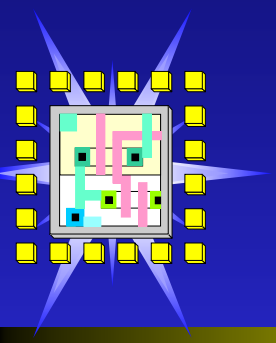


A Performance-Constrained Template-Based Layout Retargeting Algorithm for Analog Integrated Circuits

Zheng Liu Lihong Zhang

Memorial University of Newfoundland

St. John's, Canada

Overview

- Introduction
- Analog Layout Retargeting Design Flow
- Problem Formulation and Modeling
- MINLP-Based Retargeting Algorithm
- Experimental Results
- Conclusions

Introduction

- System-on-chip (SoC) application necessitates analog design automation
- Layout parasitics can be significantly sensitive to analog circuit performances
- Rough estimation during the optimization phase
- Analog circuits have become a design bottleneck for the growing mixed-signal SoC market
- Special analog automated design tools are needed for analog integrated circuits

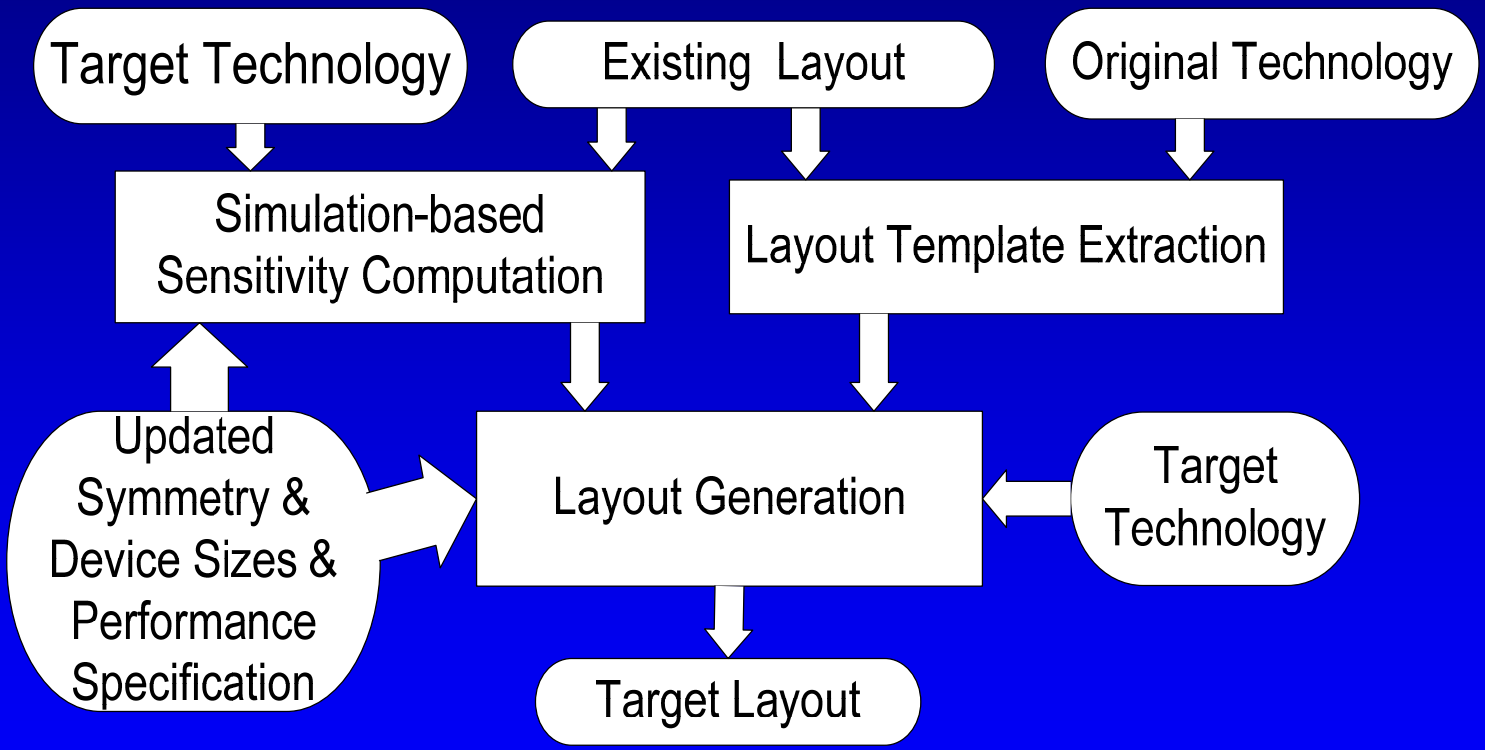
Review of Prior Work

- Analog layout optimization tools have been developed with limited design aspects
- A fully integrated constraint-driven analog layout system (PARCAR)
- Macro-cell based layout automation systems (including KOAN/ANAGRAM-II, LAYLA, and ALADIN)
- IPRAIL – Intel lecture Property Reuse Based Layout Automation

