Automated Analog Layout Methodologies for Mixed-Signal SoC Implementation

Xindi Liu

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2025

Reading Committee:
C. -J. Richard Shi, Chair
Sajjad Moazeni
Ang Li

Program Authorized to Offer Degree:
Electrical and Computer Engineering

University of Washington

Abstract

Automated Analog Layout Methodologies
for Mixed-Signal SoC Implementation

Xindi Liu

Chair of the Supervisory Committee:
C. -J. Richard Shi
Department of Electrical and Computer Engineering

This dissertation presents two automated analog layout methodologies for high-performance mixed-signal SoCs:

- A template-based generator produces device-level and block-specific layouts in minutes, with batch-mode verification enabling fast optimization. An automated layout for a transceiver driver was generated and taped out in TSMC 28nm using a template-based generator.

- The LEGO methodology builds a process-portable analog cell library, where standard-cell-like analog cells are designed for seamless integration with digital blocks, supported by a full-stack schematic-to-GDS flow accommodating various layout constraints. Using this approach, two interface designs: Advanced Interface Bus (AIB) and Electronic Integrated Circuit (EIC), were implemented in GlobalFoundries 12nm, demonstrating modularity, and rapid integration with digital blocks.

These methodologies enable efficient analog layout generation while maintaining critical performance in complex AMS designs.

To my parents

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would first like to express my sincere gratitude to my advisor, Prof. C. -J. Richard Shi, for his continuous guidance, patience, and encouragement throughout my PhD. I am especially thankful for the freedom he gave me to explore new ideas, while always providing thoughtful feedback and support when I needed direction. This balance helped me grow as an independent researcher and shaped much of the work presented in this thesis.

I am also grateful to my thesis committee, Prof. Sajjad Moazeni, Prof. Ang Li, Prof. Mark Oskin, for their valuable time and constructive feedback. Their comments and suggestions not only improved the quality of this thesis but also broadened the way I think about my research.

My heartfelt thanks go to my colleagues in the SSRL, Ailing Piao, Jiayi Wang, Jialin Wang, Aili Wang, Chang Liu, Rongjin Xu, and many others, for creating such a collaborative and supportive environment. I would like to give special thanks to Huwan Peng, with whom I graduated and shared some of the toughest moments of this journey. I especially want to thank Prof. Chixiao Chen and Dr. Chien-Jian Tseng, for their guidance and mentorship. Their advice, encouragement, and generosity with their time made a significant difference in both my research and personal growth.

Outside the lab, I am deeply thankful to my close friends, Ban Wang, Xinyi Chang, Yuchen Yao, Qianwen Sun, and Chenchen Ning, for their years of friendship, support, and encouragement throughout this journey. I would also like to thank my dance team, FifthSeason: Lu Chen, Xiaoyu Yang, Sixuan Zou, and Yifan Wang. I am deeply grateful that through our shared love of dance, we have grown from teammates into close friends. The time we shared our stories, practiced, and performed together has brought me joy and unforgettable memories. I would also like to thank my dance community, StepUp, where I began my jour-

ney in dance. I am deeply grateful to Ruiming Liu and Yilin Jin for their unwavering support and companionship during the challenging COVID period. I would also like to acknowledge Luxuan, Yi, Jinghe, Shichun, Yue, Yuqing and many others, whose encouragement, joy, and friendship have made my experience in this community truly unforgettable. Being part of this community has given me inspiration, happiness, and countless cherished memories both on and off the dance floor.

Finally, I want to express my deepest love and thanks to my mother, Yan Liu, and my father, Ying Liu, for their unwavering support and trust, even when my path was uncertain. My mother, who also holds a PhD, has been a constant role model through her hard work, compassion, and perseverance. My father sparked my interest in engineering at a young age, inspiring me to follow an academic and professional path in the field. Your love, trust, and encouragement have given me the strength to overcome challenges and the confidence to keep moving forward.

This thesis would not have been possible without the guidance, support, and inspiration from all the wonderful people I have met. To each of you, I am truly grateful and send you all my love.

Chapter 1
INTRODUCTION

1.1 Motivation

Automated analog and mixed-signal (AMS) layout generation has become increasingly important as modern SoCs demand high performance, tight matching, and rapid design turnaround. Unlike digital circuits, analog layouts are highly sensitive to device placement, symmetry, and parasitics, making manual design time-consuming and error-prone.

While automated AMS layout tools address many design bottlenecks, emerging FinFET technologies introduce new opportunity and challenges. FinFET technologies below 16nm have been widely adopted for high-speed links. The FinFET process offers superior performance by providing a higher cut-off frequency and intrinsic gain compared to planar processes. This advantage allows circuits to meet the requirements of ultra-high-speed interconnects [1] . Transitioning to FinFET technology, despite its significant advantages, presents a range of challenges that must be addressed. The manufacturing process is considerably more complex due to the three-dimensional structure of FinFETs, requiring advanced lithography and precise etching techniques to create the fin, gate, and source/drain regions. Design paradigms need to shift, necessitating updates to Electronic Design Automation (EDA) tools and methodologies to accommodate new layout rules and device characteristics. Additionally, managing increased parasitic capacitances and resistances introduced by the 3D architecture is crucial to maintaining performance. Effects of process variability are more pronounced, potentially impacting yield and reliability.

Thus, the high speed circuit design with new process technologies applied becomes increasingly complex and time-consuming. The analog design is heavily depends on the designer's experience. To achieve higher performance in analog circuits, it often requires the

implementation of diverse architectures and the optimization of transistor and passive device sizes. Besides schematic, the experienced analog designers are also required to optimize the layout since the layout quality directly affects final performance. In contrast to digital circuits, which are constructed using pre-made digital standard cells, analog and mixed-signal design often requires manual creation of the layout for each device, drawing of rectangles for front-end-of-line (FEOL) layers, such as N-well, placing the devices and connecting individual wire connections. Moreover, any modifications to the schematic can lead to substantial alterations in the floorplan, resulting in days or even weeks to achieve a new version of a DRC/LVS clean layout, so the turnaround time is significantly constrained by the layout design process.

1.2 Prior Work

Although analog EDA tools lag behind their digital counterparts, efforts to automate analog layout design have a long history. Early frameworks introduced two complementary layout generators: STUCCO, a procedure-based generator that uses pre-defined templates for fundamental analog blocks, and MOSAIC, an optimization-based generator that places these blocks using algorithms like simulated annealing and completes the interconnections. These two approaches have formed the foundation of modern analog layout automation: procedure-based and optimization-based methods.

In recent years, further advancements have been made, including strategies for synthesis-friendly analog circuits and improved layout generation frameworks. Various paradigms have been proposed with different levels of generality, and while incorporating design-specific strategies or prior knowledge can improve automation efficiency, it may also reduce the general applicability of the flow. Three methods are commonly used in analog layout automation: [2]:

- Synthesis-friendly AMS circuits: Design highly digital and modularized AMS circuits that are suitable for commercially available digital physical design tools. A synthesis-

able SAR ADC was proposed in [3]. It only requires the custom-designed comparators besides digital standard cells. The fully digital architecture allows fast synthesis and provides sufficient programmability. In addition, Acells [4] proposes abuttable analog cells that can be automatically placed and routed like digital standard cells, achieving similar performance to manual Pcell-based layouts with minimal area overhead.

- Procedure-based layout generation: Generate layouts based on the pre-designed templates in a procedural approach. BAG2 [5] offers process-independent schematic and layout generators within a generator-based design framework. However, the reference generator heavily depends on the designer's knowledge of the circuit, including the use of loss functions to determine optimal transistor sizes and layout constraints. This design flow requires huge effort to develop a new design.

- Optimization-based layout synthesis: Formulate the layout generation as a constrained optimization problem and tends to model the layout quality as the objectives. MAGICAL [7] presents an analog IC layout system for generating layouts from unannotated netlists. However, the framework only considers symmetry constraint, including device symmetry and system symmetry, which are insufficient for high-performance analog circuits.

1.3 Overview of the Dissertation

To address these concerns, two of the framework as been proposed, to enable fast circuit designs:

- **Template driven layout generator and batch-mode verification suite** The template-driven layout generator, implemented in SKILL and parameterized with circuit and layout specifications, can produce DRC/LVS-clean layouts that replicate human-designed layouts. Its integrated verification flow not only ensures correctness

but also supports post-layout optimization, enabling rapid iteration and refinement of device placement, routing, and overall circuit performance.

- **LEGO analog cells library and constraint-driven layout generation framework** Designed in the style of digital standard cells, these blocks enable compatibility with automated place-and-route while meeting analog constraints. A custom layout generator applies user-defined rules for efficient schematic-to-layout translation. The methodology is demonstrated in two case studies: AIB and EIC, highlighting its adaptability to diverse mixed-signal design needs.

In this dissertation, we first explore the template-driven layout generator for design of a combo Phy transceiver in Chapter 2. It is designed in TSMC 28nm and that supports multi-standards, including LPDDR4x [8] and UCIE v1.0 [9] and supports multi signal modulation, including NRZ, PAM4 and Chord. A template-driven layout generator and batch mode verification for high-speed transmitter is designed for fast layout design and post-layout optimization.

Then, a constraint-driven, digital approach layout generator and a analog standard cell, named as LEGO-cell is introduced in Chapter 3. Different from traditional analog layout generator that target to create layout as manual layout, the LEGO-based framework extends the industry standard digital electronic design automation (EDA) tools and flows to support analog and mixed-signal design. This is made feasible by (1) introducing LEGO cells, parametrized "analog" standard cells following the rules of digital standard cell layouts, and (2) developing a full stack framework from schematic to GDS.

Two application are implemented with LEGO cell and framework, (1) AIB 2.0 custom block IP, and (2) EIC receiver frontend, which will be discussed in Chapter 4 and 5. The AIB implementation focus on applying the minimum-length transistors and passive devices, which also wrapped as standard cells. Major layout constraints are implemented in the framework, including P&R symmetry and non-default-rule(NDR) routing, eg. routing metal width. The experimental results shows the efficiency and compatibility of the LEGO-based

analog layout and other digital layout. Then, EIC application extends the framework to long-channel transistors and passive devices of wider selection of physical sizes. Also, more layout constraints are considered in the framework, including device array and dummy devices for matching devices.

Chapter 2

TEMPLATE DRIVEN LAYOUT GENERATOR FOR HIGH-SPEED TRANSMITTER FOR PLANNER TECHNOLOGY

This chapter introduces a template-based automation flow for reconfigurable UCIe/LPDDR interface layout design. By capturing design, layout, and simulation parameters within parameterized templates, the flow systematically explores transistor sizing and floorplan options. The approach not only ensures compliance with UCIe and LPDDR specifications but also shortens the schematic-to-layout iteration time from hours to minutes, demonstrating both efficiency and design consistency.

2.1 Introduction

High-speed analog and mixed-signal circuit design is complex and time-consuming, with tapeout-quality implementations heavily dependent on designer experience. Achieving higher performance often requires diverse circuit architectures, careful optimization of transistor and passive device sizes, and layout techniques that directly impact final performance. Unlike digital design, which leverages pre-characterized standard cells, analog layout remains largely manual. Designers must create device geometries (e.g., wells, diffusion, and gate structures), place devices, and route interconnects. Any schematic modification can cascade into major floorplan changes, often requiring days or weeks to regenerate a DRC/LVS-clean layout. Consequently, the design turnaround time is severely constrained by the layout phase.

To address this challenge, various analog layout automation frameworks have been explored. Early works introduced template-based procedural layout generation and optimization-based synthesis methods. More recent efforts such as BAG2 [5], ALIGN [6], and MAG-

ICAL [7] represent three major paradigms: (i) synthesis-friendly AMS circuits that are highly digital and modularized, (ii) procedure-based layout generation using reusable templates, and (iii) optimization-based layout synthesis that formulates layout as a constrained optimization problem. While these approaches have demonstrated progress, they remain limited: many target only small analog macros such as comparators and amplifiers, rely heavily on designer-specified knowledge, or lack silicon-proven results. In particular, extending such methods to high-speed PHY design that must seamlessly integrate with large digital systems remains a significant challenge.

In this chapter, we present the design of a combo PHY transmitter in TSMC 28nm that supports multiple standards, including LPDDR4 [8] and UCIe [9], as well as multiple modulation formats, including NRZ, PAM4, and CPAM4. Our contributions are threefold:

- We propose a parameterized template-based layout generator that reduces schematic-to-layout turnaround time from hours to minutes while matching the quality of manual layouts.

- We develop a batch-mode verification suite enabling DRC-clean GDSII generation, parasitic extraction, and post-layout simulation in a unified flow.

- We provide silicon-proven results demonstrating data rates of 4.6 Gb/s/pin (NRZ), 7 Gb/s/pin (PAM4), and 5.25 Gb/s/pin (CPAM4), compliant with both UCIe and LPDDR4 standards.

The rest of the chapter is organized as follows. Section 2.2 introduces the transmitter architecture, template-based layout generator, and validation flow. Section 2.3 and 2.4 describes the device level and block level layout generator in detail. Section 2.5 presents the automatic scripting flow. Section 2.6 presents silicon measurement results, and Section 2.7 concludes the paper.

2.2 Framework: Template-based Generators

The template-based layout generator has the flexibility to place and route as manually designed layout, and the target of this layout generator is to generate layout with circuit parameters, eg. transistor sizes, number of segmentation of the passive devices, and layout parameters, eg. aspect ratio, for the same topology of a design.

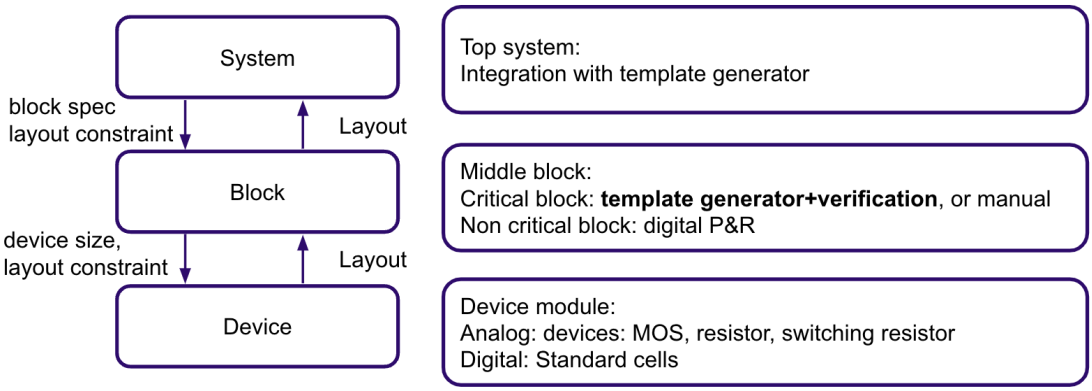
The layout generator follows a traditional top-down bottom-up analog design flow, as shown in figure 2.1 (a). For the system level design, the design architecture is chosen based on the functionality and specifications. The layout parameters, including aspect ratio and pin locations, are constraint by the floorplan to accommodate with other blocks on the same chip.

For the device level design, the transistors and resistors are generated with parametric device generators and wrappers. The device is fully parameterized to compile with any size's devices. The transistor generator connects the sources, drains and gates, as well as creating the FEOL rectangles to bias the body to power or ground, depending on if it is a PMOS or NMOS. The resistor generator generate the resistor array and route the unit resistors in parallel or series, depends on designers requirement.

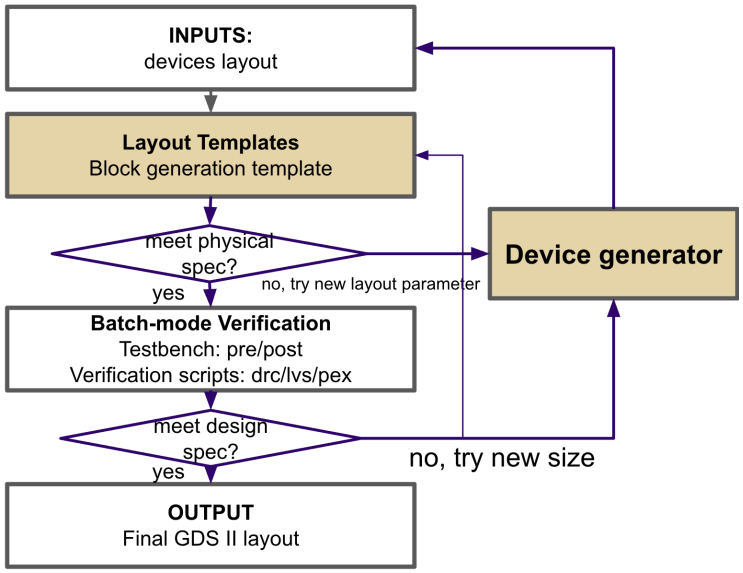
The block generator is designed for each critical analog block with different functionality. The specifications are derived from system level specs and layout template can be used for fast layout generation and verification. Block layout template as a list of parameters for optimization, including layout parameters to define the physical constraints and information of its sub-blocks. Critical net list can be defined to set the routing rules as double width, double pitch rule and digital control signal/non-critical signal should follow minimum width, minimum pitch.

