1 Lecture 1b: DFM For Dummies

1.1 @luk036

2023-09-06

1.2 Abstract

DFM is an engineering practice that optimizes the manufacturing ease and production cost of ICs while meeting performance, power, and reliability requirements. As ICs become more miniaturized and complex, the manufacturing process becomes sensitive to variations and defects, which can degrade chip quality and functionality. DFM helps address various manufacturing issues. DFM can be applied to various aspects of IC design, such as circuit design, logic design, layout design, verification, and testing. The lecture covers the general guidelines and best practices of DFM for layout design. Applying DFM techniques in the physical design stage of IC development can help reduce the number of design iterations, improve collaboration with foundries, enhance product performance and functionality, and achieve faster time-to-market and lower production costs. The lecture discusses the challenges of DFM and its market share, including DFM analysis and verification, enhancement, optimization, and the algorithms that solve DFM problems. The course structure highlights the problems that arise from DFM and abstracts them into mathematical forms.

1.3 Faster, smaller & smarter

1.4 Silicon Gold Rush?

1.5 Current Transistors

- High-K dielectrics, Metal Gate (HKMG)
- "3D" gate

The significance of High-K dielectrics is that they have a higher dielectric constant than traditional silicon dioxide (SiO2) dielectrics. This allows for a thicker gate oxide layer to be used without increasing the gate capacitance,

图 1: iPhoneX

图 2: SMIC

which can improve the transistor's performance and reduce leakage current.

Metal Gate refers to the use of a metal material (such as tungsten or tantalum) for the gate electrode, instead of the traditional polysilicon material. This is significant because metal gates can provide better control over the transistor's threshold voltage, which can improve its performance and reduce variability.

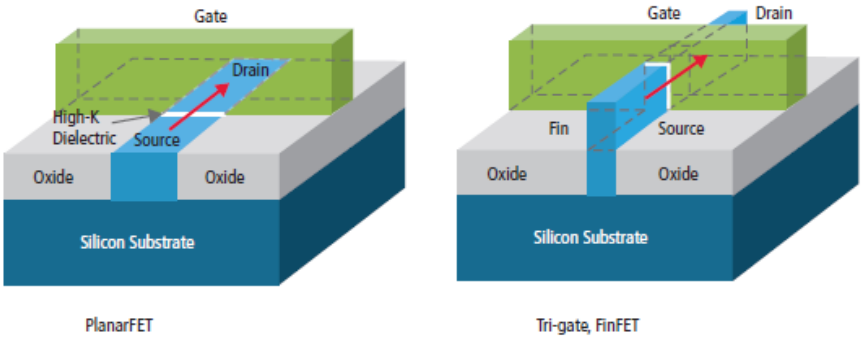


图 3: FinFET

1.6 Lithography

- Photo-resist coating
- Illumination
- Exposure
- Etching
- Impurities Doping
- Metal connection

1.7 Process-Design Gap

1.8 Problem Visualization

One of the main impacts of lithography is that it can cause variations in the dimensions and shapes of the IC's features, which can negatively impact the performance and yield of the IC. This is because lithography is a complex process that involves the use of light to transfer a pattern from a mask to

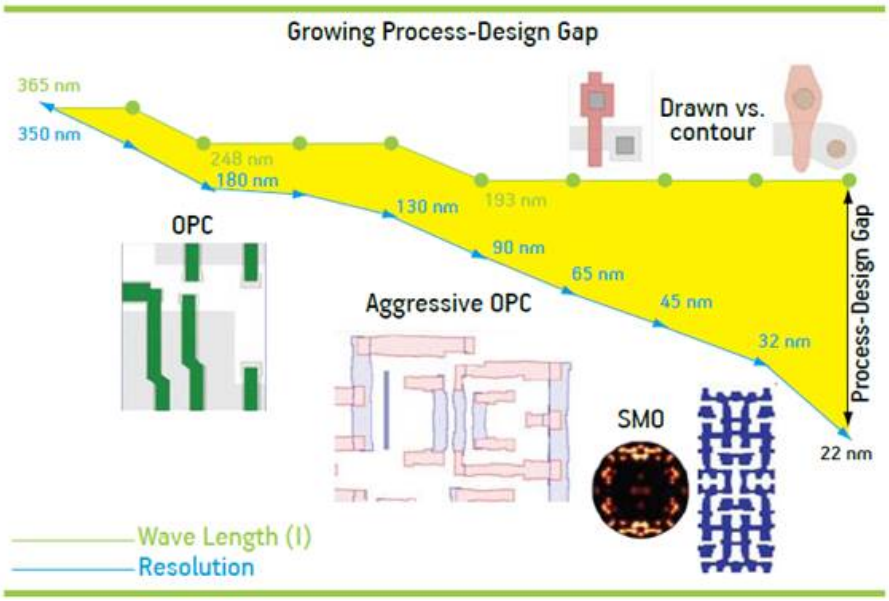


图 4: Process-Design Gap

a wafer. Variations in the intensity, wavelength, and angle of the light can cause deviations in the dimensions and shapes of the features, which can lead to process-induced variation.

1.9 Chemical Mechanical Polishing

Chemical Mechanical Polishing (CMP) is a process used in semiconductor manufacturing to planarize the surface of a wafer. CMP is one of the steps involved in the fabrication of integrated circuits, specifically in the metal connection stage.

1.10 Chemical Mechanical Polishing

In terms of bridging the Process-Design Gap, CMP can help address the issue of process-induced variation by improving the uniformity of the wafer surface. This is important because process-induced variation can cause deviations in the dimensions and electrical properties of the transistors, which

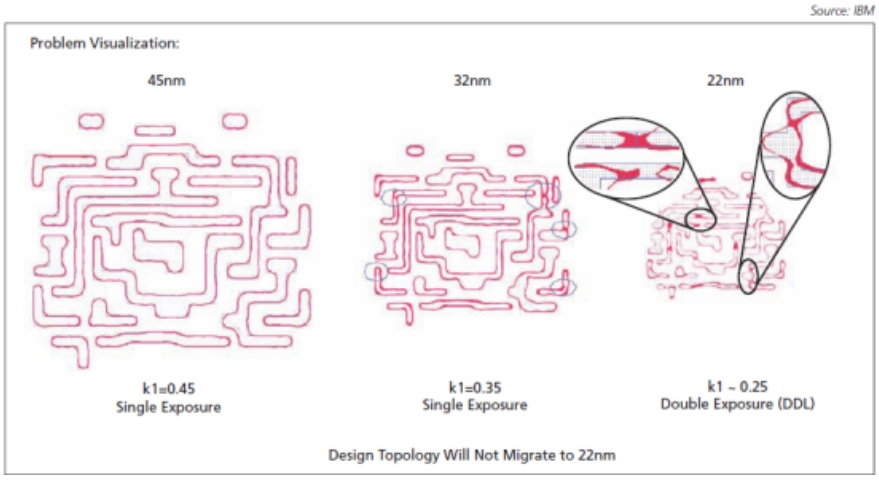

Source: IBM
Problem Visualization:
45nm
32nm
22nm
k1=0.45
Single Exposure
k1=0.35
Single Exposure
k1 ~ 0.25
Double Exposure (DDL)
Design Topology Will Not Migrate to 22nm

图 5: ibm

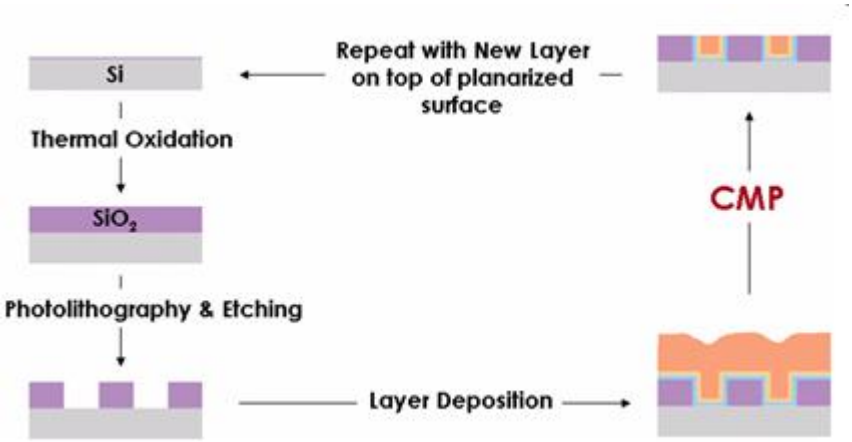

Si
Repeat with New Layer on top of planarized surface
Thermal Oxidation
SiO2
CMP
Photolithography & Etching
Layer Deposition

图 6: CMP

can negatively impact the performance and yield of the IC.

1.11 ECP & CMP

.pull-left[

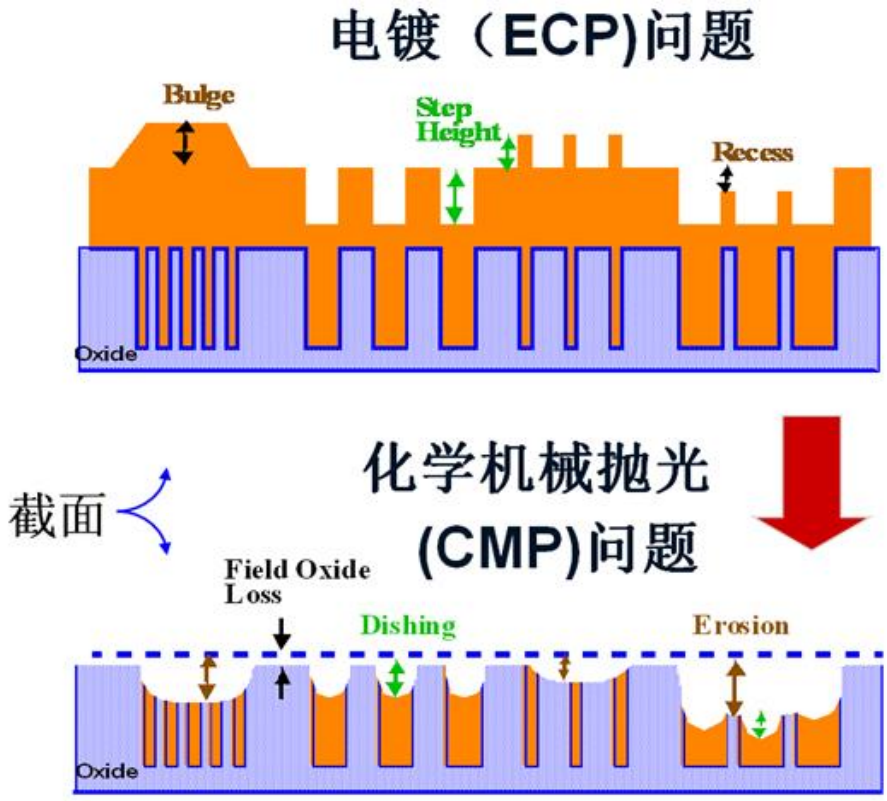


图 7: ECP

] .pull-right[

By using CMP to planarize the wafer surface, designers can reduce the variability in the thickness of the metal layers, which can improve the accuracy and consistency of the IC's electrical properties. This, in turn, can help bridge the Process-Design Gap by ensuring that the ICs are manufactured according to the intended design specifications.

]

count: false class: nord-light, middle, center

2 Process Variation

2.1 Total Thickness Variation Per Node

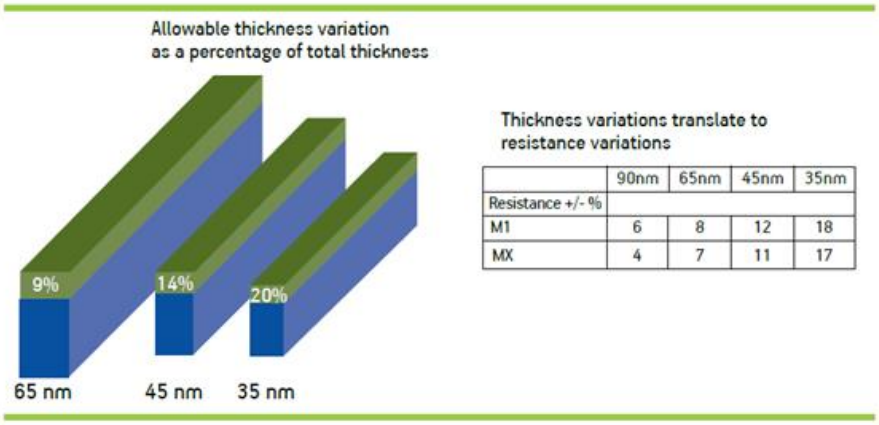


图 8: Thickness Variation

2.2 "Slippery Fish" at 45nm

- Process variation, impacting yield and performance
- More restricted design rules (RDRs)
  - +3 or more rules at 45nm
  - +100 or more rules at 32nm
  - +250 or more rules at 22nm
- More rules implies larger die size, lower performance
- 10nm is not sci-fiction due to FinFET technology

count: false class: nord-light, middle, center

3 DFM

3.1 What is DFM?

- Design for ?
- Design for Manufacturing
- Design for Manufacturability
  - Refer to a group of challenges less than 130nm
  - The engineering practice of designing integrated circuits (ICs) to optimize their manufacturing ease and production cost given performance, power and reliability requirements
  - A set of techniques to modify the design of ICs to improve their functional yield, parametric yield or their reliability

3.2 Why is it important?

- Achieving high-yielding designs in the state-of-the-art VLSI technology is extremely challenging due to the miniaturization and complexity of leading-edge products
- The manufacturing process becomes more sensitive to variations and defects, which can degrade the quality and functionality of the chips
- DFM can help to address various manufacturing issues, such as lithography hotspots, CMP dishing and erosion, antenna effects, electromigration, stress effects, layout-dependent effects and more

3.3 How is it applied?

- DFM can be applied to various aspects of IC design, such as circuit design, logic design, layout design, verification and testing
- Each aspect has its own specific DFM guidelines and best practices that designers should follow to ensure manufacturability
- For example, some general DFM guidelines for layout design are:
  - Use regular and uniform layout structures
  - Avoid narrow or long metal wires
  - Avoid acute angles or jogs in wires

- Avoid isolated or floating features
- Use dummy fill to improve planarity and density uniformity
- Use recommended design rules and constraints from foundries

3.4 What are the benefits?

- By applying DFM techniques in the physical design stage of IC development, designers can:
  - Reduce the number of design iterations
  - Improve the collaboration with foundries
  - Enhance the product performance and functionality
  - Achieve faster time to market and lower production costs

3.5 DFM Market Share 2008

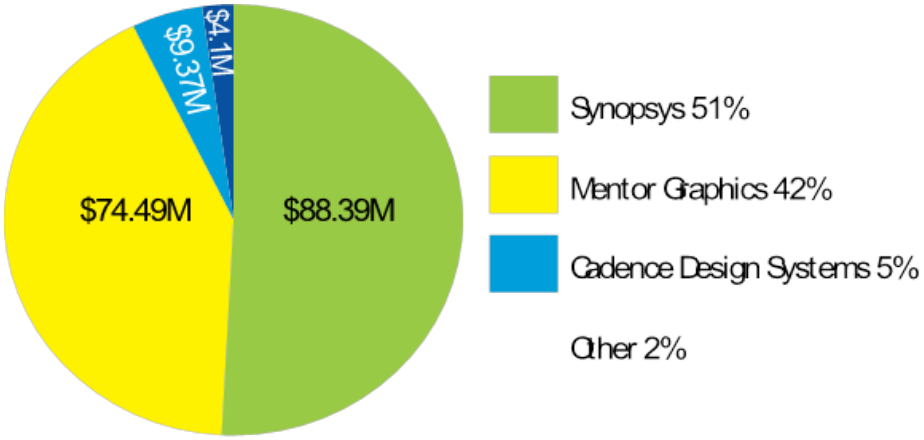


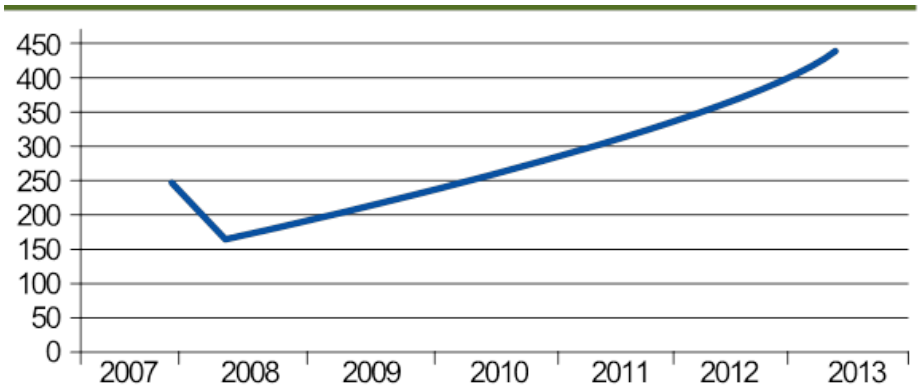
图 9: Market Share

3.6 DFM Forecast 2009 in $M

3.7 Increasing Importance of DFM

3.8 DFM Analysis and Verification

- Critical area analysis
- CMP modeling



	2007	2008	2009	2010	2011	2012	2013
RET	$208	$131	$144	$158	$174	$192	$211
DFY	$37	$46	$62	$84	$115	$156	$212
Total DRM	$246	$176	$206	$243	$289	$347	$423

图 10: forecast

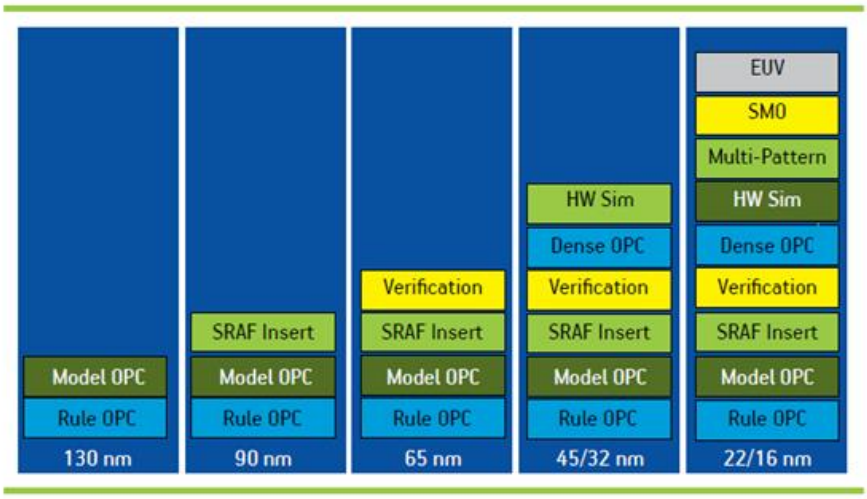


图 11: trend

- Statistical timing analysis
- Pattern matching
- Lithography simulation
- Lithographic hotspot verification

