc?
  - Not more than ~10 pages, not less than 4 pages, pdf format
- [30 pts] Public benchmarks
  - We will select a set of small benchmarks, all with one single primary output
  - First: just put model #1 in for all gates, Tell us your nums for final distrib, cost
  - Second: optimize cost and variance while hitting a specified time for mean of primary output
- [10 pts] Private benchmarks
  - I will select a set of small private benchmarks, all with one single primary output
  - And we will run your code on and see what it can do
  - Aim for dunx1
  - Windows, MacOS binaries OK. You still need to turn in source code.
  - Your responsibility to provide logistics
- [20 pts] Coolness
  - Too many options…
  - Contest entries are very cool

Project : Statistical STA Grading

- Coolness [20pts]
  - As always, more capacity, more speed, more accuracy
  - Can handle more than 1 primary output– we'll make a few benchmarks
  - *Incremental* static timing – being able to rapidly flip one gate to a different model, and then update overall timing.
    - You can find some papers (from IBM etc.) about how to do it.
    - Not mandatory to do it. It's tricky, but you'll see huge speedups
  - Clever heuristics for doing the overall optimization
    - Simulated annealing will work fine, it just won't scale to big designs or be fast. It's the low-risk choice.
    - Heuristics based on "find worst gate & improve it" will be faster – but riskier. This is still an open area of research in the CAD biz.
  - Nice graphics
    - These things make nice histograms, if you can plot them automatically for us, that's cool. So is doing your own Monte

Project Deadlines

- Deadline 1 (See early slides)
  - Slowest, cheapest circuit--- Non-optimized project ready
  - Fill in the table and submit

Bench mark	Critical Path	Critical Path Delay	Cost ($)	Run Time (s)
Cct 1	Input->gate1->…->gate n-> output	$g_0\ g_1\ g_2\ g_R$	X	Y
Cct 2	Input->gate1->…->gate n-> output	$i_0\ i_1\ i_2\ i_R$	W	Z
…	…	…		

- Deadline 2
  - Optimize for fastest (and cheapest) circuit---Project Report due
- Deadline 3
  - 5-10 minute presentations describing optimization techniques

Project Summary

- Big picture
  - You get to build a basic STA engine
  - Groups of two possible
  - You will do late-mode ATs using statistical distributions in a simple, standard form
  - This is mainly bookkeeping and some nasty eqns
  - After you get this up, you get to try to optimize some (small) netlists
- What's next
  - Next lecture(s) after this one are on simulated annealing
  - You can contribute to the generation of stat files with your bright ideas for coolness points