Overview

- Introduction
- Analog Layout Retargeting Design Flow
- Problem Formulation and Modeling
- MINLP-Based Retargeting Algorithm
- Experimental Results
- Conclusions

Analog Layout Retargeting Design Flow



Overview

- Introduction
- Analog Layout Retargeting Design Flow
- Problem Formulation and Modeling
- MINLP-Based Retargeting Algorithm
- Experimental Results
- Conclusions

Problem Formulation & Modeling

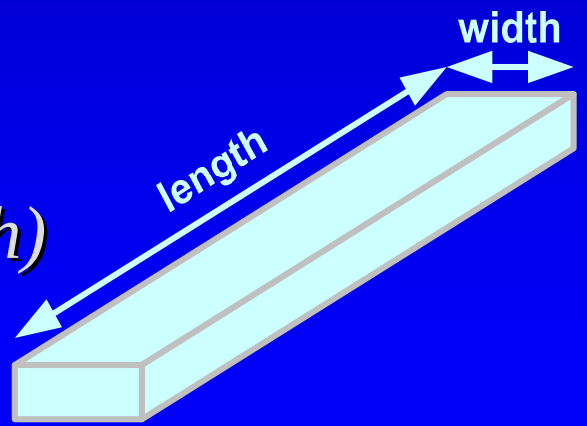
- Interconnect Parasitic Model

  Parasitic resistance and capacitance for a tile on a layer can be mathematically represented with its *length* and *width*:

$$R = \rho_{sh} \times (length / width)$$

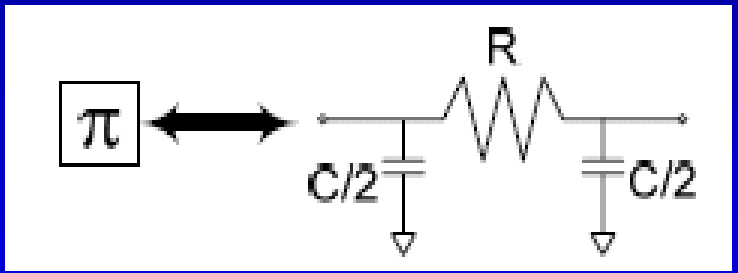
$$C_{sub} = c_a \times (length \times width) + c_{sw} \times (2 \times length)$$

$$C_{coup} = c_c \times (length / distance)$$



Interconnect Modeling

- Resistance-capacitance (RC) π -model is used to represent resistance and capacitance of a net



$$R = \rho_{sh} \times (x_r - x_l) / (y_r - y_l)$$

$$C_{sub} = c_a \times (x_r - x_l) \times (y_r - y_l) + c_{sw} \times 2 \times (x_r - x_l)$$

$$C_{coup} = c_c \times (x_r - x_l) / distance$$

Performance and Matching Constraints

- To ensure the desired circuit performance, the performance deviation must be restricted within a maximum allowed tolerance

$$\sum S_{jres} \times R_j(x,y) + \sum S_{jcap} \times C_j(x,y) \le \Delta W_{i\_\max}$$

- Matching parasitic constraints are indispensable for the parasitic-aware optimization problem

$$P_{n1}(x, y) = P_{n2}(x, y)$$

Sensitivity Computation

- Performance sensitivity is utilized to quantify the dependence of circuit performance with respect to parasitics

$$S_{ij} = \partial W_i / \partial p_j$$

- The segmental sensitivity of W_i with respect to p_j is modeled using finite-difference approximation as

$$S_{ij} = [W_{ij}(p_1) - W_{ij}(p_2)] / (p_1 - p_2)$$

Central-Difference Sensitivity

- Finite-difference approximation is not able to generally represent the expected sensitivity when p_1 is far away from p_2
- To manage desired performance, we advance the calculation to central-difference by assuming $p_1 = p_{j\_worst} + \Delta$ and $p_2 = p_{j\_worst} - \Delta$

$$S_{ij} = [W_{ij}(p_{j\_worst} + \Delta) - W_{ij}(p_{j\_worst} - \Delta)] / 2\Delta$$

