Circuit Design via Geometric Programming

Stephen Boyd Seung Jean Kim S. S. Mohan
Mark Horowitz Dinesh Patil

ICCAD Tutorial 11/11/04

Outline

- Basic approach
- Geometric programming & generalized geometric programming
- Digital circuit design applications
- Analog circuit design applications
- Conclusions

Basic Approach

Basic approach

1. formulate circuit design problem as **geometric program** (GP), an optimization problem with special form

2. solve GP using specialized, tailored method

- this tutorial focuses on step 1 (a.k.a. **GP modeling**)
- step 2 is **technology**

Why?

- we can solve even large GPs very effectively, using recently developed methods
- so once we have a GP formulation, we can solve circuit design problem effectively

we will see that

- GP is especially good at handling a large number of concurrent constraints
- GP formulation is useful even when it is approximate

Trade-offs in optimization

- general trade-off between **generality** and **effectiveness**
- generality
  - number of problems that can be handled
  - accuracy of formulation
  - ease of formulation
- effectiveness
  - speed of solution, scale of problems that can be handled
  - global vs. local solutions
  - reliability, baby-sitting, starting point

Example: least-squares vs. simulated annealing

least-squares

- large problems reliably (globally) solved quickly
- no initial point, no algorithm parameter tuning
- solves very restricted problem form
- with tricks and extensions, basis of vast number of methods that work (control, filtering, regression, . . .)

simulated annealing

- can be applied to any problem (more or less)
- slow, needs tuning, babysitting; not global in practice
- method of choice for some problems you can't handle any other way

Where GP fits in

somewhere in between, closer to least-squares . . .

- like least-squares, large problems can be solved reliably (globally), no starting point, tuning, . . .
- solves a class of problems broader than least-squares, less general than simulated annealing
- **formulation takes effort, but is fun and has high payoff**

Geometric Programming

Monomial & posynomial functions

$x = (x_1, \ldots, x_n)$: vector of positive optimization variables

- function g of form

$$g(x) = cx_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n},$$

  with $c > 0$, $\alpha_i \in \mathbf{R}$, is called **monomial**

- sum of monomials, $i.e.$, function f of form

$$f(x) = \sum_{k=1}^{t} c_k x_1^{\alpha_{1k}} x_2^{\alpha_{2k}} \cdots x_n^{\alpha_{nk}},$$

  with $c_k > 0$, $\alpha_{ik} \in \mathbf{R}$, is called **posynomial**

Examples

with x, y, z variables,

- 0.23, $2z\sqrt{x/y}$, $3x^2y^{-.12}z$ are monomials (hence also posynomials)
- $0.23 + x/y$, $2(1+xy)^3$, $2x+3y+2z$ are posynomials
- $2x+3y-2z$, $x^2+\tan x$ are neither

Geometric program (GP)

a special form of optimization problem:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 1, \quad i = 1, \ldots, m \\ & g_i(x) = 1, \quad i = 1, \ldots, p \end{array}$$

f_i are posynomials and g_i are monomials

- a highly nonlinear constrained optimization problem
- **but, can be solved extremely efficiently**
  - dense 1000 vbles, 10000 constraints: one minute on PC
  - sparse 1M vbles, 10M constraints: one hour on PC

Example

$$
\begin{array}{ll}
\text{minimize} & x^{-1}y \\
\text{subject to} & 2x^{-1} \leq 1, \\
& (1/3)x \leq 1, \\
& x^2y^{-1/2} + 3y^{1/2}z^{-1} \leq 1, \\
& xy^{-1}z^{-2} = 1
\end{array}
$$

- this one could be solved by hand, or by sweeping values of x, y, and z
- but a GP with 1000 variables (which is easily solved if you know how) cannot be solved by hand or sweeping

Posynomial and monomial algebra

- monomials closed under products, division, positive scaling, powers (hence, inverse), $e.g.$,

$$\left(2x^{-0.2}y^{1.1}\right)\left(0.3xy^{-0.3}z^{2}\right) = 0.6x^{0.8}y^{0.8}z^{2}$$

- posynomials closed under sums, products, positive scaling, division by monomials, positive integer powers

Simple GP extensions

- maximizing a monomial objective g
  - same as minimizing g^{-1}, a monomial (hence also posynomial)
- monomial-monomial equality constraint $g_1 = g_2$
  - same as monomial equality constraint $g_1/g_2 = 1$
- posynomial-monomial inequality constraint $f \leq g$
  - same as posynomial inequality constraint $f/g \leq 1$

Example

- maximize volume of box with width w, height h, depth d
- subject to limits on wall and floor areas, aspect ratios h/w, d/w

$$\begin{array}{ll} \text{maximize} & hwd \\ \text{subject to} & 2(hw+hd) \leq A_{\text{wall}}, \quad wd \leq A_{\text{flr}} \\ & \alpha \leq h/w \leq \beta, \quad \gamma \leq d/w \leq \delta \end{array}$$

in standard GP form:

$$\begin{array}{ll} \text{minimize} & h^{-1}w^{-1}d^{-1} \\ \text{subject to} & (2/A_{\text{wall}})hw + (2/A_{\text{wall}})hd \leq 1, \quad (1/A_{\text{flr}})wd \leq 1 \\ & \alpha h^{-1}w \leq 1, \quad (1/\beta)hw^{-1} \leq 1 \\ & \gamma wd^{-1} \leq 1, \quad (1/\delta)w^{-1}d \leq 1 \end{array}$$

Trade-off analysis

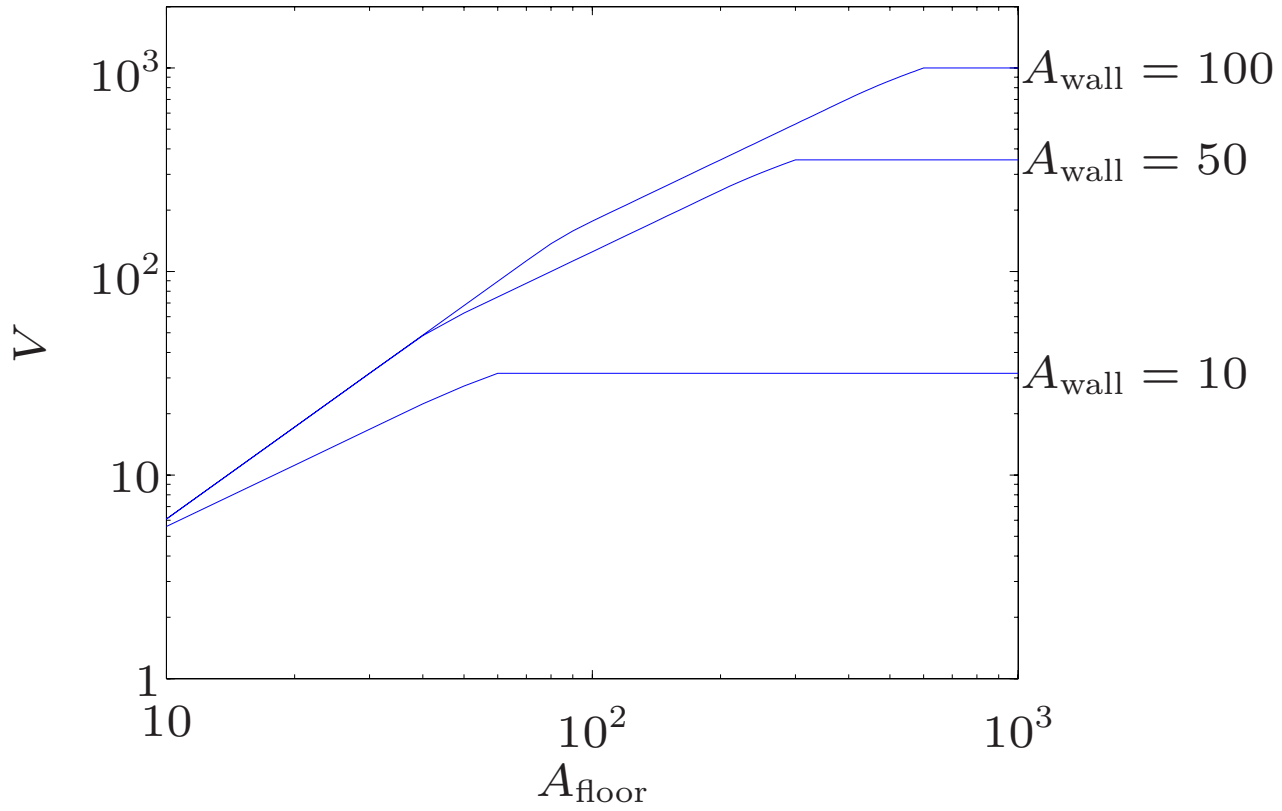
(no equality constraints, for simplicity)

- form perturbed version of original GP, with changed righthand sides:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq u_i, \quad i = 1, \ldots, m \end{array}$$

- $u_i > 1$ ($u_i < 1$) means ith constraint is relaxed (tightened)
- let $p(u)$ be optimal value of perturbed problem
- plot of p vs. u is (globally) **optimal trade-off surface** (of objective against constraints)

Trade-off curves for maximum volume box example



- maximum volume V vs. A_{flr}, for $A_{\text{wall}} = 10,\ 50,\ 100$
- h/w, d/w aspect ratio limits 0.5, 2

Sensitivity analysis

- optimal sensitivity of ith constraint is

$$S_i = \left. \frac{\partial p/p}{\partial u_i/u_i} \right|_{u=1}$$

- S_i predicts fractional change in optimal objective value if ith constraint is (slightly) relaxed or tightened

- very useful in practice; give quantitative measure of how tight a binding constraint is

- when we solve a GP **we get all optimal sensitivities at no extra cost**

Example

- minimize circuit delay, subject to power, area constraints (details later)

$$\begin{array}{ll} \text{minimize} & D(x) \\ \text{subject to} & P(x) \leq P^{\max}, \quad A(x) \leq A^{\max} \end{array}$$

- both constraints tight at optimal $x^\star$: $P(x^\star) = P^{\max}$, $A(x^\star) = A^{\max}$
- suppose optimal sensitivities are $S^{\text{pwr}} = -2.1$, $S^{\text{area}} = -0.3$
- we predict:
  - for 1% increase in allowed power, optimal delay decreases 2.1%
  - for 1% increase in allowed area, optimal delay decreases 0.3%

How GPs are solved

the practical answer: **none of your business**

more politely: **you don't need to know**

it's **technology:**

- good algorithms are known
- good software implementations are available

How GPs are solved

- work with log of variables: $y_i = \log x_i$
- take log of monomials/posynomials to get

$$\begin{array}{ll} \text{minimize} & \log f_0(e^y) \\ \text{subject to} & \log f_i(e^y) \leq 0, \quad i = 1, \ldots, m \\ & \log g_i(e^y) = 0, \quad i = 1, \ldots, p \end{array}$$

- $\log f_i(e^y)$ are **convex** functions
- $\log g_i(e^y)$ are affine functions, $i.e.$, linear plus a constant
- solve (nonlinear) **convex optimization problem** above using interior-point method

Current state of the art

- basic interior-point method that exploits sparsity, generic GP structure
- approaching efficiency of linear programming solver
  - sparse 1000 vbles, 10000 monomial terms: few seconds
  - sparse 10000 vbles, 100000 monomial terms: minute
  - sparse 10^6 vbles, 10^7 monomial terms: hour

  (these are order-of-magnitude estimates, on simple PC)

History

- GP (and term 'posynomial') introduced in 1967 by Duffin, Peterson, Zener

- engineering applications from the very beginning
  - early applications in chemical, mechanical, power engineering
  - digital circuit transistor and wire sizing with Elmore delay since 1984 (Fishburn & Dunlap's TILOS)
  - analog circuit design since 1997 (Hershenson, Boyd, Lee)
  - other applications in finance, wireless power control, statistics, . . .

- extremely efficient solution methods since 1994 or so (Nesterov & Nemirovsky)

Generalized Geometric Programming

Handling positive fractional powers

- suppose f_1, f_2 are posynomials
- we can handle $f_1 + f_2^3 \leq 1$ directly, since LHS is posynomial
- we can't handle $f_1 + f_2^{3.1} \leq 1$, since $f_2^{3.1}$ isn't posynomial
- **trick:** replace inequality $f_1 + f_2^{3.1} \leq 1$ with two (posy) inequalities

$$f_1 + t^{3.1} \leq 1, \qquad f_2 \leq t$$

  t is new variable (called dummy or slack)

Handling maximum

- suppose f_1, f_2, f_3 are posynomials
- can't handle $f_1 + \max\{f_2, f_3\} \leq 1$ since $\max\{f_2, f_3\}$ isn't posynomial
- **trick:** replace $f_1 + \max\{f_2, f_3\} \leq 1$ with three (posy) inequalities

$$f_1 + t \leq 1, \qquad f_2 \leq t, \qquad f_3 \leq t$$

  t is new slack variable

- can be applied recursively, together with fractional power trick

Example

$$\begin{array}{ll} \text{minimize} & xyz + 4x^{-1}y^{-3/2} \\ \text{subject to} & \max\{x, y\} + z \leq 1 \\ & y(x^{1/2} + 3z)^{1/2} + z^2 \leq 1 \end{array}$$

equivalent to GP

$$\begin{array}{ll} \text{minimize} & xyz + 4x^{-1}y^{-3/2} \\ \text{subject to} & t_1 + z \leq 1, \quad x \leq t_1, \quad y \leq t_1 \\ & yt_2^{1/2} + z^2 \leq 1, \quad x^{1/2} + 3z \leq t_2 \end{array}$$

(t_1 and t_2 are new variables)

Generalized posynomials

f is a **generalized posynomial** if it can be formed using addition, multiplication, positive power, and maximum, starting from posynomials

examples:

- $\max\left\{1 + x_1, 2x_1 + x_2^{0.2}x_3^{-3.9}\right\}$
- $\left(0.1x_1x_3^{-0.5} + x_2^{1.7}x_3^{0.7}\right)^{1.5}$
- $\left(\max\left\{1 + x_1, 2x_1 + x_2^{0.2}x_3^{-3.9}\right\}\right)^{1.7} + x_2^{1.1}x_3^{3.7}$
- $4x_1^{-0.1}x_2^{2.7}\max\left\{\max\left\{1 + x_1, 2x_1 + x_2^{0.2}x_3^{-3.9}\right\} + x_2^{1.1}, x_1x_2x_3\right\}$

Generalized geometric program (GGP)

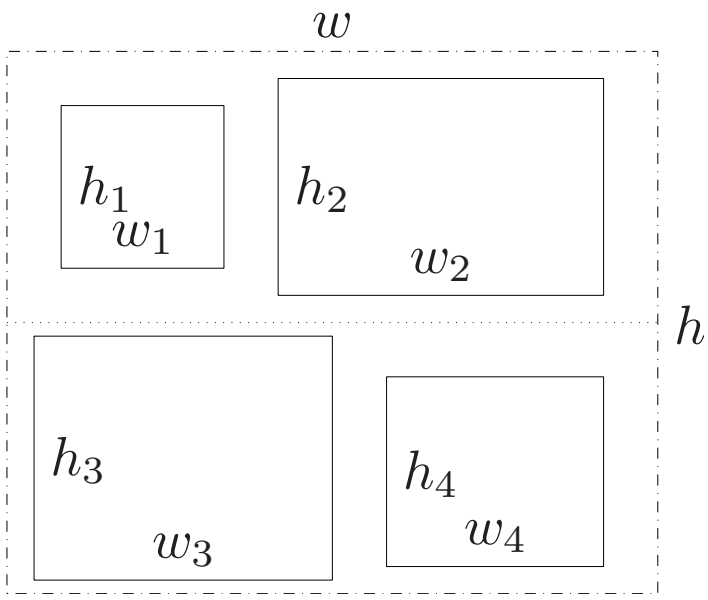
$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 1, \quad i = 1, \ldots, m \\ & g_i(x) = 1, \quad i = 1, \ldots, p \end{array}$$

f_i are **generalized posynomials**, g_i are monomials

- using tricks, can convert GGP to GP, then solve efficiently
- **conversion tricks can be automated**
  - parser scans problem description, forms GP
  - GP solver solves GP
  - solution transformed back (dummy variables eliminated)