The testbenches for schematic and post-layout simulation is built for each critical block and ocean scripts are provided accordingly to printout human-readable results for simulation and optimization. Once the Makefile is configured, the validation flow can be run as:

```
make pr; make strmout; make drc; make lvs; make pex; make presim; make postSim
```

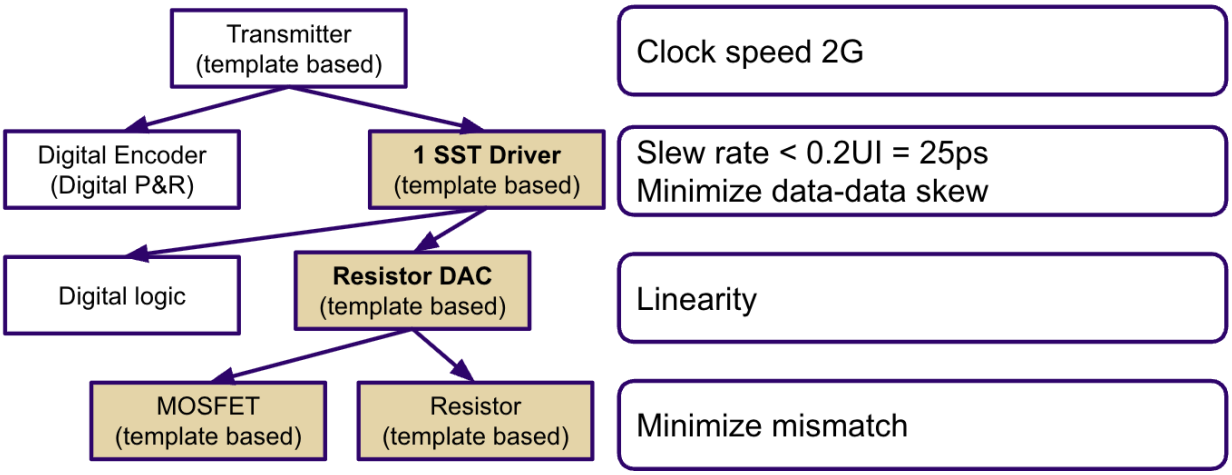


(a)

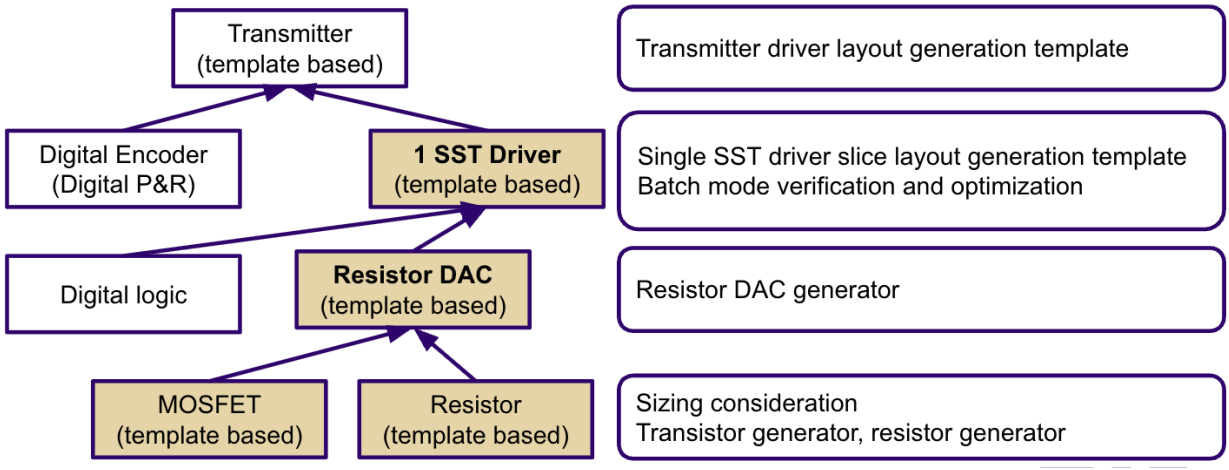


(b)

Figure 2.1: Template Generator Architecture (a) top to bottom (b) layout generator and optimization of block



(a)



(b)

Figure 2.2: Hierarchy as (a) top-down spec (b) bottom-up layout generator

System	Block	Device	signal type	method
TX 4 channel			mixed	template
TX 1 channel			mixed	template
	ESD		analog	custom design
	single driver slice		analog	template
	Encoder		digital	Digital P&R
	HS(high speed single to differential)		analog	custom design
	level shifter		analog	custom design
	driver single leg		analog	template
		MOSFET	analog	template
		Resistor	analog	template

Table 2.1: Hierarchy of TX design and the layout method

In this work, these transceiver blocks serve as a case study for template-driven layout generator. The transmitter layouts, including single devices as in the device level, switching resistor array as in the block level, and TX driver as system level, are systematically mapped into several templates. Table 2.1 shows a summary of the hierarchy and the size of the design. Design parameters, such as the number of legs and termination resistances, introduce layout constraints that must be respected. By applying the template-based flow, functional layouts are generated rapidly while ensuring correct connectivity, device matching, symmetry, and adherence to PVT-aware constraints. This approach demonstrates how complex, high-speed I/O circuits can be efficiently and reliably laid out using a template-driven methodology, without manual placement of individual devices.

name	supported parameter range
MOSFET type	PMOS/NMOS
wfg(width per finger)	0.65 um - 3um
l(length)	0.03-1um
nf(number of finger):	2-16
m(multiplier)	integer

Table 2.2: MOSFET basic parameters

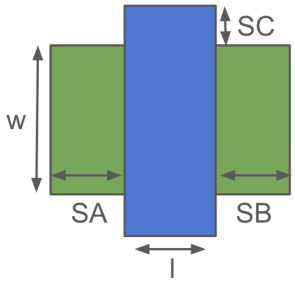


Figure 2.3: MOSFET layout

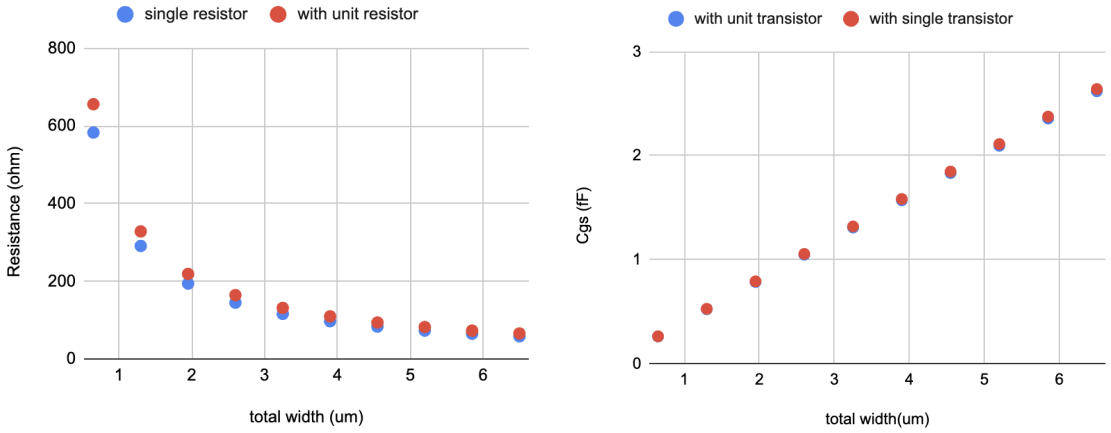
2.3 Device Level Generator

2.3.1 Transistor Generator

Table 2.2 shows the basic parameters of a MOSFET. For a CMOS technology, the device can be PMOS or NMOS and the technology limits the minimum and maximum size of each device can choose. In 28nm technology, the minimum width and length can be 0.1um and 0.03um. Then, the total width can be calculated as

$$w = wfg \times nf \times m$$

The resistance of a MOSFET, particularly the on-resistance (Ron), is an important parameter in determining the performance of the transistor when it is operating in the linear region.



(a) (b)

Figure 2.4: Resistance and gate capacitance of NMOS, with fixed length

The on-resistance is proportional to W/L and can be formulated as

$$Rfet = \frac{1}{\mu C_{\text{ox}} \left(\frac{W}{L}\right) (V_{GS} - V_t)}$$

, where u is the mobility, Cox is intrinsic gate capacitance and is a constant parameter under given technology, VGS is the gate-source voltage and Vt is the threshold voltage. Figure 2.4 (a) show the resistance with NFET with fixed length and the figure can be used as a look-up table to find the size of transistor with given resistance.

The gate capacitance (Cg) is critical in determining its switching speed and power consumption. The gate capacitance is proportional to W and L and figure 2.4 (b) shows the gate-source capacitance. The gate capacitance and can be also formulated as

$$Cgs(linear) : Cgs = Cgd = 1/2Cox * W * L + Cox$$

$$Cg(saturation) : Cgd = Cov, Cgs = 2/3 * Cox * W * L + Cov$$

Hence, the lookup table and the functions can be used to find the dimensions and estimate the gate capacitance for optimization with given Ron resistance.

name	supported parameter range
width	0.5 - 2um
length	0.65 - 2um
number of segment	integer
connection method	parallel/series

Table 2.3: Resistor basic parameters

Algorithm 1 Transistor Sizing and Floorplan Procedure

1: **Step 1: Transistor Sizing Based on** R_{on} Set $w = w_{min}$ (maximize routing width). Set $l = l_{min}$ (minimize delay and power) . Set $m = 4$, $R = 164\ \Omega$

2: **Step 2: Width Scaling for Exact** R_{on} Adjust width

3: (Optional) Scale width and length together to reduce mismatch

4: **Step 3: Area Estimation** Compute transistor area:

$$A = (w + SA + SB) \times (l + 2 \cdot SC) \times m$$

5: **Step 4: Floorplan Dimensions** Calculate floorplan width and height based on aspect ratio

6: **Step 5: Connectivity** Connect all transistor terminals (source, drain, gate, bulk)

2.3.2 Resistor Generator

The resistor layout generator enforces layout constraints to ensure accurate electrical characteristics, including precise resistance values. It also incorporates mismatch reduction techniques, such as scaling and uniform placement, to minimize variations between unit resistors. These constraints help maintain both performance consistency and manufacturability across the generated layout.

The poly resistors are widely used in integrated circuits, due to its precision, temper-

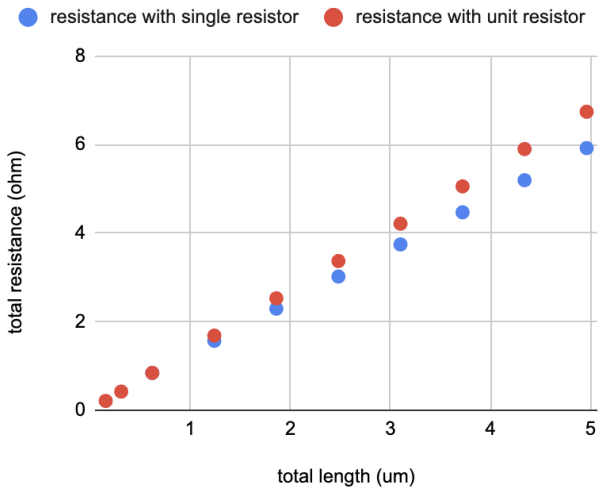


Figure 2.5: Resistor resistance and total length with fixed width

ature stability, integration efficiency, and process compatibility. Table 2.3 shows the basic parameters and range of its values for poly resistor in 28nm. The resistance R of a resistor is determined by its physical dimensions (width W and length L) and the material properties (resistivity ρ and thickness t) and figure 2.5 shows the total resistance and the total length of resistor with fixed minimum width. For a resistor made from a uniform material, the resistance can be expressed by the following equation

$$R = \rho\frac{L}{A} = \rho\frac{L}{Wt} = R_s\frac{L}{W}$$

The resistors in this template are implemented using unit resistors to minimize random error, since the variation in R_i is proportional to $1/\sqrt{n}$ [14]. Research has shown that for large resistance values, resistors are typically decomposed into smaller unit segments, which are then arranged in parallel and connected in series to improve accuracy—from about 70% to as high as 95%. In addition, special care is taken in routing: because corners contribute significant resistance, the layout employs maximum metal width with ample vias and utilizes both M1 and M2 layers to reduce parasitic effects.

Resistor mismatch is often quantified using standard deviation (σ) and it is determined

by the area proportionality constant A_R (process-dependent) and width and length .The mismatch can be expressed as:

$$\frac{\Delta R}{R} = \frac{A_R}{\sqrt{WL}}$$

For large values, the resistors are usually decomposed into smaller units that are mapped out in parallel and connected in series, which could improve the accuracy from 70% to 95%. Besides basic resistor parameters, the layout quality, for example aspect ratio, also affect the resistance accuracy The corners contribute significant resistance, so the routing is using with maximum width and via and using both M1 and M2 to reduce the parasitic resistance on the wire.

Thus, the dimensions can be found with the lookup table and the equations, setting the width as minimum and scale the width and length with the requirement of mismatch.

The switch and resistor dimension will affect the parasitic, area, mismatching, thus affect the speed and data reliability, so it is necessary to take all the circuit parameters in to account during layout. Given the total resistance of one leg and layout aspect ratio as a:b(width:length), assuming total resistance is 4k Ω, the design consideration is as in algorithm 3:

2.4 Block Level Generator

2.4.1 Switch Resistor Leg

In the single driver slice, the 240 Ω calibration resistor is generated by turning on or off the switching resistor array, as shown in figure 2.6. In a 5 binary-weighted resistor DAC, the total resistance per leg, including a resistor and a transistor as a switch, should be R, 2R, 4R, 8R, 16R. Single driver slice has 5 pull up legs, 1 fixed pull-up leg, 5 pull down legs and 1 fixed pull down leg. In real designs, the on-resistance of the switch is chosen as 1/20 of the total leg resistance. Each of the switch and the resistor attached are divided as a switch leg.

Algorithm 2 Resistor Sizing and Floorplan Procedure

1: **Step 1: Initial Resistor Estimation** Find the closest total length/number of unit resistors based on resistance.

2: **Step 2: Resistance Scaling** Scale resistor length to reach exact total resistance. Example: length +18.6%

3: (Optional) Scale both resistor length and width to improve mismatch tolerance.

4: **Step 3: Area Estimation** Compute area based on unit resistor size and number:

$$A = (w_{res} + w_{dummy}) \times (l_{res} + 2 \cdot l_{pin}) \times m$$

5: **Step 4: Floorplan Dimensions** Calculate width and height of floorplan:

$$w_{fp} = \frac{1}{\sqrt{\frac{aA}{b}}}, \quad h_{fp} = \frac{1}{\sqrt{\frac{bA}{a}}}$$

6: **Step 5: Array Organization** Determine array organization:

$$column = \frac{w_{fp}}{L}, \quad row = \frac{m}{column}$$

7: **Step 6: Routing** Route resistors in S-pattern if connected in series. Route resistors in 11-pattern if connected in parallel

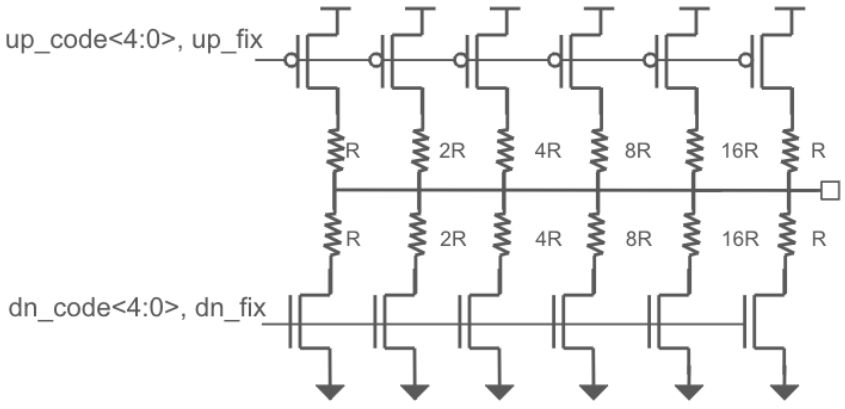


Figure 2.6: Pull up and pull down

Algorithm 3 Switching Resistor Leg Layout Procedure

1: **Input:** Target resistance R_{target}, minimum length L_{min}, unit resistor width W, unit transistor size (w, l), safety margins SA, SB, SC, SR, constants a, b

2: **Step 1: Resistor Estimation** Assume unit resistor with W_{min}, L_{min}, and $m = 4$. From Fig. 2.5, find the closest number of unit resistors m_{res} based on R_{target}. Compute R_{leg}

3: **Step 2: Transistor Estimation** From Fig. 2.4, find number of unit transistors m_{mos} so that

$$R_{mos,total} = \frac{1}{20} \times R_{res,total}, \quad R_{mos} = 164\ \Omega$$

4: **Step 3: Resistance Scaling** Scale resistor length to match R_{target}:

$$L = L_{min} \times \left(1 + \frac{\Delta R}{R_{res}}\right)$$

.

5: **Step 4: Mismatch Reduction** Scale both resistor width W and multiplier m by same integer factor. Example: $W \to 2W$, $m :\to 2m$

6: **Step 5: Area Estimation** Estimate area A:

$$A = (w + SA + SB)(l + SC)m_{mos} + (W + SR)(L + SR)m_{res}$$

7: **Step 6: Floorplan Dimensions** Compute width and height:

$$w_{fp} = \frac{1}{\sqrt{\frac{aA}{b}}}, \quad h_{fp} = \frac{1}{\sqrt{\frac{bA}{a}}}$$

8: **Step 7: Array Organization** Find transistor array column and row:

$$column = \frac{w_{fp}}{length}, \quad row = \frac{m_{mos}}{column}$$

9: **Step 8: Routing** Route resistors in S-pattern and connect all transistor source/drain terminals

10: **Output:** Resistor and transistor sizing, floorplan dimensions, array configuration

# of unit resistor	8 in series	2 in series	2 in parallel
to unit R ratio (target)	16	4	1
area	16.7328	6.6168	17.6592
aspect ratio	0.4957	0.3134	0.4535
resistance (pre)	7.77 kΩ	1.944 kΩ	0.486 k
to unit R ratio (pre)	15.99	4.00	1.00
resistance (post)	8.528 kΩ	2.134 kΩ	0.486 k
to unit R ratio (post)	15.89	3.98	1.00

Table 2.4: Experiential result with 16R, 4R, and 1R switching resistor

The same switch resistor template can be applied to both pull up and pull down system by changing the type of the switches. Table 2.4 shows the experimental result of target resistance of 16R, 4R, and 1R. Post simulation shows the resistance to unit ratio is 15.9, 3.98, which meets the design requirement. The post-layout simulation shows the pull up and pull down resistor DAC has a resolution of 2.3 Ω and 4.7 Ω in TT corner.

2.4.2 Single SST Driver

Each driver consists of 12 switch resistor legs along with 12 bits of logic for pull-up and pull-down control. The layout is constrained by a maximum floorplan height of 25 µm, and the design ensures that delays along all data paths are balanced to minimize signal skew. Additionally, the slew rate is limited to less than 0.2 UI to maintain signal integrity.

Table 2.5 shows the parameters in the single driver templates, including both circuit parameters and layout parameters. Algorithm 27 shows the template step by step. Each logic gate along the data path can be tuned. The physical information of each switch-resistor is set as layout parameters in single driver, so that the driver layout can be updated with any modification of the legs. The location of each major block is set to initialize the floorplan

design parameter			layout parameter		
name	number	range	name	number	paramter
pull up nor driving strength	6	1-3	leg physical size	12	width, height
pull up inverter	6	1-9	logic location	1	x, y
pull down nand	6	1-3	leg location	1	x,y
pull down inverter	6	1-9	critical nets	1 list	up code, dn code

Table 2.5: Parameters of single driver

and the routing to between the major blocks will be calculated automatically based on the floorplan.

SST driver is built with one pull up resistor DAC, one pull down resistor DAC and logic to encode the data and calibration bit to control code of the resistor DAC, to match the 240Ω resistance. The transistors and resistors used the minimum size to reduce parasitic and delay time. Then, the speed is limited by the data slew, data skew and delay of the logic paths. In this design, the data slew rate is limited to 0.2UI, which is 25ps and the data skew should be minimized. Thus, it is critical to size the logic gates and on the data line.

Figure 2.7 shows the logic path of 1-bit data to the calibration bit on resistor DAC. The ZQ calibration bit and data is encoded with a inverter, a tri-state buffer, a NAND/NOR gate and an inverter, and the encoded data directly connects with the switch of the switch resistor. However, as discussed in the previous section, transistor gate capacitance depends on its width and length and different resistor leg uses various sizes, which introduce different load capacitance to the data lane, causing mismatch on the path delay and introducing data skew.

The single driver slice is simulated and optimized for 20+ iterations. Figure 2.8 shows the optimized data skew between each bits. The post-layout simulation shows the skew is improved by 50% by logic sizing and floorplan optimization.