- Sensitivity computation is conducted across parasitic upper bounds

Segmental Sensitivity

- For less sensitive parasitics, the central-difference approach can be used to generate plain upper-bound sensitivities to approximately model the general impact of parasitics on performance
- For sensitive nets, the sensitivities themselves are very large and may vary significantly along with changing parasitic values
- A piecewise sensitivity model is proposed to accurately represent performance sensitivities for sensitive parasitics

Piecewise Sensitivity Model

- Sensitivity analysis is conducted to identify the sensitive parasitics by running multiple simulations
- Optimization flexibility ranges of these parasitics are then divided into a number of small segments

$$\alpha = |S_{jmin} / S_{jmax}|$$

$$N_{seg} = N(\alpha, S_{jmax})$$

- Within each segment, the central-difference sensitivity method is used to calculate its upper-bound sensitivity
- Piecewise sensitivity can be built up as a linear function of binary-integer variables and segmental sensitivities

Overview

- Introduction
- Analog Layout Retargeting Design Flow
- Problem Formulation and Modeling
- MINLP-Based Retargeting Algorithm
- Experimental Results
- Conclusions

MINLP-Based Retargeting Algorithm

- The parasitic-aware analog layout retargeting and optimization can be formulated as a two-dimensional compaction problem
- By computing segmental sensitivities, the binary-integer piecewise sensitivities construct a set of coefficients
- Linear approximation is used for quick performance-deviation evaluation

Mixed Integer Non-linear Programming

$$\text{Minimize} \quad (x_{rr} - x_{ll}) \times (y_{rr} - y_{ll}),$$

Subject to

$$\sum_{j=1}^{N}\left[\left(\sum_{n=1}^{N_{res}} B_n S_n\right)\times R_j(x,y) + \left(\sum_{n=1}^{N_{cap}} B_n S_n\right)\times C_j(x,y)\right] \le \Delta W_{i\_\max}$$

$$P_{m1}(x,y) = P_{m2}(x,y),$$

$$\sum_{n=1}^{N_{res}} B_n = \sum_{n=1}^{N_{cap}} B_n = 1,\ B_n \in \{0,1\},$$

$$B_n\ (P - P_{n1}) \ge 0\ \&\&\ B_n\ (P - P_{n2}) \le 0,$$

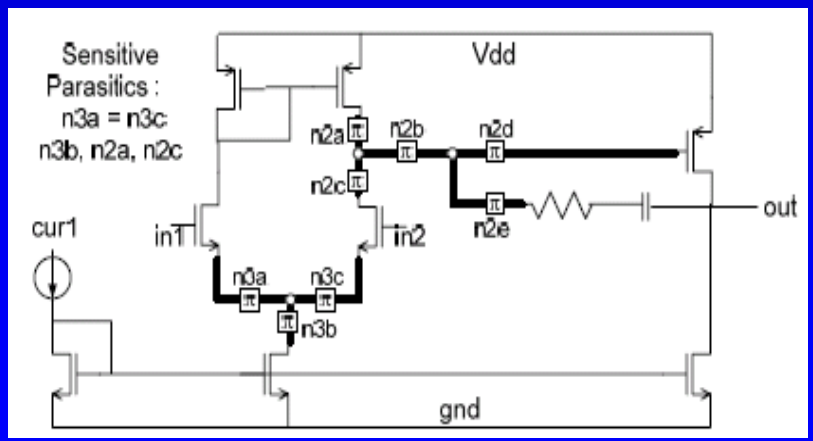
- where x_{rr}, x_{ll}, y_{rr}, and y_{ll} represent the boundaries
- $\sum_{n=1}^{N_{res}} B_n S_n$ and $\sum_{n=1}^{N_{cap}} B_n S_n$ refer to piecewise sensitivities of all the required performances