Floor planning

- configure cell widths, heights
- minimize bounding box area
- fixed cell areas
- aspect ratio constraints



$$
\begin{array}{ll}
\text{minimize} & hw \\
\text{subject to} & h_i w_i = A_i, \quad 1/\alpha_{\max} \leq h_i/w_i \leq \alpha_{\max}, \\
& \max\{h_1, h_2\} + \max\{h_3, h_4\} \leq h, \\
& \max\{w_1 + w_2, w_3 + w_4\} \leq w
\end{array}
$$

. . . a GGP

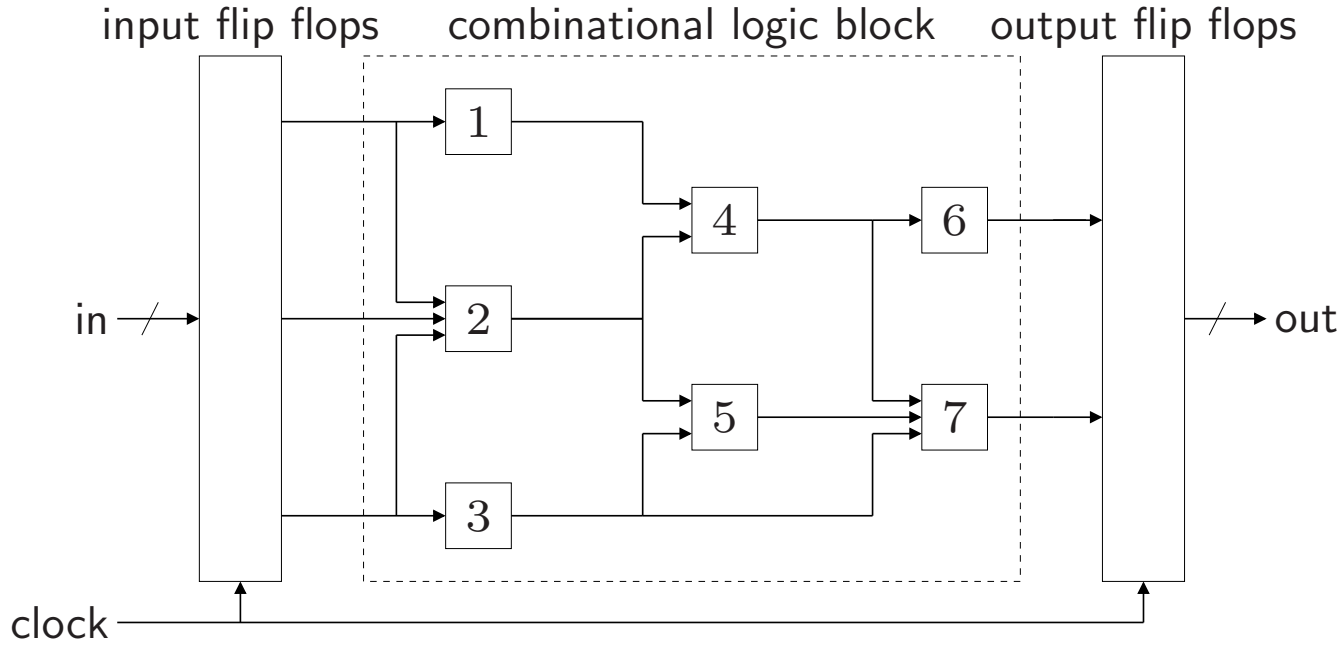
Mixed-integer geometric program

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 1, \quad i = 1, \ldots, m \\ & g_i(x) = 1, \quad i = 1, \ldots, p \\ & x_i \in \mathcal{D}_i, \quad i = 1, \ldots, k \end{array}$$

- f_i are generalized posynomials, g_i are monomials
- $\mathcal{D}_i$ are discrete sets, *e.g.*, $\{1, 2, 3, 4, \ldots\}$ or $\{1, 2, 4, 8 \ldots\}$
- **very hard** to solve exactly; all methods make some compromise (compared to methods for GP)
- **heuristic methods** attempt to find good approximate solutions quickly, but cannot guarantee optimality
- **global methods** always find the global solution, but can be extremely slow

Digital Circuit Design Applications

Gate scaling

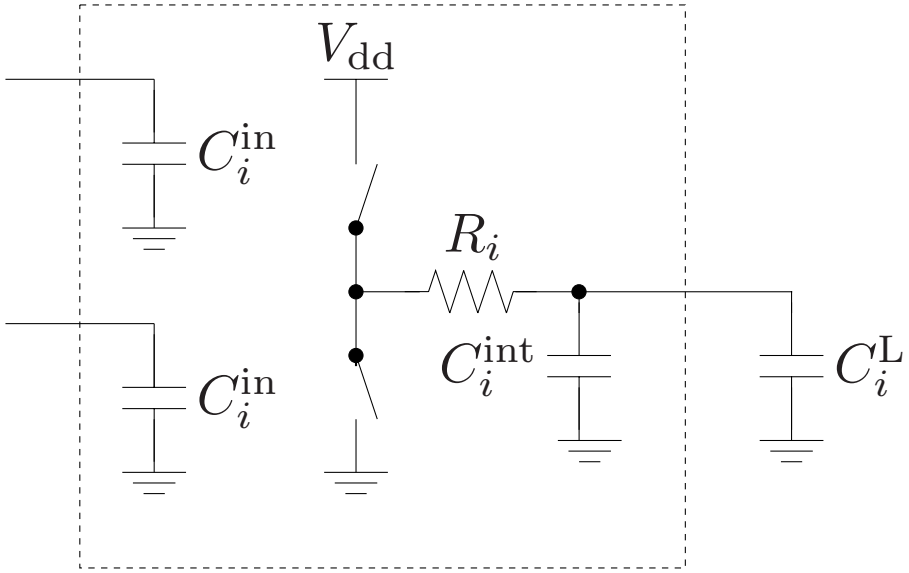


- combinational logic; circuit topology & gate types given
- gate sizes (scale factors $x_i \geq 1$) to be determined
- scale factors affect total circuit area, power and delay

Area & power

- total circuit area: $A = (a_1 x_1 + \cdots + a_n x_n)\bar{A}$
  - $\bar{A}$: area of unit scaled inverter
  - a_i: area of unit scaled gate i (in units of $\bar{A}$)
- total power (dynamic + static): $P = (b_1 x_1 + \cdots + b_n x_n) f_{\mathrm{clk}} \bar{E}$
  - f_{clk}: clock frequency
  - $\bar{E}$: energy lost per transition by unit scaled inverter driving no load
- A and P are linear functions of x, with positive coefficients, hence posynomials

RC gate delay model



- input & intrinsic capacitances, driving resistance, load capacitance

$$C_i^{\mathrm{in}} = \bar{C}_i^{\mathrm{in}} x_i, \qquad C_i^{\mathrm{int}} = \bar{C}_i^{\mathrm{int}} x_i, \qquad R_i = \bar{R}_i / x_i, \qquad C_i^{\mathrm{L}} = \sum_{j \in \mathrm{FO}(i)} C_j^{\mathrm{in}}$$

RC gate delay model

- model

$$\bar{C}_i^{\text{in}} = \alpha_i \eta \bar{C}, \qquad \bar{C}_i^{\text{int}} = \beta_i \bar{C}, \qquad R_i = \gamma_i \bar{R}/x_i$$

  - $\bar{C}$: intrinsic capacitance of unit scaled inverter
  - η: (input capacitance of unit scaled inverter)/$\bar{C}$
  - $\bar{R}$: driving resistance of unit scaled inverter

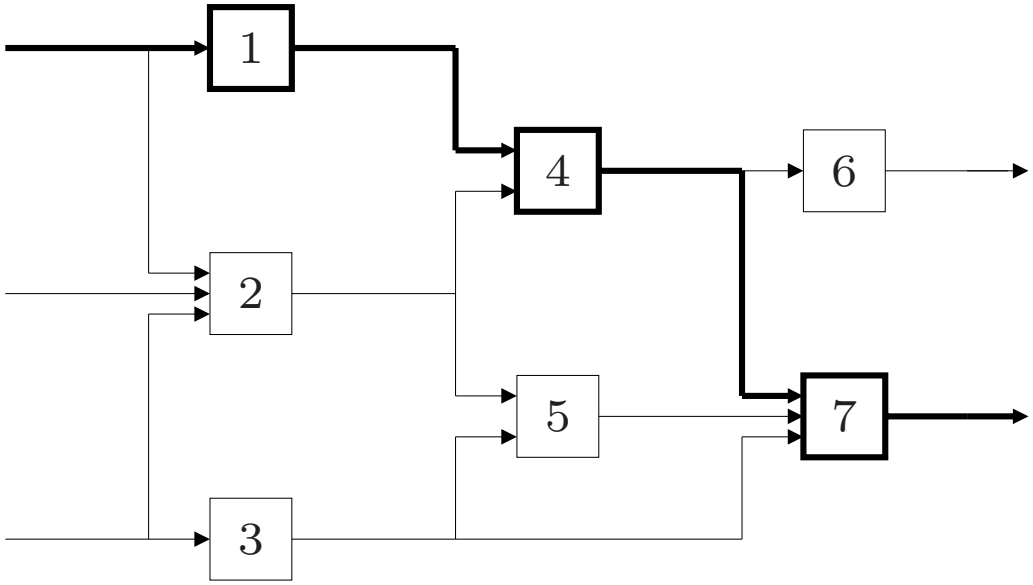
- RC gate delay:

$$D_i = 0.69 R_i (C_i^{\text{L}} + C_i^{\text{int}}) = \left(\gamma_i \beta_i + (\gamma_i / x_i) \sum_{j \in \text{FO}(i)} \eta \alpha_j x_j \right) \bar{D}$$

  $\bar{D} = 0.69 \bar{R} \bar{C}$: delay of unit scaled inverter with no load

- D_i are **posynomials** (of scale factors)

Path and circuit delay



- delay of a path: sum of delays of gates on path
  . . . **posynomial**

- circuit delay: maximum delay over all paths
  . . . **generalized posynomial**

Basic gate scaling problem

$$\begin{array}{ll} \text{minimize} & D \\ \text{subject to} & P \leq P^{\max}, \quad A \leq A^{\max} \\ & 1 \leq x_i, \quad i = 1, \ldots, n \end{array}$$

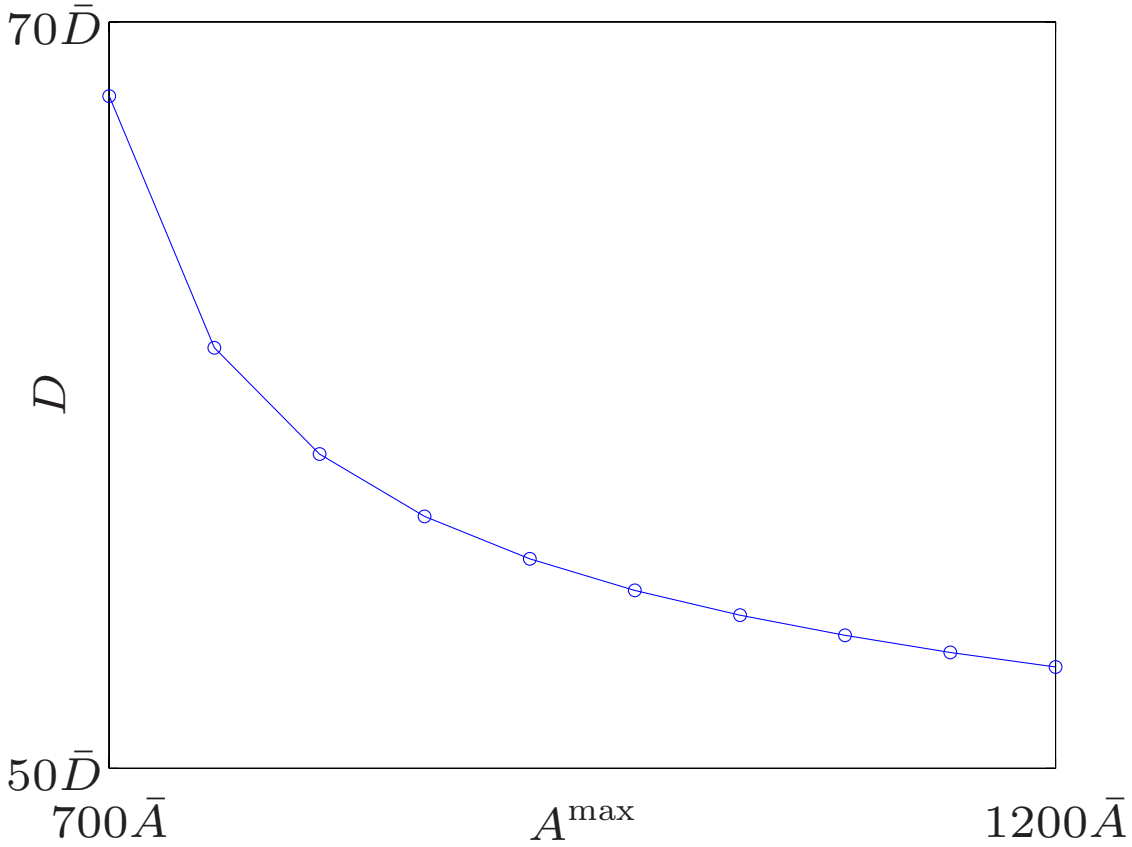
. . . a **GGP**

extensions/variations:

- minimize area, power, or some combination
- add other constraints
- optimal trade-off of area, power, delay