3.9 2D Pattern Matching in DRC+

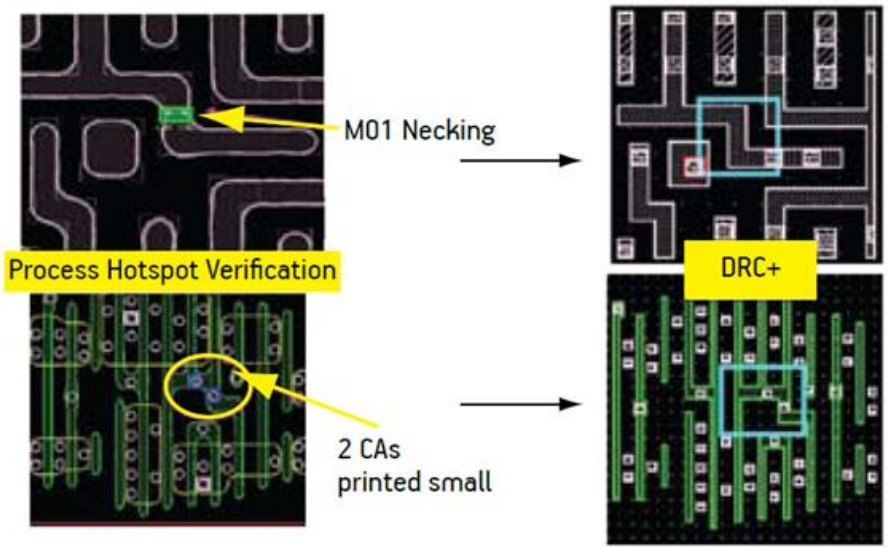


图 12: DRC+

3.10 Contour Based Extraction

3.11 DFM Enhancement and Optimization

- Wire spreading
- Dummy Filling
- Redundant Via Insertion
- Optical proximity correlation (OPC)
- Phase Shift Masking (PSM)
- Double/Triple/Multiple Patterning
- Statistical timing and power optimization

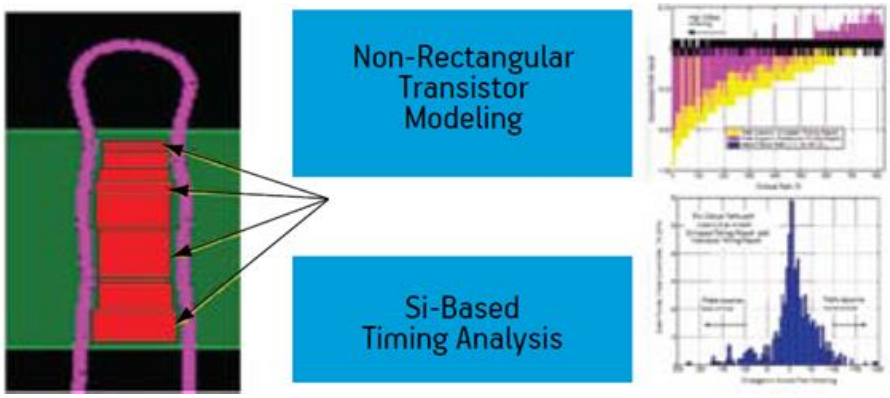


图 13: contour

3.12 Dummy Filling

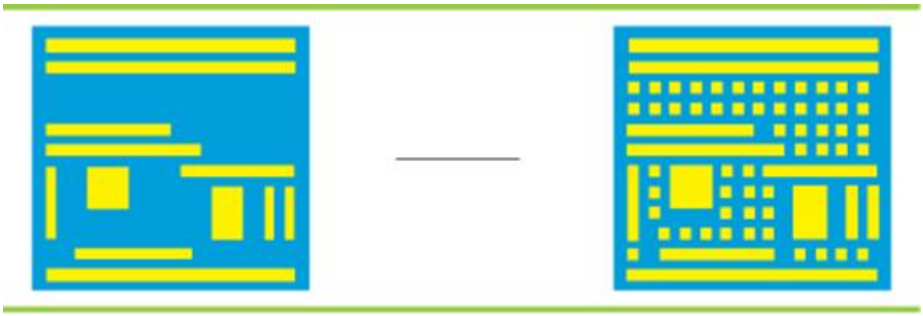


图 14: filling

3.13 “Smart” Filling

3.14 Redundant Via Insertion

- Also known as double via insertion.
- Post-routing RVI (many EDA tools already have this feature)
- Considering RVI during routing

???

Looks good, right?

But actually only few people are using this!

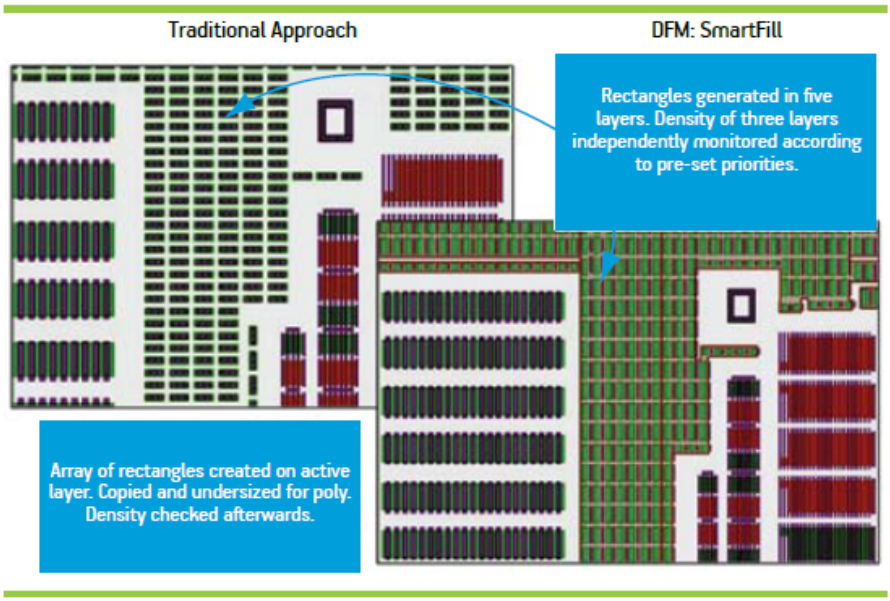


图 15: “Smart” Filling

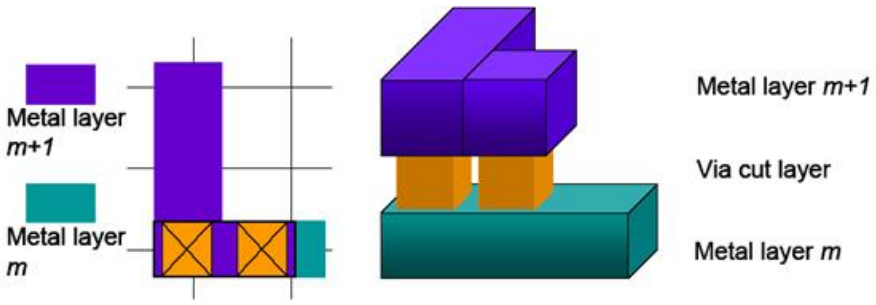


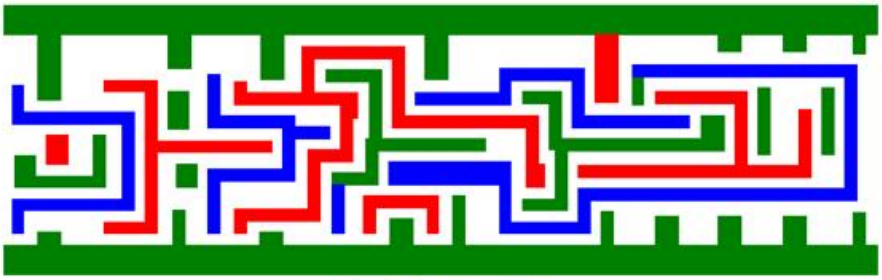
Fig. 1 Illustration for redundant via insertion.

图 16: RVI

Why?

3.15 Multiple Patterning (MPL)

- Instead of exposing the photoresist layer once under one mask, MPL exposes it twice by splitting the mask into "k" parts, each with features less dense.



(b) Triple patterning

图 17: MPL

3.16 What are the challenges of DFM?

- DFM is not a fixed set of rules, but rather a flexible and evolving methodology that depends on the product requirements, the manufacturing technology and the industry standards
- DFM can also be combined with other design methodologies, such as DFT, DFR, DFLP and DFS, to create a holistic approach to product development
- DFM requires strong capabilities in research, supply chain, talent, IP protection and government policies

3.17 Course Structure

- Describe the DFM problems that arise from.
- Abstract the problems in mathematical forms
- Describe the algorithms that solve the problems
- Discuss the alternative algorithms and possible improvement.

- Discuss if the algorithms can be applied to other area.
- Only describe the key idea in lectures. Details are left for paper reading if necessary.

3.18 Not covered

- Algorithms for 3D problems
- Packaging
- Machine Learning/AI Based algorithm

4 Lecture 2c: Introduction to Convex Programming

4.1 @luk036

2023-09-27

4.2 Abstract

This lecture provides an introduction to the convex programming and covers various aspects of optimization. The lecture begins with an overview of optimization, including linear and nonlinear programming, duality and convexity, and approximation techniques. It then delves into more specific topics within continuous optimization, such as linear programming problems and their standard form, transformations to standard form, and the duality of linear programming problems. The lecture also touches on nonlinear programming, discussing the standard form of an NLPP (nonlinear programming problem) and the necessary conditions of optimality known as the Karush-Kuhn-Tucker (KKT) conditions. Convexity is another important concept explored in the document, with explanations on the definition of convex functions and their properties. The lecture also discusses the duality of convex optimization problems and their usefulness in computation. Finally, the document briefly mentions various unconstrained optimization techniques, descent methods, and approximation methods under constraints.

4.3 Overview

- Introduction
- Linear programming
- Nonlinear programming
- Duality and Convexity
- Approximation techniques
- Convex Optimization
- Books and online resources.

4.4 Classification of Optimizations

- Continuous
  - Linear vs Non-linear
  - Convex vs Non-convex
- Discrete
  - Polynomial time Solvable
  - NP-hard
    - Approximatable
    - Non-approximatable
- Mixed

4.5 Continuous Optimization

4.6 Linear Programming Problem

- An LPP in standard form is:

$$\min\{c^{\mathsf{T}}x \mid Ax = b, x \geq 0\}.$$

- The ingredients of LPP are:
  - An $m \times n$ matrix A, with $n > m$
  - A vector $b \in \mathbb{R}^m$
  - A vector $c \in \mathbb{R}^n$

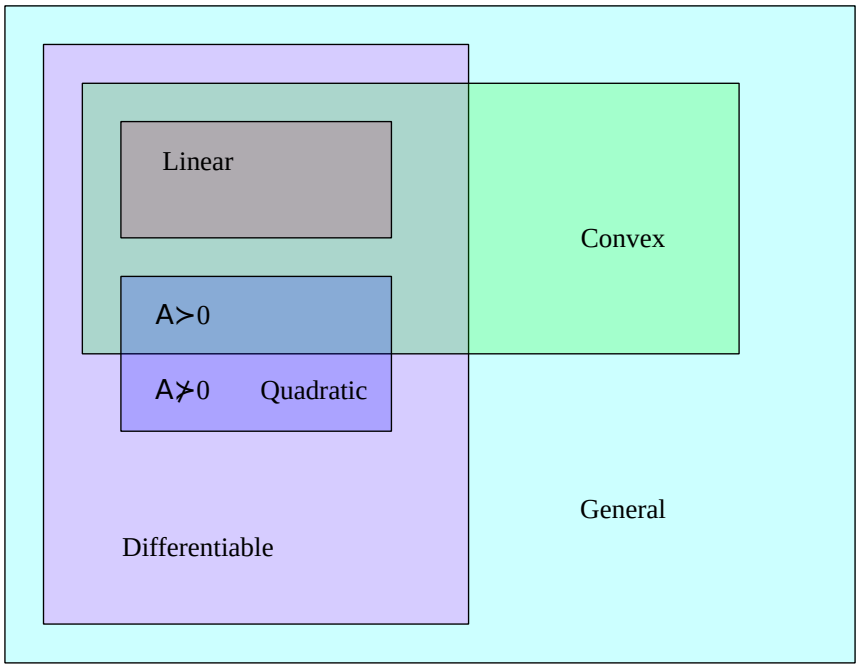


图 18: classification

4.7 Example

$$\begin{array}{lll} \text{minimize} & 0.4x_1 + 3.4x_2 - 3.4x_3 & \\ \text{subject to} & 0.5x_1 + 0.5x_2 & = 3.5 \\ & 0.3x_1 - 0.8x_2 + 8.4x_2 & = 4.5 \\ & x_1, x_2, x_3 \geq 0 & \end{array}$$

4.8 Transformations to Standard Form

- Theorem: Any LPP can be transformed into the standard form.
- Variables not restricted in sign:
  - Decompose x to two new variables $x = x_1 - x_2, x_1, x_2 \geq 0$
- Transforming inequalities into equalities:
  - By putting slack variable $y = b - Ax \geq 0$
  - Set $x' = (x, y), A' = (A, 1)$
- Transforming a max into a min
  - max(expression) = min(−expression);

4.9 Duality of LPP

- If the primal problem of the LPP: $\min\{c^\mathsf{T} x \mid Ax \geq b, x \geq 0\}$.
- Its dual is: $\max\{y^\mathsf{T} b \mid A^\mathsf{T} y \leq c, y \geq 0\}$.
- If the primal problem is: $\min\{c^\mathsf{T} x \mid Ax = b, x \geq 0\}$.
- Its dual is: $\max\{y^\mathsf{T} b \mid A^\mathsf{T} y \leq c\}$.

4.10 Nonlinear Programming

- The standard form of an NLPP is

$$\min\{f(x) \mid g(x) \leq 0, h(x) = 0\}.$$

- Necessary conditions of optimality, Karush- Kuhn-Tucker (KKT) conditions:
  - $\nabla f(x) + \mu \nabla g(x) + \lambda \nabla h(x) = 0$,
  - $\mu g(x) = 0$,
  - $\mu \geq 0, g(x) \leq 0, h(x) = 0$

4.11 Convexity

- A function f: $K \subseteq \mathbb{R}^n \mapsto R$ is convex if K is a convex set and $f(y) \geq f(x) + \nabla f(x)(y - x),\ y, x \in K$.
- **Theorem**: Assume that f and g are convex differentiable functions. If the pair (x, m) satisfies the KKT conditions above, x is an optimal solution of the problem. If in addition, f is strictly convex, x is the only solution of the problem.

  (Local minimum = global minimum)

4.12 Duality and Convexity

- Dual is the NLPP:

  $$\max\{\theta(\mu, \lambda) \mid \mu \geq 0\},$$

  where $\theta(\mu, \lambda) = \inf_x[f(x) + \mu g(x) + \lambda h(x)]$
- Dual problem is always convex.
- Useful for computing the lower/upper bound.

4.13 Applications

- Statistics
- Filter design
- Power control
- Machine learning
  - SVM classifier
  - logistic regression

class: nord-light, middle, center

5 Convexify the non-convex's

5.1 Change of curvature: square

Transform:

$$0.3 \leq \sqrt{x} \leq 0.4$$

into:

$$0.09 \leq x \leq 0.16\,.$$

Note that $\sqrt{\cdot}$ are **monotonic concave** functions in $(0, +\infty)$.

Generalization: - Consider $|H(\omega)|^2$ (power) instead of $|H(\omega)|$ (magnitude).
- square root -> Spectral factorization

5.2 Change of curvature: square

Transform:

$$x^2 + y^2 \geq 0.16, \quad \text{(non-convex)}$$

into:

$$x' + y' \geq 0.16, \quad x', y' \geq 0$$

Then:

$$x_{\text{opt}} = \pm\sqrt{x'_{\text{opt}}}, \quad y_{\text{opt}} = \pm\sqrt{y'_{\text{opt}}}.$$

5.3 Change of curvature: sine

Transform:

$$\sin^2 x \leq 0.4, \quad 0 \leq x \leq \pi/2$$

into:

$$y \leq 0.4, \quad 0 \leq y \leq 1$$

Then:

$$x_{\text{opt}} = \sin^{-1}(\sqrt{y_{\text{opt}}}).$$

Note that $\sin(\cdot)$ are monotonic concave functions in $(0, \pi/2)$.

5.4 Change of curvature: log

Transform:

$$\pi \leq x/y \leq \phi$$

into:

$$\pi' \leq x' - y' \leq \phi'$$

where $z' = \log(z)$.

Then:

$$z_{\text{opt}} = \exp(z'_{\text{opt}}).$$

Generalization: - Geometric programming

5.5 Change of curvature: inverse

Transform:

$$\log(x) + \frac{c}{x} \leq 0.3,\ x > 0$$

into:

$$-\log(y) + c \cdot y \leq 0.3,\ y > 0\,.$$

Then:

$$x_{\text{opt}} = y_{\text{opt}}^{-1}.$$

Note that $\sqrt{\cdot}$, $\log(\cdot)$, and $(\cdot)^{-1}$ are monotonic functions.

5.6 Generalize to matrix inequalities

Transform:

$$\log(\det X) + \text{Tr}(X^{-1}C) \leq 0.3,\ X \succ 0$$

into:

$$-\log(\det Y) + \text{Tr}(Y \cdot C) \leq 0.3,\ Y \succ 0$$

Then:

$$X_{\text{opt}} = Y_{\text{opt}}^{-1}.$$

5.7 Change of variables

Transform:

$$(a + b \cdot y)x \leq 0,\ x > 0$$

into:

$$a \cdot x + b \cdot z \leq 0,\ x > 0$$

where $z = yx$.

Then:

$$y_{\text{opt}} = z_{\text{opt}} x_{\text{opt}}^{-1}$$

5.8 Generalize to matrix inequalities

Transform:

$$(A + BY)X + X(A + BY)^T \prec 0,\ X \succ 0$$

into:

$$AX + XA^T + BZ + Z^T B^T \prec 0,\ X \succ 0$$

where $Z = YX$.

Then:

$$Y_{\text{opt}} = Z_{\text{opt}} X_{\text{opt}}^{-1}$$

5.9 Some operations that preserve convexity

- $-f$ is concave if and only if f is convex.
- Nonnegative weighted sums:
  - if $w_1, \ldots, w_n \geq 0$ and $f_1, \ldots, f_n$ are all convex, then so is $w_1 f_1 + \cdots + w_n f_n$. In particular, the sum of two convex functions is convex.
- Composition:
  - If f and g are convex functions and g is non-decreasing over a univariate domain, then $h(x) = g(f(x))$ is convex. As an example, if f is convex, then so is $e^{f(x)}$, because e^x is convex and monotonically increasing.
  - If f is concave and g is convex and non-increasing over a univariate domain, then $h(x) = g(f(x))$ is convex.
  - Convexity is invariant under affine maps.

5.10 Other thoughts

- Minimizing any quasi-convex function subject to convex constraints can easily be transformed into a convex programming.

- Replace a non-convex constraint with a sufficient condition (such as its lower bound). Less optimal.
- Relaxation + heuristic
- Decomposition

5.11 Unconstraint Techniques

- Line search methods
- Fixed or variable step size
- Interpolation
- Golden section method
- Fibonacci's method
- Gradient methods
- Steepest descent
- Quasi-Newton methods
- Conjugate Gradient methods

5.12 General Descent Method

1. **Input**: a starting point $x \in \text{dom } f$
2. **Output**: x^*
3. **repeat**
    1. Determine a descent direction p.
    2. Line search. Choose a step size $\alpha > 0$.
    3. Update. $x := x + \alpha p$
4. **until** stopping criterion satisfied.

5.13 Some Common Descent Directions

- Gradient descent: $p = -\nabla f(x)^\mathsf{T}$
- Steepest descent:
    - $\triangle x_{nsd} = \text{argmin}\{\nabla f(x)^\mathsf{T} v \mid \|v\| = 1\}$
    - $\triangle x = \|\nabla f(x)\| \triangle x_{nsd}$ (un-normalized)
- Newton's method:
    - $p = -\nabla^2 f(x)^{-1} \nabla f(x)$

- Conjugate gradient method:
  - p is "orthogonal" to all previous p's
- Stochastic subgradient method:
  - p is calculated from a set of sample data (instead of using all data)
- Network flow problems:
  - p is given by a "negative cycle" (or "negative cut").