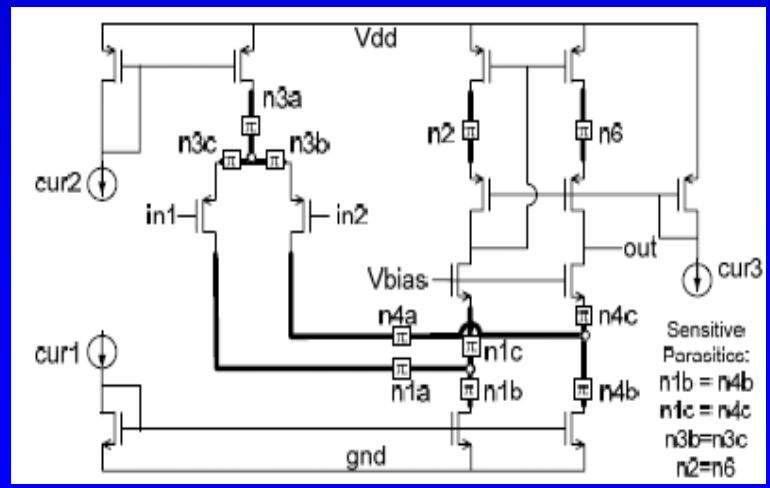
Overview

- Introduction
- Analog Layout Retargeting Design Flow
- Problem Formulation and Modeling
- MINLP-Based Retargeting Algorithm
- Experimental Results
- Conclusions

Experimental Results

- The proposed algorithm and design flow have been implemented in C++ and integrated into an automated layout tool
- Two analog circuits: a two-stage Miller-compensated operational amplifier (opamp) and a single-ended folded cascode opamp.
- The retargeting of these opamps was from a 0.25µm CMOS process to a 0.18µm CMOS process with updated performance specifications.





Sensitivity Computation

- Sensitivity computation was conducted using the central-difference scheme (S-CD) compared to the traditional technique advocated in PARCAR (called S-TT)
- The comparisons of some generated performance sensitivities and simulated circuit performance
- To simplify the comparison, only the performance of AC gain is considered and sensitivities of the critical parasitics are reported

	Parasitic Res. (Ω)	Sensitivities of Gain (dB/ Ω)		Realized Performance	
		S-CD	S-TT	S-CD	S-TT
2-stage opamp	R3a	-0.0324	-0.0020	Gain= 64.0	Gain= 60.1
	R3c	Match	Match		
	R2c	-0.1450	-0.0296		
Folded-cascode opamp	R3a	-0.0046	-0.0007	Gain= 60.6	Gain= 57.5
	R3b	-0.0096	-0.0012		
	R3c	Match	Match		
	R2	-0.0068	-0.0021		
	R6	Match	Match		

Comparison of Three Schemes

- Parasitic-aware layout retargeting was performed using three schemes

  PMI: proposed performance-constrained mixed-integer method

  PB: parasitic-bound based retargeting (called PB)

  PS: a similar flow using single upper-bound sensitivities (non-piecewise) and being solved by nonlinear programming

Res.	PB-Bound (Ω)	PS Sensitivity (dB/Ω)	PMI-Piecewise Sensitivities (dB/Ω) (B_i: 0-1 variable)
R3a	4.1	-0.032	$-0.022B_1-0.032B_2$
R3c	4.1	-0.032	$-0.022B_1-0.032B_2$
R3b	29.4	-0.068	$-0.043B_3-0.052B_4-0.068B_5$
R2a	24.0	0.031	$0.031B_6$
R2b	50.2	-0.001	$-0.001B_7$
R2c	19	-0.14	$-0.115B_8-0.128B_9-0.132B_{10}-0.145B_{11}$
R2e	112.9	0.001	$0.001B_{12}$

Cap.	PB Bound (fF)	PS Sensitivity (degree/ fF)	PMI-Piecewise Sensitivities (degree/fF) (B_i: 0-1 variable)
C1b	2.7	-0.043	$-0.035B_1-0.043B_2$
C4b	2.7	-0.043	$-0.035B_1-0.043B_2$
C1c	2.7	-0.043	$-0.038B_3-0.043B_4$
C4c	2.7	-0.043	$-0.038B_3-0.043B_4$
C3b	17.1	-0.008	$-0.005B_5-0.007B_6-0.008B_7$
C3c	17.1	-0.008	$-0.005B_5-0.007B_6-0.008B_7$
C2	8.0	0.0014	$0.003B_6+0.021B_7+0.005B_8+0.0014B_9$
C6	8.0	0.0014	$0.003B_6+0.021B_7+0.005B_8+0.0014B_9$