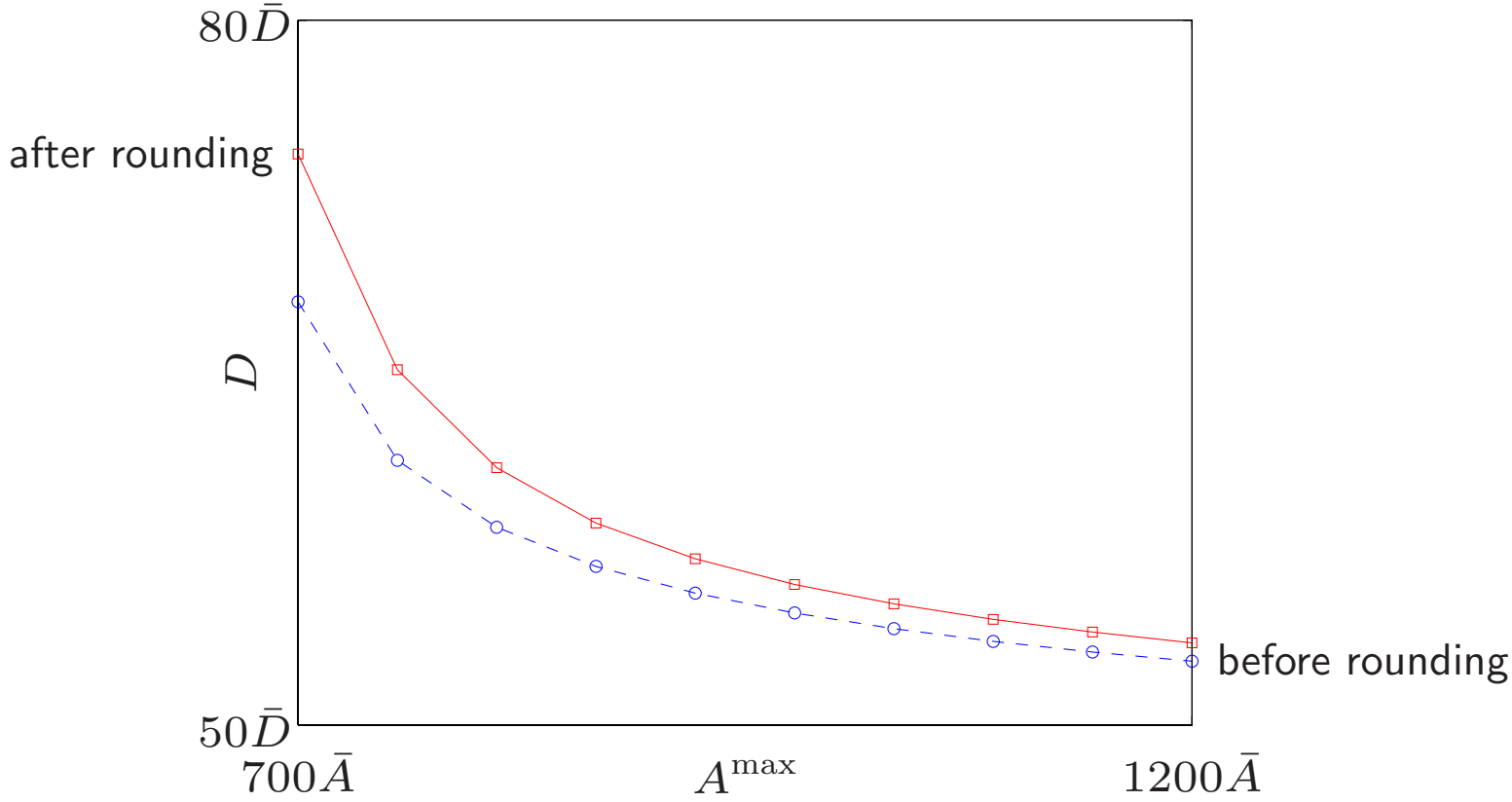
Example: Ladner-Fisher 32-bit adder

- 451 gates (scale factors); RC gate delay model
- typical optimization time: few seconds on PC



Ladner-Fisher 32-bit adder with integer scale factors

- add constraints $x_i \in \{1, 2, 3, \ldots\}$
- simple rounding of optimal continuous scalings



Sparse GP gate scaling problem

$$
\begin{array}{ll}
\text{minimize} & D \\
\text{subject to} & T_j \leq D \quad \text{for } j \text{ an output gate} \\
& T_j + D_i \leq T_i \quad \text{for } j \in \mathrm{FI}(i) \\
& P \leq P^{\max}, \quad A \leq A^{\max} \\
& 1 \leq x_i, \quad i = 1, \ldots, n
\end{array}
$$

- T_i are upper bounds on signal arrival times
- **extremely sparse GP**; can be solved very efficiently

Better (generalized posynomial) models

can greatly improve model, while retaining GP compatibility
(hence efficient global solution)

- area, delay, power can be any generalized posynomials of scale factors, $e.g.$,

$$D_i = a_i + b_i (C_i^{\mathrm{L}})^{1.05} x_i^{-0.9}, \qquad P_i = c_i + d_i (C_i^{\mathrm{L}})^{1.2} + e_i x_i^{1.1}$$

- these can be found by more refined analysis, or fitting generalized posynomials to simulation/characterization data

Distinguishing gate transitions

- can distinguish rising and falling transitions, with different delay, energy, C^{in}, for each gate input/transition

- (bounds on) signal arrival times can be propagated through recursions, $e.g.$,

$$T_i^{\mathrm{r}} = \max_{j \in \mathrm{FI}(i)} \left\{ T_j^{\mathrm{r}} + D_{ji}^{\mathrm{rr}},\ T_j^{\mathrm{f}} + D_{ji}^{\mathrm{fr}} \right\}, \quad T_i^{\mathrm{f}} = \max_{j \in \mathrm{FI}(i)} \left\{ T_j^{\mathrm{r}} + D_{ji}^{\mathrm{rf}},\ T_j^{\mathrm{f}} + D_{ji}^{\mathrm{ff}} \right\}$$

- gate scaling problem more complex, but **still a GGP**
(hence can be efficiently solved)

Modeling signal slopes

- associate (worst-case) output signal transition time τ with each gate
- model delay, energy, input capacitance as (generalized posynomial) functions of scale factor, load capacitance, input transition time
- propagate output transition time using (generalized posynomial) function of scale factor, load capacitance, input transition time
- common model:

$$D_i = a_i C_i^{\mathrm{L}}/x_i + \kappa_i \tau_i^{\mathrm{in}}, \qquad E_i = b_i(C_i^{\mathrm{L}} + c_i x_i) + \lambda_i x_i \tau_i^{\mathrm{in}}, \qquad \tau_i = \nu_i D_i$$

- gate scaling problem **still a GGP**

Arrival time propagation with soft maximum

- can even generalize max function used to propagate signal arrival times
- replace with soft maximum, $e.g.$, $(T_1^p + \cdots + T_k^p)^{1/p}$ (say, $p \approx 10$)
- can account for increased delay when inputs switch simultaneously
- can choose soft maximum function by fitting simulation data
- gate scaling problem **remains a GGP**

Design with a standard library

- circuit topology is fixed; choose size for each gate from **discrete library**
- a combinatorial optimization problem, difficult to solve exactly

- GP approach
  - for each gate type in library, fit given library data to find GP-compatible models of delay, power, . . .
  - size with **continuous** fitted models, using GP
  - snap continuous scale factors back to standard library

Robust design over corners

- have K corners or scenarios, $e.g.$, combinations of
  - process parameters
  - supply voltage
  - temperature
- for each corner have (slightly) different models for delay, power, . . .
- **robust design** finds gate scalings that work well for **all corners**

Robust design over corners

- basic (worst-case) robust design over corners:

$$\begin{array}{ll} \text{minimize} & \max\{D^{(1)},\ldots,D^{(K)}\} \\ \text{subject to} & P^{(1)}(x) \leq P^{\max},\ldots,P^{(K)}(x) \leq P^{\max} \\ & A \leq A^{\max} \\ & 1 \leq x_i, \quad i=1,\ldots,n \end{array}$$

- many variations, $e.g.$, minimize average delay over corners,

$$(1/K)\left(D^{(1)}+\cdots+D^{(K)}\right)$$

- results in (very large, but sparse) **GGP**

Multiple-scenario design

- have K scenarios or operating modes, with K models for P, D, . . .
- scenarios are combinations of
  - supply & threshold voltages
  - clock frequency
  - specifications & constraints
- like corner-based robust design, but scenarios are **intentional**
- find one set of gate scalings that work well in all scenarios

Example

- find single set of gate scalings to support both high performance mode and low power mode
  - in high performance mode: $P^{\text{fast}} \leq \bar{P}^{\text{fast}}$, $D^{\text{fast}} \leq \bar{D}^{\text{fast}}$
  - in low power mode: $P^{\text{slow}} \leq \bar{P}^{\text{slow}}$, $D^{\text{slow}} \leq \bar{D}^{\text{slow}}$

$$
\begin{array}{ll}
\text{minimize} & A \\
\text{subject to} & P^{\text{slow}} \leq \bar{P}^{\text{slow}}, \qquad D^{\text{slow}} \leq \bar{D}^{\text{slow}} \\
& P^{\text{fast}} \leq \bar{P}^{\text{fast}}, \qquad D^{\text{fast}} \leq \bar{D}^{\text{fast}} \\
& 1 \leq x_i, \quad i = 1, \ldots, n
\end{array}
$$

. . . a GGP

Dual mode design example

- random netlist, 100 gates, average fanout 3
- alpha-power law delay model; dynamic + leakage power model
- dual mode
  - low power (slow): $f_{\mathrm{clk}}^{\mathrm{slow}} = \bar{f}_{\mathrm{clk}}$, $V_{\mathrm{dd}}^{\mathrm{slow}} = 1.0$, $V_{\mathrm{th}}^{\mathrm{slow}} = 0.4$
  - high performance (fast): $f_{\mathrm{clk}}^{\mathrm{fast}} = 2\bar{f}_{\mathrm{clk}}$, $V_{\mathrm{dd}}^{\mathrm{fast}} = 2.0$, $V_{\mathrm{th}}^{\mathrm{fast}} = 0.2$
- objective is area; different power/delay specs for each mode

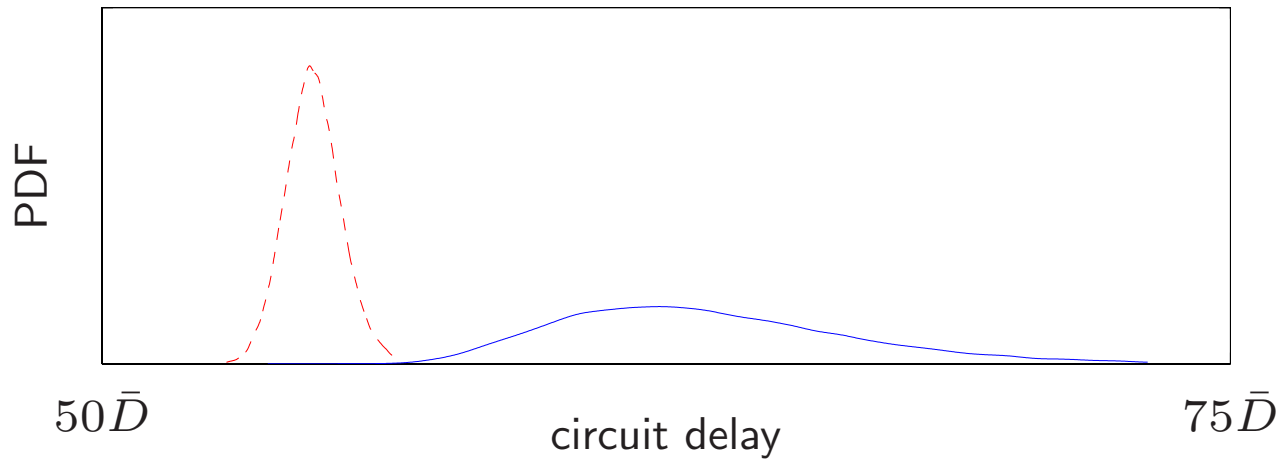
Dual mode design example

	A	D (slow)	D (fast)	P (slow)	P (fast)
specification	–	$30\bar{D}$	$15\bar{D}$	$500\bar{P}$	$5000\bar{P}$
design for slow only	$330\bar{A}$	30	**22**	290	2700
design for fast only	$370\bar{A}$	**38.4**	15	440	4060
dual mode design	$380\bar{A}$	25	15	444	4062

- $\bar{D}$: delay of unit scaled inverter driving no load, in fast mode
- $\bar{P} = f_{\mathrm{clk}}\bar{E}$: dynamic power dissipated by unit scaled inverter driving no load, transition frequency f_{clk}, supply voltage 1.0V

Statistical parameter variation

- circuit peformance depends on random device and process parameters
- hence, performance measures like P, D are random variables $\mathbf{P}$, $\mathbf{D}$
- delay **D** is max of many random variables; often skewed to right
- **distributions** of $\mathbf{P}$, $\mathbf{D}$ depend on gate scalings x_i



- related to (parametric) yield, DFM, DFY . . .

Statistical design

- measure random performance measures by 95% quantile (say)

$$\begin{array}{ll} \text{minimize} & \mathbf{Q}^{\cdot 95}(\mathbf{D}) \\ \text{subject to} & \mathbf{Q}^{\cdot 95}(\mathbf{P}) \leq P^{\max}, \qquad A \leq A^{\max} \\ & 1 \leq x_i, \quad i = 1, \ldots, n \end{array}$$

- **extremely difficult** stochastic optimization problem; almost no analytic/exact results

- but, (GP-compatible) heuristic method works well

Heuristic for statistical design

- assume generalized posynomial models for gate delay mean $D_i(x)$ and variance $\sigma_i(x)^2$
- $e.g.$, $\sigma_i(x) = \eta_i x_i^{-1/2} D_i(x)$ (Pelgrom's model)
- optimize using surrogate gate delays

$$\tilde{D}_i(x) = D_i(x) + \kappa_i \sigma_i(x)$$

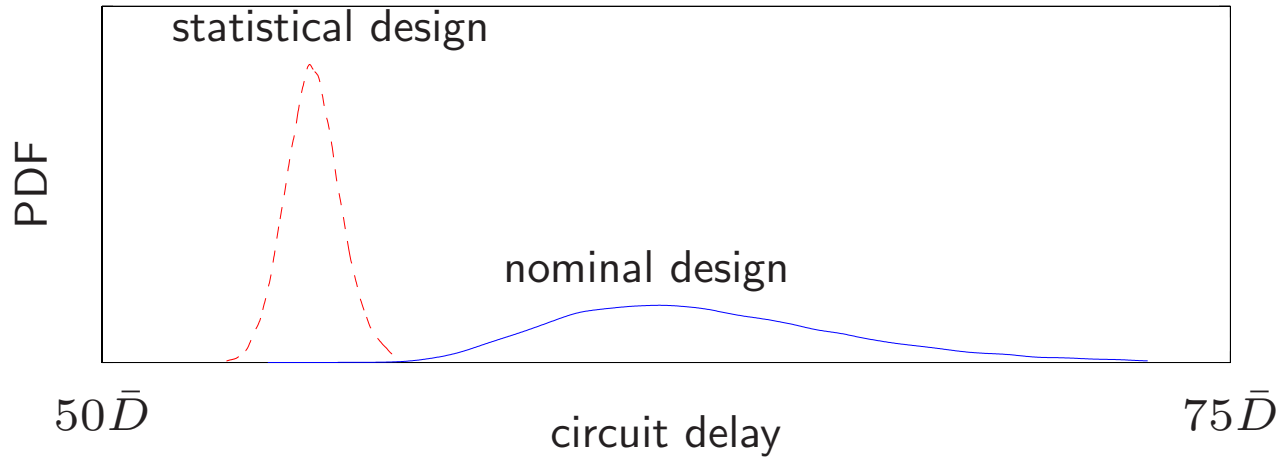
  $\kappa_i \sigma_i(x)$ are **margins** on gate delays (κ_i is typically 2 or 3)

- verify statistical performance via Monte Carlo
  (can update κ_i's and repeat)