5.14 Approximation Under Constraints

- Penalization and barriers
- Dual method
- Interior Point method
- Augmented Lagrangian method

5.15 Books and Online Resources

- Pablo Pedregal. Introduction to Optimization, Springer. 2003 (O224 P371)
- Stephen Boyd and Lieven Vandenberghe, Convex Optimization, Dec. 2002
- Mittlemann, H. D. and Spellucci, P. Decision Tree for Optimization Software, World Wide Web, http://plato.la.asu.edu/guide.html, 2003

6 Non-Parametric Spatial Correlation Estimation

6.1 Abstract

This lecture discusses non-parametric spatial correlation estimation and its importance in analyzing the variability in semiconductor devices. The intra-die variation in these devices can exhibit spatially correlated patterns, which require accurate statistical analysis during the design stage. Anisotropic models are used to allow for variations in gate length, which exhibit stronger

correlation in the horizontal direction than the vertical direction. Non-parametric approaches make sense for correlation functions, as earlier studies that used parametric forms were limited by the assumptions made about the correlation function. This lecture goes on to describe random fields and the properties of correlation functions before diving into problem formulation and solutions using maximum likelihood estimation and least squares estimation.

6.2 Overview

- Motivation:
  - Why is spatial correlation important?
  - Why anisotropic models?
  - Why do non-parametric approaches make sense?
- Problem Formulation
- Non-parametric estimation
  - Least squares estimation
  - Maximum Likelihood estimation
- Numerical experiment
- Conclusion

6.3 Why Spatial Correlation?

- As the minimum feature size of semiconductor devices continues to shrink,
  - Process variations are inevitable. It is desirable to develop more accurate statistical analysis during the design stage.
- Intra-die variation exceeds inter-die variation
  - Becomes dominant over total process variation
  - Often exhibits spatially correlated patterns.
- Applications:
  - Statistical timing analysis -> Clock Skew Scheduling
  - Power/leakage minimization

6.4 Why Anisotropic Model?

- Isotropic assumption assumes that the correlation depends only on the distance between two random variables. It was made to simplify the computation.
- Certain variations, such variations in gate length, exhibit significantly stronger correlation in the horizontal direction than in the vertical direction.

6.5 Why Non-Parametric Approaches?

- In earlier studies, the parametric form of the correlation function was simple, such as an exponential, Gaussian or Matérn function:
- Pros: guaranteed to be **positive definite**.
- Cons:
  - non-convex; may be stuck in a local minimum
  - The actual correlation function may not necessarily be of this form.
  - isotropic model

6.6 Related research

- Piecewise linearization method (imprecise, not positive definite)
- Parametric method (non-convex, too smooth, isotropic)
  - Exponential function
  - Gaussian function
  - Matérn function
- Non-parametric method
  - Polynomial fitting
  - B-spline

6.7 Random Field

- Random field is an indexed family of random variables denote as $\{\tilde{z}(s) : s \in D\}$, where $D \subseteq \mathrm{R}^d$

- Covariance $C(s_i, s_j) = \text{cov}(\tilde{z}(s_i), \tilde{z}(s_j)) = \text{E}[(\tilde{z}(s_i) - \text{E}[\tilde{z}(s_i)])(\tilde{z}(s_j) - \text{E}[\tilde{z}(s_j)])]$
- Correlation $R(s_i, s_j) = C(s_i, s_j)/\sqrt{C(s_i, s_i)C(s_j, s_j)}$
- The field is stationary, or homogeneous, if the distribution is unchanged when the point set is translated.
- The field is isotropic if the distribution is invariant under any rotation.
- In HIF, let $d = \|s_i - s_j\|_2$:
  - $C(s_i, s_j) = C(d)$
  - $R(s_i, s_j) = C(d)/C(0) = \sigma^2\rho(d)$

6.8 Properties of Correlation Function

- Even function, i.e. $\rho(\vec{h}) = \rho(-\vec{h}) \implies$ its Fourier transform is real.
- Positive definiteness (PD) $\implies$ its Fourier transform is positive (Bochner's theorem).
- Monotonicity: correlations are decreasing against h
- Nonnegativeness: no negative correlation
- Discontinuity at the origin: nugget effect.

The nugget effect refers to the discontinuity at the origin in the correlation function of spatially correlated patterns. It indicates the presence of a small, non-zero correlation value between points that are very close to each other. In other words, it represents the variance component that cannot be explained by spatial correlation and is attributed to purely random variation.

6.9 Problem Formulation

- Intra-die variation $\tilde{z} = z_{det} + \tilde{z}_{cor} + \tilde{z}_{rnd}$
  - z_{det}: deterministic component
  - $\tilde{z}_{cor}$: correlated random component
  - $\tilde{z}_{rnd}$: purely random component
- Given M samples $(z_1, z_2, ..., z_M) \in \mathbb{R}^n$.
- Measured covariance matrix Y:
  - $Y = (1/M)\sum_{i=1}^{M} z_i z_i^{\mathsf{T}}$ (unlikely PD)

- In MATLAB, simply call `cov(Zs',1)` to obtain Y.
- In Python, simple call `np.cov(Zs, bias=True)` to obtain Y.

6.10 Nearest PD Matrix Problem

- Given Y. Find a nearest matrix Σ that is positive definite.

$$\begin{array}{ll} \text{minimize} & \|\Sigma - Y\|_F \\ \text{subject to} & \Sigma \succeq 0 \end{array}$$

  where $\|\Sigma - Y\|_F$ denotes the Frobenius norm, $A \succeq 0$ denotes A is positive semidefinite.
- Note:
  1. the problem is convex
  2. the problem can be solved easily using CVX

6.11 Maximum Likelihood Estimation

- Maximum likelihood estimation (MLE):

$$\begin{array}{ll} \text{maximize} & \log\det\Sigma^{-1} - \text{Tr}(\Sigma^{-1}Y) \\ \text{subject to} & \Sigma \succeq 0 \end{array}$$

  where $\text{Tr}(A)$ denotes the trace of A.
- Note: 1st term is concave , 2nd term is convex

6.12 Maximum Likelihood Estimation (cont'd)

- Having $S = \Sigma^{-1}$, the problem becomes convex :

$$\begin{array}{ll} \text{minimize} & -\log\det S + \text{Tr}(SY) \\ \text{subject to} & S \succeq 0 \end{array}$$

- Note: the problem can be solved easily using MATLAB with the CVX package, or using Python with the cvxpy package.

6.13 Matlab Code of CVX

```
function Sig = log_mle_solver(Y);
ndim = size(Y,1);
```

```
cvx_quiet(false);
cvx_begin sdp
    variable S(ndim, ndim) symmetric
    maximize(log_det(S) - trace(S*Y))
    subject to
        S >= 0;
cvx_end
Sig = inv(S);
```

6.14 Python Code

```
from cvxpy import *
from scipy import linalg

def mle_corr_mtx(Y):
  ndim = len(Y)
  S = Semidef(ndim)
  prob = Problem(Maximize(log_det(S) - trace(S*Y)))
  prob.solve()
  if prob.status != OPTIMAL:
      raise Exception('CVXPY Error')
  return linalg.inv(S.value)
```

6.15 Correlation Function (I)

- Let $\rho(h) = \sum_i^m p_i \Psi_i(h)$, where
  - p_i's are the unknown coefficients to be fitted
  - Ψ_i's are a family of basis functions.
- Let $\{F_k\}_{i,j} = \Psi_k(\|s_i - s_j\|_2)$.
- The covariance matrix $\Omega(p)$ can be recast as:

$$\Omega(p) = p_1 F_1 + \cdots + p_m F_m$$

- Note 1: affine transformation preserved convexity

- Note 2: inverse of matrix unfortunately **cannot** be expressed in convex form.

6.16 Correlation Function (II)

- Choice of $\Psi_i(h)$:
  - Polynomial $P_i(h)$:
    - Easy to understand
    - No guarantee of monotonicity; unstable for higher-order polynomials.
  - B-spline function $B_i(h)$
    - Shapes are easier to control
    - No guarantee of positive definite

6.17 Correlation Function (III)

- To ensure that the resulting function is PD, additional constraints can be imposed according to Bochner's theorem, e.g.:
  - $\text{real}(\text{FFT}(\{\Psi_i(h_k)\})) \geq 0$

Bochner's theorem states that a continuous function is a valid covariance function if and only if its Fourier transform is a non-negative measure. In other words, a function can be a valid covariance function if and only if its Fourier transform is positive definite. This theorem is important in spatial statistics because it provides a way to check whether a given covariance function is valid or not.

6.18 Non-Parametric Estimation

- Least squares estimation

$$\begin{array}{ll} \min_{\kappa,p} & \|\Omega(p) + \kappa I - Y\|_F \\ \text{s.t.} & \Omega(p) \succeq 0, \kappa \geq 0 \end{array}$$

  Note: convex problem

- Maximum likelihood estimation (MLE):

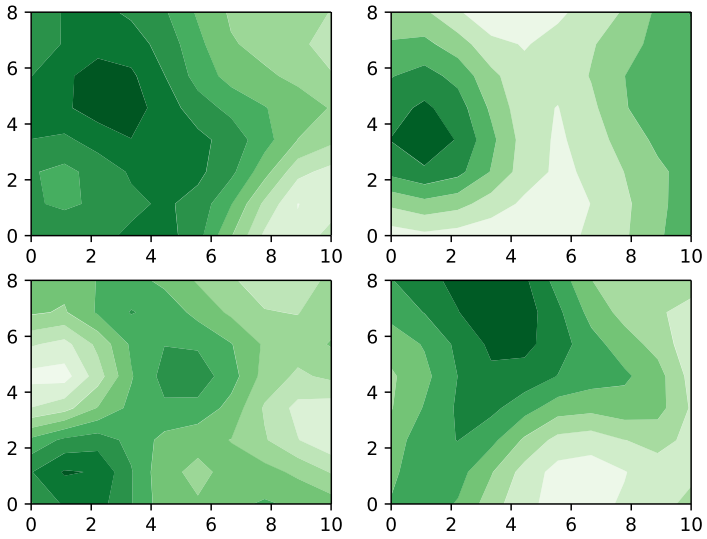
$$\begin{array}{ll} \min_{\kappa,p} & \log\det(\Omega(p)+\kappa I)+\mathrm{Tr}((\Omega(p)+\kappa I)^{-1}Y) \\ \text{s.t.} & \Omega(p)\succeq 0, \kappa \geq 0 \end{array}$$

  Note:

  - The 1st term is concave , the 2nd term is convex
  - However, the problem is **geodesically convex**.
  - If enough samples are available, then $Y \succeq 0$. Furthermore, the MLE is a convex problem in $Y \preceq \Omega(p)+\kappa I \preceq 2Y$

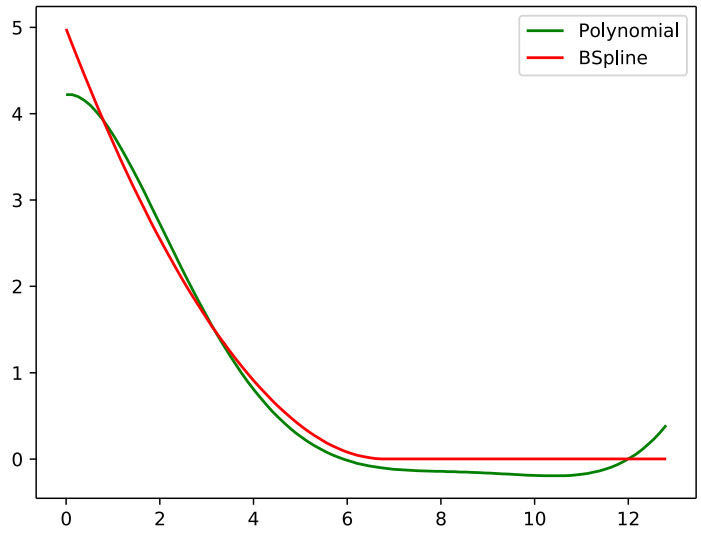
6.19 Isotopic Case I

.pull-left[



Data Sample

] .pull-right[

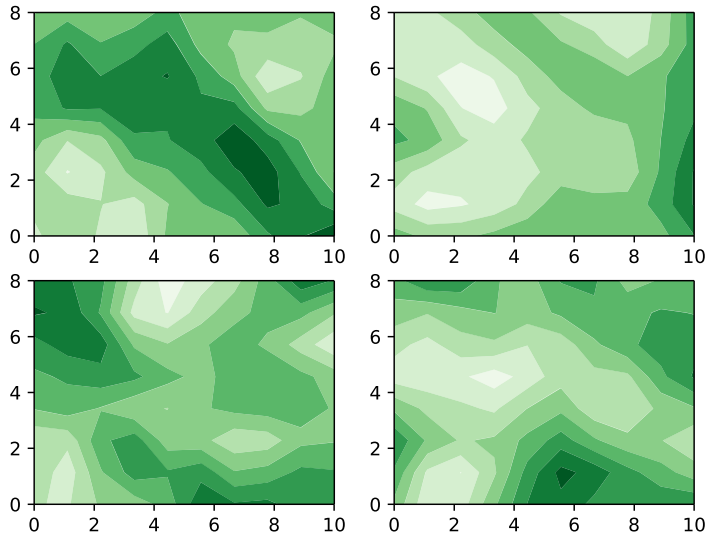


Least Square Result

]

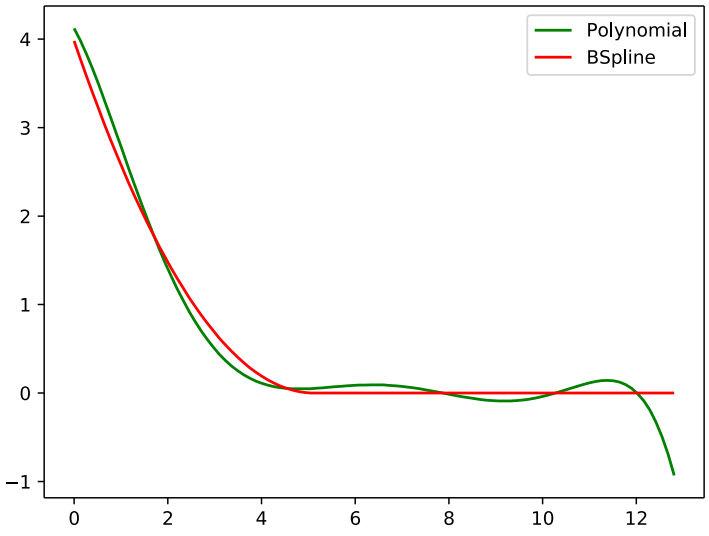
6.20 Isotopic Case II

.pull-left[



Data Sample

] .pull-right[



Least Square Result

]

6.21 Convex Concave Procedure

- Let $\Sigma = \Omega + \kappa I$. Log-likelihood function is:
  - $\log\det\Sigma^{-1} - \mathrm{Tr}(\Sigma^{-1}Y)$
- Convexify the first term using the fact:
  - $\log\det\Sigma^{-1} \geq \log\det\Sigma_0^{-1} + \mathrm{Tr}(\Sigma_0^{-1}(\Sigma - \Sigma_0))$
  - minimize: $-\log\det\Sigma_0^{-1} + \mathrm{Tr}(\Sigma_0^{-1}(\Sigma - \Sigma_0)) + \mathrm{Tr}(\Sigma^{-1}Y)$
- At each iteration k, the following convex problem is solved:

$$\begin{array}{ll} \min & \mathrm{Tr}(\Sigma_k^{-1}(\Sigma - \Sigma_k)) + \mathrm{Tr}(SY) \\ \text{s.t.} & \begin{pmatrix} \Sigma & I_n \\ I_n & S \end{pmatrix} \succeq 0, \kappa \geq 0 \end{array}$$

  Note: Convergence to an optimal solution is not guaranteed, but is practically good.

6.22 MATLAB Code

```
% Geometric anisotropic parameters
alpha = 2;      % scaling factor
theta = pi/3;   % angle
Sc = [1   0; 0   alpha];
R = [sin(theta) cos(theta); -cos(theta) sin(theta)];
T = Sc*R;
Sig = ones(n,n);
for i=1:n-1,
   for j=i+1:n,
     dt = s(j,:)' - s(i,:)';
     d = T*dt;  % become isotropic after the location transformation
     Sig(i,j) = exp(-0.5*(d'*d)/(sdkern*sdkern)/2);
     Sig(j,i) = Sig(i,j);
   end
end
```

6.23 Anisotopic Data

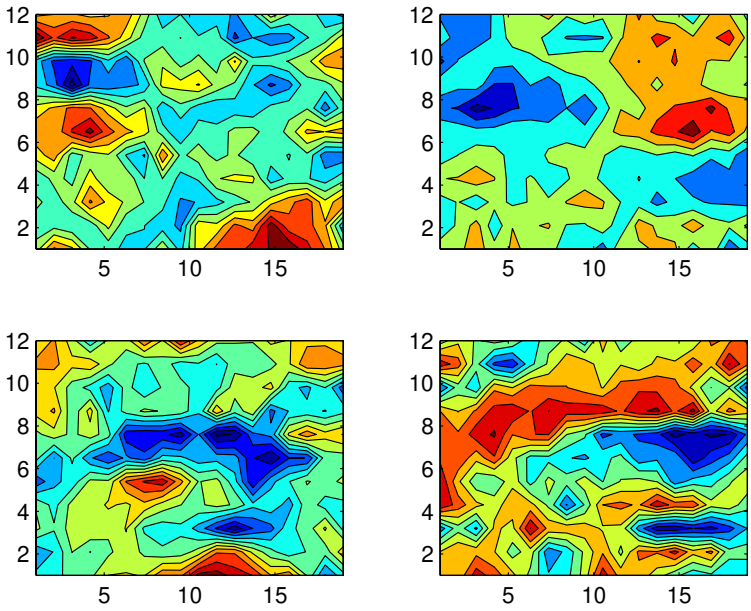


图 19: img

6.24 Isotropic Result

6.25 Anisotropic Result

6.26 Future Work

- Porting MATLAB code to Python
- Real data, not computer generated data
- Barycentric B-spline.
- Sampling method optimization.