Comparison of Extracted Parasitics

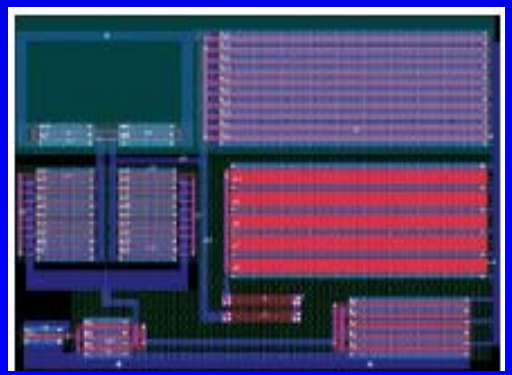
Res. (Ω)	PB	PS	Sensitivity-Range (PS)	PMI	Sensitivity-Range (PMI)
R3a	3.98	3.91	(4.0, 4.1)	3.90	(2.0, 4.0)
R3c	3.98	3.91	(4.0, 4.1)	3.90	(2.0, 4.0)
R3b	6.10	3.98	(29.3, 29.4)	3.98	(0.1, 9.8)
R2a	2.78	2.79	(23.9, 24.0)	2.73	(0, 24)
R2b	2.52	2.46	(50.1, 50.2)	2.35	(0, 50.2)
R2c	3.94	3.43	(18.9, 19.0)	2.07	(0, 4.75)
R2e	3.75	3.90	(112.8, 112.9)	3.90	(0, 112.9)

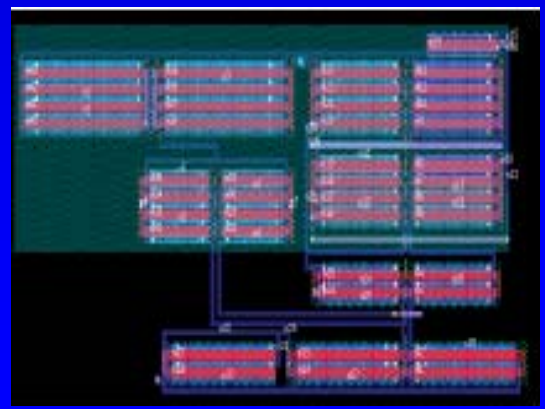
Cap. (fF)	PB	PS	Sensitivity-Range (PS)	PMI	Sensitivity-Range (PMI)
C1b	0.35	0.33	(2.6, 2.7)	0.33	(0, 1.35)
C4b	0.35	0.33	(2.6, 2.7)	0.33	(0, 1.35)
C1c	0.42	0.42	(2.6, 2.7)	0.42	(0, 1.35)
C4c	0.42	0.42	(2.6, 2.7)	0.42	(0, 1.35)
C3b	2.3	2.0	(17.0, 17.1)	2.1	(0, 5.7)
C3c	2.3	2.0	(17.0, 17.1)	2.1	(0, 5.7)
C2	1.9	1.1	(7.9, 8.0)	2.7	(2.0, 4.0)
C6	1.9	1.1	(7.9, 8.0)	2.7	(2.0, 4.0)

Performance Comparison

		Gain (dB)	BW (MHz)	PM (°)	GM (dB)	Area (μm^2)
Two-stage opamp	Spec.	60.0	100	90.0	10.0	-
	Ideal	64.5	107.3	90.5	16.8	-
	PB	63.9	105.1	90.5	16.7	3084
	PS	64.0	105.0	90.4	16.7	2920
	PMI	64.3	106.4	90.5	16.7	2815
Folded-cascode opamp	Spec.	60.0	60.0	60.0	10.0	-
	Ideal	60.7	63.7	61.2	10.4	-
	PB	60.6	63.7	60.3	10.2	2320
	PS	60.6	63.4	60.6	10.4	2262
	PMI	60.6	63.7	61.1	10.4	2190

	Two-Stage Opamp			Cascode Opamp		
	PB	PS	PMI	PB	PS	PMI
#Nodes	1050			1103		
#Parasitic Tiles	26			38		
Template Extraction	16.5s	16.5s	16.5s	20.4s	20.4s	20.4s
Parasitic Solving	42.7s	36.1s	40.3s	131.4s	69.3s	99.2s
Layout Generation	69.9s	63.3s	67.5s	212.0s	149.9s	179.8s





Overview

- Introduction
- Analog Layout Retargeting Design Flow
- Problem Formulation and Modeling
- MINLP-Based Retargeting Algorithm
- Experimental Results
- Conclusions

Conclusions

- A performance-constrained parasitic-aware automatic layout retargeting algorithm was presented
- Different from the conventional sensitivity analysis, we proposed a general piecewise central-difference based scheme by using any simulators for sensitivity computation
- Performance constraints due to parasitics are included in the formulated mixed-integer nonlinear problem rather than through indirect parasitic-bound constraints
- Experimental results show the proposed retargeting algorithm achieves less layout area and significant reduction of execution time