Algorithm 4 Resistor DAC and Control Logic Generation Flow

Set pull-up leg coordinates (x_leg_up, y_leg_up)
for each leg on the pull-up side **do**
    Call `createSingleLeg` with circuit parameters
    Place leg at (x_leg_up, y_leg_up)
    Increment x_leg_up by the width of a leg
end for
Set pull-down leg coordinates (x_leg_dn, y_leg_dn)
for each leg on the pull-down side **do**
    Call `createSingleLeg` with circuit parameters
    Place leg at (x_leg_dn, y_leg_dn)
    Increment x_leg_dn by the width of a leg
end for
Set logic coordinates (x_logic, y_logic)
for each control logic of the leg **do**
    Create control logic instance
    Place at (x_logic, y_logic)
    Increment y_logic by the height of the control logic
end for
Define routing corridor between legs and logic as $((x_1, y_1), (x_2, y_1), (x_2, y_2), (x_3, y_2), (x_3, y_3), (x_1, y_3))$
for each routing between the leg and logic **do**
    Determine starting and ending coordinates of each wire
    Create metal routing on horizontal M2 and vertical M3
    Create vias based on via rules
end for
Create FEOL layers for all devices in the same area
Route power and ground
Create pins and labels

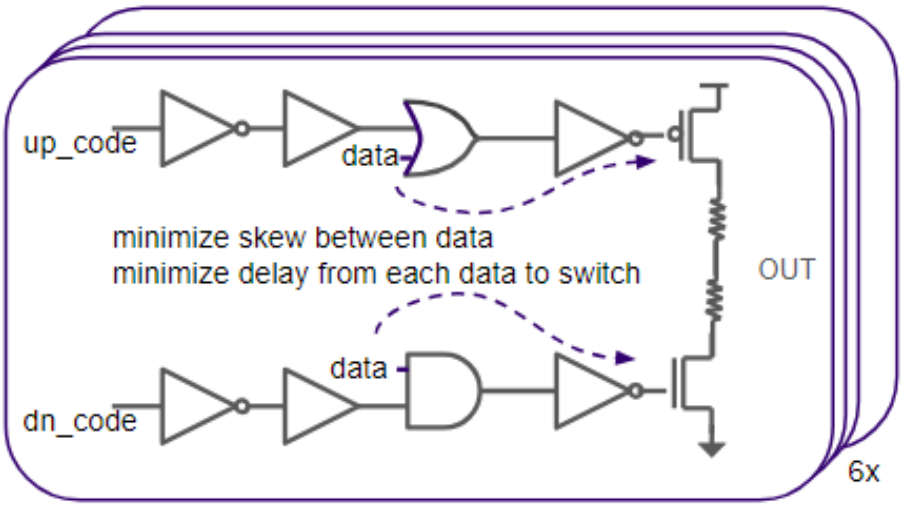


Figure 2.7: The data path of logic control

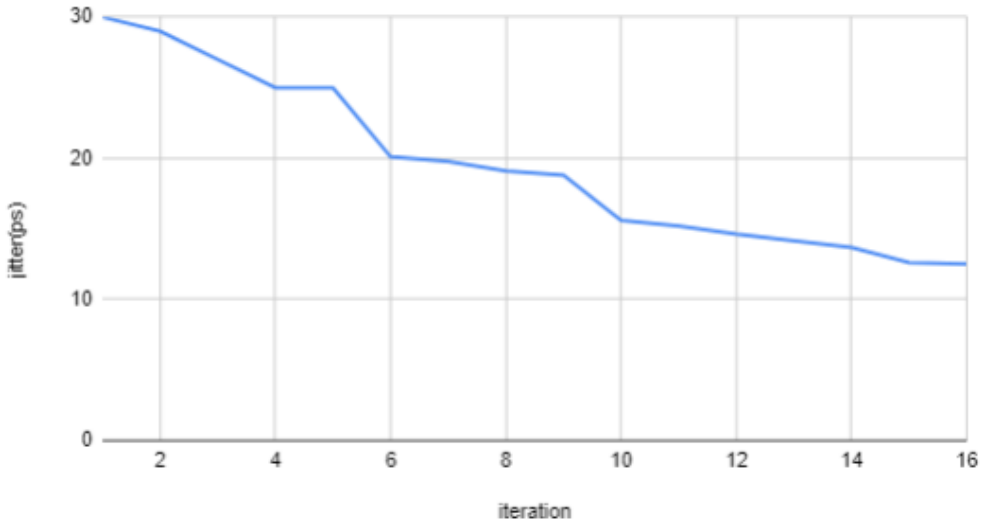


Figure 2.8: Optimized skew from 30ps to 12ps under 16 iterations

2.5 Automatic Script

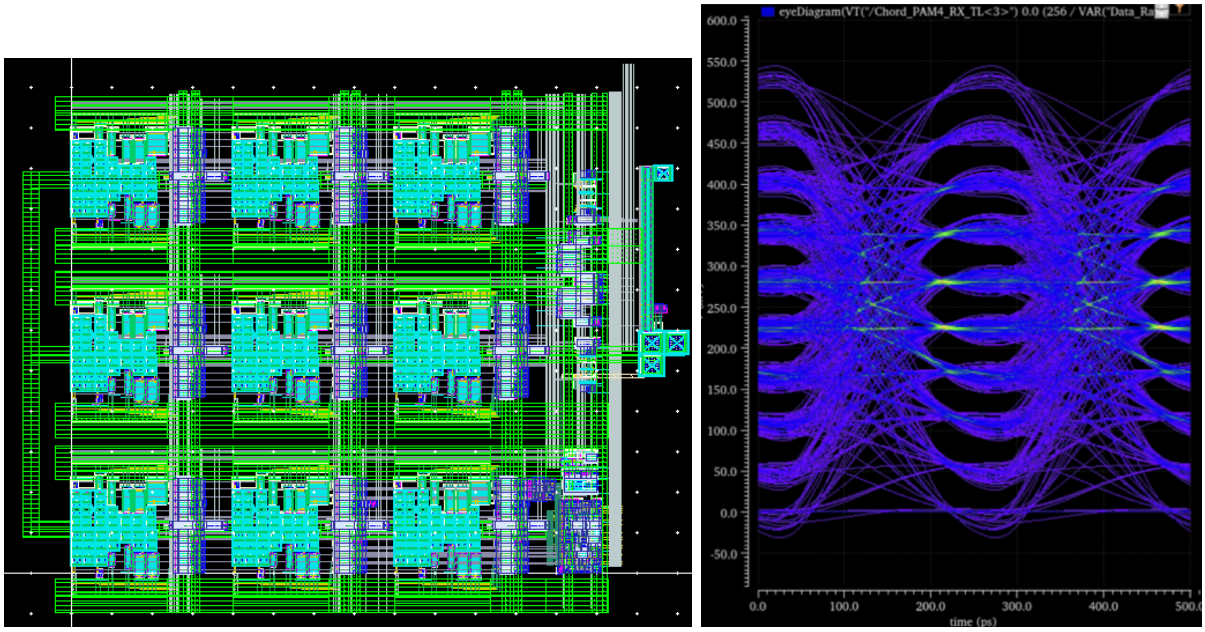
The verification is using scripting and the configurations is saved in a Makefile. The designer can define the library name, design name and design parameters included in the templates. All the validation flow refers to the definition in this file. The batch mode validation step is organized as follows:

```
make pr         # Run template with design parameters and generate a layout
                # in Virtuoso
make strmout    # Stream out the GDS II layout from Virtuoso and
                # merge the layout of the subblocks
make viewgds    # View the layout
make drc        # Run design rule check
make lvs        # Run layout vs schematic check
make pex        # Parasitic extraction
make presim     # Pre-layout simulation with testbench Spectre netlist
                # and Ocean script
make postSim    # Post-layout simulation with parasitic extracted from
                # "make pex"
```

2.6 Experimental Results

The top transmitter integrates the 9 driver slices with digital encoder and the layout is shown in figure 2.9 (a) with size 83um x 72um. The layout parameters, including all the sub-blocks physical sizes and their desired locations are set as layout parameters. Critical data list are set to follow the routing rules. Pin location of the sub-blocks are read as input for top-level routing.

To generate the transmitter that meet the designer's spec, the initial schematic is provided by the designer and the parameterized templates are initially implemented bottom up, for all the devices, blocks and the whole system. Then, the verification flow are created and



(a) (b)

Figure 2.9: Transmitter (a) layout (b) simulated PAM-10 eye diagram

performed for the critical blocks, including SST driver. After the first iteration of the layout implementation and post layout verification, the designer can tune the parameterized device sizes or re-synthesis the pure digital logic according to the simulation results to optimized the design to meet the spec. With the parameterized template, the iteration time from schematic to layout is shortened from days to hours. Figure 2.10 shows the silicon proven results, showing the robustness of the template-based generator.

2.7 Summary

This chapter introduces a template-driven flow for the automatic generation and optimization of reconfigurable UCIe/LPDDR interface layouts. The method leverages templates with design and layout parameters, enabling fast exploration of transistor sizing and floorplan variations. By reusing these templates, layouts can be generated and optimized automatically

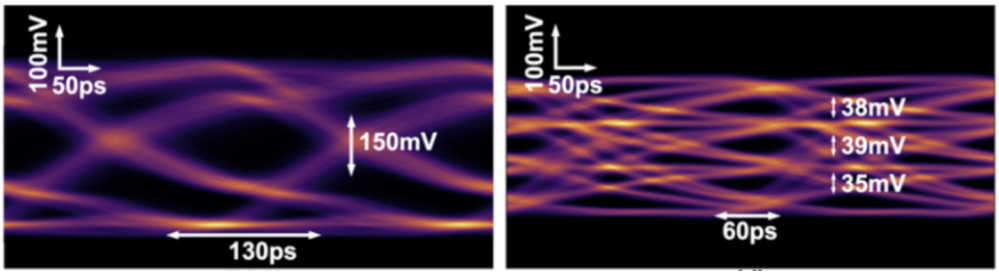


Figure 2.10: Silicon measurements: left NRZ mode data = 4.6Gb/s/pin, right PAM4 mode data = 7Gb/s/pin

with quality comparable to manual designs. The approach reduces the schematic-to-layout iteration time from hours to minutes, highlighting the efficiency gains of template-based layout automation.

Chapter 3

LEGO: ANALOG CELL LIBRARY AND CONSTRAINT DRIVEN LAYOUT GENERATOR FOR FINFET

This chapter presents LEGO cells, digital standard cell like parameterized cells for analog and mixed-signal active, passive, and physical-only devices, and a Python-based LEGO cell generator. Seamlessly integrating with digital standard cells and industry-standard digital place-and-route (P&R) tools, LEGO cells enable automated analog and mixed-signal design and layout. LEGO cells have been implemented in a 12nm FinFET process, and tested on a set of analog circuits. Experimental results have demonstrated that the proposed LEGO cell approach can reduce layout time from days to minutes while maintaining the layout quality as shown by post-layout simulation.

3.1 Introduction

3.1.1 FinFET Process Technology Overview

This work is implemented and validated in the GlobalFoundries 12 nm FinFET technology. Compared with planar CMOS, FinFET devices offer improved electrostatic control, reduced leakage current, and higher drive current at comparable dimensions, enabling superior energy efficiency and performance in advanced nodes.

A key distinction from planar MOSFETs is that FinFET dimensions are quantized by the number of fins, typically ranging from 2 to 10 fins for practical designs. This discrete sizing model significantly reduces the range of available transistor widths compared to planar devices, where continuous width scaling from approximately 100 nm to over 2 µm is common. While this quantization simplifies some aspects of design-rule compliance, it also limits fine-grained sizing flexibility and necessitates careful device matching and biasing strategies.

The technology further supports multiple threshold-voltage options and a range of passive devices (resistors, capacitors, diodes), but imposes stringent layout rules related to fin quantization, multi-patterning, well spacing, and device orientation. These constraints must be carefully addressed in any analog or mixed-signal layout automation flow.

3.1.2 Case Study Context

Two representative mixed-signal interface designs are used in this work to demonstrate the proposed layout methodology:

Advanced Interface Bus (AIB): A high-bandwidth, low-power parallel interface developed for chiplet interconnects, which will be discussed in Chapter 4.

Electronic Integrated Circuit (EIC): A multi-voltage, mixed-signal block, which will be discussed in Chapter 5.

These circuits are chosen because they differ in operating voltage, device types, and layout complexity, offering a broad spectrum of constraints for testing the proposed methodology. The detailed circuit design is outside the scope of this thesis; they serve solely as case studies for evaluating the automation flow.

3.1.3 Graph Methodology Background

Graph is made up of nodes that are connected with edges. Degree is the number of edges that is connected with a node. For example, the graph below has 3 nodes and 2 edges, where the node B has degree of 2 and both node A and node C have degree of 1. Then, for each nodes, suppose the degree sequence for a node is [degree1, degree2, degree3],]the signature S is defined as S = (π + degree1)(π + degree2)(π + degree3), where π can be any transcendental number(e, π, etc..) In this example, suppose π is 1 for simple illustration, then S(A) = (1+2), S(B) = (1+1)*(1+1) = 4 and S(C) = (1+2)

To find equivalence of two nodes, simply compare signatures of nodes.

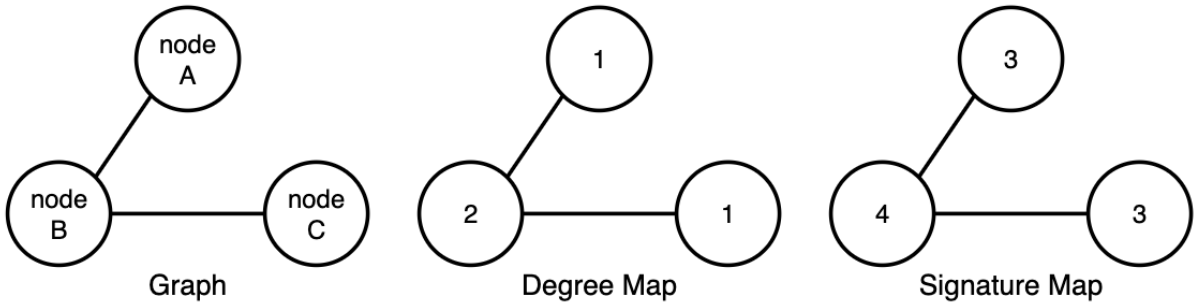


Figure 3.1: Example of signature map

3.1.4 Summary

The chapter is as follows: Section 3.2 describes the LEGO library device generation, including the design choice of FinFETs and resistors. Section 3.3 discuss the framework from design to layout, including process migration, schematic-to-Verilog mapper, constraint retriever and analog layout script generator. Section 3.4 shows preliminary experiments result, starting with a simple differential pair and moving to more crucial and complex designs, eg. comparator, CTLE and a digital delay line. Section 3.5 concludes the chapter.

3.2 Device Generation

LEGO is a cell library with parameterized active devices, physical only devices and passive devices. The active devices include parameterized PFETs and NFETs with various sizes. The physical only devices include boundary cell and decap cells with various gate sizes. And passive devices consist of capacitors and resistors with various widths and lengths. Additional LEGO standard cells include logic cells not found in the digital standard cell library, or digital standard cells with device sizes not included in the provided library. The category is shown as in the table 3.1. Each category features a variety of parameters and ranges.

Table 3.1: The category of LEGO cells.

Cell Type	Description
Active Cells and Physical-only Cells	
pFET/nFET w/ min. gate length	nfin = 2, 3, 4, nf = 1, 2, 4 vt: lvt/rvt/slvt/hvt
pFET/nFET w/ non-min. gate length	nfin = 4, nf = 1, l = L1/L2/L3/L4 vt = lvt/rvt/slvt/hvt/eg/egv
Boundary Cells	Type: left, right
Tap Cells	Tyep: p-type/n-type
Diode	minimum area
Passive Devices	
Capacitor	Type: MIM/MOM, Capacitance: 8f F - 2p F
Resistor	Type: RMRES/NWRES, Resistance: 270 Ω - 15k Ω

3.2.1 Transistor: Active FinFET

To support various custom analog designs, an analog library needs to cover transistors of all width and length. In 28nm technology, the width ranges from 100nm to 10um and the length ranges from 30nm to 1um. Also, the devices should support various threshold voltage, eg. slvt, lvt, rvt, hvt. Since both width and length could vary continuously within the range, millions of cells need to be included in one analog library. However, FinFET technology restricts the total gate finger width by the number of fins (nfin), fixed fin pitch (Pfin), and number of fingers (nf), which greatly decreased the number of cells need to be generated:

$$W = n_{fin} \times P_{fin} \times n_f. \tag{3.1}$$

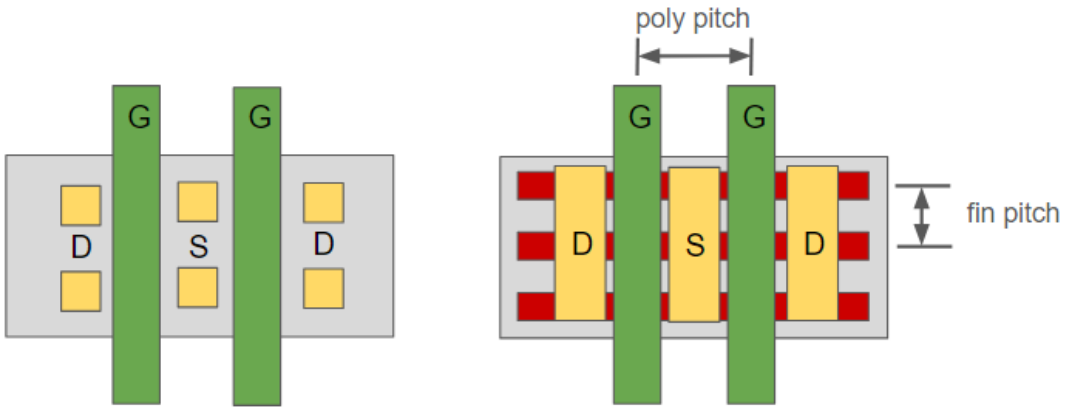


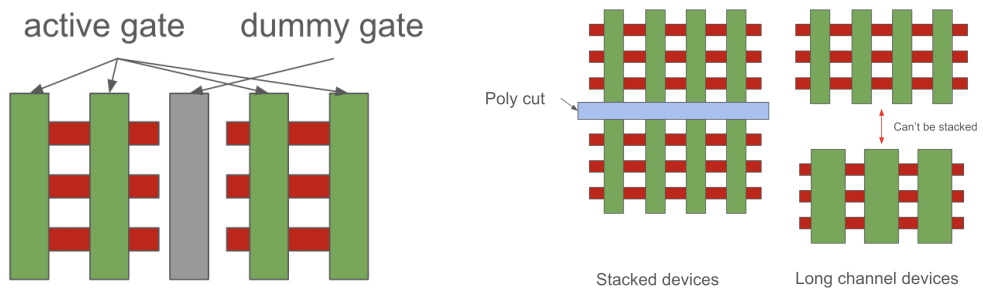
Figure 3.2: Simplified layout of a bulk device and a FinFET device

The structure of the FinFET device is composed of a horizontal fin (red), vertical poly (green), and source-drain diffusion (gray), as depicted in Figure 3.2.