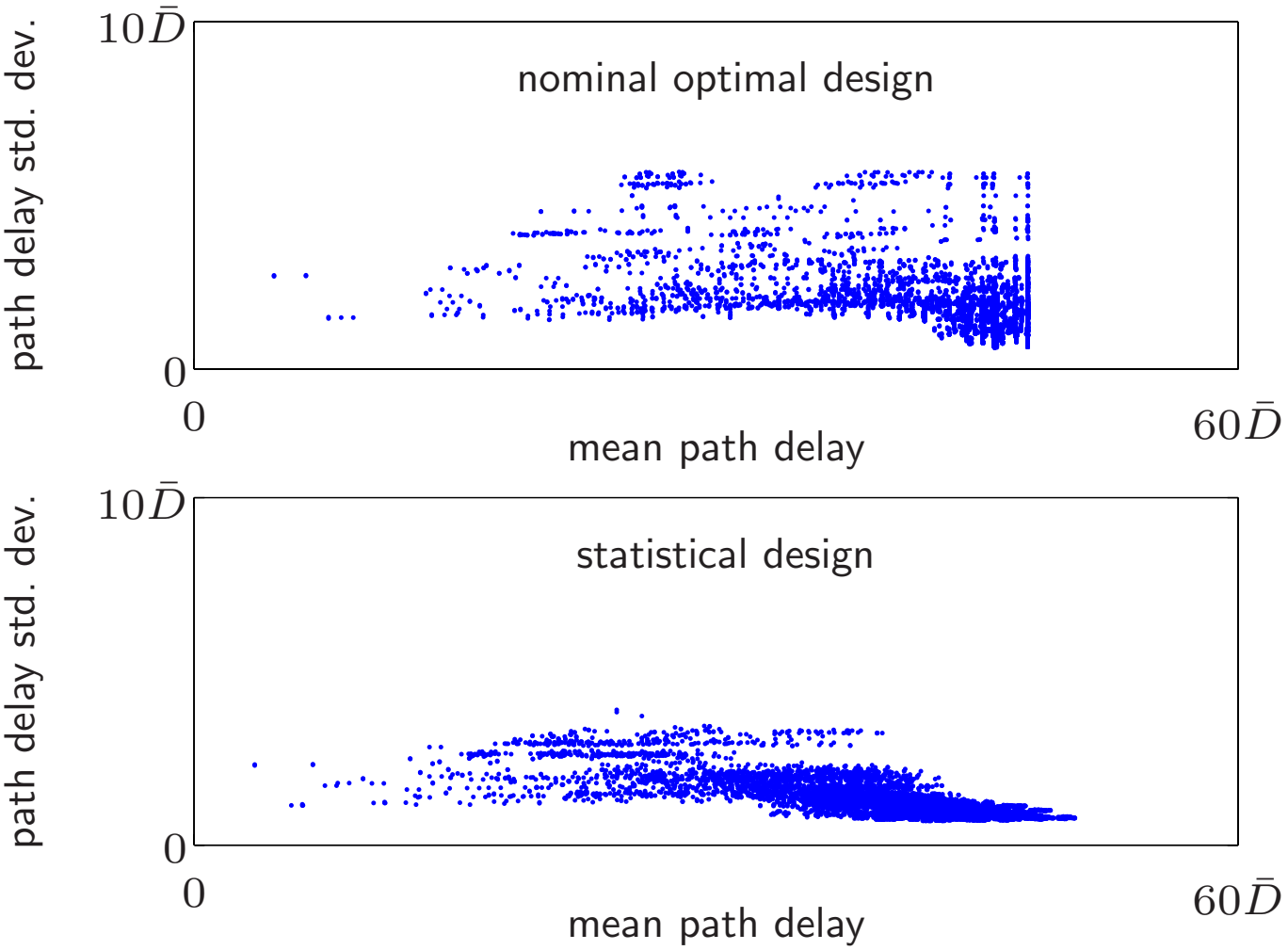
Heuristic for statistical design

heuristic statistical design

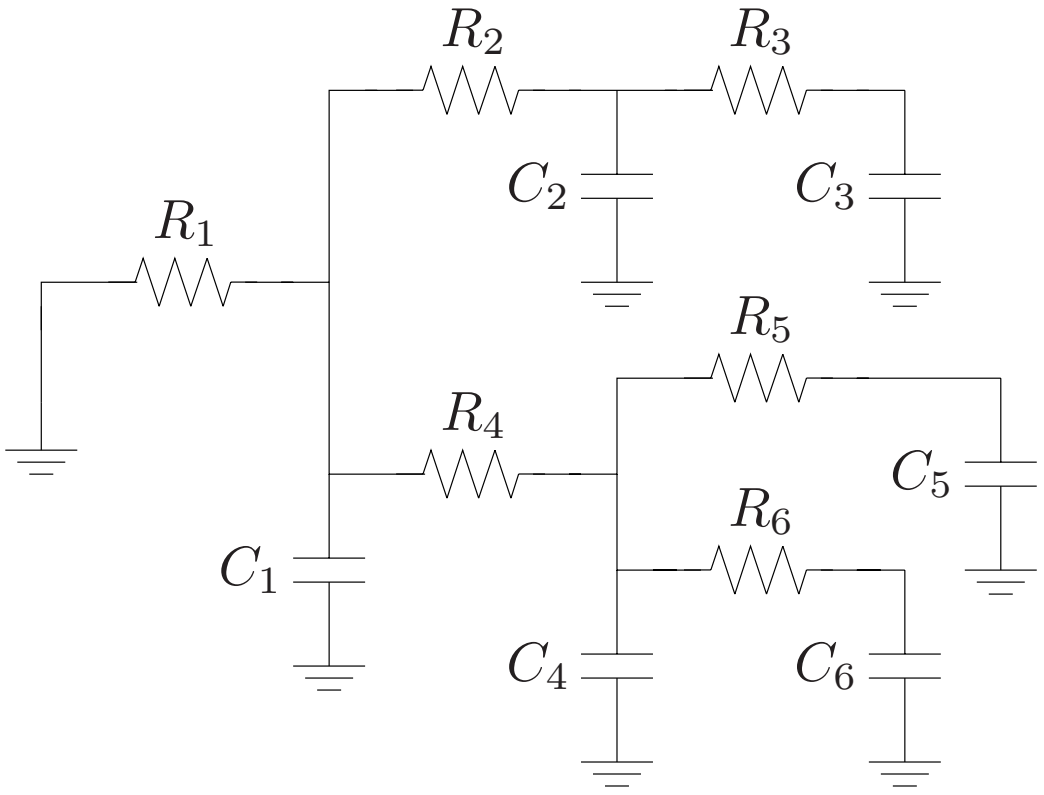
- often far superior to design obtained ignoring statistical variation
- not very sensitive to details of process variation statistics (distribution shape, correlations, . . .)
- below: Ladner-Fisher 32-bit adder, Pelgrom variance model



Path delay mean/std. dev. scatter plots



RC tree optimization



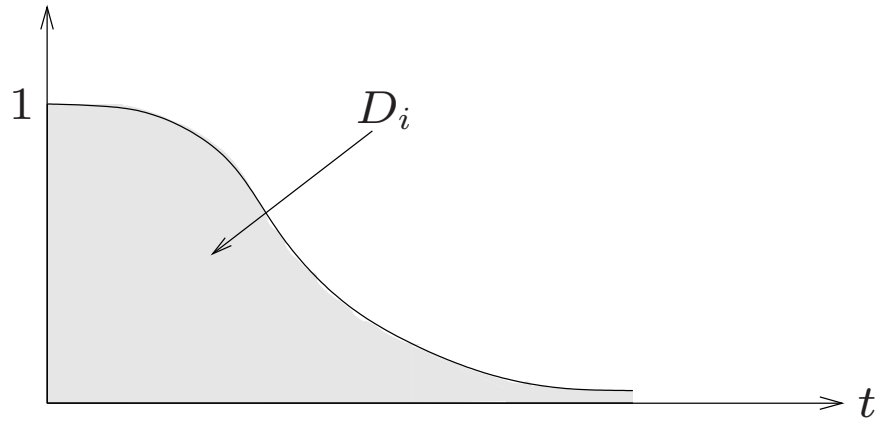
- R_is and C_is are generalized posynomials of some underlying variables x

Elmore delay

- **Elmore delay** at node i:

$$D_i = \int_0^\infty v_i(t)\, dt$$

area under voltage curve, when voltages are initialized as $v_i(0) = 1$



- Elmore delay of RC tree is $D = \max\{D_1, \ldots, D_N\}$

Elmore delay expression

- analytic expression for Elmore delay D_i

$$D_i = \sum_{j \in \mathbf{P}(i)} R_i C_i^{\text{tot}}$$

- $\mathbf{P}(i)$ is path from root to node i
- C_i^{tot} is the total capacitance downstream from node i (including C_i)
- D_i is **posynomial** of x
- D is **generalized posynomial** of x

RC tree optimization

- minimize RC tree delay subject to (generalized posynomial) constraints

$$\begin{array}{ll} \text{minimize} & D \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \end{array}$$

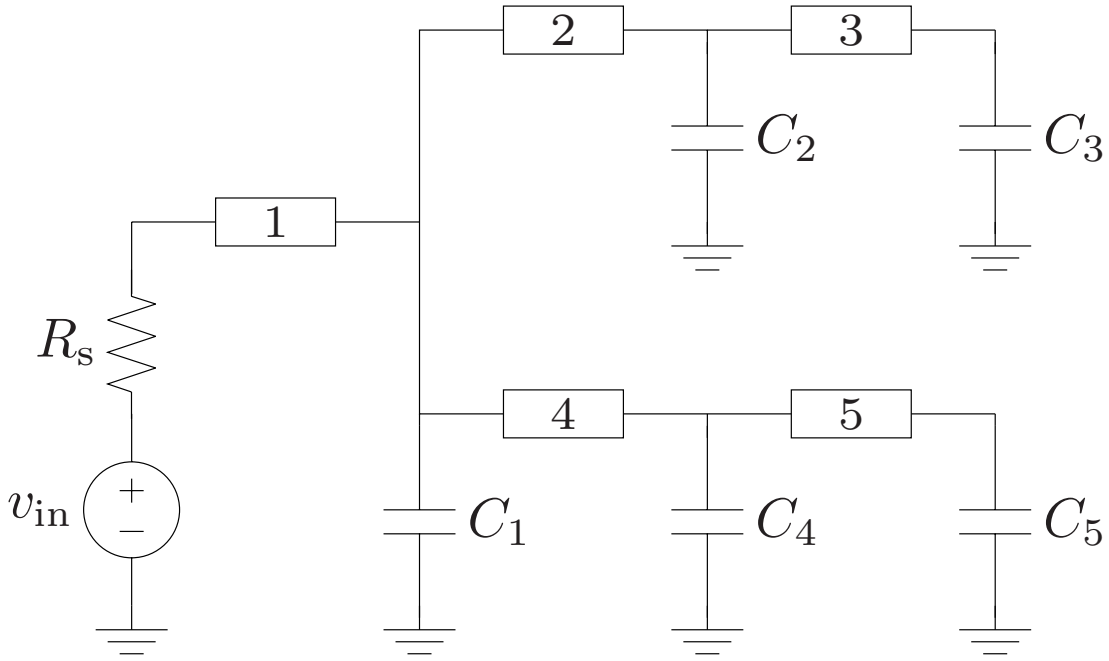
. . . **a GGP**

- sparse formulation:

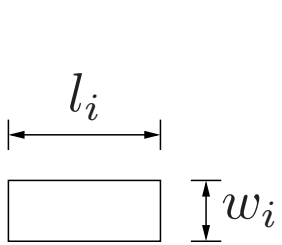
$$\begin{array}{ll} \text{minimize} & s \\ \text{subject to} & s \geq D_i, \quad i = 1, \ldots, n \\ & C_j^{\text{tot}} \geq \sum_{i \in \mathbf{Child}(j)} C_i^{\text{tot}} + C_j, \quad i = 1, \ldots, n \\ & D_i \geq D_{\mathbf{Par}(k)} + R_i C_i^{\text{tot}}, \quad i = 1, \ldots, n \\ & f_i(x) \leq 0, \quad i = 1, \ldots, m \end{array}$$

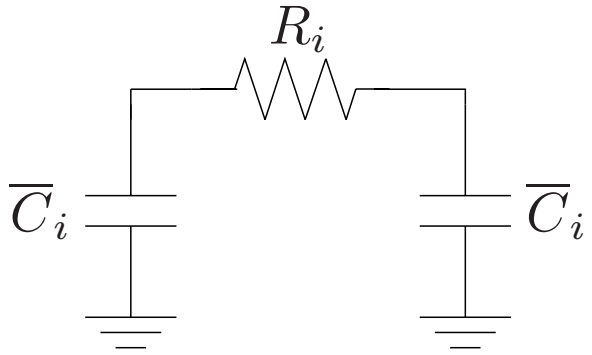
Wire sizing

- choose wire segment widths $w_i, \ldots, w_N$ in an interconnect network
- optimize delay, area



π model for wire segment





- wire resistance and capacitances

$$R_i = \alpha_i \frac{l_i}{w_i}, \qquad \overline{C}_i = \beta_i l_i w_i + \gamma_i l_i,$$

- with π model, interconnect network becomes RC tree, with R_is and C_is posynomial functions of wire segment widths w_i

Wire sizing via GP

$$\begin{array}{ll} \text{minimize} & D \\ \text{subject to} & w_i^{\min} \leq w_i \leq w_i^{\max}, \quad i = 1, \ldots, N \\ & l_1 w_1 + \cdots + l_N w_N \leq A^{\max} \end{array}$$

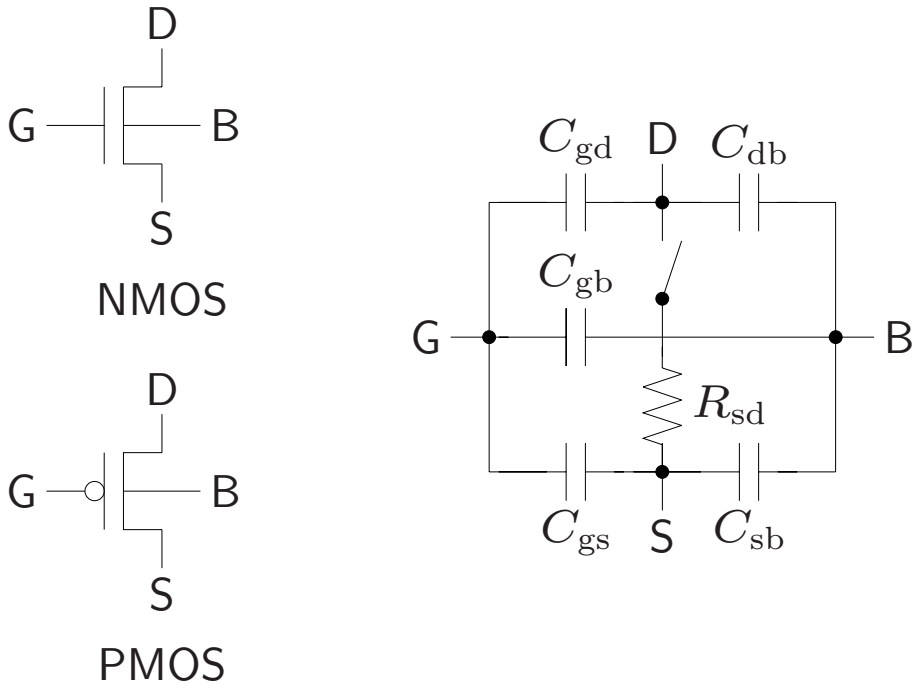
. . . **a GGP**

- can easily optimize interconnect network with 10000 wires, using sparse GP formulation
- can use more accurate generalized posynomial models of R_i, C_i