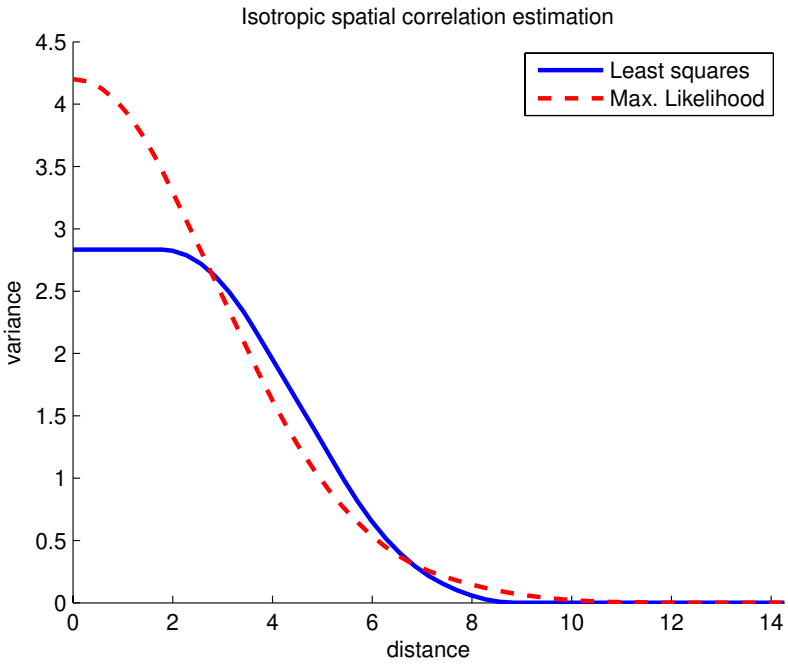


图 20: img

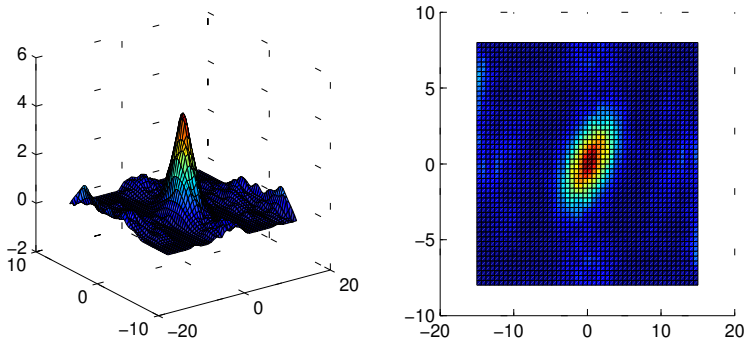


图 21: img

7 Lecture 05a - Clock Skew Scheduling Under Process Variations

7.1 @luk036

2022-10-19

7.2 Abstract

The main topic of the lecture is clock skew scheduling under process variations. The lecture discusses various techniques and methods for optimizing clock skew in order to improve circuit performance or minimize timing failures.

The lecture begins with an overview of the problem and the background of clock skew scheduling. It then explains the concept of clock skew and the difference between zero skew and useful skew designs. The importance of meeting timing constraints, such as setup time and hold time, is discussed along with the potential issues that can occur if these constraints are violated.

The lecture presents different approaches to clock skew scheduling, such as traditional scheduling, yield-driven scheduling, and minimum cost-to-time ratio formulation. It also explores various methods for finding the optimal clock period and the corresponding skew schedule, including linear programming and the use of the Bellman-Ford algorithm.

Further in the lecture, primitive solutions and their shortcomings are discussed, such as pre-allocating timing margins and utilizing least center error square (LCES) problem formulation. The lecture also presents more advanced techniques like Slack Maximization (EVEN) and Prop-based methods, which distribute slack along the most timing-critical cycle based on Gaussian models. The drawbacks of these methods are highlighted, particularly their assumptions about gate delay distributions.

Finally, the lecture discusses statistical static timing analysis (SSTA) and the use of statistical methods for accounting for process variations. The

concept of the most critical cycle is introduced, and the lecture provides experimental results to demonstrate the effectiveness of the different clock skew scheduling techniques.

Overall, the lecture explores various techniques and methods for optimizing clock skew scheduling under process variations, highlighting the challenges and potential solutions to improve circuit performance or minimize timing failures.

7.3 Keywords

- Static timing analysis, STA 静态时序分析
- Statistical STA 统计静态时序分析
- Clock skew 时钟偏差/偏斜
- Zero skew design 零偏差设计
  - Critical paths 关键路径
  - Negative slack 负时序裕量
- Useful skew design 有效偏差设计
  - Critical cycles 关键环
  - Negative cycles 负环
- Clock skew scheduling (CSS) 时钟偏差安排/规划
- Yield-driven CSS 产品率驱动时钟偏差安排

7.4 Overview

- Background
- Problem formulation
- Traditional clock skew scheduling
- Yield-driven clock skew scheduling
- Minimum cost-to-time ratio formulation

7.5 Sequential Logic

- Local data path

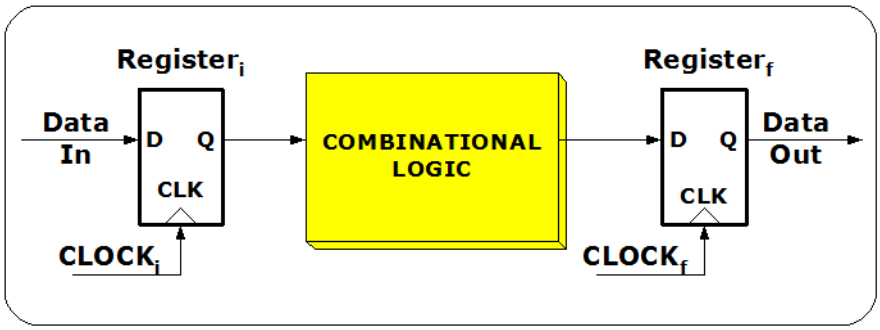


图 22: image

7.6 Sequential Logic (cont'd)

- Graph

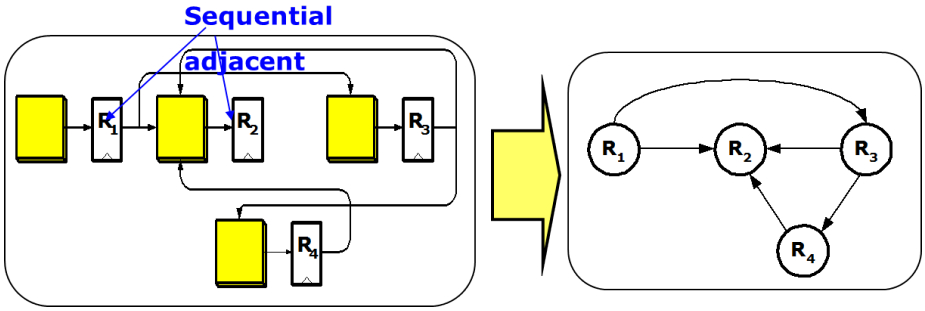


图 23: image

7.7 Clock Skew

- $T_{\text{skew}}(i, f) = t_i - t_f$, where
  - t_i: clock signal delay at the initial register
  - t_f: clock signal delay at the final register

7.8 Timing Constraint

- Setup time constraint

$$T_{\text{skew}}(i, f) \leq T_{\text{CP}} - D_{if} - T_{\text{setup}} = u_{if}$$

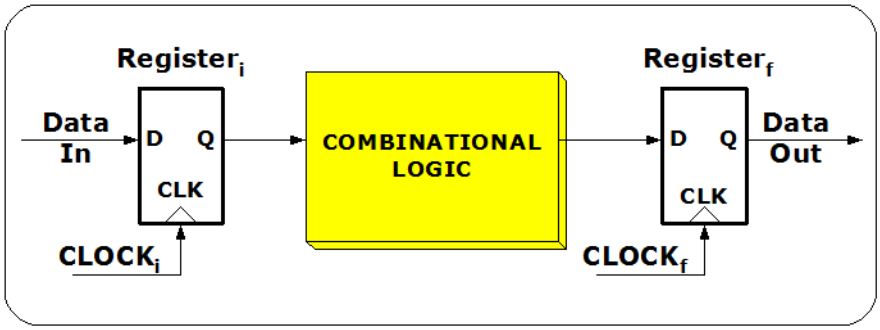


图 24: image

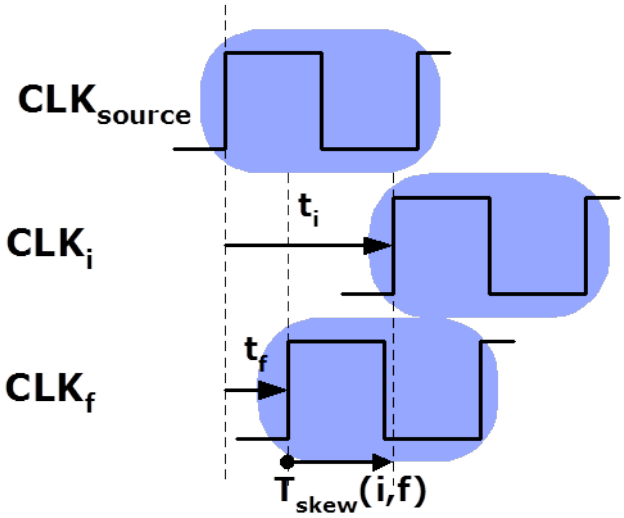


图 25: image

While this constraint destroyed, cycle time violation (zero clocking) occurs.

- Hold time constraint

$$T_{\text{skew}}(i, f) \geq T_{\text{hold}} - d_{if} = l_{if}$$

While this constraint destroyed, race condition (double clocking) occurs.

7.9 Zero skew vs. Useful skew

- Zero skew ($t_i = t_f$) : Relatively easy to implement.
- Useful skew. Improve:
  - The performance of the circuit by permitting a higher maximum clock frequency, or
  - The safety margins of the clock skew within the permissible ranges.
- Max./min. path delays are got from static timing analysis (STA).

7.10 Timing Constraint Graph

- Create a graph by
  - replacing the hold time constraint with a *h-edge* with cost $-(T_{\text{hold}} - d_{ij})$ from FF_i to FF_j, and
  - replacing the setup time constraint with an s-edge with cost $T_{\text{CP}} - D_{ij} - T_{\text{setup}}$ from FF_j to FF_i.
- Two sets of constraints stemming from clock skew definition:
  - The sum of skews for paths having the same starting and ending flip-flop to be the same;
  - The sum of clock skews of all cycles to be zero

7.11 Timing Constraint Graph (TCG)

7.12 Timing Constraint Graph (TCG)

Assume $T_{\text{setup}} = T_{\text{hold}} = 0$

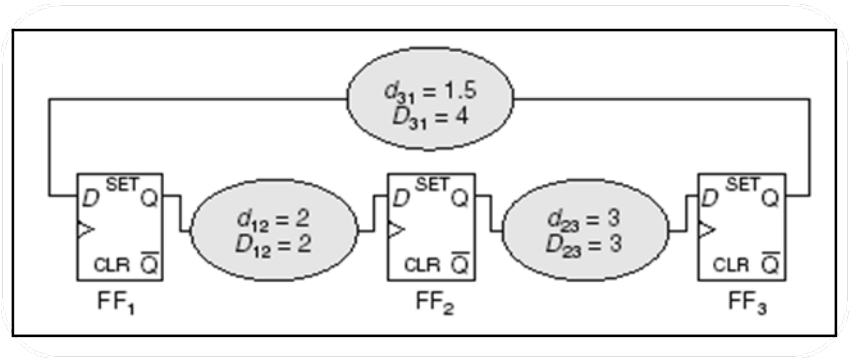


图 26: Example circuit

Clock period T_{CP} is feasible if and only if current graph contains no negative cost cycles.

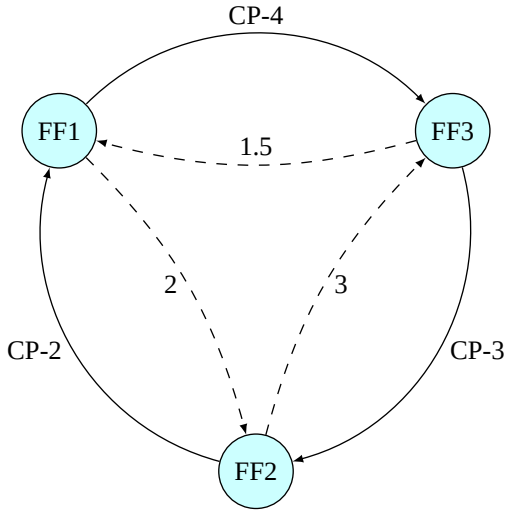


图 27: TCG

7.13 Minimize Clock Period

- Linear programming (LP) formulation

$$\begin{aligned} &\text{minimize} && T_{\text{CP}} \\ &\text{subject to} && l_{ij} \leq T_i - T_j \leq u_{ij} \end{aligned}$$

where FF_i and FF_j are sequential adjacent

- The above constraint condition is so-called **system of difference constraints** (see Introduction to Algorithms, MIT):
- Note: easy to check if a feasible solution exists by detecting negative cycle using for example Bellman-Ford algorithm.

7.14 Basic Bellman-Ford Algorithm

.font-sm.mb-xs[

```
function BellmanFord(list vertices, list edges, vertex source)
    // Step 1: initialize graph
    for each vertex i in vertices:
        if i is source then u[i] := 0
        else u[i] := inf
        predecessor[i] := null

    // Step 2: relax edges repeatedly
    for i from 1 to size(vertices)-1:
        for each edge (i, j) with weight d in edges:
*           if u[j] > u[i] + d[i,j]:
*               u[j] := u[i] + d[i,j]
*               predecessor[j] := i

    // Step 3: check for negative-weight cycles
    for each edge (i, j) with weight d in edges:
        if u[j] > u[i] + d[i,j]:
            error "Graph contains a negative-weight cycle"
return u[], predecessor[]
```

]

7.15 Problems with Bellman-Ford Algorithm

- The algorithm is originally used for finding the shortest paths.

- Detecting negative cycle is just a side product of the algorithm.
- The algorithm is simple, but…
  - detects negative cycle at the end only.
  - has to compute all `d[i,j]`.
  - Restart the initialization with `u[i] := inf`.
  - requests the input graph must have a source node.

Various improvements have been proposed extensively.

7.16 Minimize clock period (I)

- Fast algorithm for solving the LP:
  - Use binary search method for finding the minimum clock period.
  - In each iteration, Bellman-Ford algorithm is called to detect if the timing constraint graph contains negative weighted edge cycle.
- Note: Originally Bellman-Ford algorithm is used to find a shortest-path of a graph.

7.17 Minimize clock period (II)

- When the optimal clock period is solved, the corresponding skew schedule is got simultaneously.
- However, many skew values are on the bounds of feasible range.

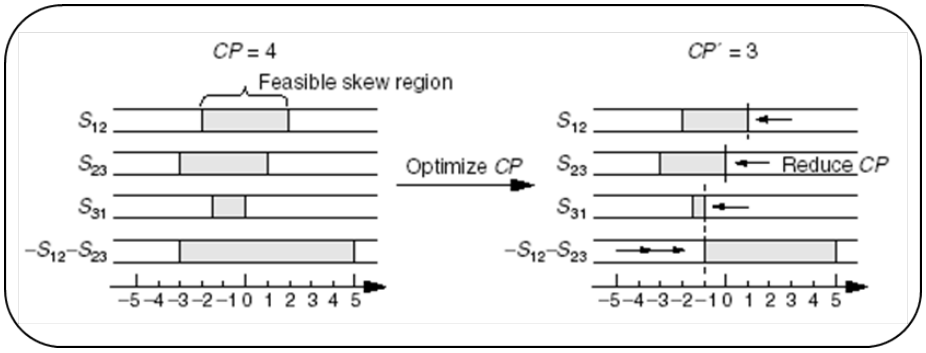


图 28: Timing uncertainty emerges under process variations

7.18 Yield-driven Clock Skew Scheduling

- When process variations increase more and more, timing-failure-induced yield loss becomes a significant problem.
- Yield-driven Clock Skew Scheduling becomes important.
- Primary goal of this scheduling is to minimize the yield loss instead of minimizing the clock period.

7.19 Timing Yield Definition

- The circuit is called functionally correct if all the setup- and hold-time constraints are satisfied under a group of determinate process parameters.
- Timing Yield = (functional correct times) / sample number * 100%

7.20 Primitive solution (1)

- Pre-allocate timing margins (usually equivalent to maximum timing uncertainty) at both ends of the FSR's (Feasible Skew Region).

$$l_{ij} \leq s_{ij} \leq u_{ij} \implies l_{ij} + \Delta d \leq s_{ij} \leq u_{ij} - \Delta d$$

- Then perform clock period optimization.

7.21 Problems with this method

- The maximum timing uncertainty is too pessimistic. Lose some performance;
- Δd is fixed; it does not consider data path delay differences between cycle edges.

7.22 References (1)

- "Clock skew optimization", IEEE Trans. Computers, 1990
- "A graph-theoretic approach to clock skew optimization", ISCAS'94

- "Cycle time and slack optimization for VLSI-chips", ICCAD'99
- "Clock scheduling and clocktree construction for high performance Asics", ICCAD'03
- "ExtensiveSlackBalance: an Approach to Make Front-end Tools Aware of Clock Skew Scheduling", DAC'06

7.23 Primitive solution (2)

- Formulate as LCES (Least Center Error Square) problem
  - A simple observation suggests that, to maximize slack, skew values should be chosen as close as possible to the middle points of their FSR's.

$$l_{ij} + lm_k(u_{ij} - l_{ij}) \leq s_{ij} \leq u_{ij} - um_k(u_{ij} - l_{ij})$$

$$\begin{array}{ll} \text{minimize} & \sum_k (0.5 - \min(lm_k, um_k))^2 \\ \text{subject to} & 0 \leq lm_k \leq 0.5 \\ & 0 \leq um_k \leq 0.5 \end{array}$$

7.24 References (2)

- Graph-based algorithm
  - (J. L. Neves and E. G. Friedman, "Optimal Clock Skew Scheduling Tolerant to Process Variations", DAC'96)
- Quadratic Programming method
  - (I. S. Kourtev and E. G. Fredman, "Clock skew scheduling for improved reliability via quadratic programming", ICCAD'99)

Shortcoming: might reduce some slacks to be zero to minimum **total** CES. This is not optimal for yield.

7.25 Primitive solution (3)

- Incremental Slack Distribution

  - (Xinjie Wei, Yici CAI and Xianlong Hong, "Clock skew scheduling under process variations", ISQED'06)

- Advantage: check all skew constraints
- Disadvantage: didn't take the path delay difference into consideration

7.26 Minimum Mean Cycle Based

- **Even**: solve the slack optimization problem using a minimum mean cycle formulation.
- **Prop**: distribute slack along the most timing-critical cycle proportional to path delays
- **FP-Prop**: use sensitizable-critical-path search algorithm for clock skew scheduling.

7.27 Slack Maximization (EVEN)

- Slack Maximization Scheduling

$$\begin{aligned} &\text{maximize} && t \\ &\text{subject to} && T_j - T_i \leq \mu_{ij} - t \end{aligned}$$

- Equivalent to the so-called minimum mean cycle problem (MMC), where

$$t^* = \frac{1}{|C|} \sum_{(i,j) \in C} \mu_{ij}$$

  C: critical cycle (first negative cycle)

- Can be solved efficiently by the above method.

7.28 Even - iterative slack optimization

- Identify the circuit's most timing-critical cycle,
- Distribute the slack along the cycle,
- Freeze the clock skews on the cycle, and
- Repeat the process iteratively.