The number of fins must have a minimum value of 2 and must be integers, with the fin pitch fixed according to the technology being utilized. In addition to a fixed fin pitch, the device also features a fixed poly pitch, typically constructed with a minimum length. Transistors with greater lengths will have larger poly pitches, but they are located in separate areas.

To enhance layout density, transistors can be stacked using cut layers. Figure 3.3 (a) illustrates an example of a single diffusion break in between two devices and figure 3.3 (b) demonstrates how transistors can be stacked with a horizontal poly cut to prevent gate shorting. However, transistors with different lengths cannot be stacked even with a poly cut and must be placed apart from each other.

To ensure compatibility with standard digital and industrial CAD tools, device layout is generated based on the design grid and following standard cell design principles. As depicted in Figure 3.4, each device is surrounded by metal 1 (light blue) power rail(VDD) on top, ground rail(VSS) on the bottom, and dummy gates on the side. The poly cut layer is hidden under the power/ground rail. Besides power/ground rail, metal 1 pins are also positioned



(a) (b)

Figure 3.3: FinFET design rules (a) Dummy gates (b) Short channel devices and Long channel devices placement

vertically for source and drain terminals, and a horizontal M1 is used to connect all drain terminals . The cell width must be a multiple of minimum poly pitch, and all cells share the same height, with exceptions of some complex cells be designed with double height.

Given the standard cell layout constraints, the LEGO library offers FinFET devices with nfin ranges from 2 to 5 and nf ranges from 1 to 2. The limitation of the nfin range is determined by the fixed standard cell height. However, due to the discrete nature of total width in FinFET technology, transistors with any width size can be segmented and mapped to the LEGO cells.

While standard digital cells utilize transistors with minimum length, analog circuits require the use of long-channel devices. In order to place the cells on grid, we constrained the gate length by the minimum poly pitch, and the supported gate length is carefully selected as:

$$Gate\_length \times nf = min\_cpp \times N - 2 \times offset \tag{3.2}$$

$$min\_length < Gate\_length < max\_length, \tag{3.3}$$

where min_cpp is poly pitch of minimum gate length N, must be an integer, offset is the minimum area kept to connect source/drain Min_length and max_length are the minimum

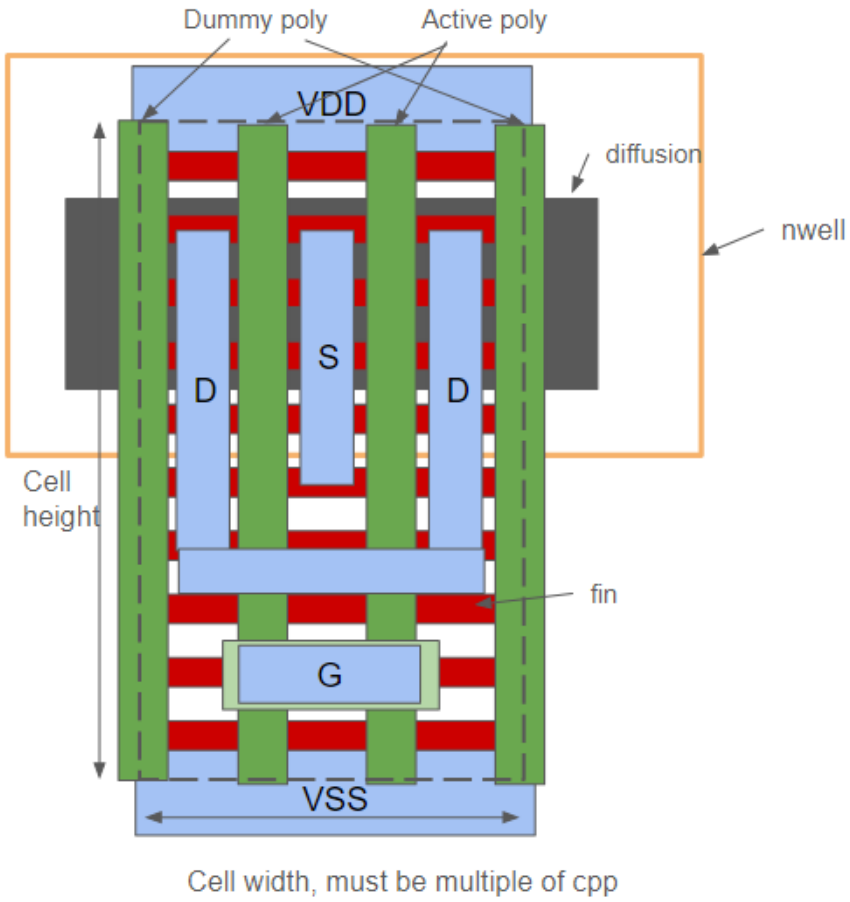


Figure 3.4: PFET transistor as a LEGO cell. The layout follows all standard cell layout rules

Area overhead(%)	l1	l 2	l3	l4
nfin = 2	18.7	8.17	8.92	7.73
nfin = 3	12.15	4.93	5.65	4.50
nfin = 4	3.00	5.16	5.88	4.73

Table 3.2: Average layout area overhead of LEGO cell to Pcell in PDK

and maximum gate length under target technologies

With all the constraints above, we support 4 levels of the gate length, l_min, l1, l2, l3 in 12nm technology. Long-channel devices require non-minimum length dummy gates, which are implemented using LEGO boundary cells. Additionally, the source/drain pins are positioned on the cell boundary to facilitate device merging and increase area utilization.

Transistor with large width can be decomposed to multiple transistors with smaller width, as long as the nfin $\times$ nf $\times$ m stays the same. In traditional analog layout, designers also tend to divide larger transistor to smaller devices connected in parallel and arrange in different pattern to minimize systematic mismatch [17].

To evaluate layout density, we compare the areas of devices using LEGO cells and p-cells provided with the 12nm PDK. The layout of the FinFET device need to consider not only the active area, including fin, poly and diffusion, but also metal for source/drain routing, gate connection, dummy gates, diffusion breaks, and poly cuts between the active areas.

Then, table 3.2 compares the overhead of various nfins. LEGO cells and a Pcell are compared with the same size, eg. nfin = 3, l = minimum length. Both of the devices has poly cuts on the top and bottom, dummy gates on left and right, drain routing connected on metal 1 and horizontal gate routing. The area is calculate by multiplying the pitch between two vertical dummy gates and the pitch between two horizontal poly cuts, and we noticed the overhead is reduced with larger size devices, due to device merging.

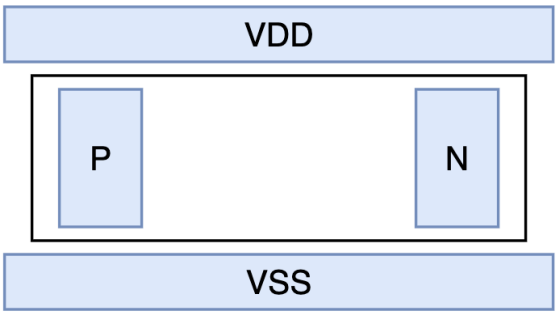


Figure 3.5: Layout of resistors with VDD/VSS

3.2.2 Resistor and Capacitor

Passive devices, including capacitors and resistors, are crucial components in analog design. The cells for passive devices are designed according to the needs of circuit designers. The height of passive devices is limited by the standard cell height. Figure 3.5 illustrates the layout of resistor, where each resistor's height is limited to one standard cell height, and the resistance depends on the width of each resistor. Similar to active device cells, the resistor has power on top and ground on the bottom of the cell.

3.3 LEGO Framework

The LEGO layout generator flow consists of several key modules, including process migration, schematic-to-Verilog mapper, constraint finder and analog layout script generator. Figure **??** illustrates the overall framework of the flow, which integrates these modules to enable automated AMS layout generation. Each module plays a specific role in the process, and this section will discuss their functionality and implementation in detail.

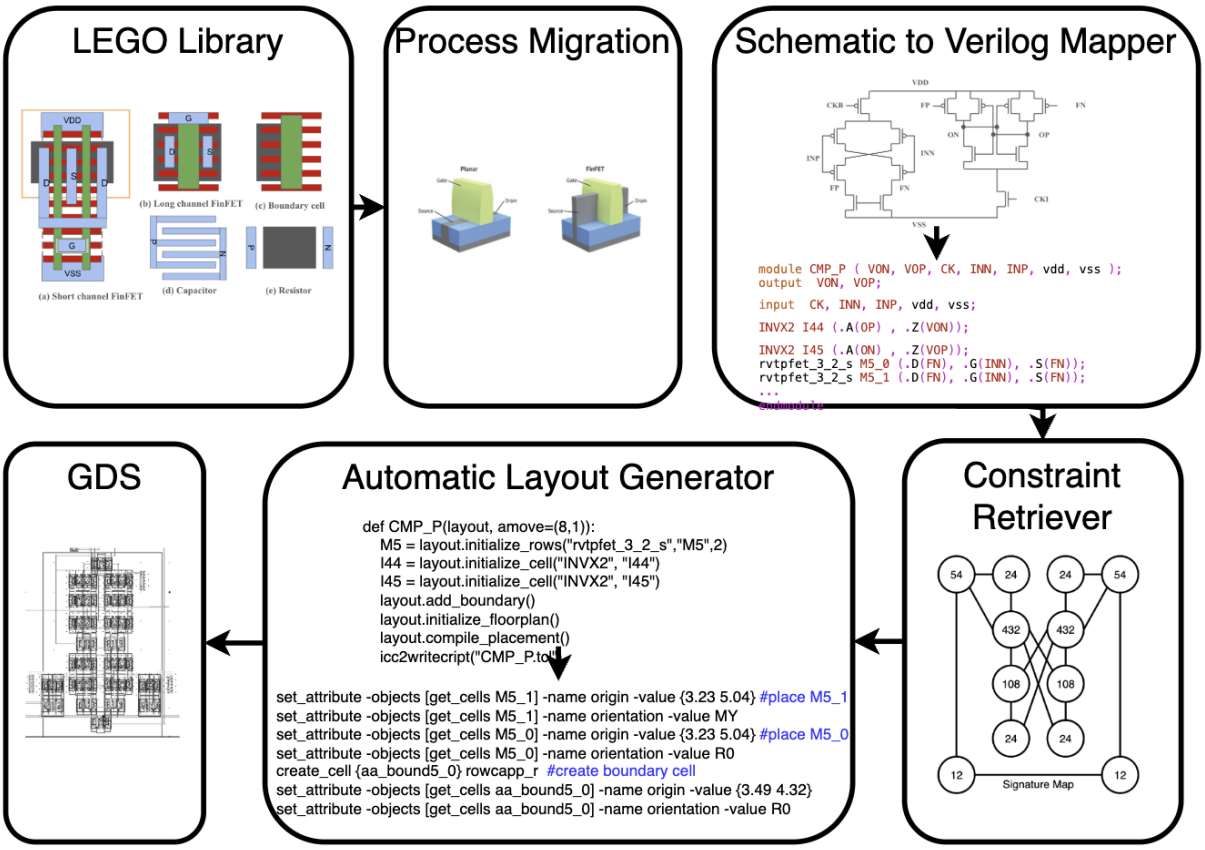


Figure 3.6: The LEGO Framework

3.3.1 Process Migration

Active device sizing

To transfer from 28nm to 12nm technology, we first find the process and technology parameters associated with the MOS models. Assuming all the devices are operating in saturation region, the current is as follows:

$$I_{DS} = \frac{1}{2} * \mu_n C_{ox} * \frac{W}{L}(V_{GS} - V_T)^2 * (1 + V_{DS}) \tag{3.4}$$

To migrate the process keeping the same performance, we estimated a constant gm to size the device in target technology. We target to achieve the same drain to source current with the same V_GS and V_DS. The choice of gm/ID is based on its strong relation to the performance of analog circuits [13].

The gm/ID relation can be generated in two ways, either using MOS transistor model to calculate the continuous representation of the transistor current, or from measurement of a transistor. With the limited selection of LEGO cell physical sizes, we choose to use the measure the FinFETs with 4 different gate lengths. The saturation current is proportional to W/L as equation 3.5 and we define a scaling factor, S, as the constant parameter from one technology to another technology. Then we run the simulation with device length as L1, L2 and L3

$$I_{DS\_T28} \propto \frac{W_{T28}}{L_{T28}} \propto \frac{S * W'_{GF12}}{L_{GF12}} \propto I_{DS\_GF12} \tag{3.5}$$

For each gate length, we simulated the saturation current Ids under strong reversion (VGS = VDD) and VDS = VGS-VTH and VDS = VDD. Table 3.3 shows the Ids simulation with selected gate length. Notice that the W/L ratio under the simulation are kept the same between two technologies, where W_T28/L_T28 = W_GF12'/L_GF12. Then we can estimate the scaling factor,S, under each length level, by dividing the saturation currents.

$$S \approx \frac{I\_ds\_T28}{I\_ds\_GF12} \tag{3.6}$$

length	VDS(GF12)		VDS(T28)		Scaling Factor
	0.35V	0.6V	0.35V	0.6V	
l_{min}	65u	88u	69u	75u	1
$l1$	52u	62u	19u	23u	1/3
$l2$	29u	33u	2.5u	2.6u	1/10
$l3$	29u	33u	1.84u	1.86u	1/18

Table 3.3: I_{DS} simulation result

With the scaling factor, the physical sizes of the devices can be calculated. The length of the devices are selected first. The minimum length in 28nm is scaled to the minimum length in 12nm and the non-minimum length are scaled down accordingly and estimated to the nearest L1, L2 or L3. The width is selected as equation 3.7 by multiplying the scaling factor with the initial width in target technology.

$$W_{GF12} = S \times W'_{GF12} = S \times \frac{W_{T28} \times L_{GF12}}{L_{T28}} \tag{3.7}$$

Passive device

The passive devices are mapped to the corresponding devices of the target library with the same electrical sizes. The resistor is mapped to rmres and capacitor are mapped to alternative polarity MOM capacitor(apmom), for area efficiency in 12nm FinFET technology. To find the physical sizes, we estimate the relation between electrical size and the physical size by the equation 3.8

$$\text{Resistance} = A \times \frac{\text{physical\_length}}{\text{physical\_width}} + B \tag{3.8}$$

$$\text{Capacitance} = A \times \text{physical\_width} \times \text{physical\_length} + B \tag{3.9}$$

, where the capacitor has fixed finger width, finger space and metal layer usage. Simulation under target technology is performed to find the constant vector A and B, and used to estimate the physical length and width for each passive devices.

We limit the physical width to be as close as multiples of standard cell height to maximize the area utilization and keeps the default aspect ratio, width/length, as 1.

Digital standard cell

The digital standard cells are mapped to the cells with same functionality and same driving strength in the target library, for example, the smallest inverter of 28nm is mapped to the smallest inverter in 12nm, the largest inverter of 28nm is mapped to the largest inverter in 12nm, and the inverters in between will be mapped in scale, respectively. For the cells that are not included in the target library, either the logic is not included or the logic gate with required devices' size are not included, LEGO standard cells will be designed following the standard cell design rule and included in the LEGO library.

3.3.2 Schematic-to-Verilog Mapper

The schematic-to-Verilog mapper reads the un-annotated SPICE netlist and maps the schematic to a Verilog netlist that can be used by industrial digital P&R tools. The mapper identifies devices by parsing the SPICE netlist, and assigns corresponding LEGO cells based on key parameters, including nfin, nf, and vt type. Passive components such as resistors are matched to the nearest available LEGO cell by quantizing their values to the supported existing passive LEGO cells with closest electrical and physical values.

3.3.3 Constraint Finder

Graph theory in analog circuit

Each device in an analog circuit is labeled as a node and nets that connect devices are labeled as edges. For example, in the figure below, PM0 is connected with PM1, NM10 and

NM4, so degree(PM0) = 3. Then signature of PM0 is calculated with S(PM0) = (1+degree(PM1))*(1+degree(NM4))*(1+degree(NM10)) =(1+2)*(1+5)*(1+2) = 54 After completing the signature map, find equivalent nodes by comparing signatures of nodes. In this case, PM0 and PM3 can be characterized as equivalent, since they have the same signature 54.

Placement: Current-flow

Current flow constraint is implemented with monotonic routing from signal paths from power to ground, which was proved to reduce both the interconnect wire length and routing induced parasitics, improving the post layout circuit performance. In this work, current flow constraint is used to limit the topological relations between devices. In this case, first find current path from VDD to VSS to construct current-flow constraints from figure 3.7. All current-flow constraints should be monotonic from power to ground. Placement constraint relates to y coordinates of each cell can be represented and one of the layout representations that satisfy such constraint is shown in (c).