Device sizing

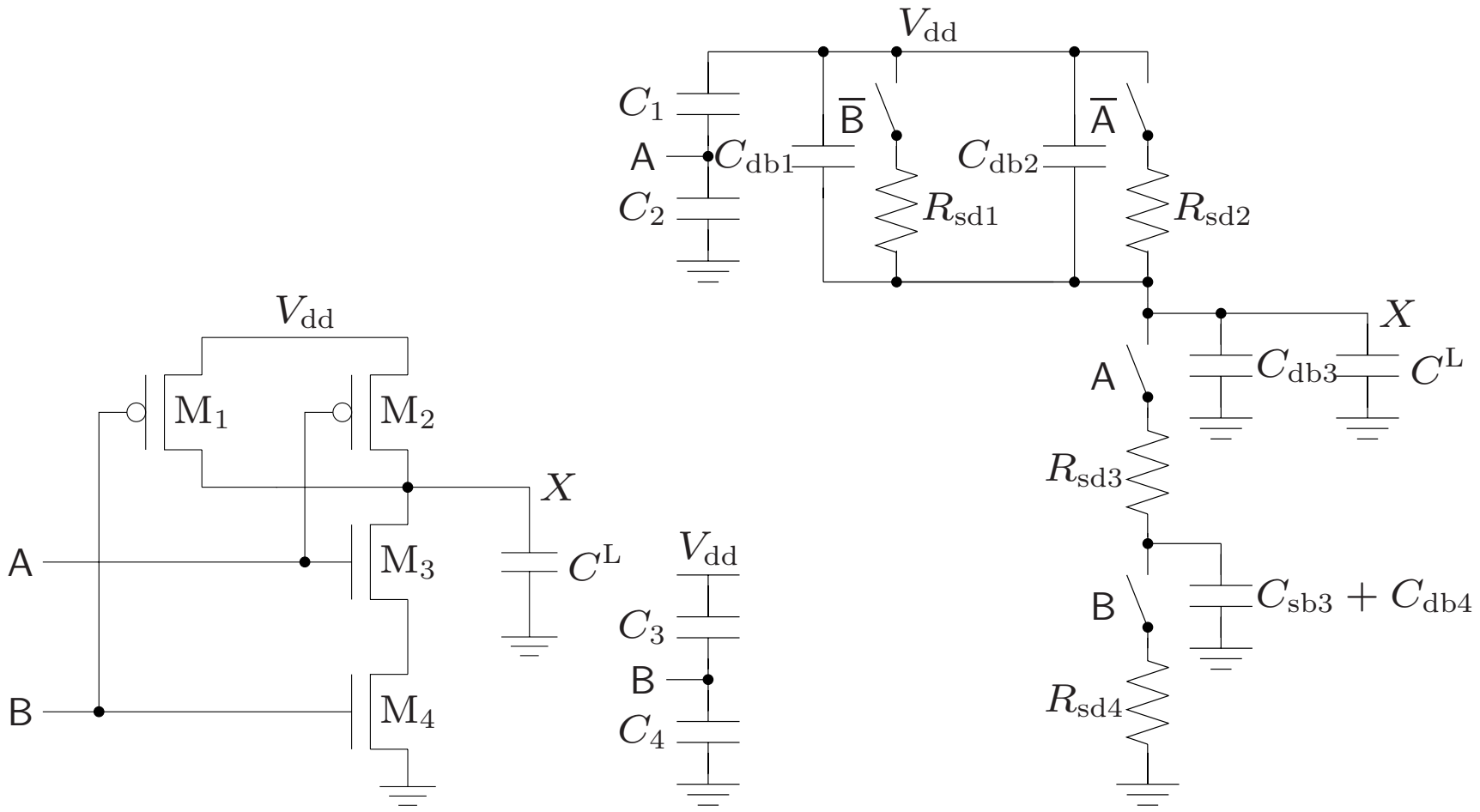
- devices (and wire segments) are sized individually
- replace each device with switch-level RC model
- each transition is associated with RC tree
- use Elmore delay to measure delay of transition
- . . . **problem is GGP**

Switch-level RC device model



- crude linear approximation of device, for delay and power optimization
- R, all Cs are generalized posynomials of device width
- we'll ignore C_{gd} (but can be incorporated via Miller effect . . .)

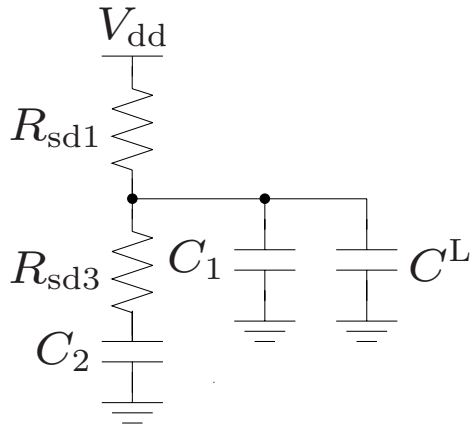
Example: 2-input NAND



$$C_1 = C_{\rm gb2}+C_{\rm gs2},\ \ C_2 = C_{\rm gb3}+C_{\rm gs3},\ \ C_3 = C_{\rm gb1}+C_{\rm gs1},\ \ C_4 = C_{\rm gb4}+C_{\rm gs4}$$

Example transition

- transition: B falls from V_{dd} to zero; A remains at V_{dd}
- associated RC tree:



  $C_1 = C_{\mathrm{db1}} + C_{\mathrm{db2}} + C_{\mathrm{db3}}$, $C_2 = C_{\mathrm{sb3}} + C_{\mathrm{db4}}$
- Elmore delay: $D = R_{\mathrm{sd1}}(C^{\mathrm{L}} + C_1 + C_2)$
- energy lost: $E = (C^{\mathrm{L}} + C_1 + C_2)V_{\mathrm{dd}}^2/2$

Device, supply and threshold voltage optimization

- **goal:** jointly optimize device sizes, supply and threshold voltages via GGP
- **need to:** model delay, power as generalized posynomial functions of device sizes, supply and threshold voltages

Generalized posynomial delay model

- alpha-power law model

$$D = \frac{V_{\mathrm{dd}}}{(V_{\mathrm{dd}} - V_{\mathrm{th}})^{\alpha}} h(w, C^{\mathrm{L}}, \tau^{\mathrm{in}})$$

  h is generalized posynomial

- generalized posynomial approximation

$$\widehat{D} = V_{\mathrm{dd}}^{1-\alpha}(1 + V_{\mathrm{th}}/V_{\mathrm{dd}} + \cdots + (V_{\mathrm{th}}/V_{\mathrm{dd}})^5)^{\alpha} h(w, C^{\mathrm{L}}, \tau^{\mathrm{in}})$$

  error under 1% for $V_{\mathrm{dd}} \geq 2V_{\mathrm{th}}$, $1.3 \leq \alpha \leq 2$

Generalized posynomial power model

- gate dynamic power: $P_{\mathrm{dyn}} = f_{\mathrm{t}}(C^{\mathrm{L}} + C^{\mathrm{int}})V_{\mathrm{dd}}^2$
- leakage current model for NMOS: $I_{\mathrm{leak}} = awe^{-(V_{\mathrm{th}} - \gamma V_{\mathrm{dd}})/V_0}$
- simple gate leakage power model:

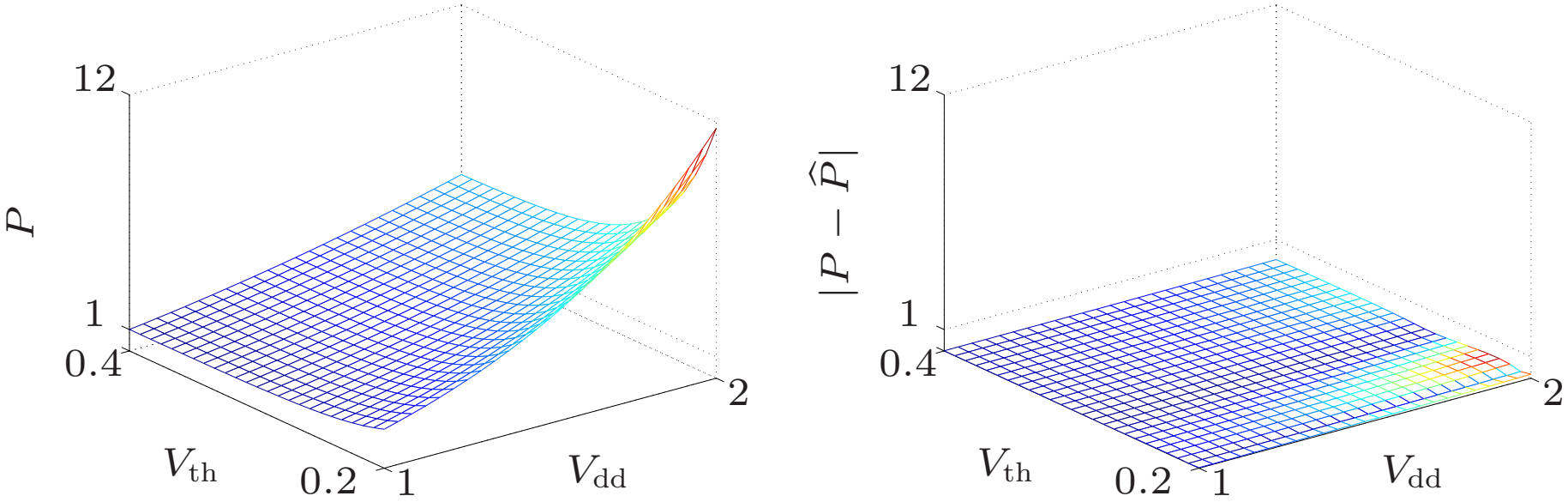
$$P_{\mathrm{leak}} = V_{\mathrm{dd}}\psi(x)e^{-(V_{\mathrm{th}} - \gamma V_{\mathrm{dd}})/V_0}$$

  ψ is generalized posynomial (from gate topology, stack effect . . .)

- **bad news:** P_{leak} (by itself) cannot be approximated by a generalized posynomial
- **good news:** the total power $P = P_{\mathrm{dyn}} + P_{\mathrm{leak}}$ can be approximated by a generalized posynomial

Example

total power $P = V_{\mathrm{dd}}^2 + 30V_{\mathrm{dd}}e^{-(V_{\mathrm{th}}-0.06V_{\mathrm{dd}})/0.039}$ (up to scaling)



- posynomial approximation
  $\widehat{P} = V_{\mathrm{dd}}^2 + 0.06V_{\mathrm{dd}}(1 + 0.0031V_{\mathrm{dd}})^{500}(V_{\mathrm{th}}/0.039)^{-6.16}$

- error under 3% (well under accuracy of model!)

Joint optimization of device sizes, $V_{\rm dd}$, & $V_{\rm th}$

basic problem, with variables: x_i, $V_{{\rm th},i}$, $V_{{\rm dd},i}$ (. . . **a GGP**)

$$
\begin{array}{ll}
\text{minimize} & D \\
\text{subject to} & P \leq P^{\max}, \qquad A \leq A^{\max} \\
& V_{\rm th}^{\min} \leq V_{{\rm th},i} \leq V_{\rm th}^{\max}, \quad i = 1, \ldots, n \\
& V_{\rm dd}^{\min} \leq V_{{\rm dd},i} \leq V_{\rm dd}^{\max}, \quad i = 1, \ldots, n \\
& \text{other constraints . . .}
\end{array}
$$

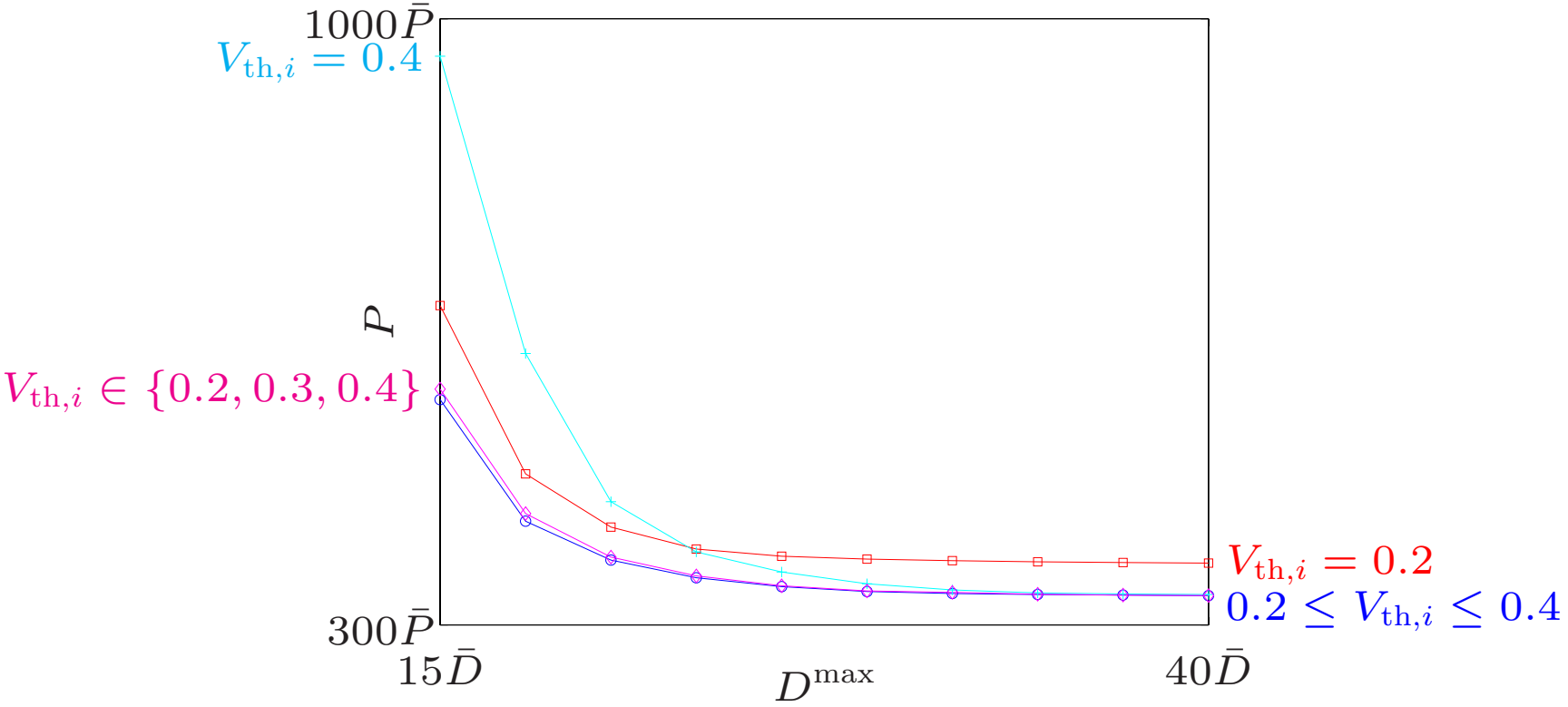
extensions/variations:

- discrete allowed $V_{\rm dd}$, $V_{\rm th}$ values (yields MIGP)
- clustering, with single $V_{\rm dd}$, $V_{\rm th}$ per cluster
- multi-scenario design: choose single set of w_i's, different $V_{\rm dd}^{(k)}$, $V_{\rm th}^{(k)}$ for each scenario $k = 1, \ldots, K$