7.29 Most timing-critical cycle

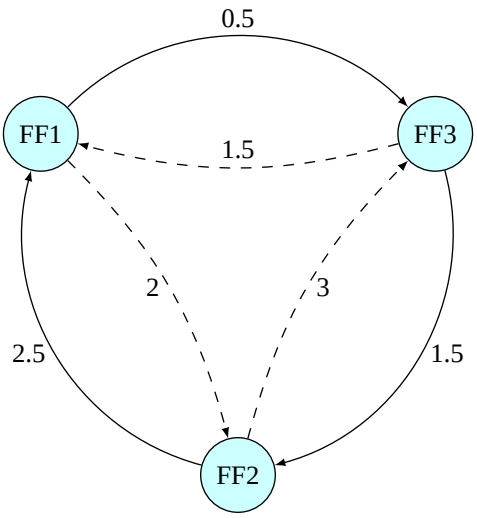


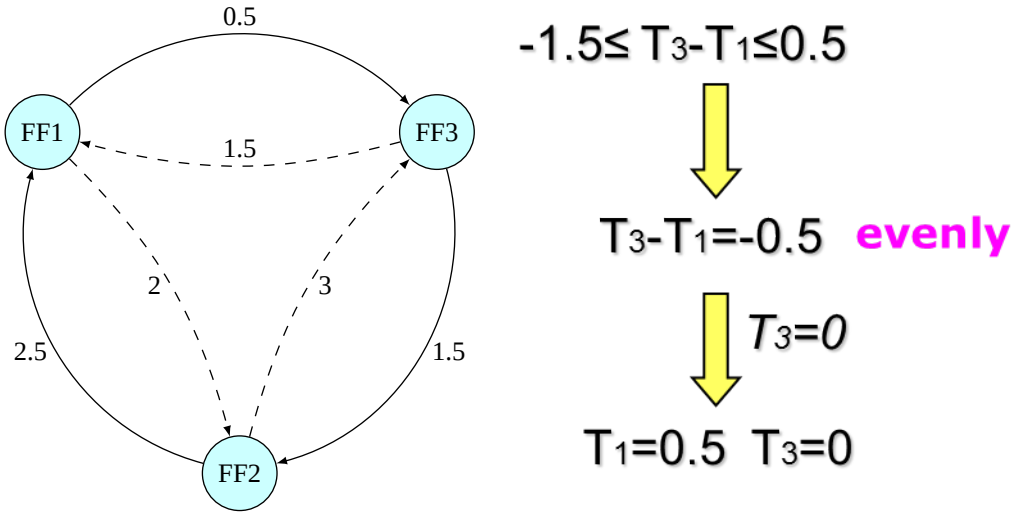
图 29: image

7.30 Identify the timing-critical cycle

- Identify the circuit's most timing-critical cycle
- Solve the minimum mean-weight cycle problem by
  - Karp's algorithm
  - A. Dasdan and R.K.Gupta, "Faster Maximum and Minimum Mean Cycle Algorithms for System-Performance", TCAD'98.

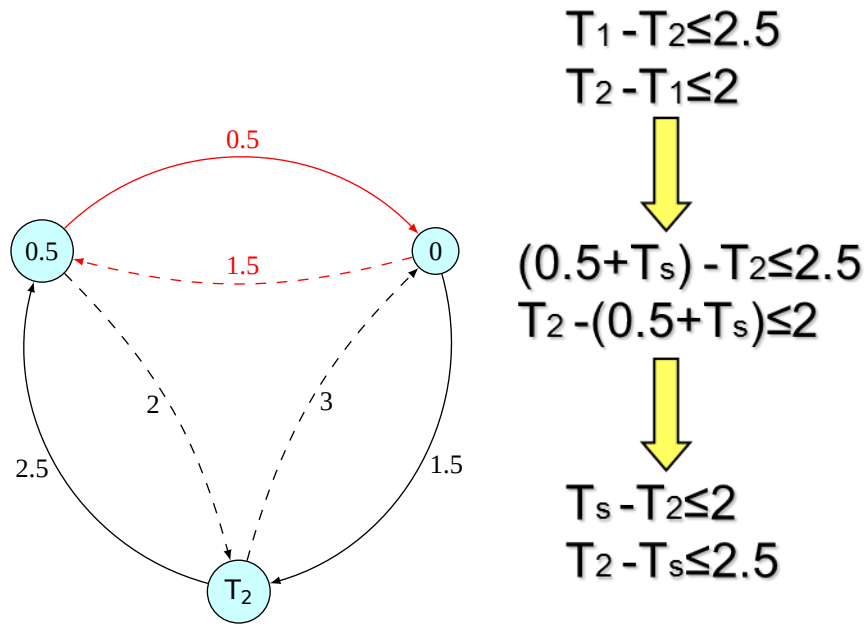
7.31 Distribute the slack

Distribute the slack evenly along the most timing-critical cycle.

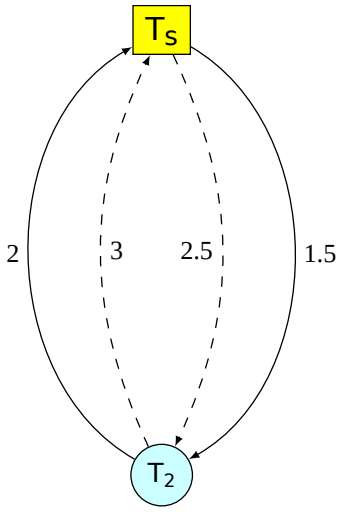


7.32 Freeze the clock skews (I)

Replace the critical cycle with super vertex.



7.33 Freeze the clock skews (II)



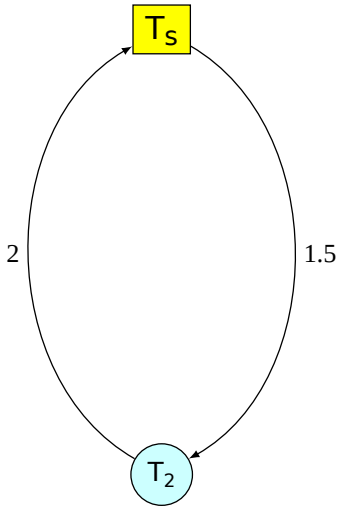
$T_1 - T_2 \leq 2.5$
$T_2 - T_1 \leq 2$

$(0.5+T_s) - T_2 \leq 2.5$
$T_2 - (0.5+T_s) \leq 2$

$T_s - T_2 \leq 2$
$T_2 - T_s \leq 2.5$

To determine the optimal slacks and skews for the rest of the graph, we replace the critical cycle with super vertex.

7.34 Repeat the process (I)



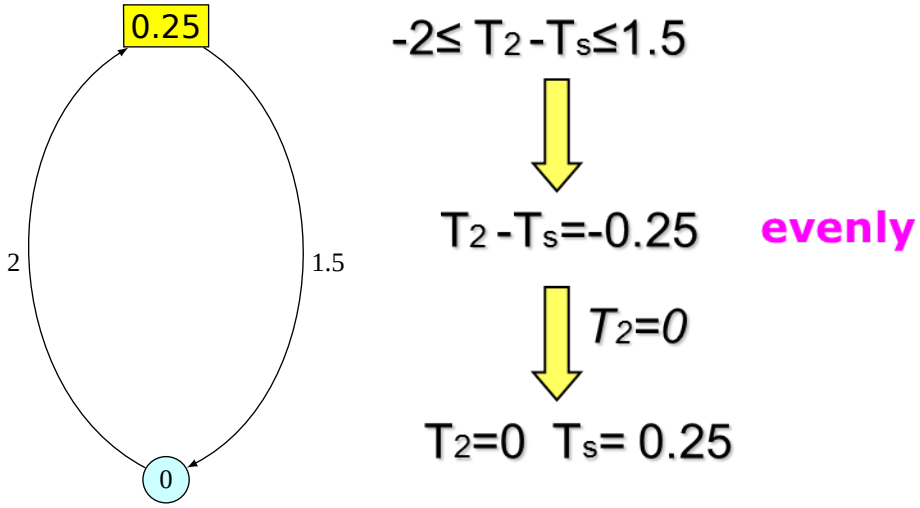
$-2 \leq T_2 - T_s \leq 1.5$

$T_2 - T_s = -0.25$ evenly

$T_2 = 0$

$T_2 = 0 \quad T_s = 0.25$

7.35 Repeat the process (II)



7.36 Final result

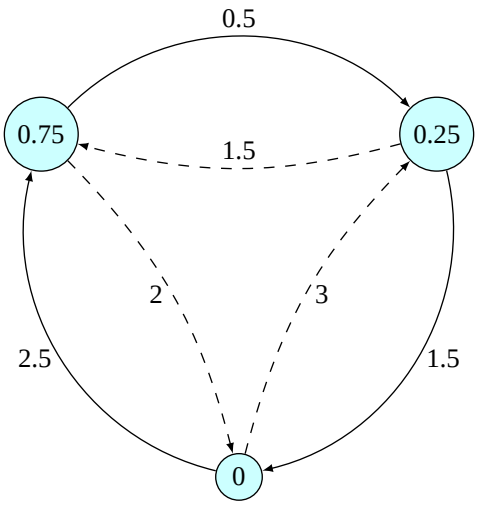


图 30: image

- $\text{Skew}_{12} = 0.75$
- Skew_{23} = -0.25
- Skew_{31} = -0.5
- $\text{Slack}_{12} = 1.75$
- $\text{Slack}_{23} = 1.75$
- $\text{Slack}_{31} = 1$

where $\text{Slack}_{ij} = T_{\text{CP}} - D_{ij} - T_{\text{setup}} - \text{Skew}_{ij}$

7.37 Problems with Even

- Assume all variances are the same.
- However, the timing uncertainty of a long combinational path is usually larger than that of a shorter path.
- Therefore, the even slack distribution along timing-critical cycles performed by **Even** is not optimal for yield if data path delays along the cycles are different.

7.38 Prop-Based on Gaussian model (I)

- Assuming there are n gates with delay $N(\mu, \sigma^2)$ in a path, then this path delay is $N(n\mu, n\sigma^2)$
- Distribute slack along the most timing-critical cycle, according to the square root of each edge's path delays (???).
- To achieve this, update the weights of s-edges and h-edges:

$$T_{\text{CP}} - (D_{ij} + \alpha\sqrt{D_{ij}}\sigma) - T_{\text{setup}}$$
$$-T_{\text{hold}} + (d_{ij} - \alpha\sqrt{d_{ij}}\sigma)$$

  where α ensures a minimum timing margin for each timing constraint.

7.39 Prop-Based on Gaussian model (II)

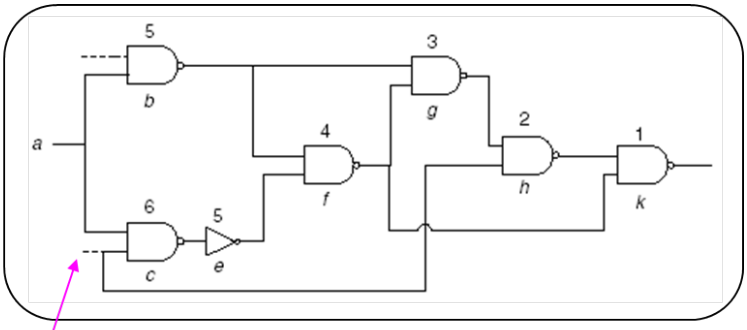
- Given a specific clock period T_{CP}, we gradually increase α and use the Bellman-Ford algorithm to detect whether it is still feasible.

- After finding the maximum α, the edges along the most timing-critical cycle will have slacks equal to the pre-allocated timing margins.
- Many edges in a circuit have sufficiently large slack. Therefore, we can perform proportional slack distribution only for the most timing-critical cycle. Assign the rest of skews using **Even**.

7.40 Problems with Prop

- Assume all gate delay has the same distribution.
- Not justify using the square root of path delay for timing margin.

7.41 FP-Prop (I)



When the signal at this terminal is "0"

False path

7.42 FP-Prop (II)

- If we do not consider false path, some non timing-critical cycles become timing-critical. Then, more slacks are distributed to these cycles, but the slacks in actually timing-critical cycles are not sufficient. As a result, the overall timing yield decreases.

7.43 Problems with FP-Prop

- Same problems as Prop

7.44 Experimental Results

		Even	*Prop*		*fp-Prop*	
Circuit	***CP***	**Yield (%)**	**Yield (%)**	**Imp. (%)**	**Yield (%)**	**Imp. (%)**
s1423	55.73	74.1	75.5	1.9	75.5	1.9
s1488	16.62	72.3	75.4	4.3	NA	NA
s1494	16.62	77.9	78.8	1.2	NA	NA
s5378	22.50	63.8	64.2	0.6	NA	NA
s9234.1	40.86	74.1	83.9	13.2	NA	NA
s13207.1	52.73	60.2	60.2	0.0	NA	NA
s35932	31.96	64.3	98.6	53.3	98.6	53.3
s38417	34.07	74.1	74.7	0.8	89.1	20.2
s38584.1	50.24	72.5	85.8	18.3	NA	NA
b04s	23.88	67.4	82.6	22.6	NA	NA
b05s	47.68	67.2	86.9	29.3	86.9	29.3
b06	4.92	63.1	75.1	19.0	NA	NA
b07s	26.23	74.8	72.7	–2.8	72.7	–2.8
b08	14.90	68.9	65.6	–4.8	65.6	–4.8
b09	7.68	62.4	78.3	25.5	NA	NA
b10	9.71	73.5	73.7	0.3	NA	NA
b11s	29.86	71.5	58.6	–18.0	80.9	13.1
b12	12.12	78.2	87.6	12.0	87.6	12.0

图 31: image

7.45 Statistical Method

- Setup time constraint

$$T_{\text{skew}}(i,f) \leq T_{\text{CP}} - \tilde{D}_{if} - T_{\text{setup}}$$

- Hold time constraint

$$T_{\text{skew}}(i,f) \geq T_{\text{hold}} - \tilde{d}_{if}$$

where $\tilde{D}_{if}$ and $\tilde{d}_{if}$ are random variable under process variations.

7.46 Statistical TC Graph

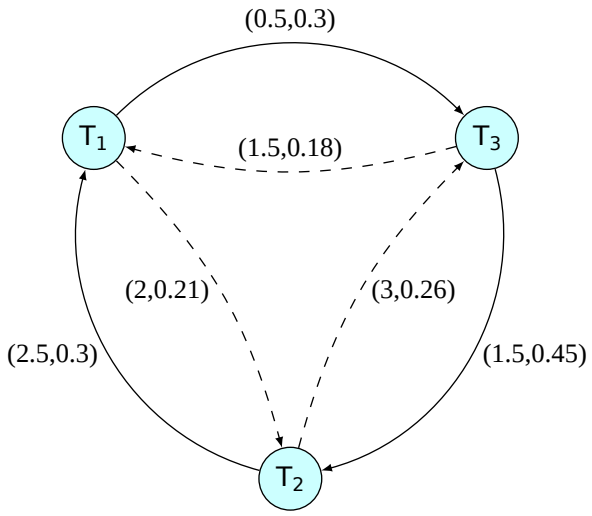


图 32: image

After SSTA, edge weight is represented as a pair of value (mean, variance).

7.47 Most Critical Cycle

- Traditional criteria: minimum mean cycle

$$\min_{C \in \mathcal{C}} \frac{\sum_{(i,j) \in C} \mu_{ij}}{|C|}$$

- New criteria:

$$\min_{C \in \mathcal{C}} \frac{\sum_{(i,j) \in C} \mu_{ij}}{\sum_{(i,j) \in C} \sigma_{ij}}$$

  (We show the correctness later)

7.48 Slack Maximization (C-PROP)

- Slack Maximization Scheduling

$$\begin{aligned} &\text{maximize} && t \\ &\text{subject to} && T_j - T_i \leq \mu_{ij} - \sigma_{ij} t \end{aligned}$$

- Equivalent to the *minimum cost-to-time ratio* problem (MMC), where:
  - $t^* = \sum_{(i,j)\in C} \mu_{ij} / \sum_{(i,j)\in C} \sigma_{ij}$
  - C: critical cycle (first negative cycle)

7.49 Probability Observation

- Prob(timing failure) turns out to be an Error function that solely depends on this ratio. Therefore, it is justified to use this ratio as critical criteria.

7.50 Whole flow

- After determining the clock arrival time at each vertex in the most critical cycle, the cycle is replaced with a super vertex v'.
- In-edge (u, v) from outside vertex u to cycle member v is replaced by an in-edge (u, v') with weight mean $\mu(u, v) - T_v$.
- Out-edge (v, u) is replaced by out-edge (v', u) with weight mean $\mu(v, u) + T_v$. However, the variance of the edge weight is not changed. And parallel edges can be remained.
- Repeat the process iteratively until the graph is reduced to a single super vertex, or the edges number is zero.

7.51 Data structure

Final result: $T_1 = T_1 + T_{s_1} + T_{s_3}$

7.52 Advantages of This Method

- Justified by probability observation.
- Fast algorithm exists for minimum cost-to-time ratio problem.
- Reduce to Even when all variances are equal.
- When a variance tends to zero, it makes sense that only minimal slack is assigned to this variable, and hence others can be assigned more.

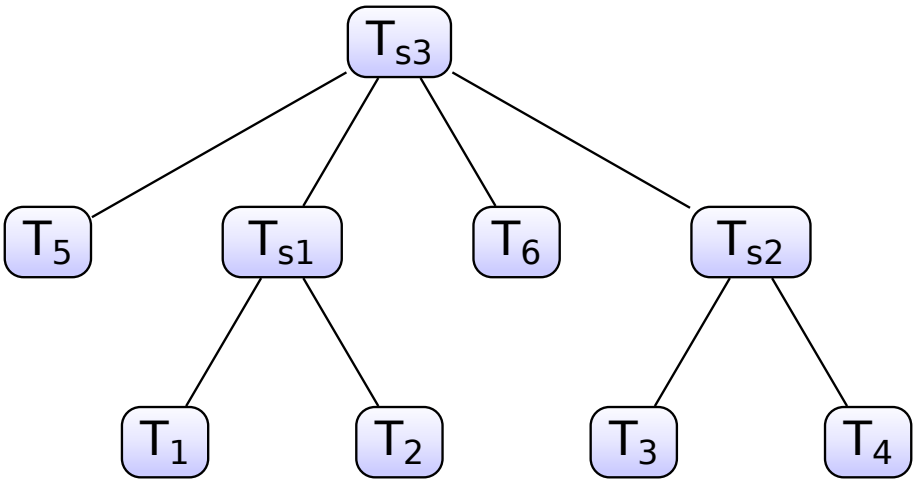
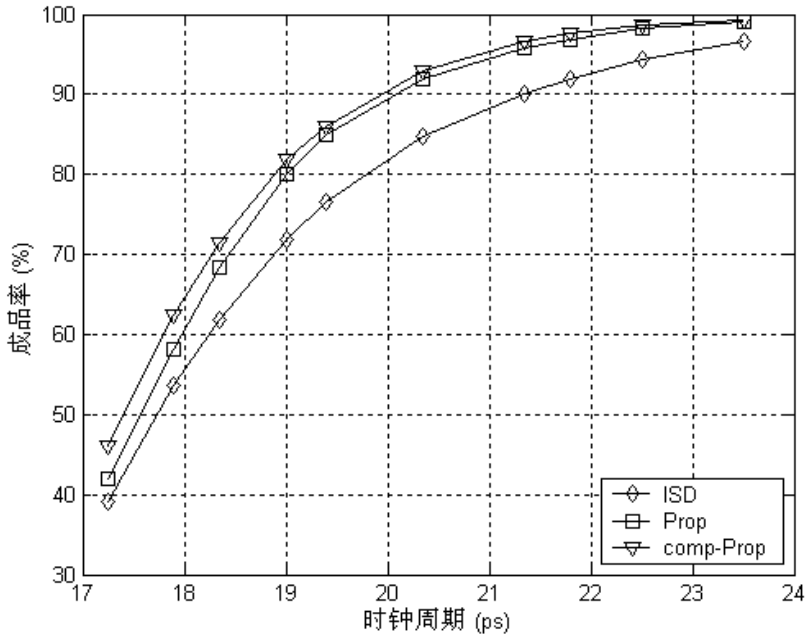


图 33: image

7.53 Results



7.54 Main Reference

- Jeng-Liang Tsai, Dong Hyum Baik, Charlie Chung-Ping Chen, and Kewal K. Saluja, "Yield-Driven, False-Path-Aware Clock Skew Scheduling", IEEE Design & Test of Computers, May-June 2005

8 Lecture 05b - Clock Skew Scheduling Under Process Variations (2)

8.1 Overview

- A Review of CSS Issues
- General Formulation
- Yield-driven Clock Skew Scheduling
- Numerical Results

8.2 Minimum Clock Period Problem

- Linear programming (LP) formulation

$$\begin{array}{ll} \text{minimize} & T_{\text{CP}} \\ \text{subject to} & l_{ij} \leq T_i - T_j \leq u_{ij} \end{array}$$

  where FF_i and FF_j are sequentially adjacent to each other.