In this case, the constraint is formulated as below:

Subject to :

$$Minimize : area = max(y) * max(x)$$

$$y_{vdd} > y_{PM0}, y_{PM0} > y_{NM10}, y_{NM10} > y_{vss}$$

$$y_{vdd} > y_{PM1}, y_{PM1} > y_{NM4}, y_{NM4} >= y_{NM9}, y_{NM9} > y_{vss}$$

$$y_{vdd} > y_{NM6}, y_{NM6} > y_{NM8}, y_{NM8} > y_{vss}$$

$$y_{vdd} > y_{PM2}, y_{PM2} > y_{NM5}, y_{NM5} >= y_{NM8}, y_{NM8} > y_{vss}$$

$$y_{vdd} > y_{NM7}, y_{NM7} > y_{NM9}, y_{NM9} > y_{vss}$$

$$y_{vdd} > y_{PM3}, y_{PM3} > y_{NM11}, y_{NM11} > y_{vss}$$

$$y_{vss} = 0, *y_{vdd}, y_0\ y_{11} areintegers$$

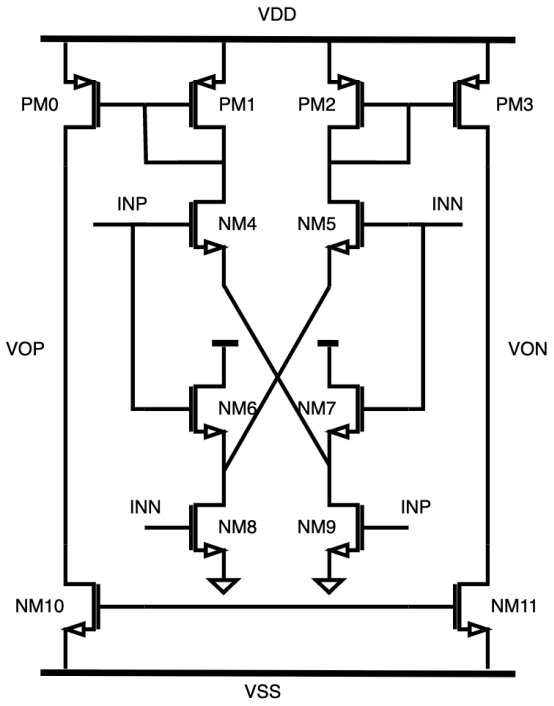
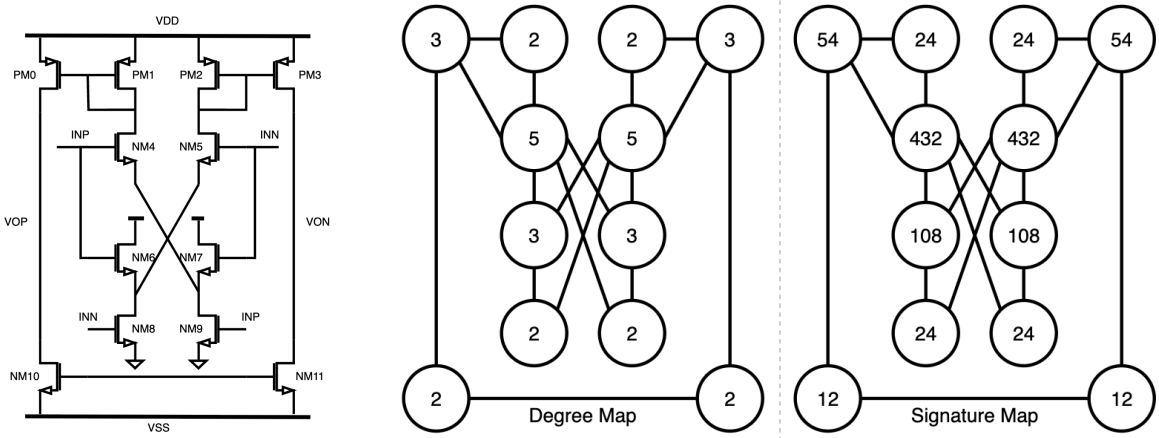


Figure 3.7: Current flow example circuit



(a) (b)

Figure 3.8: Construction of degree map and signature map from a circuit

Placement: Current flow enhanced equivalent device finder

Besides comparing signature values of two nodes to find equivalent relation, current flow paths are also used to enhance the algorithm. For two current paths, if (I)number of nodes equals, (II)for all corresponding devices, device sizes(width and length) are equal and (III) signature S of corresponding devices equal. Then these two current paths are considered as equivalence. In the case below, for example, Vdd-M_5-M_1-M_7-VSS and VDD-M_6-M_2-M_7-VSS are considered as a pair of equivalent paths. After equivalent path pairs are found, symmetry device pairs, self-symmetry devices, and symmetry nets can be found. In the following case, since Vdd-M_5-M_1-M_7-VSS and Vdd-M_6-M_2-M_7-Vss are equivalent, corresponding devices are considered as symmetry pairs(M5-M6, M1-M2), corresponding nets are considered as symmetry nets(X, Y) and if the corresponding devices points to the same device, the device is self symmetry(M7).

For each symmetry pairs(M1, M2), the constraint is formulated as follows:

$$x(M1) + x(M2) == 0, y(M1) = y(M2)$$

In this case, the constraint can be formulated as follows:

$$y_{M5} = y_{M6}, x_{M5} + x_{M6} = 0$$

$$y_{M1} = y_{M2}, x_{M1} + x_{M2} = 0$$

$$x_{M7} + x_{M7} = 0$$

3.3.4 Analog Layout Script Generator

The analog layout generator enables designers to design layouts in Python without needing in-depth knowledge of target technology parameters or concerns about script syntax, as shown in Figure 1. The generator calculates the coordinates of each cell based on layout constraints and technology parameters. Helper functions, including physical-only cells insertion, are included, to prevent design rule violations. The generated layout scripts are used in digital P&R tools, to produce a DRC/LVS clean layout.

The Python-based layout generator interfaces with IC Compiler II, enabling users to use LEGO cells and standard digital cells to script placement and routing with custom defined constraints. Equipped with built-in functions, the layout generator assists users in achieving legal placement while avoiding violations of design rules. It offers a suite of helper functions for tasks such as adding boundary cells, placing cells with layout constraints and generating routing constraints. This tool empowers designers to create scripts that can be used in industry-standard tools and create layouts without necessitating in-depth knowledge of layout design parameters and syntax, thereby speed up the design process. The main helper function includes:

Device Initialization with Layout Constraints

The typical block generation process begins with creating small device blocks, such as sets of PFETs or NFETs, with designers specifying parameters like the number of fins per finger (at least 2), gate length (L_min, L1, L2, or L3), and multiplier (must be an integer). Leveraging Python functions, the layout generator facilitates the creation and placement of numerous

devices or cells in an array pattern, for example, capacitor arrays or common centroid layouts for a large number of devices.

The designers can also define layout constraints for the devices and cells, including the addition of boundaries and symmetric placement. The tool automatically generates boundary cells based on device size and quantity. After adding constraints, the relative location and orientation are updated accordingly.

Placement Initialization

Once the smallest elements are initialized, designers begin placement initialization using the "initialize_placement" function, specifying the relative location of each small block. The tool calculates the row in which devices should be placed based on technology requirements, ensuring proper spacing between active transistors, passive devices, and active transistors with various gate lengths.

Set offset

"Move_placement" is utilized to create spaces for tap cells and IO pads in future designs, allowing for manual adjustment of sub-block locations to meet specific floorplan requirements.

Placement Compilation

Before writing the script, the "compile_placement" step automatically calculates exact x and y coordinates based on relative cell/device locations on the grid and the x/y layout grid width of the target technology. It checks and legalizes cell placement, updating cell orientations to prevent power/ground shorting. If violations are detected, error messages are generated along with suggestions for resolution. During compilation, the tool also identifies passive devices and long-channel devices, creating routing and placement blockages in those areas.

Script Out

The script is generated, translating the database into a human-readable script compatible with industry tools. The tool handles the placement of devices and generates placement and routing constraints using industrial syntax. Designers are relieved from the burden of considering front-end of line (FEOL) design rules as if designing a typical digital layout design.

3.4 Preliminary Experiments: Comparator and CTLE, DCDL

This flow has been experiment to design and simulate under GlobalFoundries 12nm FinFET technology. Three critical blocks, including dynamic comparator with calibration, CTLE and delay lines, including a find delay line and coarse delay line has been implemented. In 12nm technology, all of the devices and cells are from LEGO library and standard digital library and the layout was created with the python layout generator and Synopsys IC Compiler II. The layout is validated with using Calibre DRC/LVS/PEX/XACT and evaluated with Calibre Virtuoso ADE environmnet. The performance of the layouts generated by LEGO cells and flow under 12nm is compared with the same human designed layout in TSMC 28nm. The performance shows the post layout simulation with proposed method is a good match with the schematic design with a normal degradation of the performance.

3.4.1 Current Mirror

Figure 3.10 shows the common layout strategies and drain current mismatch has been simulated for all the strategies, including:

Symmetry ensures mirrored device placement, reducing sensitivity to thermal gradients, parasitic mismatches, and improving CMRR in differential topologies, as in figure 3.10 (a), (c), (f),

Interdigitization divides devices into smaller segments placed in alternating patterns, e.g., M1, M2, M1, M2 as in figure 3.10 (b), (e), (g), to average out local variations, suitable

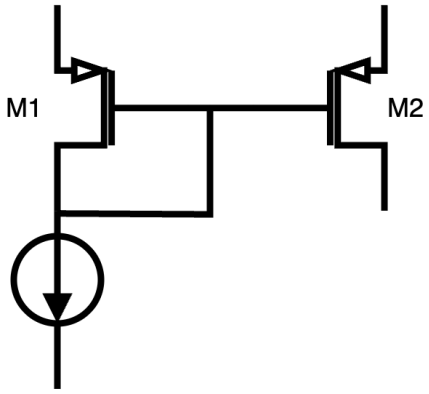


Figure 3.9: A current mirror,

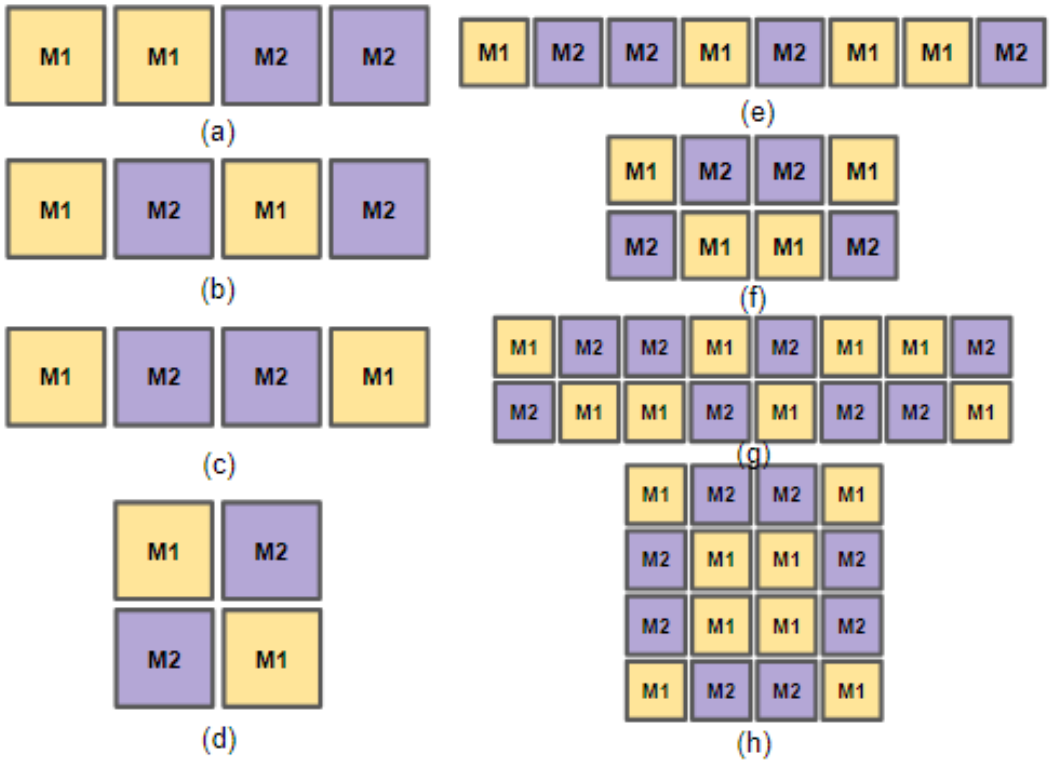


Figure 3.10: Pattern to reduce systematic mismatch

	minimum length FinFET (l=14n)		medium length FinFET(l=142n)	
pattern	default routing	symmetry routing	default routing	symmetry routing
a	30.28	19.32	2.751	7.24
b	2.47	3.22	1.93	2.42
c	10	11.27	3.79	7.23
d	3.46	2.86	1.75	0.23
e	34.27	8.12	0.87	1.67
f	5.53	3.09	0.455	0.938
g	30.36	1.22	0.791	0.555
h	19.16	10.1	0.396	0.154

Table 3.4: The differential nets length under different matching patterns

for matched resistors, capacitors, or current mirrors.

Common centroid arranges segments symmetrically around a central point to cancel linear process gradients, as in figure 3.10 (d), (h) .

We run the LEGO flow and run the placement for each of the matching patterns, and find the matching routing under each cases, listed in table 3.4

3.4.2 Comparator

The comparator with 4-bit offset calibration, as shown in figure 3.11 (a), is designed to achieve 4GHz sampling rate with offset calibration range covering 3 sigma offset variation in 28nm technology. Then, the comparator with same topology was migrated in GlobalFoundries 12 nm. The schematic consists FinFETs with various gate lengths and standard digital cells, including inverters and NAND gates.

The comparator is laid out with a comparator layout generator and the devices are placed symmetrically. Several floorplans have been generated as figure 3.11 (c)-(e). In all

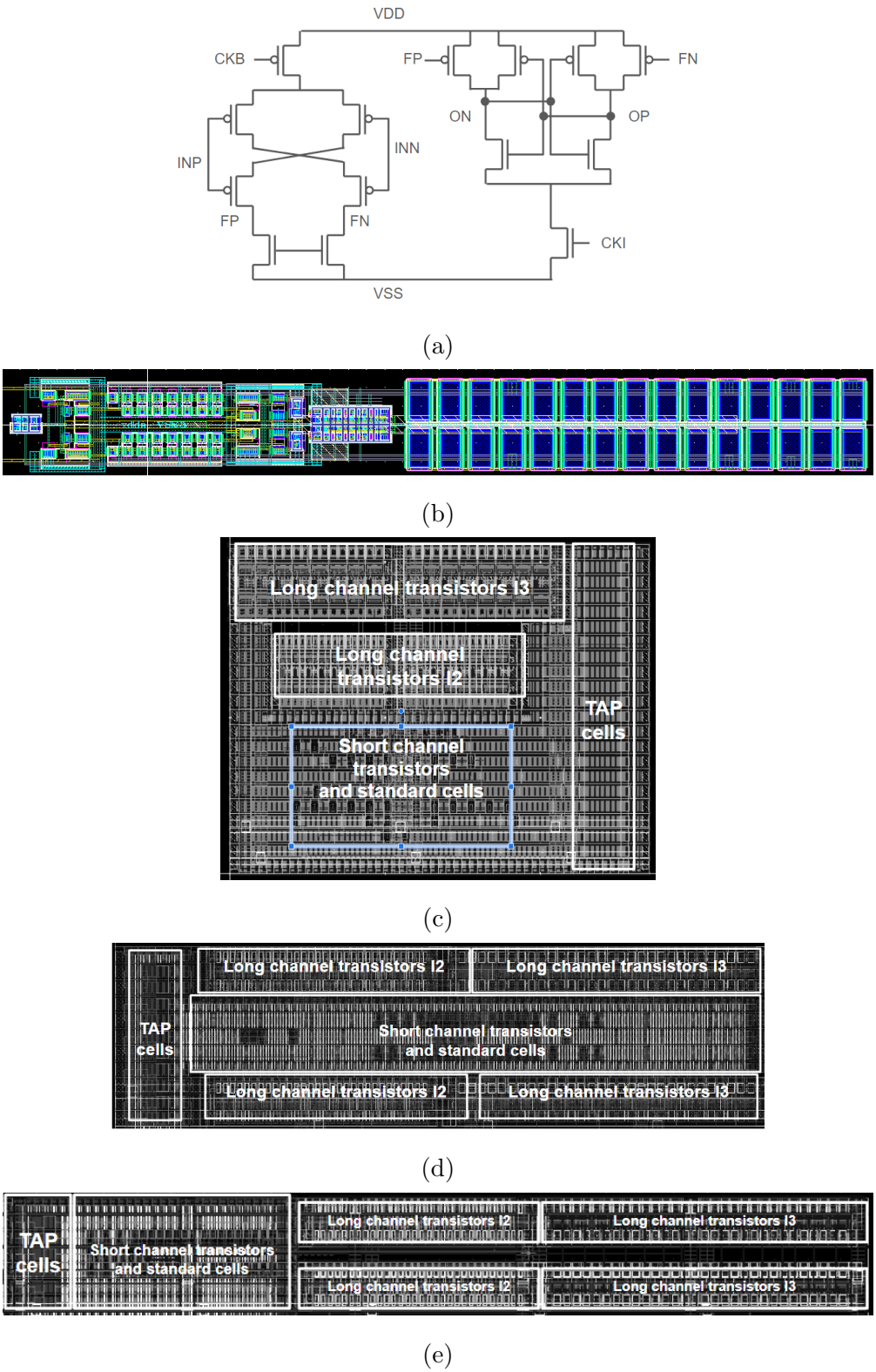


Figure 3.11: Comparator (a) circuit schematic (b) manual layout in 28nm (c)-(e) automatic layout in 12nm with various floorplan

	Area	Systematic Offset	Offset devation
28nm Schematic	-	1.39 mV	12.84mV
28nm manual layout	39 x 6u	0.8mV	14.43mV
12nm schematic	-	1.74mV	16.62mV
G12nm Automatic Layout(c)	19.3 x 5.28u	7.1m	22.23mV
G12 Automatic Layout(e)	25.7 x 3.36u	3.0mV	17.7mV

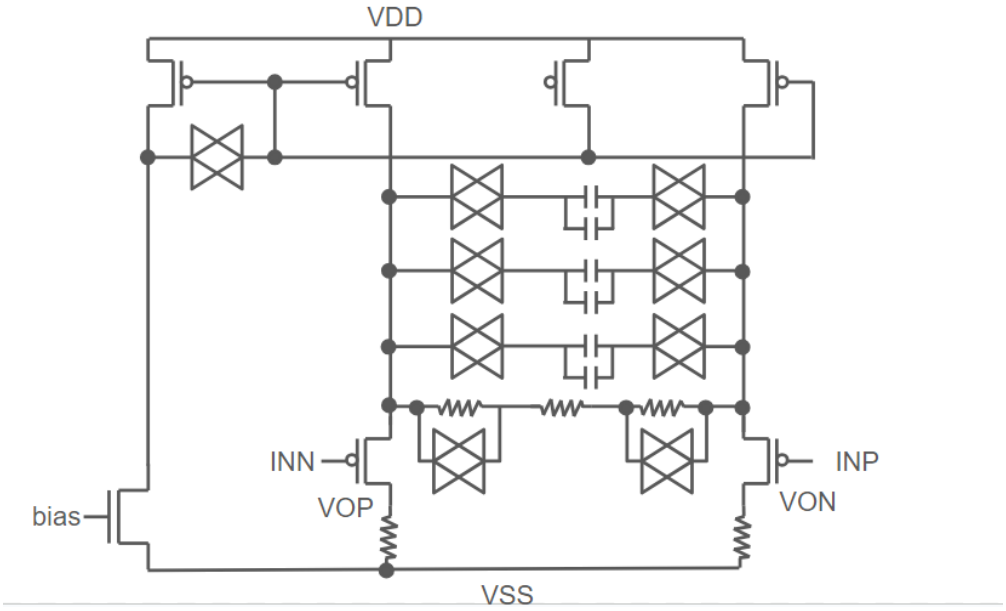
Table 3.5: CMP simulatinon result

floorplans, the matching devices are placed symmetrically and the short channel and long channel transistors are placed in different areas. The simulation of comparator are shown in table 3.5.

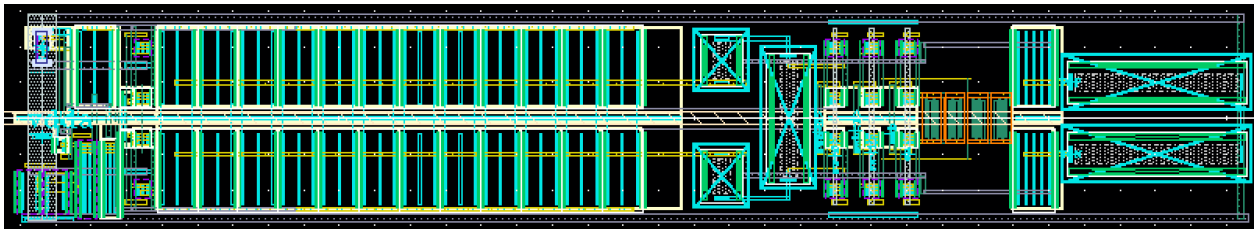
3.4.3 CTLE

CTLE (figure 3.12) is designed to amplify the signal at the nyquest frequency and cancel the noise on higher frequency. We adopt the similar CTLE architecture from the work [18] In addition to the active transistors with various gate length sizes and transmission gate provided by LEGO library, the CTLE is also built with passive devices, including resistors and capacitors with various sizes.