Joint optimization example

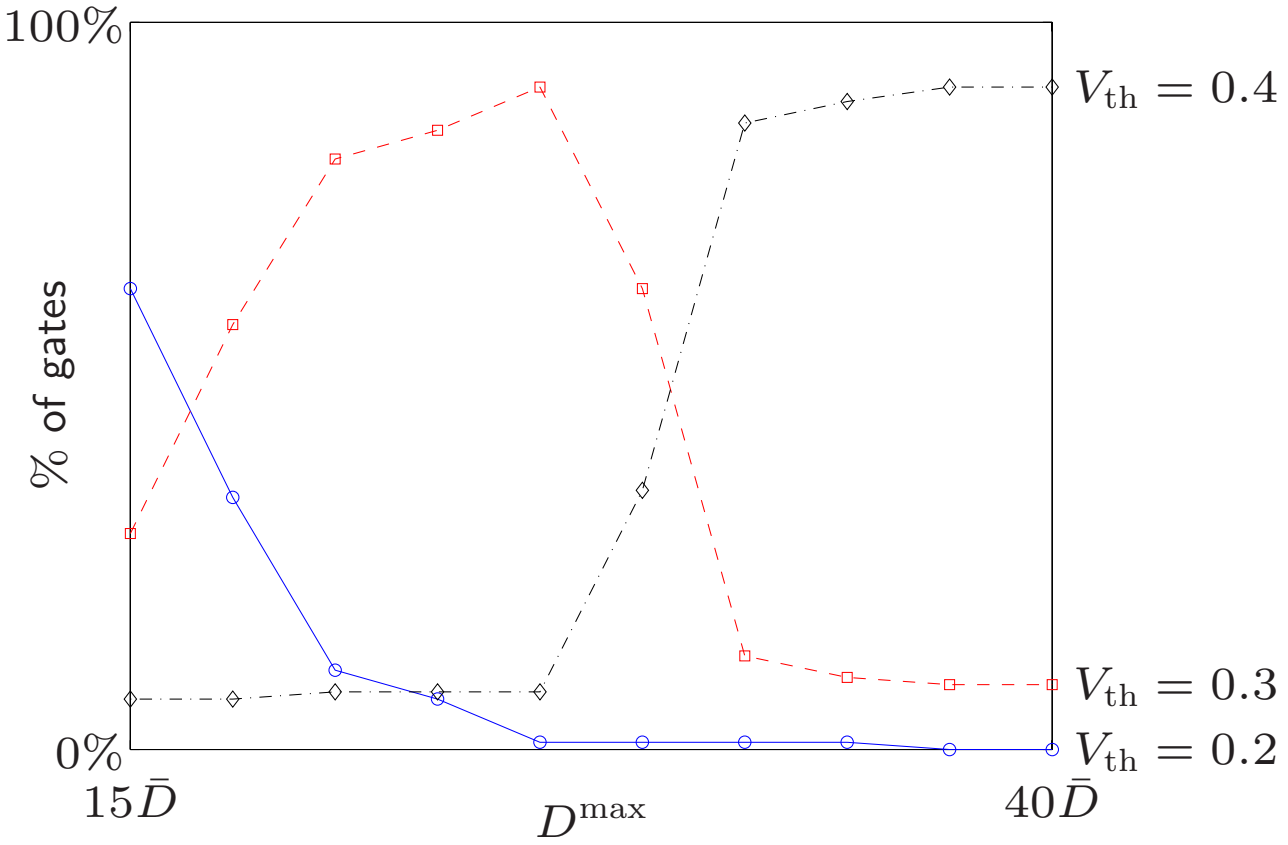
- random netlist, 100 gates, average fanout 3, alpha-power-law model
- variables: gate scale factors x_i, threshold voltages $V_{\mathrm{th},i}$
- all gates with common supply voltage
- four delay-power trade-off curves:
  - all gates low $V_{\mathrm{th},i} = 0.2$
  - all gates high $V_{\mathrm{th},i} = 0.4$
  - continuous threshold voltages $0.2 \leq V_{\mathrm{th},i} \leq 0.4$
  - discrete threshold voltages $V_{\mathrm{th},i} \in \{0.2, 0.3, 0.4\}$

Joint optimization example



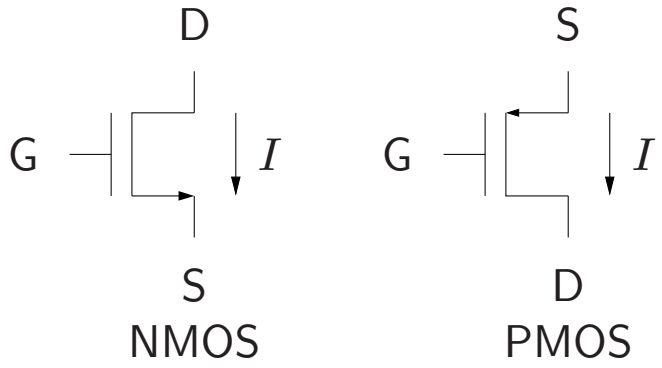
- $\bar{D}$: delay of unit scaled inverter driving no load in fast mode
- $\bar{P}$: dynamic power dissipated by unit scaled inverter driving no load

Joint optimization example



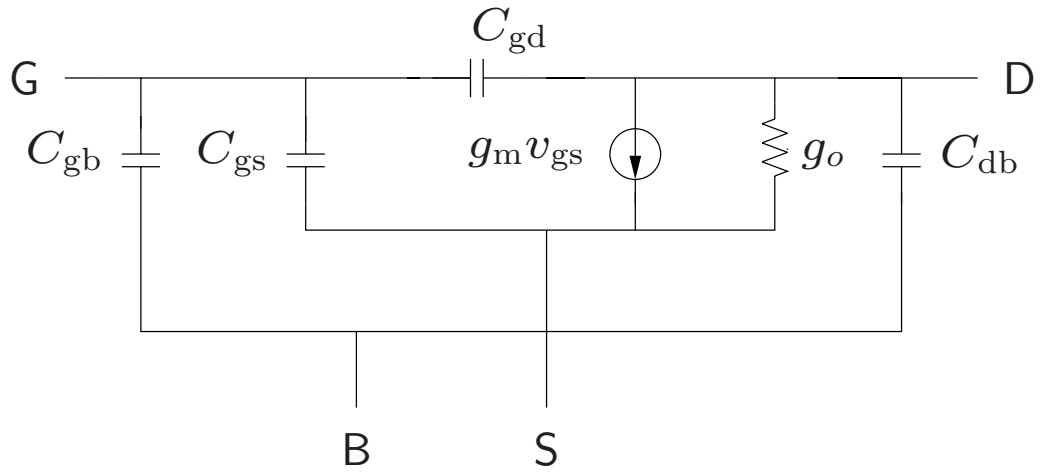
Analog Circuit Design Applications

Large signal MOS model



- gate overdrive voltage $V_{\rm gov} = V_{\rm gs} - V_{\rm th}$
- saturation condition: $V_{\rm ds} \geq V_{\rm dsat} = V_{\rm gov}$ ($V_{\rm dsat}$ is minimum drain-source voltage for device to operate in saturation)
- square-law model $I = 0.5\mu C_{\rm ox}(W/L)V_{\rm gov}^2$
- GP model variables: $I,\ L,\ W$
- $V_{\rm gov} = (\mu C_{\rm ox}/2)^{-1/2} I^{1/2} L^{1/2} W^{-1/2}$ is monomial
- $V_{\rm gs} = V_{\rm gov} + V_{\rm th}$ is posynomial

Small signal dynamic MOS model



- transconductance $g_{\rm m} = (2\mu C_{\rm ox})^{1/2} I^{1/2} L^{-1/2} W^{1/2}$ is monomial
- output conductance $g_{\rm o} = \lambda I$ is monomial
- all capacitances are (approximately) posynomial in I, L, W
- better (GP-compatible) models can be obtained by fitting data from accurate models or measurements

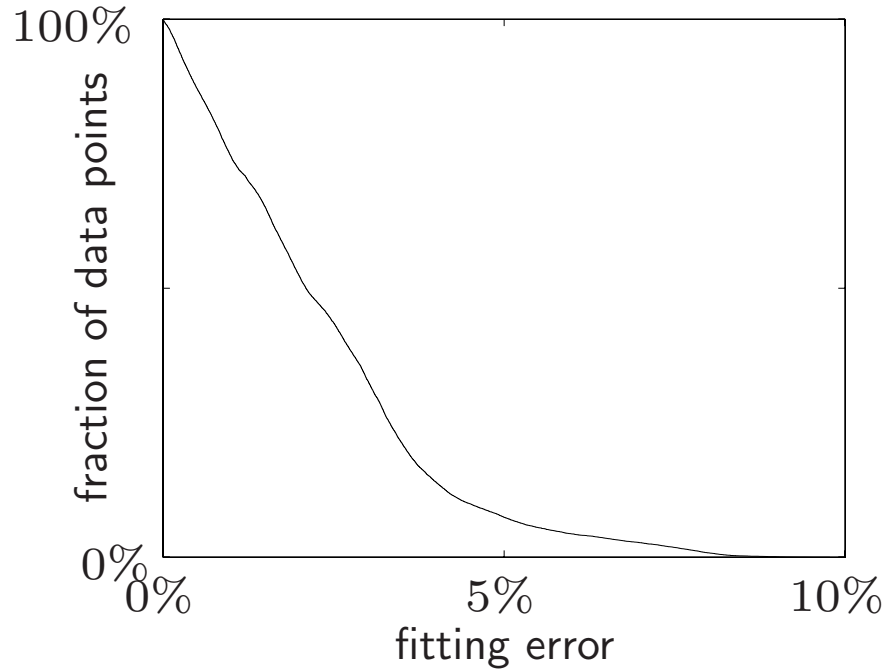
Example: monomial g_{m} model

- monomial model of g_{m} for I/O NMOS device in a 0.13μm technology
- 11000 data points (from BSIM3) over ranges
  - $0.3\mu\mathrm{m} \leq L \leq 3\mu\mathrm{m}, \quad 2\mu\mathrm{m} \leq W \leq 20\mu\mathrm{m}$
  - $0.7\mathrm{V} \leq V_{\mathrm{gs}} \leq 1.7\mathrm{V}, \quad V_{\mathrm{dsat}} \leq V_{\mathrm{ds}} \leq 1.5V_{\mathrm{gs}}$
- V_{ds} appears in data set, but not in g_{m} model
- monomial fit (using simple log-regression, SI units):

$$g_{\mathrm{m}} = 0.0278 I^{0.4798} L^{-0.511} W^{0.5632}$$

Example: monomial g_{m} model

- fitting (relative) error cumulative distribution plot:

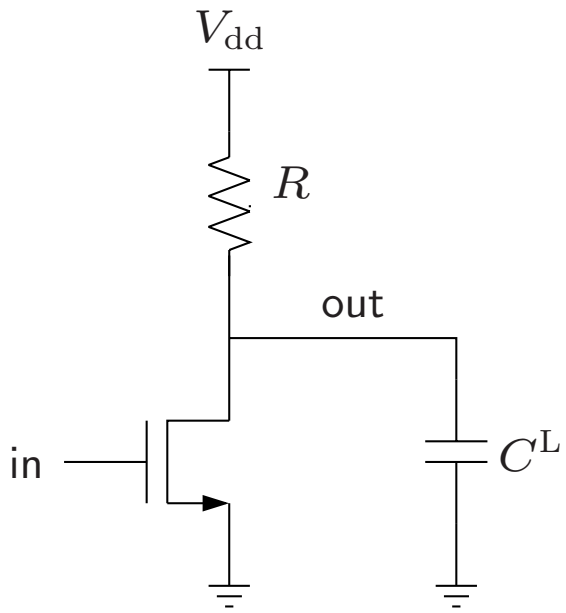


- for 90% of points, fit is better than 4%

Single transistor common source amplifier

- variables: I, L, W, R
- saturation: $V_{\rm dsat} + IR \leq V_{\rm dd}$
- gain $G = g_{\rm m}/(1/R + g_{\rm o})$
- power $P = V_{\rm dd} I$
- (unity gain) bandwidth $B = g_{\rm m}/C^{\rm L}$
- design problem:

$$\begin{array}{ll} \text{minimize} & P \\ \text{subject to} & B \geq B^{\rm min}, \quad G \geq G^{\rm min} \\ & \text{saturation} \end{array}$$



Common source amplifier design via GP

- rewrite as

$$\begin{array}{ll} \text{minimize} & P \\ \text{subject to} & B \geq B^{\min}, \quad G^{-1} \geq 1/G^{\min} \\ & V_{\mathrm{dsat}} + IR \leq V_{\mathrm{dd}} \end{array}$$

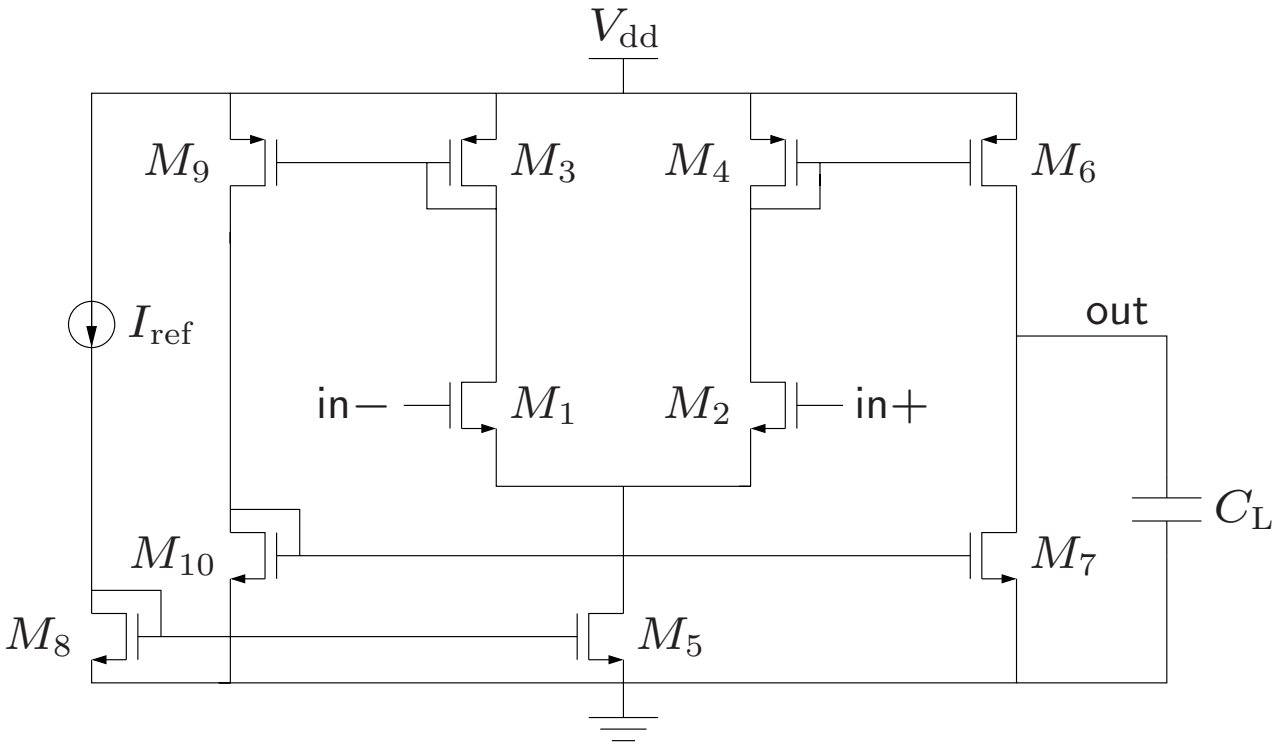
- . . . a **GP**, since P and B are monomials, and

$$G^{-1} = \frac{1/R + g_{\mathrm{o}}}{g_{\mathrm{m}}}$$

  is posynomial

- this is a simple problem; don't need GP sledgehammer . . .