- The above constraints are called *system of difference constraints* (see Introduction to Algorithms, MIT):
  - Key: it is easy to check if a feasible solution exists by detecting negative cycles using the Bellman-Ford algorithm.

8.3 System of Difference Constraints

- In some cases, you may need to do some transformations, e.g.
  - $T_i \leq \min_k\{T_k + a_{ik}\} \rightarrow T_i - T_k \leq a_{ik},\ \forall k$
  - $T_i \geq \max_k\{T_k + b_{ik}\} \rightarrow b_{ik} \leq T_i - T_k,\ \forall k$

8.4 Slack Maximization (EVEN)

- Slack Maximization Scheduling

$$\begin{array}{ll} \text{maximum} & t \\ \text{subject to} & T_j - T_i \leq \mu_{ij} - t \end{array}$$

  (Note: μ_{ij} is not equal to μ_{ji})

- is equivalent to the so-called *minimum mean cycle problem* (MMC), where:
  - $t^* = \sum_{(i,j)\in C} \mu_{ij}/|C|$,
  - C: critical cycle (first negative cycle)
- Can be efficiently solved by the parametric shortest path methods.

8.5 Slack Maximization (C-PROP)

- Slack Maximization Scheduling

$$\begin{array}{ll} \text{maximum} & t \\ \text{subject to} & T_j - T_i \leq \mu_{ij} - \sigma_{ij} t \end{array}$$

  (we show the correctness later)

- is equivalent to the *minimum cost-to-time ratio problem* (MCR), where:
  - $t^* = \sum_{(i,j)\in C} \mu_{ij} / \sum_{(i,j)\in C} \sigma_{ij}$,
  - C: critical cycle

8.6 General Formulation

- General form:

$$\begin{array}{ll} \text{maximum} & g(t) \\ \text{subject to} & T_i - T_j \leq f_{ij}(t),\ \forall (i,j) \in E \end{array}$$

  where $f_{ij}(t)$ a linear function that represents various problems defined above.

Problem	$g(t)$	$f_{ij}(t)$ (setup)	$f_{ji}(t)$ (hold)
Min. CP	$-t$	$t - D_{ij} - T_{\text{setup}}$	$-T_{\text{hold}} + d_{ij}$
EVEN	t	$T_{\text{CP}} - D_{ij} - T_{\text{setup}} - t$	$-T_{\text{hold}} + d_{ij} - t$
C-PROP	t	$T_{\text{CP}} - D_{ij} - T_{\text{setup}} - \sigma_{ij}t$	$-T_{\text{hold}} + d_{ij} - \sigma_{ij}t$

8.7 General Formulation (cont'd)

- In fact, $g(t)$ and $f_{ij}(t)$ are not necessarily linear functions. Any monotonic decreasing function will do.
- Theorem: if $g(t)$ and $f_{ij}(t)$ are *monotonic decreasing* functions for all i and j, then there is a unique solution to the problem. (prove later).
- Question 1: Does this generalization have any application?
- Question 2: What if $g(t)$ and $f_{ij}(t)$ are convex but not monotone?

8.8 Non-Gaussian Distribution

- 65nm and below, the path delay is likely to have a non-Gaussian distribution:

  Note: central limit theorem does not apply because

  - random variables are correlated (why?)
  - delays are non-negative

8.9 Timing Yield Maximization

- Formulation:
  - $\max\{\min\{\Pr\{T_j - T_i \leq \tilde{W}_{ij}\}\}\}$
  - is not exactly timing yield but reasonable.

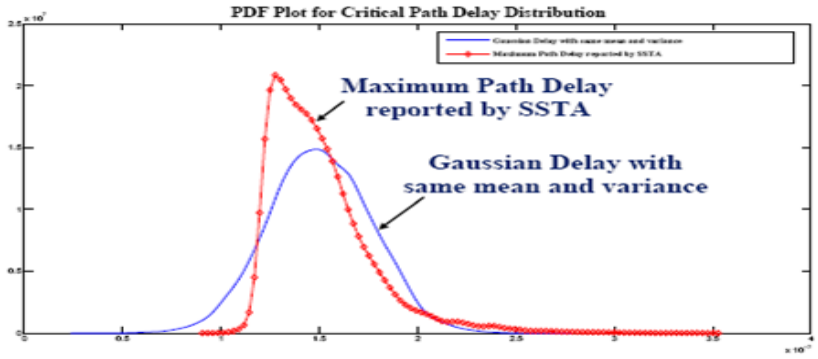


图 34: image

- It is equivalent to:

$$\begin{array}{ll} \text{maximum} & t \\ \text{subject to} & T_i - T_j \leq T_{\text{CP}} - F_{ji}^{-1}(t) \\ & T_j - T_i \leq F_{ij}^{-1}(1-t) \end{array}$$

  where $F_{ij}(\cdot)$ is CDF of $\tilde{W}_{ij}$

- Luckily, any CDF must be a monotonic increasing function.

8.10 Statistical Interpretations of C-PROP

- Reduce to C-PROP when $\tilde{W}_{ij}$ is Gaussian, or precisely

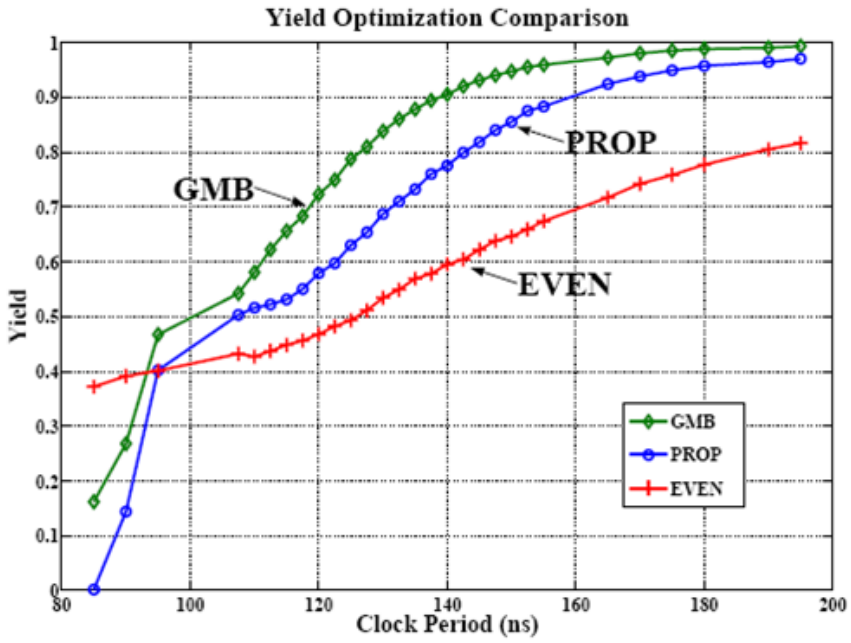
$$F_{ij}(x) = K((x - \mu_{ij})/\sigma_{ij})$$

- EVEN: identical distribution up to shifting

$$F_{ij}(x) = H(x - \mu_{ij})$$

  Not necessarily worse than C-PROP

8.11 Comparison



8.12 Three Solving Methods in General

- Binary search based
  - Local convergence is slow.
- Cycle based
  - Idea: if a solution is infeasible, there exists a negative cycle which can always be "zero-out" with minimum effort (proof of optimality)
- Path based
  - Idea: if a solution is feasible, there exists a (shortest) path from where we can always improve the solution.

8.13 Parametric Shortest Path Algorithms

- Lawler's algorithm (binary search)

- Howard's algorithm (based on cycle cancellation)
- Hybrid method
- Improved Howard's algorithm
- Input:
  - Interval [tmin, tmax] that includes t*
  - Tol: tolerance
  - G(V, E): timing graph
- Output:
  - Optimal t* and its corresponding critical cycle C

9 Clock Skew Scheduling for Unimodal Distributed Delay Models

@luk036

2022-10-26

9.1 Useful Skew Design: Why and Why not?

Bad :

- Needs more engineer training.
- Balanced clock-trees are harder to build.
- Don't know how to handle process variation, multi-corner multi-mode, ..., etc.

Good :

If you do it right,

- spend less time struggling about timing, or
- get better chip performance or yield.

9.2 What can modern STA tools do today?

- Manually assign clock arrival times to registers (all zeros by default)
- Grouping: Non-critical parts can be grouped as a single unit. In other words, there is no need for full-chip optimization.
- Takes care of multi-cycle paths, slew rate, clock-gating, false paths etc. All we need are the reported **slacks**.
- Provide 3-sigma statistics for slacks/path delays (POCV).
- However, the full probability density function and correlation information are not available.

9.3 Unimodality

- In statistics, a unimodal probability distribution or unimodal distribution is a probability distribution with a single peak.
- In continuous distributions, unimodality can be defined through the behavior of the cumulative distribution function (cdf). If the cdf is *convex* for $x < m$ and *concave* for $x > m$, then the distribution is unimodal, m being the *mode*.
- Examples
  - Normal distribution
  - Log-normal distribution
  - Log-logistic distribution
  - Weibull distribution

9.4 Quantile function

- The quantile function z_p of a distribution is the inverse of the cumulative distribution function $\Phi^{-1}(p)$.
- Close-form expression for some unimodal distributions:
  - Normal: $\mu + \sigma\sqrt{2}\text{erf}^{-1}(2p-1)$
  - Log-normal: $\exp\left(\mu + \sigma\sqrt{2}\text{erf}^{-1}(2p-1)\right)$
  - Log-logistic: $\alpha\left(\frac{p}{1-p}\right)^{1/\beta}$

  - Weibull: $\lambda(-\ln(1-p))^{1/k}$

- For log-normal distribution:
  - mode: $\exp(\mu - \sigma^2)$
  - CDF at mode: $1/2(1 + \text{erf}(-\sigma/\sqrt{2}))$

9.5 Normal vs. Log-normal Delay Model

Normal/Gaussian:

- Convertible to a linear network optimization problem.
- Supported over the whole real line. Negative delays are possible.
- Symmetric, obviously not adaptable to the 3-sigma results.

Log-normal:

- Non-linear, but still can be solved efficiently with network optimization.
- Supported only on the positive side.
- Non-symmetric, may be able to fit into the 3-sigma results. (???)

9.6 Setup- and Hold-time Constraints

- Let $T_{\text{skew}}(i, f) = t_i - t_f$, where
  - t_i: clock signal delay at the initial register
  - t_f: clock signal delay at the final register
  - Assume in zero-skew, i.e. $T_{\text{skew}}(i, f) = 0$, the reported setup- and hold-time slacks are S_{if} *and* H_{if} respectively.
- Then, in useful skew design:

$$T_{\text{skew}}(i, f) \leq S_{if} \implies t_i - t_f \leq S_{if}$$

$$T_{\text{skew}}(i, f) \geq -H_{if} \implies t_f - t_i \leq H_{if}$$

- In principle, H_{if} and $T_{\text{CP}} - S_{if}$ represent the minimum- and maximum-path delay, and should be always greater than zero.
- Let $D_{if} = T_{\text{CP}} - S_{if}$

9.7 Yield-driven Optimization

- Max-Min Formulation:
  - $\max\{\min\{\Pr\{t_j - t_i \leq \tilde{W}_{ij}\}\}\}$,
  - No need for correlation information between paths.
  - Not exactly the timing yield objective but reasonable.
- Equivalent to:

$$\begin{array}{ll} \text{maximum} & \beta \\ \text{subject to} & \Pr\{t_i - t_j \leq T_{\text{CP}} - \tilde{D}_{ij}\} \geq \beta \\ & \Pr\{t_j - t_i \leq \tilde{H}_{ij}\} \geq \beta \end{array}$$

- or:

$$\begin{array}{ll} \text{maximum} & \beta \\ \text{subject to} & t_i - t_j \leq T_{\text{CP}} - \Phi^{-1}_{D_{ij}}(\beta) \\ & t_j - t_i \leq \Phi^{-1}_{H_{ij}}(1-\beta) \end{array}$$

9.8 Yield-driven Optimization (cont'd)

- In general, Lawler's algorithm (binary search) can be used.
- Depending on the distribution, there are several other ways to solve problem.

9.9 Gaussian Delay Model

- Reduce to:

$$\begin{array}{ll} \text{maximum} & \beta \\ \text{subject to} & t_i - t_j \leq T_{\text{CP}} - (\mu^D_{ij} + \sigma^D_{ij}\sqrt{2}\text{erf}^{-1}(2\beta - 1)) \\ & t_j - t_i \leq \mu^H_{ij} + \sigma^H_{ij}\sqrt{2}\text{erf}^{-1}(2(1-\beta) - 1) \end{array}$$

- Linearization. Since $\text{erf}^{-1}(\cdot)$ is anti-symmetric and monotonic, we have:

$$\begin{aligned} \text{maximum} \quad & \beta' \\ \text{subject to} \quad & t_i - t_j \leq T_{\text{CP}} - \mu_{ij}^D - \sigma_{ij}^D \beta' \\ & t_j - t_i \leq \mu_{ij}^H - \sigma_{ij}^H \beta' \end{aligned}$$

- is equivalent to the minimum cost-to-time ratio (linear).
- However, actual path delay distributions are non-Gaussian.

9.10 Log-normal Delay Model

- Reduce to:

$$\begin{aligned} \text{maximum} \quad & \beta \\ \text{subject to} \quad & t_i - t_j \leq T_{\text{CP}} - \exp(\mu_{ij}^D + \sigma_{ij}^D \sqrt{2} \text{erf}^{-1}(2\beta - 1)) \\ & t_j - t_i \leq \exp(\mu_{ij}^H + \sigma_{ij}^H \sqrt{2} \text{erf}^{-1}(2(1-\beta) - 1)) \end{aligned}$$

- Since $\text{erf}^{-1}(\cdot)$ is anti-symmetric and monotonic, we have:

$$\begin{aligned} \text{maximum} \quad & \beta' \\ \text{subject to} \quad & t_i - t_j \leq T_{\text{CP}} - \exp(\mu_{ij}^D + \sigma_{ij}^D \beta') \\ & t_j - t_i \leq \exp(\mu_{ij}^H - \sigma_{ij}^H \beta') \end{aligned}$$

- Bypass evaluating error function. Non-linear and non-convex, but still can be solved efficiently by for example binary search on β'.

9.11 Weibull Delay Model

- Reduce to:

$$\begin{aligned} \text{maximum} \quad & \beta \\ \text{subject to} \quad & t_i - t_j \leq T_{\text{CP}} - \lambda_{ij}^D (-\ln(1-\beta))^{1/k_{ij}^D} \\ & t_j - t_i \leq \lambda_{ij}^H (-\ln(\beta))^{1/k_{ij}^H} \end{aligned}$$

10 When "Convex Optimization" Meets "Network Flow"

10.1 @luk036

2022-11-16

11 Introduction

11.1 Overview

- Network flow problems can be solved efficiently and have a wide range of applications.
- Unfortunately, some problems may have other additional constraints that make them impossible to solve with current network flow techniques.
- In addition, in some problems, the objective function is quasi-convex rather than convex.
- In this lecture, we will investigate some problems that can still be solved by network flow techniques with the help of convex optimization.

12 Parametric Potential Problems

12.1 Parametric potential problems

Consider:

$$\begin{array}{ll} \text{maximize} & g(\beta), \\ \text{subject to} & y \le d(\beta), \\ & Au = y, \end{array}$$

where $g(\beta)$ and $d(\beta)$ are concave.

Note: the parametric flow problems can be defined in a similar way.

12.2 Network flow says:

- For fixed β, the problem is feasible precisely when there exists no negative cycle
- Negative cycle detection can be done efficiently using the Bellman-Ford-like methods
- If a negative cycle C is found, then $\sum_{(i,j)\in C} d_{ij}(\beta) < 0$

12.3 Convex Optimization says:

- If both sub-gradients of $g(\beta)$ and $d(\beta)$ are known, then the *bisection method* can be used for solving the problem efficiently.
- Also, for multi-parameter problems, the *ellipsoid method* can be used.

12.4 Quasi-convex Minimization

Consider:

$$\begin{array}{ll} \text{maximize} & f(\beta), \\ \text{subject to} & y \leq d(\beta), \\ & Au = y, \end{array}$$

where $f(\beta)$ is *quasi-convex* and $d(\beta)$ are concave.

12.5 Example of Quasi-Convex Functions

- $\sqrt{|y|}$ is quasi-convex on $\mathbb{R}$
- $\log(y)$ is quasi-linear on $\mathbb{R}_{++}$
- $f(x, y) = xy$ is quasi-concave on $\mathbb{R}^2_{++}$
- Linear-fractional function:
  - $f(x) = (a^\mathsf{T} x + b)/(c^\mathsf{T} x + d)$
  - dom $f = \{x \,|\, c^\mathsf{T} x + d > 0\}$
- Distance ratio function:
  - $f(x) = \|x - a\|_2 / \|x - b\|_2$

– dom $f = \{x \mid \|x - a\|_2 \leq \|x - b\|_2\}$

12.6 Convex Optimization says:

If f is quasi-convex, there exists a family of functions ϕ_t such that:

- $\phi_t(\beta)$ is convex w.r.t. β for fixed t
- $\phi_t(\beta)$ is non-increasing w.r.t. t for fixed β
- t-sublevel set of f is 0-sublevel set of ϕ_t, i.e., $f(\beta) \leq t$ iff $\phi_t(\beta) \leq 0$

For example:

- $f(\beta) = p(\beta)/q(\beta)$ with p convex, q concave $p(\beta) \geq 0$, $q(\beta) > 0$ on dom f,
- can take $\phi_t(\beta) = p(\beta) - t \cdot q(\beta)$

12.7 Convex Optimization says:

Consider a convex feasibility problem:

$$\begin{array}{ll} \text{find} & f(\beta), \\ \text{s. t.} & \phi_t(\beta) \leq 0, \\ & y \leq d(\beta), Au = y, \end{array}$$

- If feasible, we conclude that $t \geq p^*$;
- If infeasible, $t < p^*$.

Binary search on t can be used for obtaining p^*.

12.8 Quasi-convex Network Problem

- Again, the feasibility problem ([eq:quasi]) can be solved efficiently by the bisection method or the ellipsoid method, together with the negatie cycle detection technique.
- Any EDA's applications ???

12.9 Monotonic Minimization

- Consider the following problem:

$$\begin{array}{ll} \text{minimize} & \max_{ij} f_{ij}(y_{ij}), \\ \text{subject to} & Au = y, \end{array}$$

  where $f_{ij}(y_{ij})$ is non-decreasing.

- The problem can be recast as:

$$\begin{array}{ll} \text{maximize} & \beta, \\ \text{subject to} & y \leq f^{-1}(\beta), \\ & Au = y, \end{array}$$

  where $f^{-1}(\beta)$ is non-deceasing w.r.t. β.

12.10 E.g. Yield-driven Optimization

- Consider the following problem:

$$\begin{array}{ll} \text{maximize} & \min_{ij} \Pr(y_{ij} \leq \tilde{d}_{ij}) \\ \text{subject to} & Au = y, \end{array}$$

  where $\tilde{d}_{ij}$ is a random variables.

- Equivalent to the problem:

$$\begin{array}{ll} \text{maximize} & \beta, \\ \text{subject to} & \beta \leq \Pr(y_{ij} \leq \tilde{d}_{ij}), \\ & Au = y, \end{array}$$

  where $f_{ij}^{-1}(\beta)$ is non-deceasing w.r.t. β.