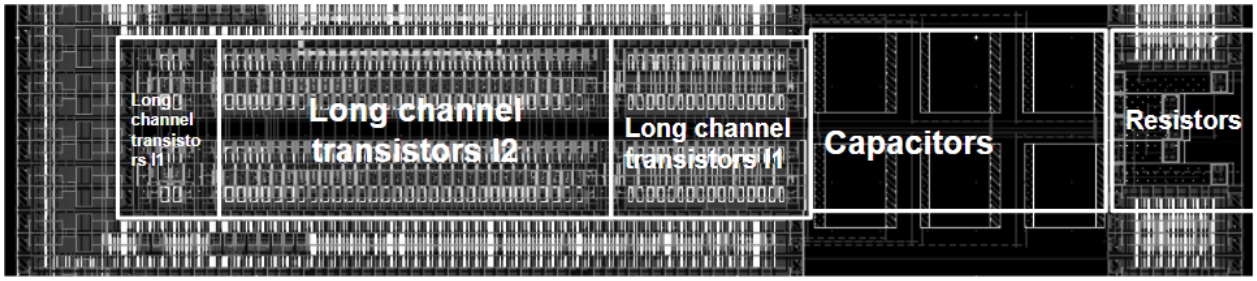
The layout generator is carefully designed, referring to the manual layout under TSMC 28nm, fig 3.12 (b), and the generated layout under 12nm is as in (c). The long channel devices are placed on the left, and the passive devices are placed on the right. Placement blockage around the long channel devices and routing blockage on passive devices are automatically generated to avoid power ground short.



(a)



(b)



(c)

Figure 3.12: CTLE (a) circuit schematic (b) manual layout in 28 nm (c) automatic layout in 12 nm

	Area	DC gain	Boost Amount
28 nm schematic	-	1.11db	1.3, 2.33, 4.31, 5.28 db
28 nm Manual layout	40 x 7u	1.06 db	1.23, 3.19, 4.31, 5.05 db
GF 12 schematic	-	0.9 db	0.01, 1.08, 2.25, 3.25 db
12 nm automatic layout	25.4 x 6.7u	0.7db	0.1, 1.6, 2.8, 3.63 db

Table 3.6: CTLE simulation result

	Area	Fine Delay Range	Coarse Delay range	Total Delay Range
28 nm schematic	-	9 - 33ps	24 - 337ps	32 - 372ps
28 nm Manual layout	97 x24um	16 - 42ps	44 - 646ps	60 - 688ps
12 nm schematic	-	10 - 33 ps	147 - 415 ps	157 - 448ps
12 nm automatic layout	92 x 12um	72 - 97p	269 - 909ps	341- 1006ps

Table 3.7: Delay simulation results

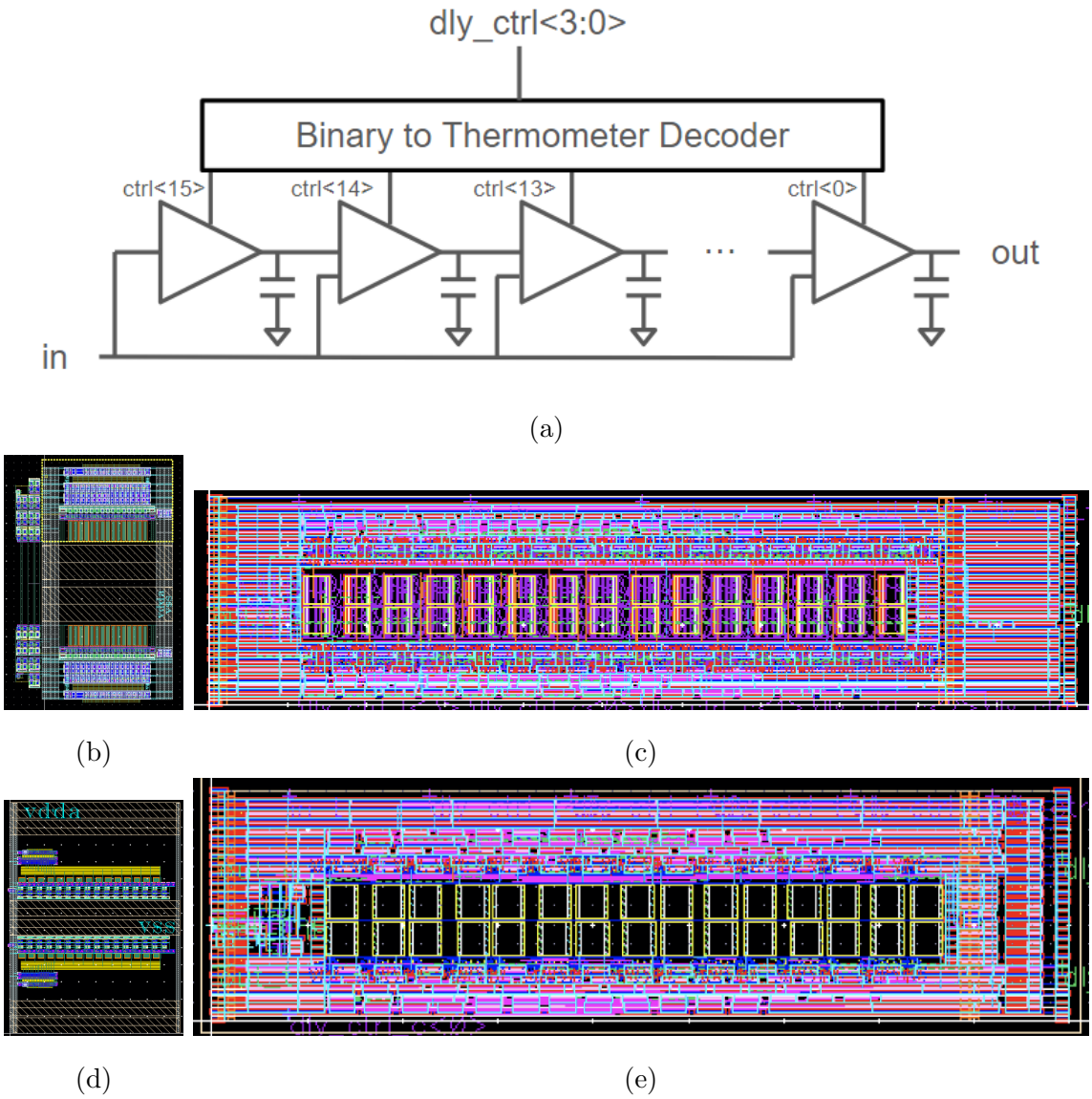


Figure 3.13: Delay line (a) circuit schematic (b) fine delay manual layout in 28 nm (c) coarse delay manual layout in 28 nm (d) Fine delay automatic Layout in 12 nm (e) coarse delay automatic layout in 12 nm

3.4.4 Digital Delay Line

Two programmable delay lines have been migrated to 12nm. The delay line has in total 8 bit input controls, with a fine delay and a coarse delay, each has 4 bits input controls. A pair of differential inputs are fed in to the delay line outputs a pair of differential delayed signal based on the delay control bit. Figure 3.13 (a) shows half of the architecture of the delay lines. Both of the coarse and fine delay lines are built with a binary to thermometer decoder, switches and capacitors. The schematic is migrated automatically from 28 nm design with manual tuning of the capacitor size to meet the minimum resolution under 12 nm.

Figure 3.13 (b)-(e) shows the layout of the fine delay and coarse delay line. The layout was generated with LEGO automatic layout generator and the generator was carefully written referring to the manual layout under 28 nm. The placement is symmetry with a horizontal symmetry line and the passive devices are placed in the middle. The APMOM are chosen to be mapped to 12 nm since they can they can achieve a minimum capacitance Cu = 3f F. And the switches are placed on top and bottom of the layout, together with the input control signals.

The minimum resolution in 28 nm was designed as 2.1ps and the fine delay achieves a resolution of 1.4ps from 10ps to 33ps and the coarse delay is adjustable from 147ps to 416ps.

The simulation of delay has been done for all the possible control values of Di = 0-255, where D[3:0] controls the fine delay and D[7:4] controls the coarse delay. The propagation delay $Ti = T\_dl + Di * \tau$ where T_dl is the minimum delay and τ is the resolution of fine delay. The performance shows a monotonic increase of time delay for increasing values of input controls Di.

3.5 Summary

This chapter has demonstrated the feasibility of designing high-quality analog/mixed-signal circuits using the digital standard cell based design flow. Following the digital standard cell layout rules, a set of analog cells, called LEGO cells, have been designed and implemented in

a 12nm FinFET process. The layouts of four typical analog/mixed-signal blocks including a comparator, an operational amplifier, a level shifter, and a bias circuit have been generated in minutes, instead of days, with the layout quality comparable with that of manual effort, as validated by post-layout simulation. Ongoing effort is directed towards using LEGO cells for automatic generation of high-speed die-to-die interface circuits.

Chapter 4

CASE STUDY: AIB - APPLICATION TO AN OPEN-SOURCE INDUSTRY-STANDARD INTERFACE

Building on the LEGO methodology presented in the previous chapter, this case study applies the flow to the Advanced Interface Bus (AIB), an open-source, industry-standard high-bandwidth interface with repeated slices and multiple voltage domains. The methodology efficiently handles repeated analog and digital blocks while enforcing symmetry, placement constraints, and routing shielding. Results demonstrate fast post-layout iteration, reliable DRC/LVS closure, and consistent device matching, validating LEGO 2.0 for high-density mixed-signal interfaces.

4.1 Overview

Over the past 25 years, device-to-device interfaces have evolved by employing increasingly complex circuits to achieve high speeds over a limited number of wires, as exemplified by PCI Express*. In contrast, AIB [19] takes a different approach, using a very wide parallel interface enabled by advanced high-density packaging. By operating each wire at a relatively low speed, the transmitter and receiver circuitry is greatly simplified, reducing silicon area and design complexity. An example of AIB application is illustrated in Fiugre 4.1.

AIB is an open-source chiplet-to-chiplet PHY level standard that enables a modular approach to system design for interconnecting a variety of chiplet modularized IP (intellectual property) blocks. This work propose the first AIB Layout Generator, in FinFET technology. The architecture of AIB is particularly well suited for evaluating an automated layout flow due to its highly regular structure, strict physical constraints, and sensitivity to timing skew.

An AIB consists of 24 AIB channel and one optional auxiliary AIB channel. Each AIB

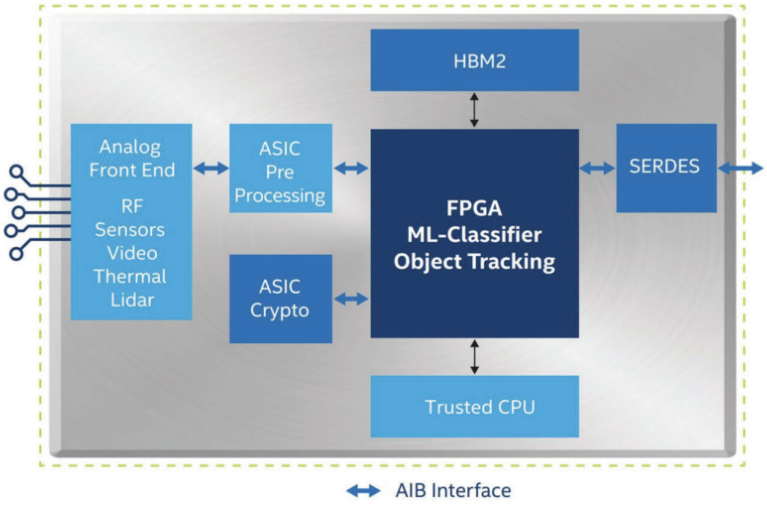


Figure 4.1: An example of AIB application [19]

channel consists of an array of IO buffer, clock buffer, as well as a programmable delay and a voltage reference circuit, as shown in figure 4.2. Each IO buffer must align precisely with a predefined bump map, enforce symmetry between differential lanes, and meet strict electrical constraints to ensure channel integrity.

The high degree of slice repetition, combined with the physical and electrical regularity, and the coexistence of both analog and digital circuits, presents an ideal opportunity for automation. The proposed automated flow is applied at the macro levels, demonstrating placement, routing, and constraint enforcement across multiple repeated blocks while accommodating mixed-signal design requirements.

The main contribution of this work is summerized as follows:

- A complete DRC/LVS clean AIB 2.0 macro P&R with LEGO analog cells and digital flow
- Layout constraints considered and encrypted in the LEGO-written scripts
- Experiment shows the effectiveness of the proposed flow and flexibility with design

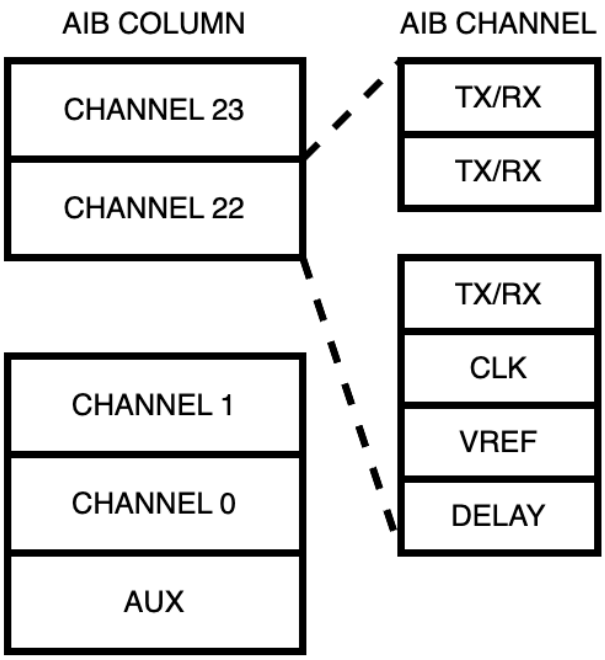


Figure 4.2: The architecture of an AIB channel, each of the channel is consisted of 102 IO buffers, including both TX/RX/CLK buffers, programmable delay and voltage reference

parameters

The remainder of the section is orginazed as follows. Section 4.2 describes the macro design and layout constraints. Section 4.3 shows the customization and features for AIB application. Section 4.4 discuss the verification setup of the AIB design, enabling the designer to verify the macros in system level. Section 4.5 shows the experimental results of the macro and its sub-modules, demonstrating the simulation comparison between the schematic, manual layout design and LEGO automatic layout. Section 4.6 summerize the chapter.

4.2 Circuit Block Descriptions and Layout Constraints

The AIB macros are composed of multiple circuit categories. Each category imposes unique layout requirements that must be captured in the automated flow. The IO buffer drives and receives data across fine-pitch interconnects with controlled impedance and minimal distortion. The clock buffer distributes low-jitter clock signals to timing-critical blocks while preserving signal integrity across process and load variations. The programmable delay enables precise timing alignment between data and clock domains, accommodating inter-die skew and channel mismatches. Finally, the voltage reference block provides a stable biasing point to ensure consistent logic thresholds and termination levels across environmental and process variations. The following subsections describe the design and implementation details of each block, emphasizing their roles in achieving the overall AIB performance targets.

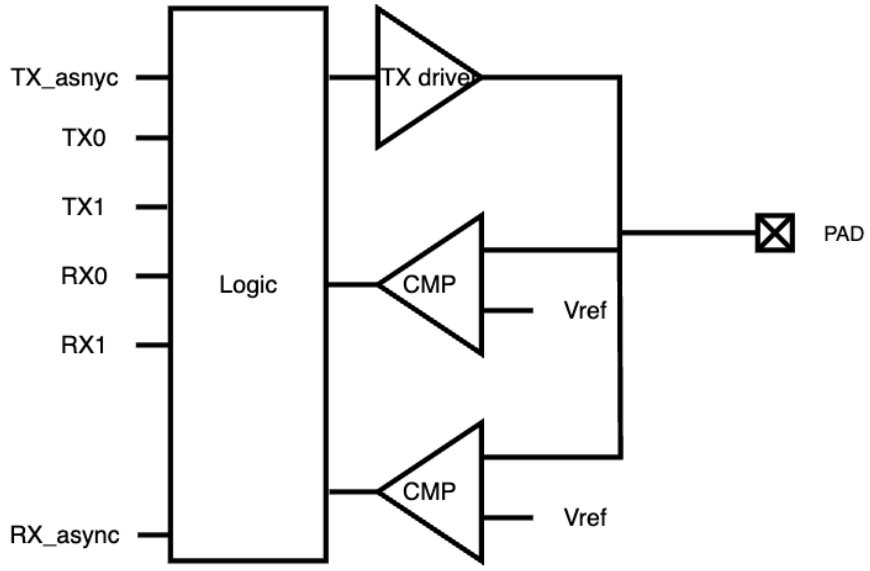
Table 4.1 shows a summary of the AIB custom block. The layout is built following the hierarchy of the table.