Current mirror opamp



- M_1, M_2 and M_3, M_4 matched pairs
- four current mirrors: M_8, M_5; M_{10}, M_7; M_9, M_3; M_4, M_6

Design problem

$$\begin{array}{ll} \text{minimize} & P \\ \text{subject to} & B \geq B^{\min}, \quad G \geq G^{\min}, \quad A \leq A^{\max} \\ & \text{other constraints } \ldots \end{array}$$

- objective & specifications:
  - P is power dissipation
  - B is unity gain bandwidth
  - G is DC gain
  - A is (active) area
- design variables: $L_1, \ldots, L_{10}$, $W_1, \ldots, W_{10}$
- given: V_{dd}, C_{L}, I_{ref}, common-mode voltage V_{cm}
- we'll formulate as GP

Power, bandwidth, gain, & area

- power: $P = V_{\mathrm{dd}}(I_8 + I_5 + I_7 + I_{10})$ . . . posynomial
- bandwidth: $B = g_{\mathrm{m},2}g_{\mathrm{m},6}/(g_{\mathrm{m},4}C_{\mathrm{L}})$ . . . monomial
- area: $A = W_1L_1 + \cdots + W_{10}L_{10}$ . . . posynomial
- gain: $G = \dfrac{g_{\mathrm{m},2}g_{\mathrm{m},6}}{g_{\mathrm{m},4}(g_{\mathrm{o},6} + g_{\mathrm{o},7})}$

  . . . G^{-1} is posynomial, so $G \geq G^{\mathrm{min}}$ can be written as $G^{-1} \leq 1/G^{\mathrm{min}}$

Dimension, matching, and current constraints

- limits on device sizes: $L_{\min} \leq L_i \leq L_{\max}$, $W_{\min} \leq W_i$, $i = 1, \ldots, 10$
- differential symmetry constraints (M_1, M_2 and M_3, M_4 matched):

$$\begin{array}{lll} W_1 = W_2, & L_1 = L_2, & I_1 = I_2, \\ W_3 = W_4, & L_3 = L_4, & I_3 = I_4, \end{array}$$

- length & gate overdrive voltage matched for current mirror pairs:

$$\begin{array}{llll} L_5 = L_8, & L_{10} = L_7, & L_3 = L_9, & L_4 = L_6 \\ V_{\mathrm{gov},5} = V_{\mathrm{gov},8}, & V_{\mathrm{gov},10} = V_{\mathrm{gov},7}, & V_{\mathrm{gov},3} = V_{\mathrm{gov},9}, & V_{\mathrm{gov},4} = V_{\mathrm{gov},6} \end{array}$$

- current relations:

$$I_1 = I_3 = I_5/2, \qquad I_8 = I_{\mathrm{ref}}, \qquad I_6 = I_7, \qquad I_9 = I_{10}$$

Saturation constraints

- diode connected devices (M_3, M_4, M_8, M_{10}) automatically in saturation
- others must have $V_{\rm ds} \geq V_{\rm dsat}$:
  - M_7: $V_{\rm dsat,7} \leq V_{\rm cm}$
  - M_6: $V_{\rm dsat,6} + V_{\rm cm} \leq V_{\rm dd}$
  - M_9: $V_{\rm dsat,9} + V_{\rm gs,10} \leq V_{\rm dd}$
  - M_5: $V_{\rm ds,5} + V_{\rm gs,1} \leq V_{\rm cm}$
  - M_1 & M_2: $V_{\rm cm} + V_{\rm gs,3} \leq V_{\rm dd} + V_{\rm th}$
- . . . all are posynomial inequalities

Node capacitances and non-dominant poles

- capacitances at nodes are posynomials, $e.g.$,

$$C^{\text{out}} = C_{\text{gd},6} + C_{\text{db},6} + C_{\text{gd},7} + C_{\text{db},7} + C_{\text{L}}$$

- non-dominant time constants are posynomials:

$$\tau_1 = \frac{C_{\text{d1}}}{g_{\text{m},3}}, \qquad \tau_2 = \frac{C_{\text{d2}}}{g_{\text{m},4}}, \qquad \tau_9 = \frac{C_{\text{d9}}}{g_{\text{m},10}}$$

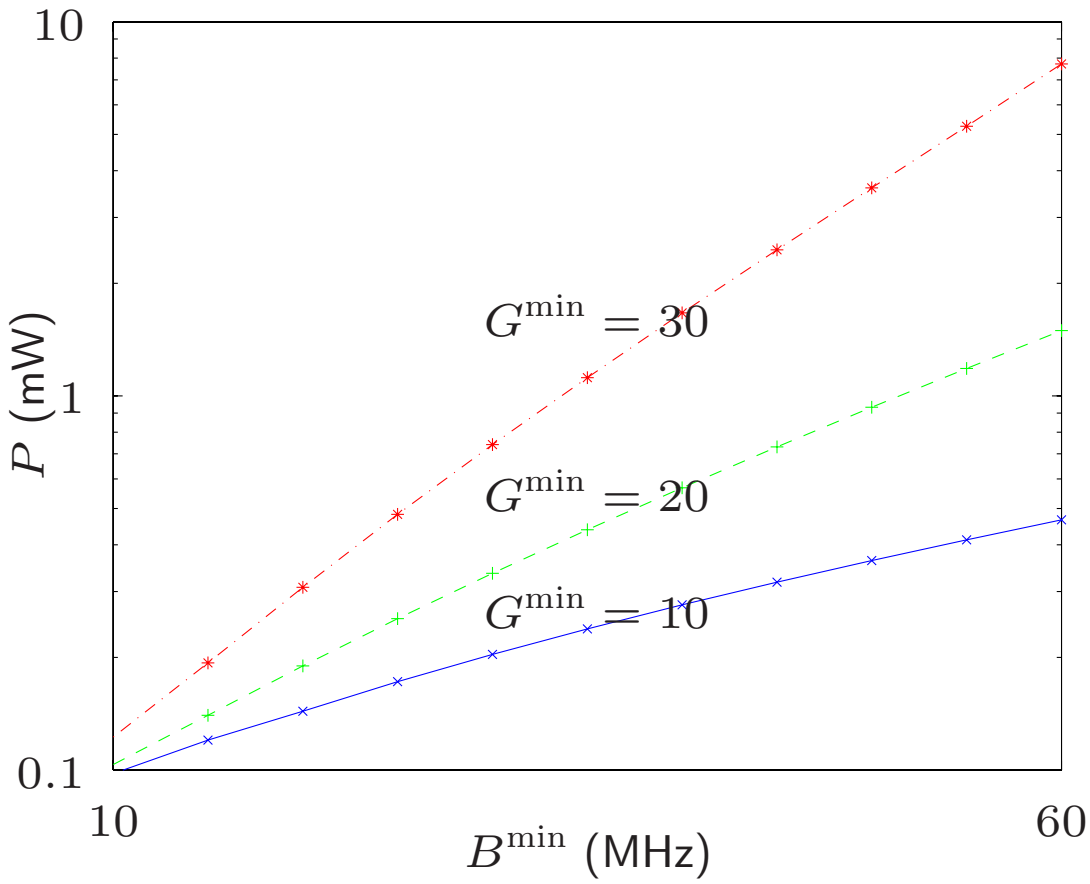
  ($C_{\text{d1}}, C_{\text{d2}}, C_{\text{d9}}$ are node capacitances at drains of M_1, M_2, M_9)

- to limit effect of non-dominant poles, make sum smaller than dominant time constant:

$$\tau_1 + \tau_2 + \tau_9 \leq \tau_{\text{dom}} = C_{\text{L}}/g_{\text{m}}$$

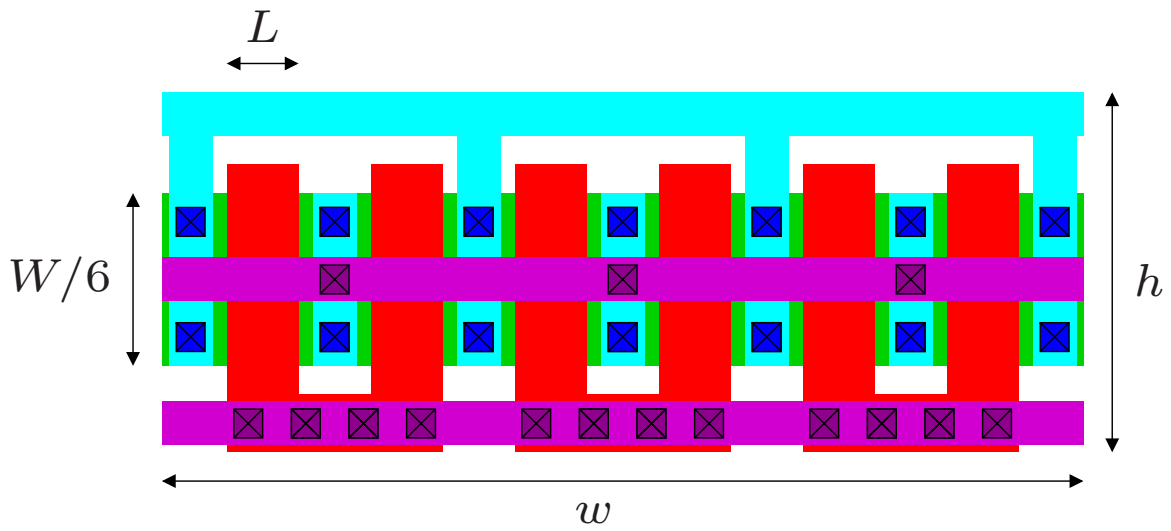
  . . . a posynomial constraint

Power versus bandwidth trade-off



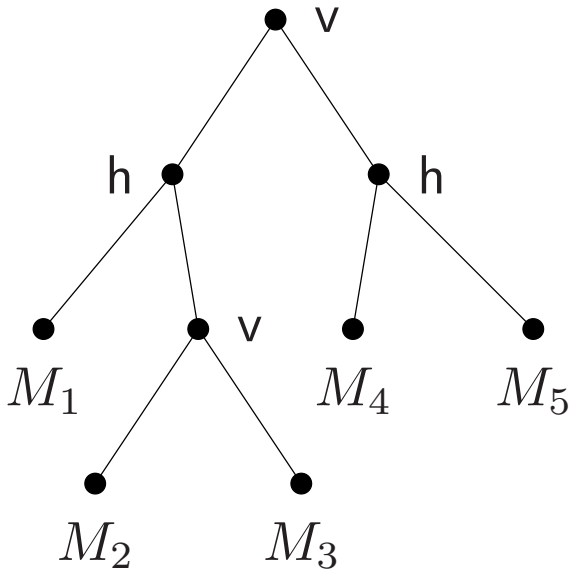
Joint electrical/physical design

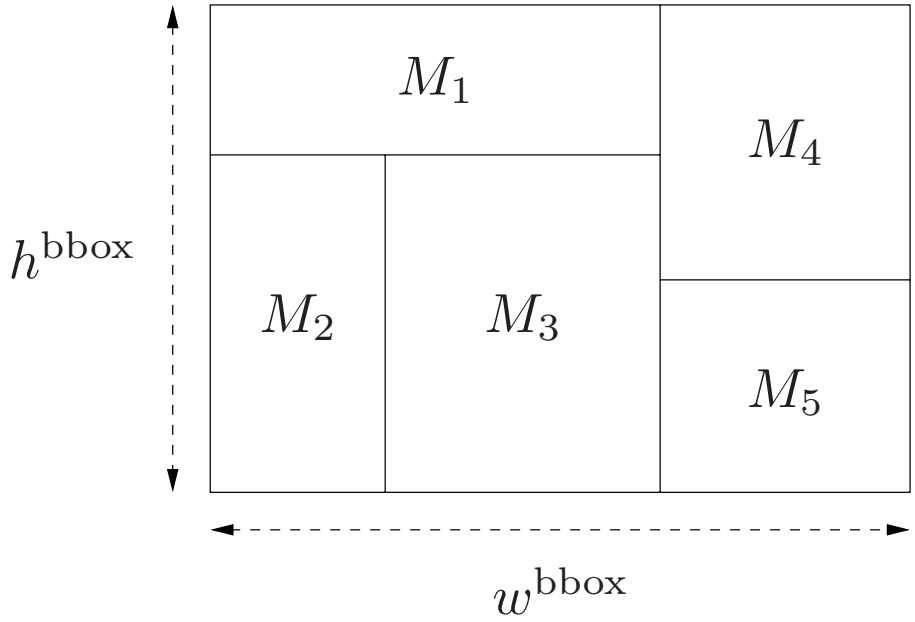
- each device has a (physical) cell width w and height h for floor planning
- devices are folded into multiple fingers
- (approximate) posynomial or monomial relations link electrical variables (I, L, W) and physical variables (w, h), $e.g.$,
  - cell area is at least $4\times$ active area: $wh \geq 4WL$
  - cell aspect ratio limited to 5:1: $1/5 \leq w/h \leq 5$



Slicing tree layout scheme

- vertical and horizontal slices fix relative placement of device cells
- leaves are device cells; root is bounding box





Slicing tree constraints

- introduce width, height for each node in slicing tree
- for each vertical slice with parent a and children b, c add constraints

$$w_a = w_b + w_c, \qquad h_a = \max\{h_b, h_c\}$$

- for each horizontal slice with parent a and children b, c add constraints

$$w_a = \max\{w_b, w_c\}, \qquad h_a = h_b + h_c$$

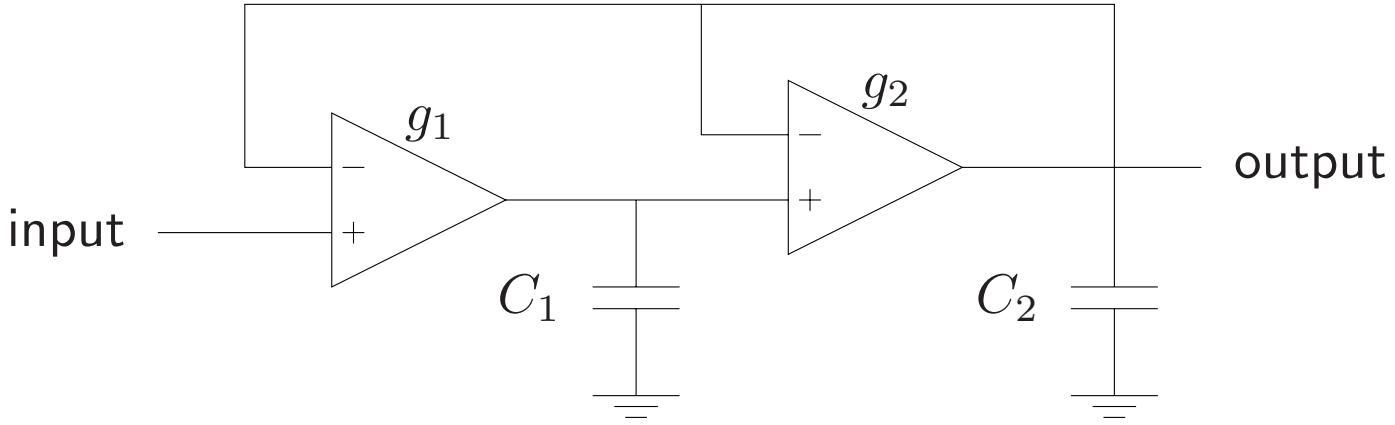
- shows width and height of bounding box and each node is generalized posynomial of device cell widths, heights
- resulting GP formulation is very sparse

Joint electrical/physical design via GP

- form **one** GP that includes
  - electrical variables, constraints ($I_i, L_i, W_i, g_{\mathrm{m},i} \ldots$)
  - physical variables, constraints ($w_i, h_i, w^{\mathrm{bbox}}, h^{\mathrm{bbox}}, \ldots$)
  - coupling constraints ($w_i h_i \geq 4 W_i L_i$, . . .)
- solve it all together
- extensions: can add
  - parasitic estimates
  - more accurate expressions for device cell dimensions
  - channels for routing

Optimal filter implementation

simple Gm-C two-pole lowpass filter



transfer function is

$$H(s) = \frac{1}{1 + t_1 s + t_1 t_2 s^2}, \qquad t_1 = C_1/g_1, \quad t_2 = C_2/g_2$$

g_i is amplifier transconductance

Noise analysis

- N_i is input referred (white) amplifier input-referred voltage density
- spectral density of output noise is

$$N(\omega)^2 = \frac{N_1^2 + \omega^2 N_2^2}{(1 - t_1 t_2 \omega^2)^2 + t_1^2 \omega^2}$$