12.11 E.g. Yield-driven Optimization (II)

- Let $F(x)$ is the cdf of $\tilde{d}$.

- Then:

$$\begin{aligned} & \beta \leq \Pr(y_{ij} \leq \tilde{d}_{ij}) \leq t \\ \Rightarrow \quad & \beta \leq 1 - F_{ij}(y_{ij}) \\ \Rightarrow \quad & y_{ij} \leq F_{ij}^{-1}(1-\beta) \end{aligned}$$

- The problem becomes:

$$\begin{array}{ll} \text{maximize} & \beta, \\ \text{subject to} & y_{ij} \leq F_{ij}^{-1}(1-\beta), \\ & Au = y, \end{array}$$

12.12 Network flow says

- Monotonic problem can be solved efficiently using cycle-cancelling methods such as Howard's algorithm.

13 Min-cost flow problems

13.1 Min-Cost Flow Problem (linear)

Consider:

$$\begin{array}{ll} \min & d^{\mathsf{T}}x + p \\ \text{s. t.} & c^- \leq x \leq c^+, \\ & A^{\mathsf{T}}x = b,\ b(V) = 0 \end{array}$$

- some c^+ could be $+\infty$ some c^- could be $-\infty$.
- A^{T} is the incidence matrix of a network G.

13.2 Conventional Algorithms

- Augmented-path based:
  - Start with an infeasible solution
  - Inject minimal flow into the augmented path while maintaining infeasibility in each iteration
  - Stop when there is no flow to inject into the path.
- Cycle cancelling based:

- Start with a feasible solution x_0
- find a better sol'n $x_1 = x_0 + \alpha\triangle x$, where α is positive and $\triangle x$ is a negative cycle indicator.

13.3 General Descent Method

1. **Input**: a starting $x \in \text{dom } f$
2. **Output**: x^*
3. **repeat**
   1. Determine a descent direction p.
   2. Line search. Choose a step size $\alpha > 0$.
   3. Update. $x := x + \alpha p$
4. **until** a stopping criterion is satisfied.

13.4 Some Common Descent Directions

- For convex problems, the search direction must satisfy $\nabla f(x)^\mathsf{T} p < 0$.
- Gradient descent:
  - $p = -\nabla f(x)^\mathsf{T}$
- Steepest descent:
  - $\triangle x^{nsd} = \text{argmin}\{\nabla f(x)^\mathsf{T} v \mid \|v\| = 1\}$.
  - $\triangle x^{sd} = \|\nabla f(x)\| \triangle x^{nsd}$ (un-normalized)
- Newton's method:
  - $p = -\nabla^2 f(x)^{-1} \nabla f(x)$

13.5 Network flow says (II)

- Here, there is a better way to choose p!
- Let $x := x + \alpha p$, then we have:

$$\begin{array}{lll} \min & d^\mathsf{T} x_0 + \alpha d^\mathsf{T} p & \Rightarrow d^\mathsf{T} < 0 \\ \text{s. t.} & -x_0 \le \alpha p \le c - x_0 & \Rightarrow \text{residual graph} \\ & A^\mathsf{T} p = 0 & \Rightarrow p \text{ is a cycle!} \end{array}$$

- In other words, choose p to be a negative cycle with cost d!
  - Simple negative cycle, or
  - Minimum mean cycle

13.6 Network flow says (III)

- Step size is limited by the capacity constraints:
  - $\alpha_1 = \min_{ij}\{c^+ - x_0\}$, for $\triangle x_{ij} > 0$
  - $\alpha_2 = \min_{ij}\{x_0 - c^-\}$, for $\triangle x_{ij} < 0$
  - $\alpha_{\text{lin}} = \min\{\alpha_1, \alpha_2\}$
- If $\alpha_{\text{lin}} = +\infty$, the problem is unbounded.

13.7 Network flow says (IV)

- An initial feasible solution can be obtained by a similar construction of the residual graph and cost vector.
- The LEMON package implements this cycle cancelling algorithm.

13.8 Min-Cost Flow Convex Problem

- Problem Formulation:

$$\begin{aligned} \min \quad & f(x) \\ \text{s. t.} \quad & 0 \le x \le c, \\ & A^\mathsf{T} x = b,\ b(V) = 0 \end{aligned}$$

13.9 Common Types of Line Search

- Exact line search: $t = \text{argmin}_{t>0} f(x + t\triangle x)$
- Backtracking line search (with parameters $\alpha \in (0, 1/2), \beta \in (0, 1)$)
  - starting from $t = 1$, repeat $t := \beta t$ until

$$f(x + t\triangle x) < f(x) + \alpha t \nabla f(x)^\mathsf{T} \triangle x$$

  - graphical interpretation: backtrack until $t \le t_0$

13.10 Network flow says (V)

- The step size is further limited by the following:
  - $\alpha_{\text{cvx}} = \min\{\alpha_{\text{lin}}, t\}$
- In each iteration, choose $\triangle x$ as a negative cycle of G_x, with cost $\nabla f(x)$ such that $\nabla f(x)^\mathsf{T} \triangle x < 0$

13.11 Quasi-convex Minimization (new)

- Problem Formulation:

$$\begin{array}{ll} \min & f(x) \\ \text{s. t.} & 0 \leq x \leq c, \\ & A^{\mathsf{T}}x = b,\ b(V) = 0 \end{array}$$

- The problem can be recast as:

$$\begin{array}{ll} \min & t \\ \text{s. t.} & f(x) \leq t, \\ & 0 \leq x \leq c, \\ & A^{\mathsf{T}}x = b,\ b(V) = 0 \end{array}$$

13.12 Convex Optimization says (II)

- Consider a convex feasibility problem:

$$\begin{array}{ll} \text{find} & x \\ \text{s. t.} & \phi_t(x) \leq 0, \\ & 0 \leq x \leq c, \\ & A^{\mathsf{T}}x = b,\ b(V) = 0 \end{array}$$

  - If feasible, we conclude that $t \geq p^*$;
  - If infeasible, $t < p^*$.
- Binary search on t can be used for obtaining p^*.

13.13 Network flow says (VI)

- Choose $\triangle x$ as a negative cycle of G_x with cost $\nabla\phi_t(x)$
- If no negative cycle is found, and $\phi_t(x) > 0$, we conclude that the problem is infeasible.
- Iterate until x becomes feasible, i.e. $\phi_t(x) \leq 0$.

13.14 E.g. Linear-Fractional Cost

- Problem Formulation:

$$\begin{array}{ll} \min & (e^{\mathsf{T}}x + f)/(g^{\mathsf{T}}x + h) \\ \text{s. t.} & 0 \leq x \leq c, \\ & A^{\mathsf{T}}x = b,\ b(V) = 0 \end{array}$$

- The problem can be recast as:

$$\begin{array}{ll} \min & t \\ \text{s. t.} & (e^{\mathsf{T}}x + f) - t(g^{\mathsf{T}}x + h) \leq 0 \\ & 0 \leq x \leq c, \\ & A^{\mathsf{T}}x = b,\ b(V) = 0 \end{array}$$

13.15 Convex Optimization says (III)

- Consider a convex feasibility problem:

$$\begin{array}{ll} \text{find} & x \\ \text{s. t.} & (e - t \cdot g)^{\mathsf{T}}x + (f - t \cdot h) \leq 0, \\ & 0 \leq x \leq c, \\ & A^{\mathsf{T}}x = b,\ b(V) = 0 \end{array}$$

  - If feasible, we conclude that $t \geq p^*$;
  - If infeasible, $t < p^*$.
- Binary search on t can be used for obtaining p^*.

13.16 Network flow says (VII)

- Choose $\triangle x$ to be a negative cycle of G_x with cost $(e - t \cdot g)$, i.e. $(e - t \cdot g)^{\mathsf{T}} \triangle x < 0$
- If no negative cycle is found, and $(e - t \cdot g)^{\mathsf{T}}x_0 + (f - t \cdot h) > 0$, we conclude that the problem is infeasible.
- Iterate until $(e - t \cdot g)^{\mathsf{T}}x_0 + (f - t \cdot h) \leq 0$.

13.17 E.g. Statistical Optimization

- Consider the quasi-convex problem:

$$\begin{aligned} \min \quad & \Pr(\mathbf{d}^\mathsf{T} x > \alpha) \\ \text{s. t.} \quad & 0 \leq x \leq c, \\ & A^\mathsf{T} x = b,\ b(V) = 0 \end{aligned}$$

- $\mathbf{d}$ is random vector with mean d and covariance Σ.
- Hence, $\mathbf{d}^\mathsf{T} x$ is a random variable with mean $d^\mathsf{T} x$ and variance $x^\mathsf{T} \Sigma x$.

13.18 Statistical Optimization

- The problem can be recast as:

$$\begin{aligned} \min \quad & t \\ \text{s. t.} \quad & \Pr(\mathbf{d}^\mathsf{T} x > \alpha) \leq t \\ & 0 \leq x \leq c, \\ & A^\mathsf{T} x = b,\ b(V) = 0 \end{aligned}$$

Note:

$$\begin{aligned} & \Pr(\mathbf{d}^\mathsf{T} x > \alpha) \leq t \\ \Rightarrow \quad & d^\mathsf{T} x + F^{-1}(1-t)\|\Sigma^{1/2} x\|_2 \leq \alpha \end{aligned}$$

(convex quadratic constraint w.r.t x)

13.19 Recall...

Recall that the gradient of $d^\mathsf{T} x + F^{-1}(1-t)\|\Sigma^{1/2} x\|_2$ is $d + F^{-1}(1-t)(\|\Sigma^{1/2} x\|_2)^{-1}\Sigma x$.

13.20 Problem w/ additional Constraints (new)

- Problem Formulation:

$$\begin{aligned} \min \quad & f(x) \\ \text{s. t.} \quad & 0 \leq x \leq c, \\ & A^\mathsf{T} x = b,\ b(V) = 0 \\ & s^\mathsf{T} x \leq \gamma \end{aligned}$$

13.21 E.g. Yield-driven Delay Padding

- Consider the following problem:

$$\begin{aligned} \text{maximize} \quad & \gamma\,\beta - c^{\mathsf{T}} p, \\ \text{subject to} \quad & \beta \le \Pr(y_{ij} \le \mathbf{d}_{ij} + p_{ij}), \\ & Au = y,\ p \ge 0 \end{aligned}$$

  - p: delay padding
  - γ: weight (determined by a trade-off curve of yield and buffer cost)
  - $\mathbf{d}_{ij}$: Gaussian random variable with mean d_{ij} and variance s_{ij}.

13.22 E.g. Yield-driven Delay Padding (II)

.pull-left[

- The problem is equivalent to:

$$\begin{aligned} \max \quad & \gamma\,\beta - c^{\mathsf{T}} p, \\ \text{s.t.} \quad & y \le d - \beta s + p, \\ & Au = y, p \ge 0 \end{aligned}$$

]

.pull-right[

- or its dual:

$$\begin{aligned} \min \quad & d^{\mathsf{T}} x \\ \text{s.t.} \quad & 0 \le x \le c, \\ & A^{\mathsf{T}} x = b,\ b(V) = 0 \\ & s^{\mathsf{T}} x \le \gamma \end{aligned}$$

]

13.23 Recall ...

- Yield drive CSS:

$$\begin{aligned} \max \quad & \beta, \\ \text{s.t.} \quad & y \le d - \beta s, \\ & Au = y, \end{aligned}$$

- Delay padding

$$\begin{aligned} \max \quad & -c^{\mathsf{T}} p, \\ \text{s.t.} \quad & y \leq d + p, \\ & Au = y,\ p \geq 0 \end{aligned}$$

13.24 Considering Barrier Method

- Approximation via logarithmic barrier:

$$\begin{aligned} \min \quad & f(x) + (1/t)\phi(x) \\ \text{s.t.} \quad & 0 \leq x \leq c, \\ & A^{\mathsf{T}} x = b,\ b(V) = 0 \end{aligned}$$

  - where $\phi(x) = -\log(\gamma - s^{\mathsf{T}} x)$
  - Approximation improves as $t \to \infty$
  - Here, $\nabla\phi(x) = s/(\gamma - s^{\mathsf{T}} x)$

13.25 Barrier Method

- **Input**: a feasible x, $t := t^{(0)}$, $\mu > 1$, tolerance $\varepsilon > 0$
- **Output**: x^*
- **repeat**
  1. Centering step. Compute $x^*(t)$ by minimizing $t\,f + \phi$
  2. Update $x := x^*(t)$.
  3. Increase t. $t := \mu t$
- **until** $1/t < \varepsilon$.

Note: Centering is usually done by Newton's method in general.

13.26 Network flow says (VIII)

In the centering step, instead of using the Newton descent direction, we can replace it with a negative cycle on the residual graph.

14 Useful Skew Design Flow

14.1 Useful Skew Design: Why vs. Why Not

14.1.1 Why not

Some common challenges when implementing useful skew design include:

- need more engineer training
- difficulty in building a balanced clock-tree
- uncertainty in how to handle process variation and multi-corner multi-mode issues ..., etc.

14.1.2 Why

If these challenges are overcome and useful skew design is implemented correctly,

- it can lead to less time spent on timing issues
- get better chip performance or yield

14.2 Clock Arrival Time vs. Clock Skew

- Clock signal runs periodically.
- Thus, absolute clock arrival time u_i is not so important.
- Instead, the skew $y_{ij} = u_i - u_j$ is more important in this scenario.

14.3 Useful Skew Design vs. Zero-Skew Design

- "Critical cycle" instead of "critical path".
- "Negative cycle" instead of "negative slack".
- If there is a negative cycle, it means that there is no positive slack solution no matter how to schedule.
- Others are pretty much the same.
- Same design principle:
  - Always tackle the most critical one first!

14.4 Linear Programming vs. Network Flow Formulation

- Linear programming formulation
  - can handle more complex constraints
- Network flow formulation
  - usually more efficient
  - return the most critical cycle as a bonus
  - can handle quantized buffer delay (???)
- Anyway, timing analysis is much more time-consuming than the optimization solving.

14.5 Target Skew vs. Actual Skew

Don't mess up these two concepts:

- Target skew:
  - the skew we want to achieve in the scheduling stage.
  - Usually deterministic (we schedule a meeting at 10:00, rather than 10:00 $\pm$ 34 minutes, right?)
- Actual skew
  - the skew that the clock tree actually generates.
  - Can be formulated as a random variable.

14.6 A Simple Case

To warm up, let us start with a simple case:

- Assume equal path delay variations.
- Single-corner.
- Before a clock tree is built.
- No adjustable delay buffer (ADB).

14.7 Network

14.7.1 Definition (Network)

A *network* is a collection of finite-dimensional vector spaces of *nodes* and *edges/arcs*:

- $V = \{v_1, v_2, \cdots, v_N\}$, where $|V| = N$
- $E = \{e_1, e_2, e_3, \cdots, e_M\}$ where $|E| = M$

which satisfies 2 requirements:

1. The boundary of each edge is comprised of the union of nodes
2. The intersection of any edges is either empty or a boundary node of both edges.

14.8 Example

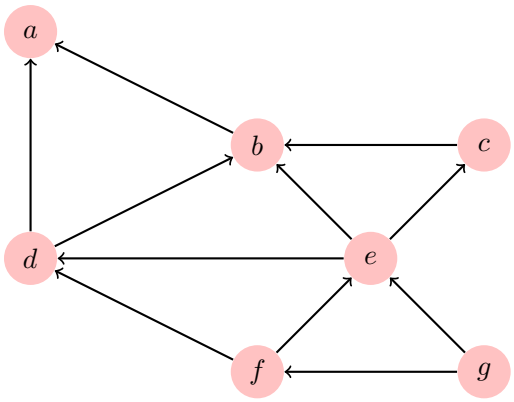


图 35: A network

14.9 Orientation

14.9.1 Definition (Orientation)

An *orientation* of an edge is an ordering of its boundary node (s, t), where

- s is called a source/initial node
- t is called a target/terminal node

14.9.2 Definition (Coherent)

Two orientations to be the same is called *coherent*

14.10 Node-edge Incidence Matrix

14.10.1 Definition (Incidence Matrix)

A $N \times M$ matrix A^{T} is a node-edge incidence matrix with entries:

$$A(i,j) = \begin{cases} +1 & \text{if } e_i \text{ is coherent with } v_j, \\ -1 & \text{if } e_i \text{ is not coherent with } v_j, \\ 0 & \text{otherwise.} \end{cases}$$

14.10.2 Example (II)

$$A^{\mathsf{T}} = \begin{bmatrix} 0 & -1 & 1 & 1 & 0 \\ 1 & 1 & 0 & -1 & -1 \\ -1 & 0 & -1 & 0 & 1 \end{bmatrix}$$

14.11 Timing Constraint

- Setup time constraint

  $$y_{\text{skew}}(i,f) \leq T_{\text{CP}} - D_{if} - T_{\text{setup}} = u_{if}$$

  While this constraint destroyed, cycle time violation (zero clocking) occurs.
- Hold time constraint

  $$y_{\text{skew}}(i,f) \geq T_{\text{hold}} - d_{if} = l_{if}$$

  While this constraint destroyed, race condition (double clocking) occurs.

14.12 Timing Constraint Graph

- Create a graph (network) by
  - replacing the hold time constraint with an *h-edge* with cost $-(T_{\text{hold}} - d_{ij})$ from FF_i to FF_j, and
  - replacing the setup time constraint with an s-edge with cost $T_{\text{CP}} - D_{ij} - T_{\text{setup}}$ from FF_j to FF_i.
- Two sets of constraints stemming from clock skew definition:

– The sum of skews for paths having the same starting and ending flip-flop to be the same;
– The sum of clock skews of all cycles to be zero

14.13 Timing Constraint Graph (TCG)

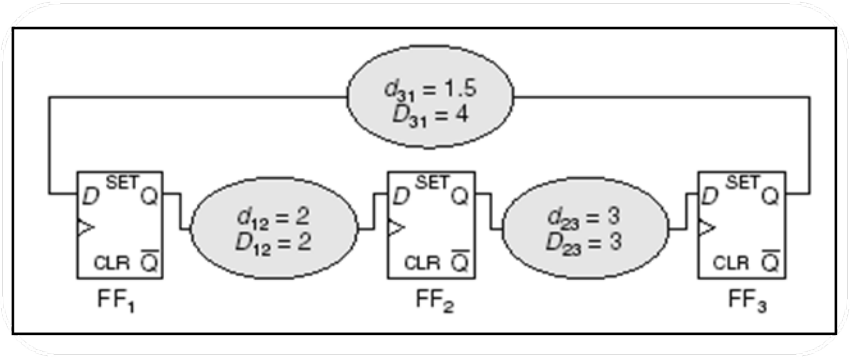
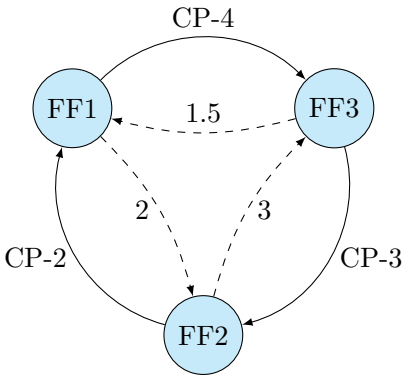


图 36: Example circuit



15 First Thing First

15.1 Meet all timing constraints

- Find y in $\{y \in \mathbb{R}^n \mid y \leq d, A\,u = y\}$
- How to solve:
  1. Find a negative cycle, fix it.