4.2.1 IO Buffer

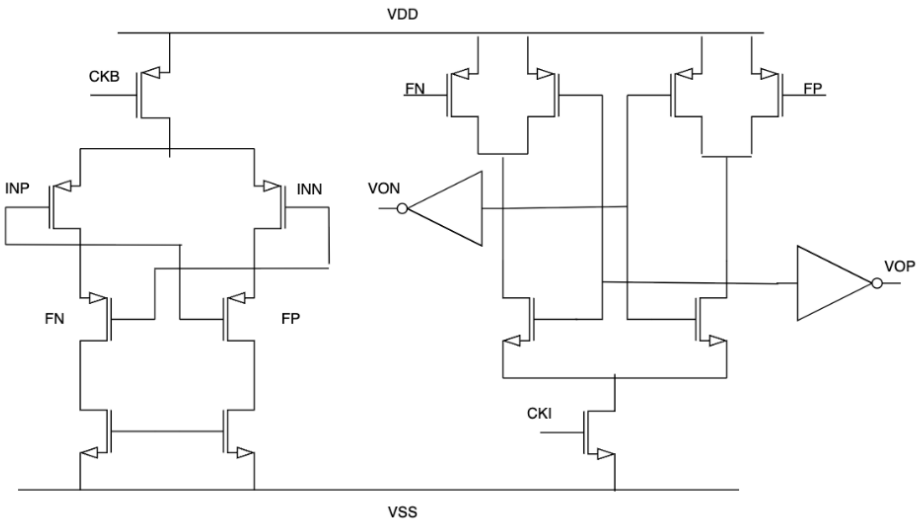
The IO Buffer has several configuration modes: (1) configured to be either IO TXs, IO RXs or side-band control and clock forwarding, (2) transfer mode can be set to either SDR or DDR mode. Each of the TX IO buffer can serialize the 2 inputs to one signal to the channel. Each slice includes a driver and electrostatic discharge (ESD) protection devices. Layout

lv1	lv2	lv3	pmos	nmos	res	cap	digital	macro
IO Buffer			1		2			4
	buffer logic						44	
	driver						14	4
		driver 1	16	28			9	
		driver 2	8	14			9	
	comparator		9	9			3	
CLK Buffer			3	3				3
	aib1 buffer		8	7				
	aib2 buffer						4	2
		dc path	6	6			3	
		ac path	2		2	2		
Delay							5	4
	course delay							41
		decoder 5:32					6	5
		delay cell					3	7
		mux 4:1					7	
		mux 8:1					7	
	fine delay						25	12
		and16					15	
		delay cell					5	2
	delay sync						8	17
Vref			4		16		7	

Table 4.1: Layout generator summary of AIB Custom blocks

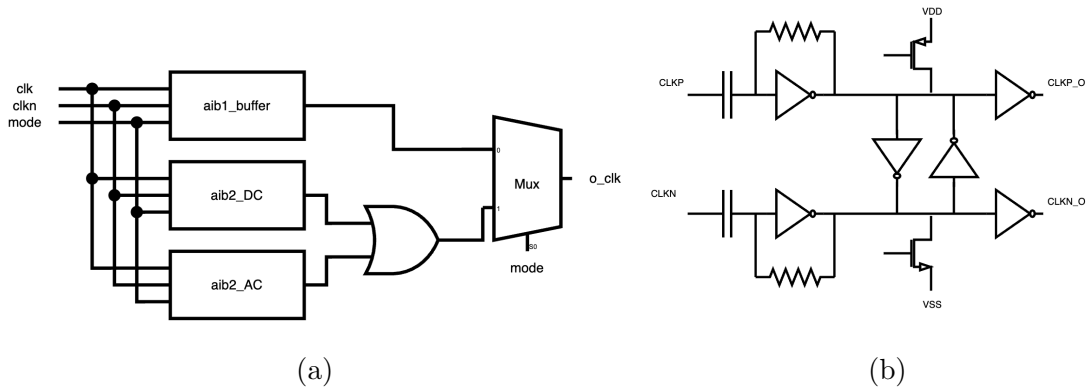


(a)



(b)

Figure 4.3: (a) IO Buffer Block Diagram (b) Comparator Schematic



(a) (b)

Figure 4.4: Clock buffer block diagram and AC path schematic

challenges include:

- **Symmetry** Enforcing symmetry for the differential pair to minimize offset, eg. differential input i_vinp and i_vinn to the comparators in figure 4.3 (c).

- **Pin Constrains** Aligning inout PAD, shown in figure 4.3 (a), and signal with the bump array at a fixed pitch.

- **High-current Routing** Routing high-current paths with wide metal width.

- **Matching Routing** Routing delay match on the configuration signals to the pull-up/pull-down PFET/NFET, as in figure 4.3 (b).

- **PG Routing** Wide power/ground straps to avoid IR drop.

- **Area Constraint** Minimize the area to reduce the routing parasitic to increase the data speed and achieve the target data rate.

4.2.2 Clock Buffer

The clock buffer provide high-speed clock signal to the channel and syncronize the clock across the channel. A DC path is designed for low speed data path and an AC path is designed to provide the clock when data rate over 1Gbps. The duty cycle is restricted to be under 3%. To achieve high clock speed, layout constraint includes:

- **Capacitor compliance** Compliance with passive devices in AC path to handle signal in high frequency with better integrity

- **Routing Shielding** Routing shielding on clock signals to prevent crosstalk and increase signal integrity.

- **Symmetry** To reduce jitter and duty cycle distortion, the differential clock signal path needs to be symmetric, as clkp to clkp_out and clkn to clkn_out in figure 4.4 (b)

4.2.3 Programmable Delay

The programmable delay is inserted into the clock path to compensate for timing skew between lanes and allows fine adjustment for process, voltage, and temperature (PVT) variations. It has 16 bit of configurations and each of the unit delay is designed to be 7ps. The 5 bit MSB controls the coarse delay path and the 11 bit LSB controls the fine delay path. The whole programmable delay is composed of two paths: the coarse delay and fine delay, as shown in figure 4.5 (a). Each of the coarse delay path and fine delay path is composed of a series of unit delay cells , as shown in figure 4.5 (b)-(c). By controlling the configuration bits, the total delay is the sum of coarse delay and fine delay. To keep the total delay monotonic and with balanced duty cycle, the layout constraint includes:

- **Matched Routing for Symmetry** Delay elements must have identical parasitics between multiple lanes from the muxes to maintain consistent timing increments.

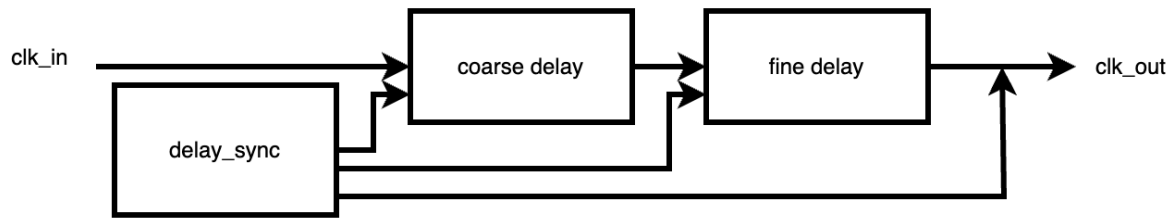


Figure 4.5: Programmable Delay block diagram

- **Minimized Skew Variation** Interconnect length and loading must be uniform to ensure the programmable delay steps are consistent across instances.

- **Shielding High Speed Signal Path** To reduce crosstalk, minimize electromagnetic interference, and maintain signal integrity in the presence of fast switching transitions.

4.2.4 Voltage Reference

The voltage reference block generates a stable low-voltage output using a resistor ladder network for biasing IO buffers in the AIB , as shown in figure 4.6. The ladder divides the reference voltage into fixed ratios, and analog switches select the tap corresponding to the desired output level. The design provides three selectable output levels, 180mV, 200mV, and 220mV to accommodate PVT variations, as well as different operating modes. It improves signal integrity and reduces sensitivity to noise or supply fluctuations. The layout constraints includes:

- **Shielding and Guard Rings** Sensitive analog nodes must be shielded from nearby switching logic; guard rings reduce substrate noise coupling.

- **Wide metal on high current path** The voltage reference draws a steady 100µA bias current, which imposes requirements on the metal routing to ensure low IR drop and

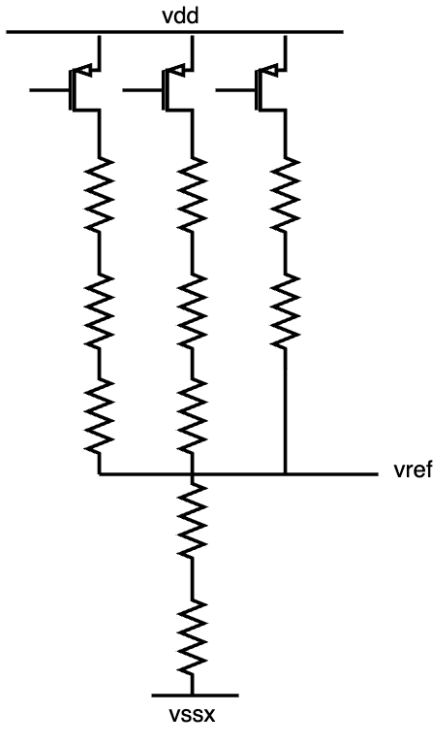


Figure 4.6: Voltage reference block diagram

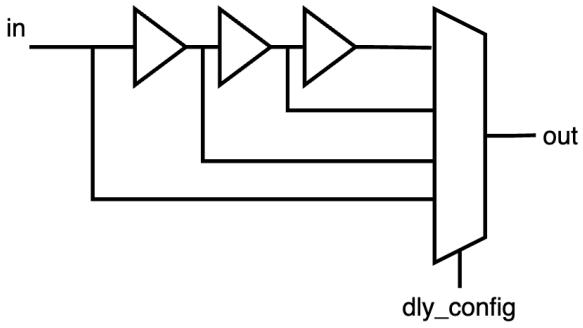


Figure 4.7: Simplified delay path schematic

reliability. To meet current density limits and minimize voltage variation, the supply and output connections are implemented with sufficiently wide metal traces.

- **Power and Ground Routing** Clean, low-impedance supply rails are necessary for stable output; separate analog supply routing may be required.

- **Passive Devices Matching** The routing of resistors introduces additional parasitic resistance, causing deviations in the effective resistance values and introduce errors in the intended resistance ratios critical for accurate voltage bias generation.

4.3 Flow Customization for AIB

4.3.1 Placement Automation with Design Parameters

The unit delay in programmable delay depends on the size of two custom clock buffer. This design involves 2 custom clock buffers on the delay path. To make the block compact, we design the custom clock buffer with minimum gate lentgh. Thus, the design parameters for this block would be p1-nfin, n1-nfin, p2-nfin, n2-nfin. We then sweep the design parameters to get the unit delay to meet the spec.

4.3.2 Matching Sensitive Devices Placement

The filler cells are placed as dummy cells around the matching sensitive devices. The matching pairs are placed first in the flow, with mirrored orientation. Then, filler cells and tap cells are placed at the empty area. The placement is optimized to be symmetric and compact for minimal routing length.

4.3.3 Routing Automation with Constraints

The vref in voltage reference block needs shielding constraints. The vref also need to carry 100uA current and the the maximum allowed current density(Jmax) for M1 is about 1mA/µA. Thus, the default minimum routing width cannot be applied to this case, non-default-routing rule (NDR) should be applied.

Shielding is applied to high-speed clock traces to reduce crosstalk, contain electromagnetic fields, and maintain signal integrity. Grounded guard traces placed adjacent to the clock route limit capacitive coupling to neighboring signals, minimizing jitter and ensuring stable timing across the chip.

4.4 Verification Setup

Verifying the functionality of custom design blocks within the AIB, with the overall system, is essential. The AIB top-level design is typically verified at the gate level using SystemVerilog, which includes all custom blocks. Gate-level simulation provides a closer approximation to real-world behavior than high-level models, as it accounts for the actual netlist structure, timing delays, and logic interactions—effects that high-level simulations often abstract away. Two common approaches are used to validate the system:

- **Gate-Level Simulation:** The synthesized gate-level netlist is imported into Virtuoso, the custom blocks are connected, and a Spectre simulation is performed. This method is highly reliable, but transient simulations can take hours or even days to complete.

Block Name	Reference Manual Design in Intel 16nm	LEGO Design in GF 12nm
IO Buffer & Clock Buffer	25.6 x 23.2	24 x 22.56
Programmable Delay	38.0 x 26.5	56.7 x 28.8
Voltage Reference	12.8 x 11.5	12.8 x 11.04
Clock Branch	6.42 x 5.8	5.6 x 5.2

Table 4.2: Area comparison between Intel 16nm reference layout and GF 12nm LEGO-based automatic layout

- **SystemVerilog Simulation:** Custom design blocks are described and modeled in SystemVerilog and simulated with VCS. While this approach is much faster, typically completing in minutes, the models may not fully capture real-world functionality.

Since most custom designs are implemented using digital standard cells, gate-level simulation is generally available and serves as a robust method for validation.

4.5 Experimental Results

In the experiments, we follow the hierarchy of the schematic and the reference manual layout design provided in Intel 16nm, create the submodule layout with updated design in GF 12nm, and integrate 4 custom blocks with 18 generated sub-blocks. The flexibility of digital P&R tool, together with prebuilt analog devices, greatly reduced the iteration time from netlist to gds.

To evaluate the result of our LEGO flow, we compared the layout between reference Intel 16nm design, manual GF12nm layout and LEGO-based GF 12nm layout, run DRC/LVS/xACT with Calibre for verification and run post-layout simulation in Virtuoso ADE environment for the key sub-blocks and the 4 custom designed macros. Table 4.2 shows the area comparison between reference Intel 16nm layout and GF 12nm layout design.

	manual	LEGO
VIN/VIP	0.01	0.002
OP/ON	0.01	0.005
FP/FN	0.01	0.008
VON/VOP	0.03	0.03

Table 4.3: Comparator C+CC mismatch of symmetry pairs

		Manual	LEGO
Size		1.68 x 3.6	3.66 x 2.88
Mismatch	VIN/VIP	0.01	0.002
	OP/ON	0.01	0.005
	FP/FN	0.01	0.008
	VON/VOP	0.03	0.03
Input Offset @ 4GHz		10mV	30mV

Table 4.4: Comparison between manual and LEGO-based layout

4.5.1 Comparator

The comparator is a key sub-block in TX driver. It it designed to run at a data speed of 4G. A manual layout and LEGO-based layout are both designed for comparison. Both layouts follows the symmetry placement and routing. The LEGO layout is 3.659 x 2.88 um, without tap cell and filler on the side. The tap cell introduces extra 1.176 um on the side and it can be shared across all the rows.

Table 4.5 shows the comparator offset simulation result of schematic, manual layout and LEGO-based automatic layout. Due to the clock routing coupling to differential signal OP/ON, the input offset increased from schematic to post-layout simulation.

	Data rate	Input Offset
Schematic	4G	1m
manual	4G	10m
LEGO	4G	30m
LEGO	2G	15m
LEGO	900M	5m

Table 4.5: Comparator offset Simulation

4.5.2 Driver

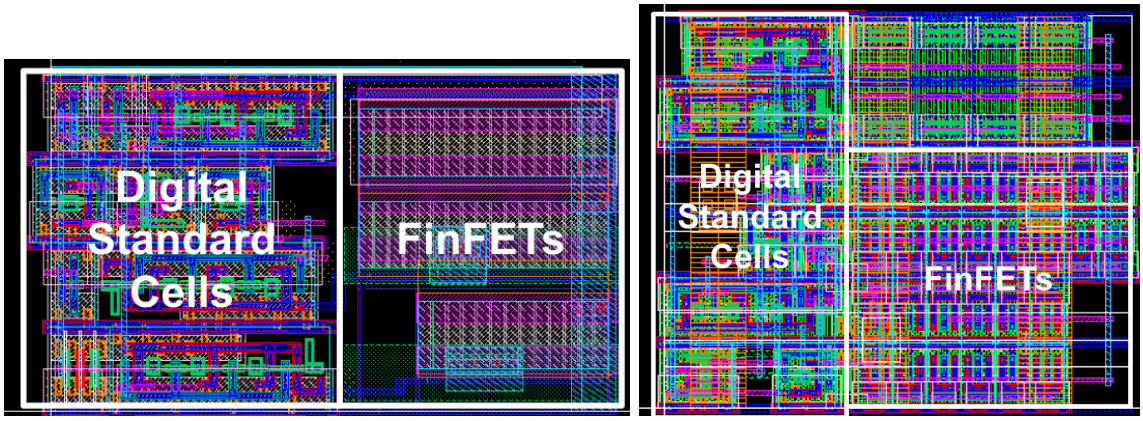
Driver restrict the data speed of the IO buffer as a TX. To reach the highest data speed, the layout need to minimize the routing parasitic, thus a compact placement is necessary to minimize the routing length.

Two iterations of the LEGO automatic layout are shown in figure 4.8. The area can be reduced by 40% with merging devices. (LEGO) The manual size is 3.318 x 1.92 and the LEGO-based layout size is 3.108 x 2.88, which has a 30% area overhead. However, the LEGO-based layout can be seamlessly integrated with other blocks and the manual designed layout would require extra gap in between to avoid DRC violation.

4.5.3 IO Buffer

Figure 4.9 shows the block diagram and layout of the IO Buffer. The IO buffer is composed of two comparators, one driver and logic. To match with the ubump location, the area budget of the IO buffer is 25.6 x 23.2 and the LEGO autogenerated layout has an area with 24 x 22.56, including decap cells, fillers and boundary cells. The area can be further reduced by removing redundant filler cells.

To align with the signal flow and comparator symmetry, the logic is placed on the left, driver placed on the right middle and the two comparators are placed symmetyically on the



(a) (b)

Figure 4.8: Driver 1 (a) manual, size 3.3 x 1.92 um (b)LEGO, size 3.1 x 2.88 um

top and bottom side of the driver.

4.5.4 Fine delay

The unit fine delay depends on the size of two custom inverters. The schematic of the inverter is shown in figure 4.11. The inverter has 4 devices and the length of each transistors are fixed as minimal width for compactness. The nfin of each transistor are tunable from 2-10, as in table **??**. The initial nfin size of p1, p2. n1, n2 are set at 7, 7, 6, 6. Since the combination of the parameters is more than 10k, the values are selected based on

- Baseline and test the impact of each parameter individually.
- Corner cases, including minimum and maximum values
- Midpoint mixes to check interactions.

Table 4.6 shows the unit delay with different nfin parameters selected. Schematic, semi-post sim and LEGO-based post-layout simulation results are compared. The table shows a potential solution map and enables the designer select the size.