- root-mean-square output noise voltage is

$$M = \left(\int_0^\infty N(\omega)^2 \, d\omega \right)^{1/2} = \left(\alpha N_1^2 + \beta N_2^2 \right)^{1/2}$$

Amplifier and capacitor implementation models

- each amplifier has **private variables** u ($e.g.$, device lengths & widths) and constraints
- transconductance g is monomial in u; area A^{amp}, power P, input-referred noise density N are posynomial in u
- each capacitor has private variables v ($e.g.$, physical dimensions) and constraints
- capacitance C is monomial in v; area A^{cap} is posynomial
- design variables are u_1, u_2, v_1, v_2

Optimal filter implementation problem

- filter is Butterworth with frequency ω_{c}:

$$t_1 = \sqrt{2}/\omega_c, \qquad t_2 = (1/\sqrt{2})/\omega_c$$

- minimize total power of implementation, subject to area, output noise limits:

$$\begin{array}{ll} \text{minimize} & P(u_1) + P(u_2) \\ \text{subject to} & t_1 = \sqrt{2}/\omega_c, \qquad t_2 = (1/\sqrt{2})/\omega_c \\ & A^{\mathrm{amp}}(u_1) + A^{\mathrm{amp}}(u_2) + A^{\mathrm{cap}}(v_1) + A^{\mathrm{cap}}(v_2) \leq A^{\max} \\ & M = (\omega_c/4\sqrt{2})(N_1^2 + 2N_2^2)^{1/2} \leq M^{\max} \end{array}$$

- a **GGP** in the variables u_1, u_2, v_1, v_2

Example

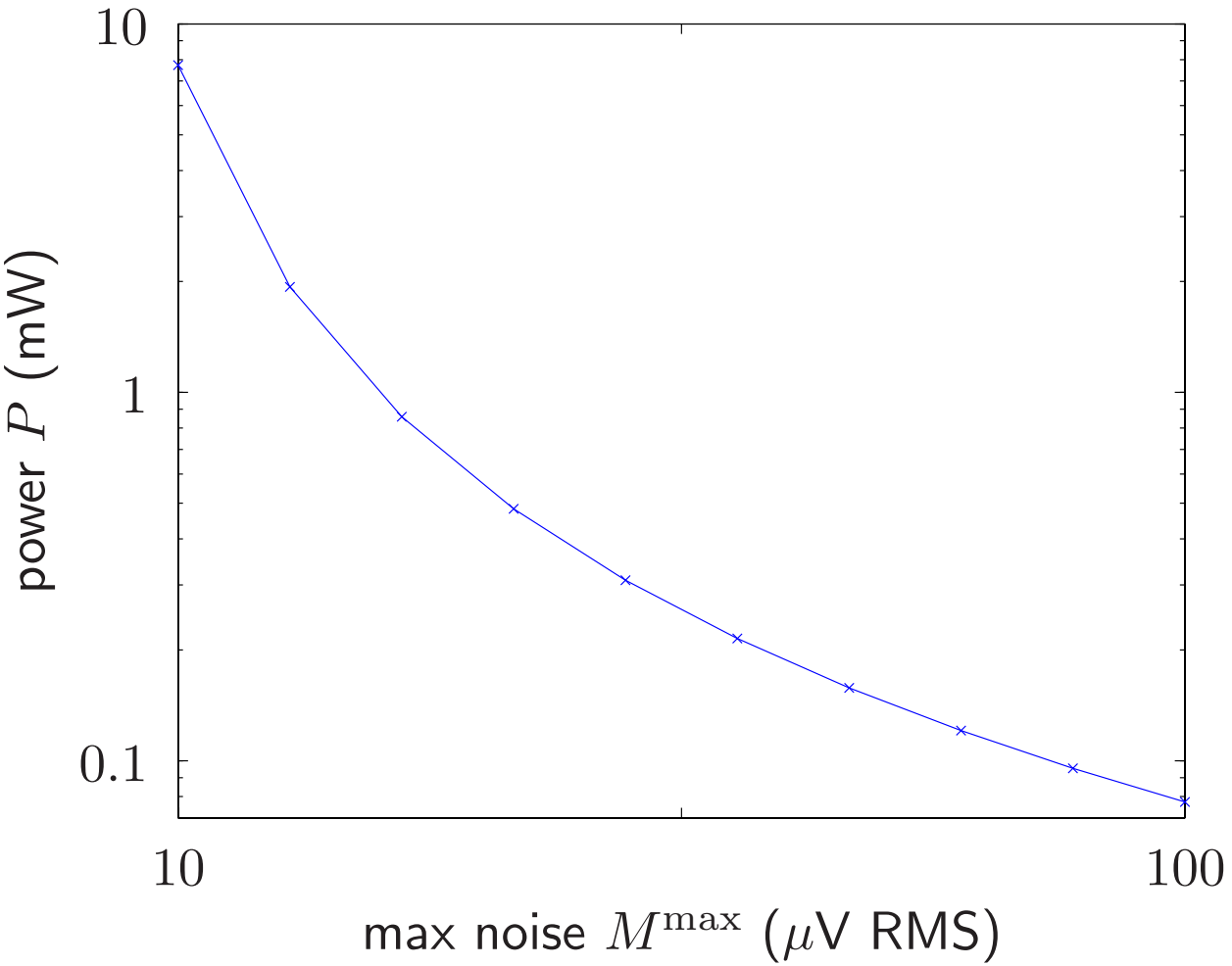
- Butterworth filter with $\omega_c = 10^8 \mathrm{rad/s}$
- private variables in amplifiers: (equivalent) L, W
- amplifier model:

$$A = WL, \qquad P = 2.5 \cdot 10^{-4} W/L,$$
$$g = 4 \cdot 10^{-5} W/L, \qquad N = \sqrt{7.5 \cdot 10^{-16} L/W}$$

  (based on simple model with $V_{\mathrm{dd}} = 2.5$, $V_{\mathrm{gov}} = 0.2$)

- private variable in capacitors is area A^{cap}; $C = 10^{-4} A^{\mathrm{cap}}$
- $A^{\max} = 4 \cdot 10^{-6}$

Power versus noise trade-off



Monomial and Posynomial Fitting

A basic property of posynomials

- if f is a monomial, then $\log f(e^y)$ is **affine** (linear plus constant)
- if f is a posynomial, then $\log f(e^y)$ is **convex**
- roughly speaking, a posynomial is convex when plotted on log-log plot
- midpoint rule for posynomial f:
  - let z be elementwise geometric mean of x, y, *i.e.*, $z_i = \sqrt{x_i y_i}$
  - then $f(z) \leq \sqrt{f(x)f(y)}$
- a converse: if $\log \phi(e^y)$ is convex, then ϕ can be approximated as well as you like by a posynomial

Convexity in circuit design context

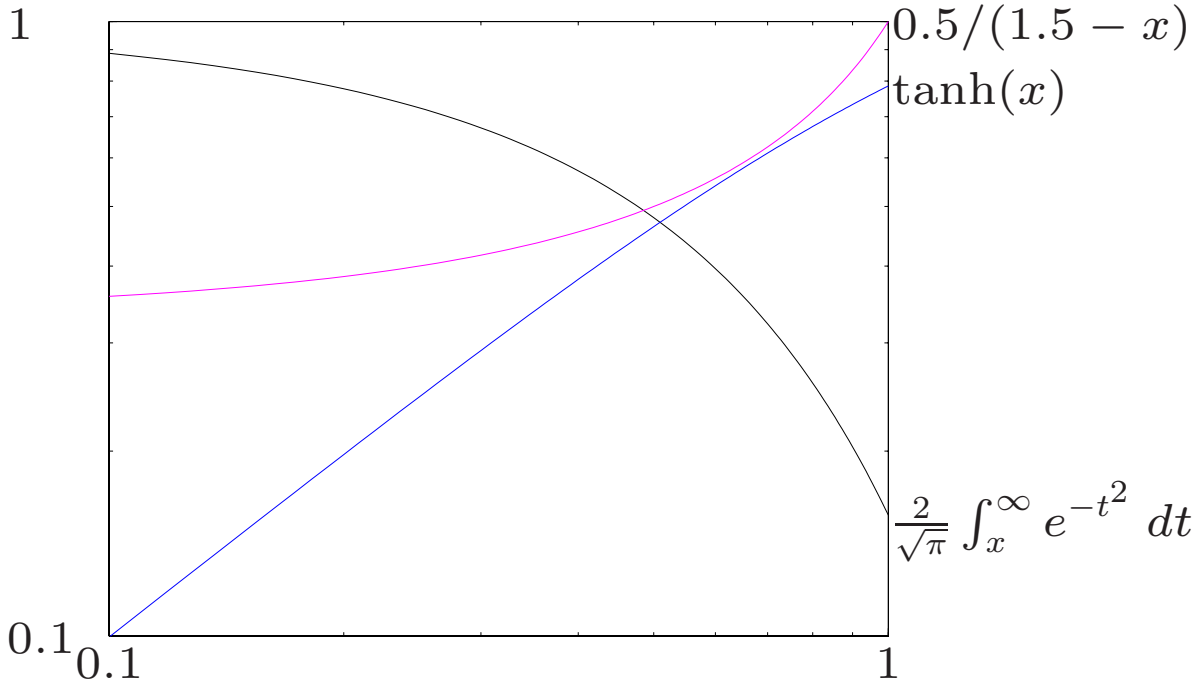
- consider circuit with design variables $W_1, \ldots, W_n$ (say) & performance measure $\phi(W_1, \ldots, W_n)$ ($e.g.$, power, delay, area)
- two designs: $W_i^{(a)}$ & $W_i^{(b)}$, with performance $\phi^{(a)}$ & $\phi^{(b)}$
- form **geometric mean** compromise design with $W_i^{(c)} = \sqrt{W_i^{(a)} W_i^{(b)}}$, performance $\phi^{(c)}$
- if ϕ is generalized posynomial, then we have $\phi^{(c)} \leq \sqrt{\phi^{(a)} \phi^{(b)}}$
- this is **not obvious**

Monomial/posynomial approximation: Theory

when can a function f be approximated by a monomial or generalized posynomial?

- form function $F(y) = \log f(e^y)$
- f can be approximated by a monomial if and only if F is nearly affine (linear plus constant)
- f can be approximated by a generalized posynomial if and only if F is nearly convex

Examples



- $\tanh(x)$ can be reasonably well fit by a monomial
- $0.5/(1.5-x)$ can be fit by a generalized posynomial
- $(2/\sqrt{\pi})\int_x^\infty e^{-t^2}\,dt$ cannot be fit very well by a generalized posynomial

What problems can be approximated by GGPs?

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 1, \quad i = 1, \ldots, m \\ & g_i(x) = 1, \quad i = 1, \ldots, p \end{array}$$

- transformed objective and inequality constraint functions $F_i(y) = \log f_i(e^y)$ must be nearly convex
- transformed equality constraint functions $G_i(y) = \log G_i(e^y)$ must be nearly affine

Monomial fitting via log-regression

find coefficient $c > 0$ and exponents $a_1, \ldots, a_n$ of monomial f so that

$$f(x^{(i)}) \approx f^{(i)}, \qquad i = 1, \ldots, N$$

- rewrite as

$$\begin{aligned} \log f(x^{(i)}) &= \log c + a_1 \log x_1^{(i)} + \cdots + a_n \log x_n^{(i)} \\ &\approx \log f^{(i)}, \qquad i = 1, \ldots, N \end{aligned}$$

- use least-squares (regression) to find $\log c$, $a_1, \ldots, a_n$ that minimize

$$\sum_{i=1}^{N} \left(\log c + a_1 \log x_1^{(i)} + \cdots + a_n \log x_n^{(i)} - \log f^{(i)} \right)^2$$

Posynomial fitting via Gauss-Newton

find coefficients and exponents of posynomial f so that

$$f(x^{(i)}) \approx f^{(i)}, \qquad i = 1, \ldots, N$$

- minimize sum of squared fractional errors

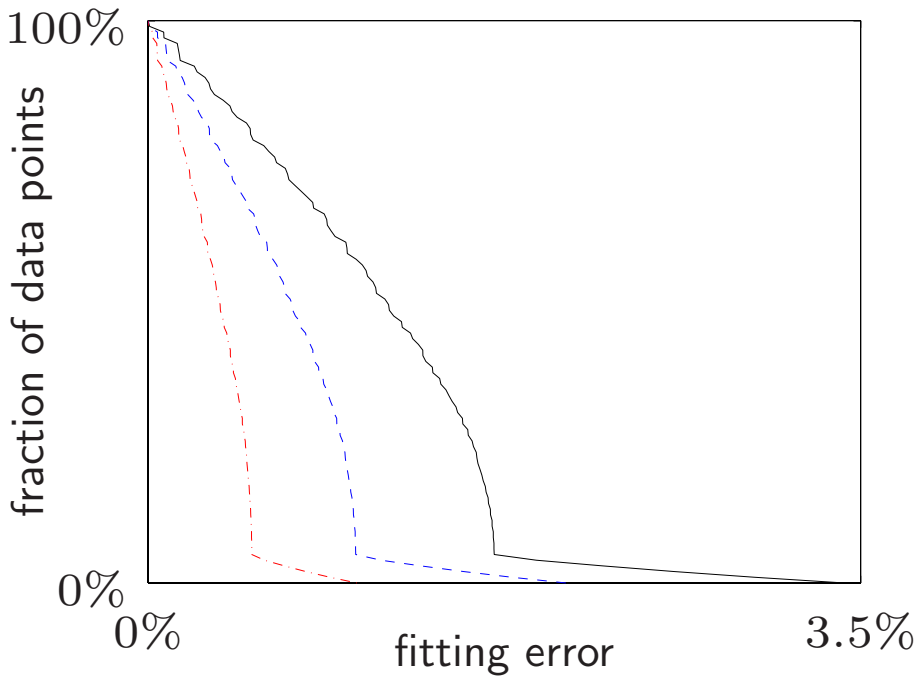
$$\sum_{i=1}^{N} \left(\frac{f^{(i)} - f(x^{(i)})}{f^{(i)}} \right)^2$$

  can be (locally) solved by Gauss-Newton method

- needs starting guess for coefficients, exponents

Posynomial fitting example

- 1000 data points from $f(x) = (1 - 0.5(x_1^2 + x_2 + x_3^{-1} - 1)^2)^{1/2}$ over $0.1 \leq x_i \leq 1$
- cumulative error distribution for 3-, 5-, and 10-term posynomial fits



Conclusions

Conclusions

(generalized) geometric programming

- comes up in a variety of circuit sizing contexts
- can be used to formulate a variety of problems
- admits fast, reliable solution of large-scale problems
- is good at concurrently balancing lots of coupled constraints and objectives
- is useful even when problem has discrete constraints

Approach

- most problems don't come naturally in GP form; be prepared to reformulate and/or approximate
- GP modeling is not a "try my software" method; it requires thinking
- our approach:
  - start with simple analytical models (RC, square-law, Pelgrom, . . .) to verify GP might apply
  - then fit GP-compatible models to simulation or measured data
  - for highest accuracy, revert to local method for final polishing

- looking for keys under street light
  (not where keys were lost, but lighting is good)

- forcing problems into GP-compatible form
  (problems aren't GPs, but solving is good)

References

- *A tutorial on geometric programming*
- *Digital circuit sizing via geometric programming*
- *Analog circuit design via geometric programming*
- *Convex optimization*, Cambridge Univ. Press 2004

(these include hundreds of references)

available at `www.stanford.edu/~boyd/research.html`