2. Iterate until no negative cycle is found.

- Bellman-Ford-like algorithm (and its variants are publicly available):
  - Strongly suggest "Lazy Evaluation":
    - Don't do full timing analysis on the whole timing graph at the beginning!
    - Instead, perform timing analysis only when the algorithm needs.
  - Stop immediately whenever a negative cycle is detected.

15.2 Delay Padding (DP)

- Delay padding is a technique that fixes the timing issue by intentionally **solely** "increasing" delays.
- Usually formulated as:
  - Find p, y in $\{p, y \in \mathbb{R}^n \mid y \leq d + p, A\,u = y, p \geq 0\}$
- If the objective is to minimize the sum of p, then the problem is the dual of the standard *min-cost flow* problem, which can be solved efficiently by the *network simplex* algorithm (publicly available).
- Beautiful right?

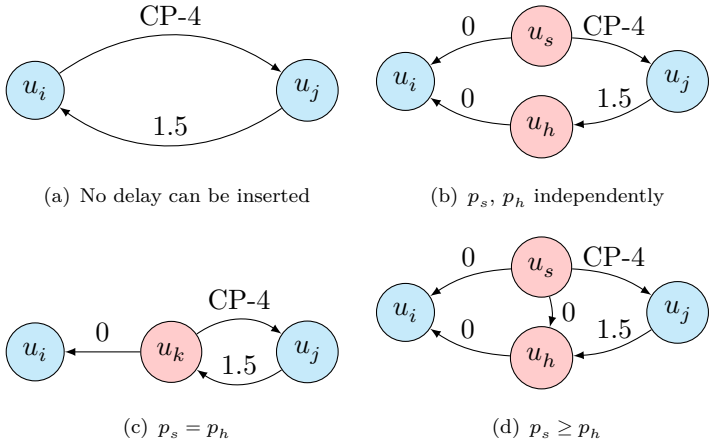
15.3 Delay Padding (II)

- No, the above formulation is impractical.
- In modern design, "inserting" a delay may mean swapping a faster cell with a slower cell from the cell library. Thus, no need to minimize the sum of p.
- More importantly, it may not be possible to find a position to insert delay for some delay paths.
- Some papers consider only allowing insert delays to the max-delay path only. Some papers consider only allowing insert delays to both the max- and min-delay paths together only. None of them are perfect.

15.4 Delay Padding (III)

- My suggestion. Instead of calculating the necessary $p's$ and then look for the suitable position to insert, it is easier (and more flexible) to determine the position first and then calculate the suitable values.
- It can be achieved by modifying the timing graph and solve a feasibility problem. Easy enough!
- Quantized delay can be handled too (???).

15.5 Four possible ways to insert delay



(a) No delay can be inserted

(b) p_s, p_h independently

(c) $p_s = p_h$

(d) $p_s \geq p_h$

图 37:

15.6 Delay Padding (cont'd)

- If there exists a negative cycle in the modified timing graph, it implies that the timing problem cannot be fixed by simply the delay padding technique.
  - Then, try decrease D_{ij}, or increase T_{CP}
- Be aware of the min-delay path is still the min-delay path after a certain amount of delay is inserted (how???).

16 Variation Issue

16.1 Yield-driven Clock Skew Scheduling

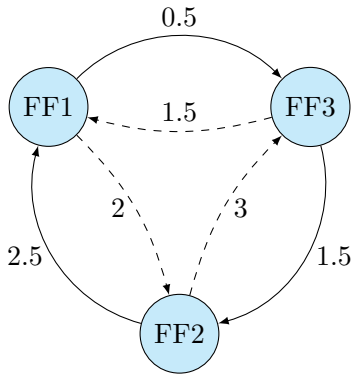
- Assume all timing issues are fixed.
- Now, how to schedule the arrival times to maximize yield?
- According to the critical-first principle, we seek for the most critical cycle first.
- The problem can be formulated as:
  - $\max\{\beta \in \mathbb{R} \mid y \leq d - \beta, A\,u = y\}$.
- It is equivalent to the *minimum mean cycle* problem, which can be solved efficiently by for example *Howard's algorithm* (publicly available).

16.2 Minimum Balancing Algorithm

- Then we evenly distribute the slack on this cycle.
- To continue the next most critical cycle, we contract the first one into a "super vertex" and repeat the process.
- The process stops when the timing graph remains only a single vertex.
- The overall method is known as *minimum balancing* (MB) algorithm in the literature.

16.3 Example: Most timing-critical cycle

The most vulnerable timing constraint



16.4 Example: Distribute the slack

- Distribute the slack evenly along the most timing-critical cycle.

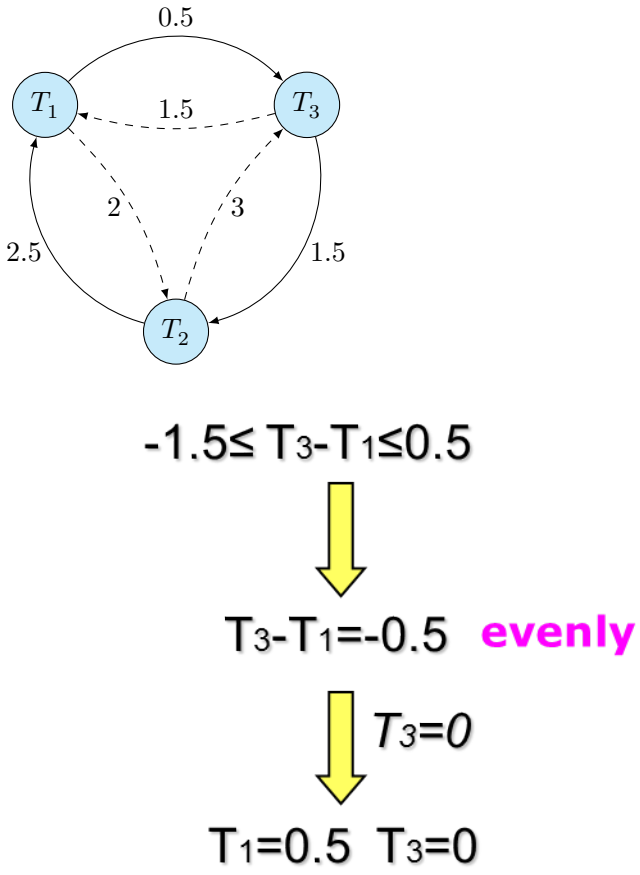


图 38: img

16.5 Example: Distribute the slack (cont'd)

- To determine the optimal slacks and skews for the rest of the graph, we replace the critical cycle with a super vertex.

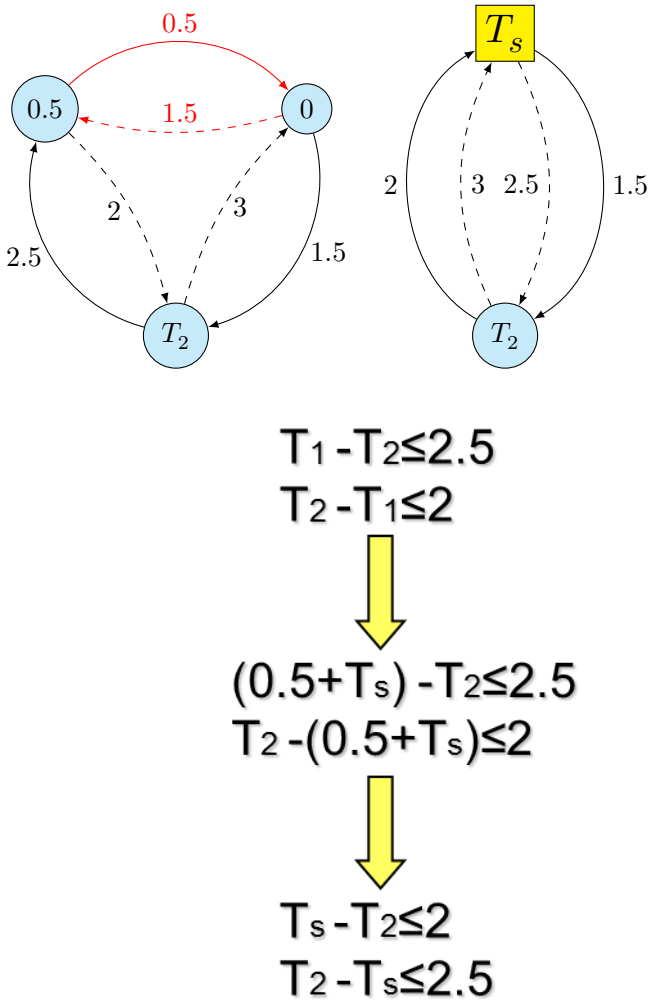

0.5
0.5
1.5
0
2
3
2.5
1.5
T_2
T_s
2
3
2.5
1.5
T_2
T1 -T2≤2.5
T2 -T1≤2
(0.5+Ts) -T2≤2.5
T2 -(0.5+Ts)≤2
Ts -T2≤2
T2 -Ts≤2.5

图 39: img

16.6 Repeat the process iteratively

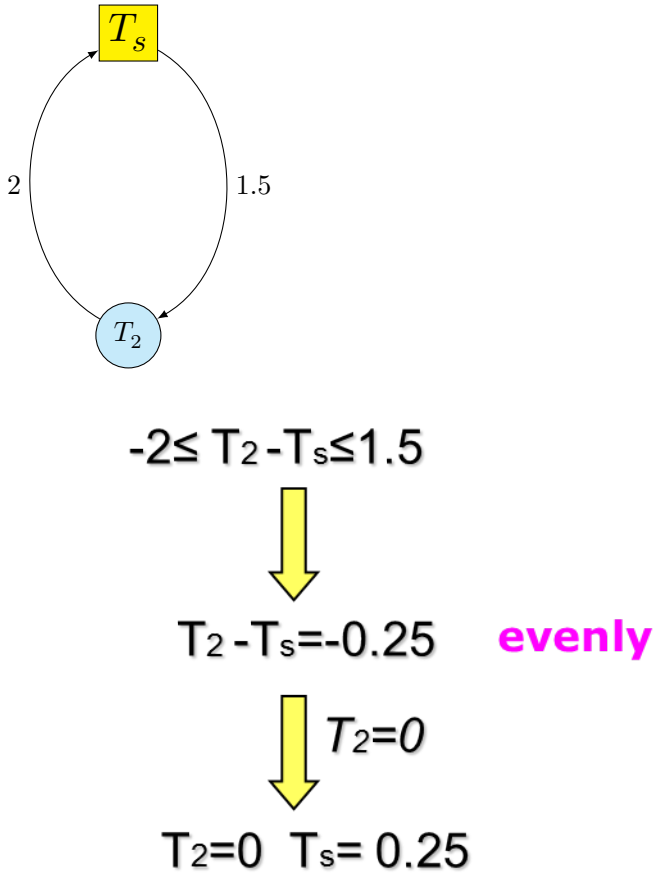


图 40: img

16.7 Repeat the process iteratively (II)

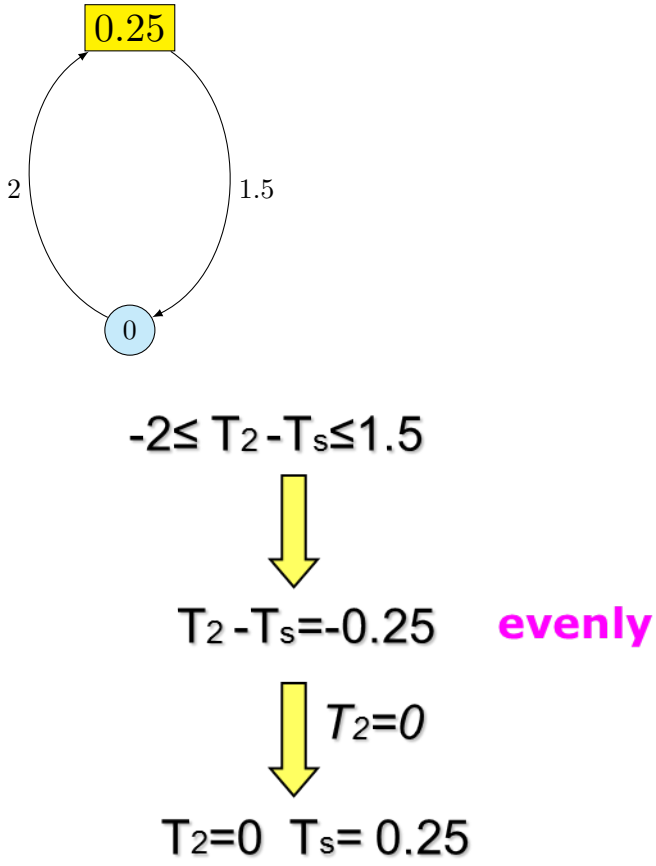


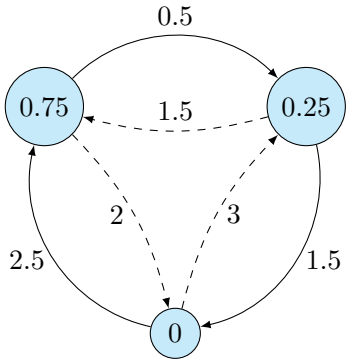
图 41: img

16.8 Final result

- $\text{Skew}_{12} = 0.75$
- Skew_{23} = -0.25
- Skew_{31} = -0.5
- $\text{Slack}_{12} = 1.75$
- $\text{Slack}_{23} = 1.75$

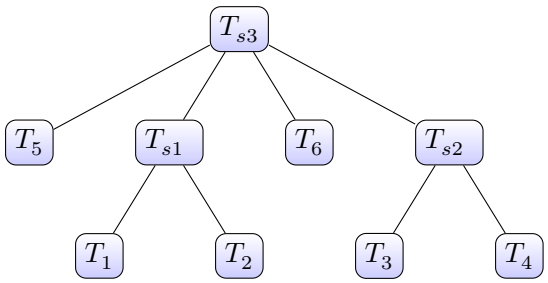
- $\text{Slack}_{31} = 1$

  where Slack_{ij} = CP - D_{ij} - T_{setup} - Skew_{ij}



16.9 What the MB algorithm really give us?

- The MB algorithm not only give us the scheduling solution, but also a tree-topology that represents the order of "criticality"!



16.10 Clock-tree Synthesis and Placement

- I strongly suggest that the topology of the clock-tree precisely follows the order of "criticality"!
  - since the lower branch of clock-tree has smaller skew variation.
- I also suggest that the placer should follow the topology of the clock-tree:
  - Physically place the registers of the same branch together.
  - The locality implies stronger correlation of variations and implies even smaller skew variation due to the cancellation effect.

– Note that the current SSTA does not provide the correlation information, so this is the best you can do!

16.11 Second Example: Yield-driven Clock Skew Scheduling

- Now assume that SSTA (or STA+OCV, POCV, AOCV) is performed.
- Let $(\bar{d}, s)$ be the (mean, variance) of $\mathbf{d}$
- The most critical cycle can be obtained by solving:
  - $\max\{\beta \in \mathbb{R} \mid y \leq \bar{d} - \beta s, A\,u = y\}$
- It is equivalent to the minimum cost-to-time ratio cycle problem, which can be solved efficiently by for example Howard's algorithm (publicly available).
- Gaussian distribution is assumed. For arbitrary distribution, see my DAC'08 paper.

16.12 What About the Correlation?

- In the above formulation, we minimum the maximum possibility of timing violation of each *individual* timing constraint. So only individual delay distribution is needed.
- Yes, the objective function is not the true timing-yield. But it is reasonable, easy to solve, and is the best you can do so far.

17 Multi-Corner Issue

17.1 Meet all timing constraints in Multi-Corner

- Assume no Adjustable Delay Buffer (ADB)
- Find y in $\{y \in \mathbb{R}^n \mid y \leq d^{(k)}, A\,u = y, \forall k \in [1..K]\}$
- Equivalent to finding y in $\{y \in \mathbb{R}^n \mid y \leq \min_k\{d^{(k)}\}, A\,u = y\}$
- Feasibility problem
- How to solve:
  1. Find a negative cycle, fix it.
  2. Iterate until no negative cycle is found.

- Better avoid fixing the timing issue corner-by-corner. Inducing ping-pong effect.

17.2 Delay padding (DP) in Multi-Corner

- The problem CANNOT be formulated as a network flow problem. But still you can solve it by a linear programming formulation.
- Or, decompose the problem into sub-problems for each corner.
- Again use the modified timing graph technique.
- Then, y's are shared variables of sub-problems.
- If we solve each sub-problem individually, the solution will not agree with each other. Induce *ping-pong effect*.
- Need something to drive the agreement.

17.3 Delay Padding (DP) in Multi-Corner (cont'd)

- Follow the idea of *dual decomposition*: If a solution is above the average. then introduce a punishment cost. If a solution is below the average, then introduce a rewarding cost.
- Then, each subproblem is a min-cost potential problem, which can be solved efficiently.
- If some subproblems do not have feasible solutions, it implies that the problem cannot be fixed by simply delay padding.
- The process repeats until all solutions converge. If not, it implies that the problem cannot be fixed by simply delay padding.

17.4 Yield-driven Clock Skew Scheduling

- $\max\{\beta \in \mathbb{R} \mid y \leq d^{(k)} - \beta s, A\,u = y, \forall k \in [1..K]\}$
- More or less the same as in Single Corner.

18 Clock-Tree Issue

18.1 Clock Tree Synthesis (CTS)

- Construct merging location

  - DME algorithm, Elmore delay, buffer insertion
- Some research on *bounded-skew DME algorithm*. But the algorithm is too complicated in my opinion.
- If the previous stage is over-optimized, the clock tree is hard to implement. If it happens, some budgeting techniques should be invoked (engineering issue)
- After a clock tree is constructed, more detailed timing (rather than Elmore delay) can be obtained via timing analysis.

18.2 Co-optimization Issue

- After a clock tree is built, we have a clearer picture.
- Should I perform the re-scheduling? And how?
- Some papers suggest adding a factor to the timing constraint, say:

$$1.2u_i - 0.8u_j \leq w_{ij}$$

.
- Then the formulation is not a kind of network-flow, but may still be solvable by linear programming.
- Need to investigate more deeply.

19 Adjustable Delay Buffer Issue

19.1 Adjustable delay buffers in Multi-Mode

- Assume adjustable delay buffers are added solely to the clock tree
- Hence, each mode can have a different set of arrival times.
- Easier for clock skew scheduling, harder for clock-tree synthesis.

19.2 Meet timing constraint in Multi-Mode:

- find $y^{(m)}$ in $\{y^{(m)} \in \mathbb{R}^n \mid y^{(m)} \leq d^{(m)}, A\,u^{(m)} = y^{(m)}, \forall m \in [1..M]\}$
- Can be done in parallel.
- find a negative cycle, fix it (do not need to know all $d_i^{(m)}$ at the beginning) for every mode in parallel.

19.3 Delay Padding (DP) in Multi-mode

- Again use a modified timing graph technique.
- NOT a network flow problem. Use LP, or
- Dual decomposition -> min-cost potential problem for each mode
  - Only p's are shared variables.
  - Initial feasible solution obtained by the single-mode method
    - A negative cycle => problem cannot be fixed by DP
- Not converge => problem cannot be fixed by DP
  - Try decrease D_{ij}, or increase T_{CP}

19.4 Yield-driven Clock Skew Scheduling

- $\max\{\beta \in \mathbb{R} \mid y^{(m)} \leq d^{(m)} - \beta s, A\, u^{(m)} = y^{(m)}, \forall m \in [1..M]\}$
- Pretty much the same as Single-Mode.

19.5 Difficulty in ADB Multi-Mode Design

- How to design the clock-tree?
- What is the order of criticality?
- How to determine the minimum range of ADB?