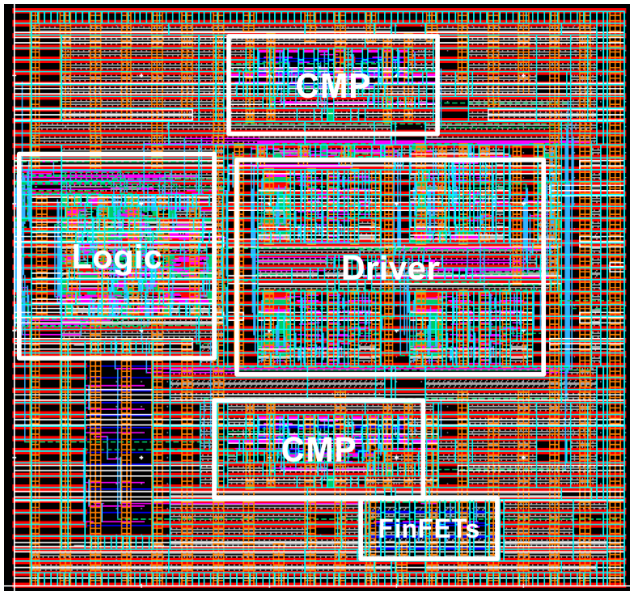


Figure 4.9: LEGO-based IO Buffer Layout with annotated components

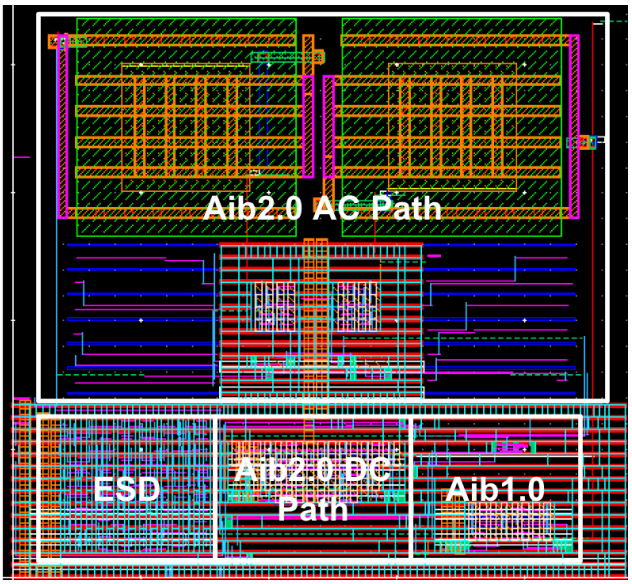


Figure 4.10: LEGO-based generated clock buffer layout, with annotated components

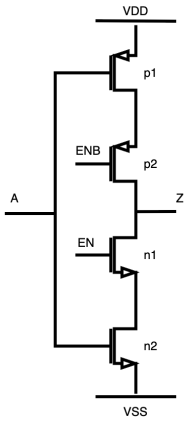


Figure 4.11: Fine delay inverter schematic, the length are fixed at minimum length and the nfin are adjustable with range of 2-10

p1	p2	n1	n2	pre rising/falling	semi-post rising/falling	post rising/falling
7	7	6	6	3.6ps/3.7ps	12.5ps/13.6ps	14.0ps/15.0ps
6	6	6	6	3.4ps/3.6ps	13.0ps/13.4ps	14.5ps/15.5ps
2	7	6	6	3.3ps/3.5ps	12.5ps/13.7ps	14.0ps/15.0ps
10	7	6	6	4.0ps/3.7ps	13.4ps/13.2ps	14.8ps/15.0ps
7	2	6	6	2.7ps/5.4ps	12.8ps/13.6ps	14.2ps/15.2ps
7	10	6	6	3.6ps/3.9ps	13.0ps/13.5ps	14.5ps/15.2ps
2	2	2	2	3.0ps/3.0ps	12.5ps/13.0ps	14.0ps/14.5ps
10	10	10	10	4.0ps/4.2ps	13.2ps/13.5ps	14.8ps/15.0ps
2	10	2	10	4.4ps/5.7ps	13.5ps/13.8ps	15.0ps/15.8ps
5	7	8	10	3.2ps/4.5ps	12.5ps/13.6ps	14.0ps/15.0ps
8	4	7	9	3.0ps/5.0ps	12.5ps/13.6ps	14.0ps/15.0ps

Table 4.6: Unit fine delay with parametric device sizes

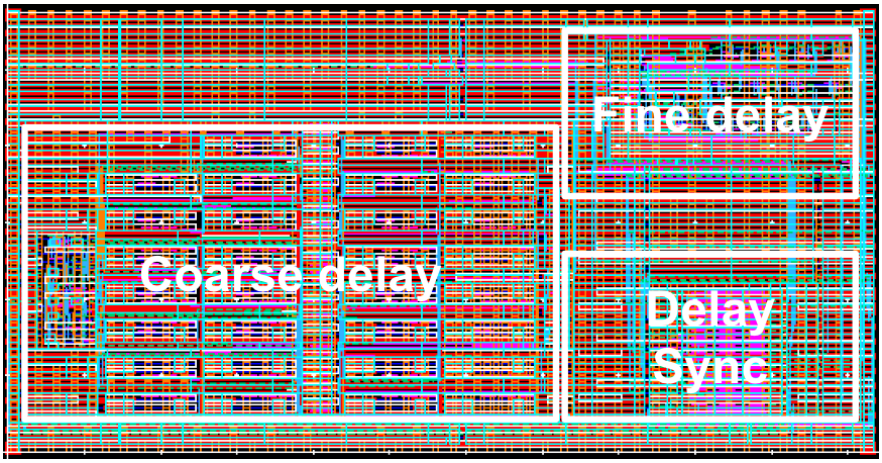


Figure 4.12: LEGO-based programmable delay layout, with annotated components

4.5.5 Coarse delay

The tap cells are shared on the top level of coarse delay and removed from each coarse delay cells to reduce the total area. The schematic simulation shows the unit coarse delay as 82.6ps, semi-post simulation as 244ps and LEGO-based layout as 294ps.

4.5.6 Delay

Figure 4.12 shows the block diagram and layout of the programmable delay. The post-layout simulaiton of unit fine delay is 18.5 and the 11 unit fine delay cell gives the maximum fine delay as 203ps. The unit coarse delay is 294ps, which result monotonic overall delay. The duty cycle ranges between 50.5-53.

4.5.7 Voltage reference

The voltage reference block constraints includes routing shielding, wide metal routing and resistor matching. The schematic and layout are shown in figure 4.13. The size of the layout is 14.2 x 6.2, with boundary cells included. Table 4.7 shows the simulation results between schematic and post layout simulation. The design target to provide stable voltage of three

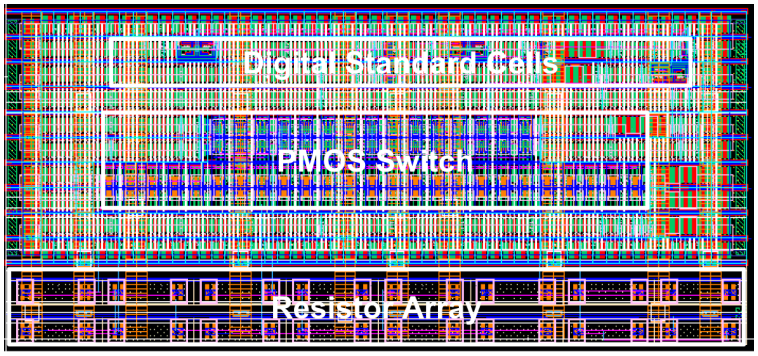


Figure 4.13: LEGO-based voltage reference Layout, with annotated components

	vrefl(mV)	vrefm(mV)	vrefh(mV)
target	180	200	220
pre	179.1	201.2	219.6
post	178.7	204.5	217.3

Table 4.7: Voltage reference simulation results between pre and post layout

voltage levels, vrefh=220m, vrefm=200m, vrefl=180mV.

4.6 Summary

This section presents a practical application on AIB 2.0 custom-designed blocks. Using the proposed LEGO analog cell library in combination with digital placement and routing tools, functional and compact layouts are generated in minutes. The framework significantly reduces schematic-to-layout iteration time and provides analog designers with greater flexibility for optimization, enabling fast post-layout exploration and efficient constraint management.

Chapter 5

CASE STUDY: EIC - EXTENDING TO LONG-CHANNEL DEVICES

Following the AIB case study presented in the previous chapter, this case study applies the LEGO methodology to the EIC design, which has higher performance requirements. The framework extends to long-channel devices and multi-power domain support, while maintaining symmetry, device matching, and routing constraints. Results demonstrate that LEGO 2.0 efficiently generates layouts, enables fast post-layout iteration, and provides a scalable, automated solution for complex analog and mixed-signal circuits with demanding performance targets.

5.1 Overview

The EIC chiplet interfaces with external components through high-density uBump connections and standard high-speed interfaces such as AIB. It serves as a complex test case for the LEGO 2.0 methodology, demonstrating the framework's ability to handle high-performance, mixed-signal interconnects across multiple domains. The design presents a range of layout constraints and performance requirements, providing an opportunity to evaluate the efficiency and scalability of the automated layout flow.

The original LEGO methodology (LEGO 1.0) introduced a digital standard-cell–inspired approach to automate analog layout, enabling compact and repeatable placement. However, LEGO 1.0 was limited to:

- Minimum-length devices.
- Single power domain.

- Simple passive integration.

While effective for certain applications such as AIB, these limitations restrict its applicability to more diverse and demanding designs and with an extended version of LEGO methdology, the LEGO 2.0 supports the following:

Routing Parasitics with NDR Interconnect resistance and capacitance can degrade high-speed performance and disturb biasing.

Multi-Power Domain Integration Requires careful power routing and isolation between domains.

Non-Standard Devices High-voltage transistors, long-channel devices, and passive components present unique layout challenges that complicate automation.

The main contribution of this chapter is summarized as follows:

- A complete DRC/LVS clean EIC macro P&R with LEGO analog cells and digital flow

- New feature introduced to expand the application with long-channel devices, diodes and multiple power domain

- Experiment shows the effectiveness of the proposed flow and flexibility with design parameters

The remainder of this chapter is organized as follows. Section 5.2 presents a high-level description and outlines the layout constraints of its key analog/mixed-signal blocks. Section 5.3 details the customization and additional features of the LEGO 2.0 framework for the EIC, extending the methodology demonstrated in the AIB application. Section 5.4 presents experimental results for the key blocks, including comparisons between schematic, semi-layout, and fully automated LEGO-generated layouts. Section 5.5 concludes the chapter
.

lv1	lv2	# pmos	# nmos	#res	#cap	#diode	# digital	# macro
CMP		9	4				7	5
Bias		39	34	5		4		8
TIA				8				11
	LS	3	3				2	
	OPAMPN	9	9		3			
	OPAMPP	6	5		1			
	VPD_SEL	7	8	5	1			5
	TCL_RFB_P	10	10	4			7	6
	Power Detector	2	5	9	1			1
	TIA_RFB_P	8	8	4			12	
C2C		5	5	4			33	

Table 5.1: Summary of EIC custom blocks

5.2 Circuit Block Descriptions

EIC is composed of six main building blocks: Transimpedance Amplifier (TIA), Delay-Lock Loop (DLL), Clock-Data-Recover (CDR), RX Clock Generation (RXCKGEN), Serial-to-Parallel(S2P) converter, and Thermal Control Loop (TCL).

This work focus on the key blocks in the EIC, including a level shifter, comparator, C2C and TIA, which is the major mixed-signal block in both TX and RX side. Table 5.1 shows the hierarchy and a summary of all the blocks using LEGO framework.

5.2.1 Level Shifter

A level-up level shifter converts digital core voltage 0.8V to higher analog voltage 1.7V. Two different type of devices are used in this block, core devices, which operates under core voltage 0.8V and eg devices, which operate under high voltage 1.7V. Two cross-coupled

PMOS powered by the higher voltage domain, ensures the signal reaches full VDDH level. The layout constraint and challenges includes:

- **Speed/delay**: LS add extra delay on the signal path and it must be designed to meet time constraints
- **Signal integrity**: Need clean transition without overshot and undershoot
- **Area**: The level shifter is used extensively across multiple blocks; therefore, a marginal increase in its area can result in a substantial expansion of the overall layout area.

5.2.2 Comparator in High Voltage Domain

Similar with the comparator in AIB, this comparator compares the differential inputs and output rail to rail logic 1 or 0 as the compaison result. However, different from the comparator in AIB, this comparator is operating under higher voltage area 1.7V, thus the devices have are using long-channel devices, while the AIB comparator uses minimum-length (l=14nm) devices.

- Symmetric placement: The differential pairs, including input pairs and output pairs, require symmetric placement
- Symmetric routing: The differential signals require symmetric routing, including same metal layer and length, and same parasitic to the clock signal.

5.2.3 Power Detector

Figure 5.1 shows the Power Detector circuit [22]. It is simply a common source amplifier and output a voltage, inversely proportional to its differential input swing where $Aout = R * (Ain)^2$. A capacitor C is intentionally placed at the output node to filter out the high frequency signal component. With both the AC and DC path design, the thermal control

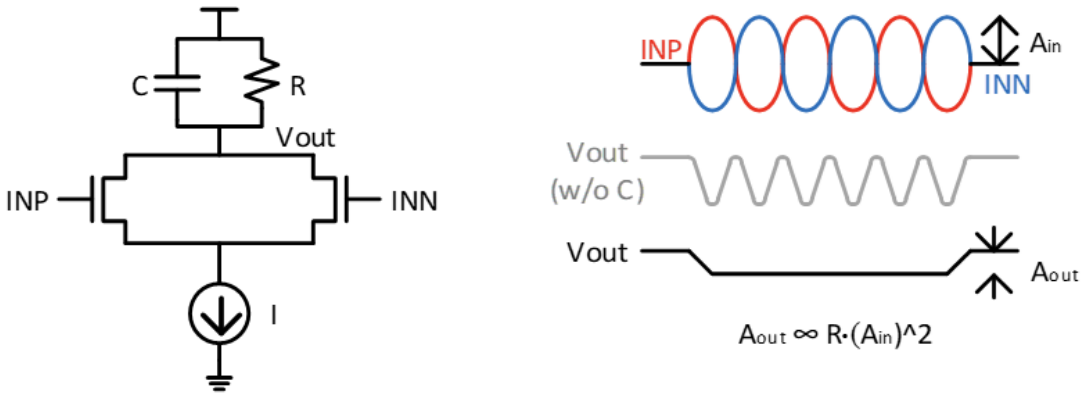


Figure 5.1: Power Detector Block Diagram

loop can switch the MUX in front of the ADC to detect the average or OMA PD current when stabilizing TCL.

- High Current: The current I runs up to 50uA and needs wide metal to carry large current.

- Input pair matching: To keep the same slew path and mininmum offset, input differential pair need to be symmetric

5.2.4 Supply Independent Biasing Circuit

The bias circuit takes 8uA and 35uA reference current and provides bias voltage to other blocks, including unity gain buffer (UGB), TCL, DLL, Some of the layout constraints include:

- Device matching: Current mirror devices need to match to current cell in TCL block

- Diode compliance: Diode is introduced in the Bias circuit, so diode devices are add to the LEGO library

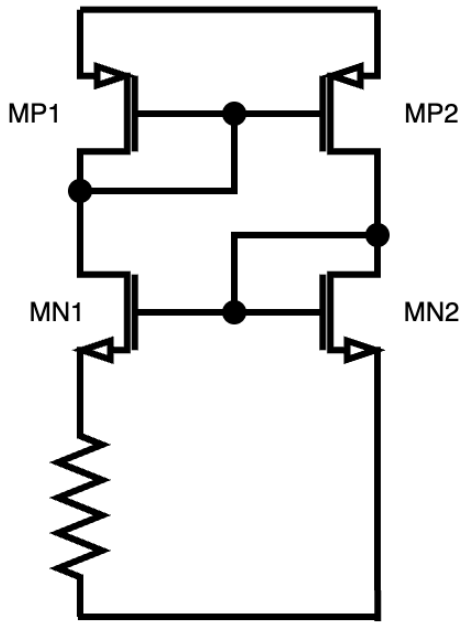


Figure 5.2: Supply independent biasing schematic

- Multiple power domain: The bias is working under both core voltage and high voltage domain. Multiple power strips are added to the scrip and minimum distance constraint between core an high-voltage devices need to be added.

- IR Drop: The current on each branch varies from 8uA to 64uA and the total current is above 500uA. To provide precise bias voltage, the IR drop needs to be under 3%, which is 54mV, so that the total resistance on pg needs to be under 108ohm. Assuming the sheet metal resistance of M3 is 28ohm/um, which restrict the ratio of PG length and width to 3.8.

5.2.5 CML to CMOS Converter (C2C)

The C2C converter [21], which performs level conversion from low (CML) level to high (CMOS) level, is a critical path that causes severe duty-cycle distortion and phase mismatch.

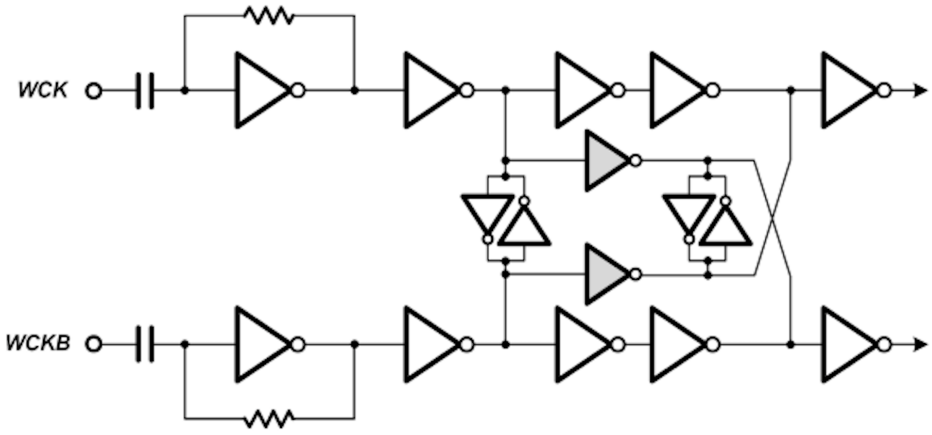


Figure 5.3: C2C schematic [21]

The layout constraints include:

- Symmetry layout: The clock path from differential clock_in to differential clock_out should be symmetry

5.2.6 OPAMPN

The opamp takes differential input and amplified to a single end output. The layout constraints includes:

- Input pair LLE: Dummy devices should be placed around matching devices and matching devices should be placed symmetrically

- Input pair WPE: Identical well proximity for matching devices.

5.2.7 TIA

TIA frontend is composed of 6 macros, logics and a high-speed signal path. The 6 macros includes opamp, power detector, etc. Layout constraints includes:

- High speed path: minimize routing length,

- Routing shileding: including all the voltage reference signals: Vref, Vbp_fil, Vpd_bias
- Routing matching of the symmetrical nets

5.3 Flow Customization for EIC

The automated flow incorporates optimizations to support advanced layout requirements such as non-minimum-length devices, multiple power domains, and array generation for passive devices. These enhancements ensure that the flow can be applied to a broader set of mixed-signal and high-performance circuits.

5.3.1 Integration with the long-channel Devices and device merging

Non-minimum length transistors are often used for analog performance (higher output resistance, better matching, reduced short channel effects). To integrate with the long-channel devices, several challenges merged:

- Device parameterization: automated placement must adapt to variable gate lengths. In GF12 technology, the minimum gate length is 14nm, 16nm and 80nm-200nm.
- DRC compliance: longer devices can alter poly pitch and routing density. The core devices could use minimum routing pitch, eg. 0.032nm for M1, while the nets connecting to Enclosed Gate(EG) devices require larger routing pitch, eg. 0.054nm for M1. NDR must be applied to such nets.
- Device merging/abutment: We allow long-channel devices to be merged and abutted during placement. This enhances cell performance by reducing parasitics and increases layout density by enabling the placer to arrange devices into complex merged structures .
- Matching & symmetry: ensure common-centroid placement still works with longer devices. In this flow, the matching-sensitive devices are placed first with dummy cells

placed around the devices.

5.3.2 Multiple power domain

Since the EIC operates under three power domains: core 0.8V, high-voltage 1.5V and 1.7V, the flow is extended to handle multiple power domains, automatically generating separate supply grids and isolation structures. Placement rules ensure that domain boundaries are physically isolated, while automated insertion of level shifters ensures functional signal transfer across domains.

5.3.3 Passive devices arrays

The flow includes passive device array generators that implement common-centroid and interdigitated arrangements automatically. Dummy devices are inserted where required to minimize edge effects, and routing is symmetrically balanced to preserve device matching across the array.

- Matching: resistor/capacitor elements require common-centroid or interdigitated arrangements.
- Routing symmetry: parasitic balance in connections.

5.4 Experimental Result

The EIC is designed under GF12LP technology. The operation temperature is 75-90 degree with data rate per channel as 16Gps. The core Power supply is 0.8V and the multiple power domain, including 1.5V and 1.7V, are applied to the circuit. The TIA is built hieratically with 3 major modifications on the floorplan. Other custom designed blocks, including comparator, bias and C2C are critical analog/mixed-signal components of the EIC.

In the experiments, we follow the hierarchy of the schematic, create an initial floorplan based on the signal flow and bump locations, create the submodule layout with area restrictions and pin locations, and integrate the top-level TIA with the generated sub-blocks. The

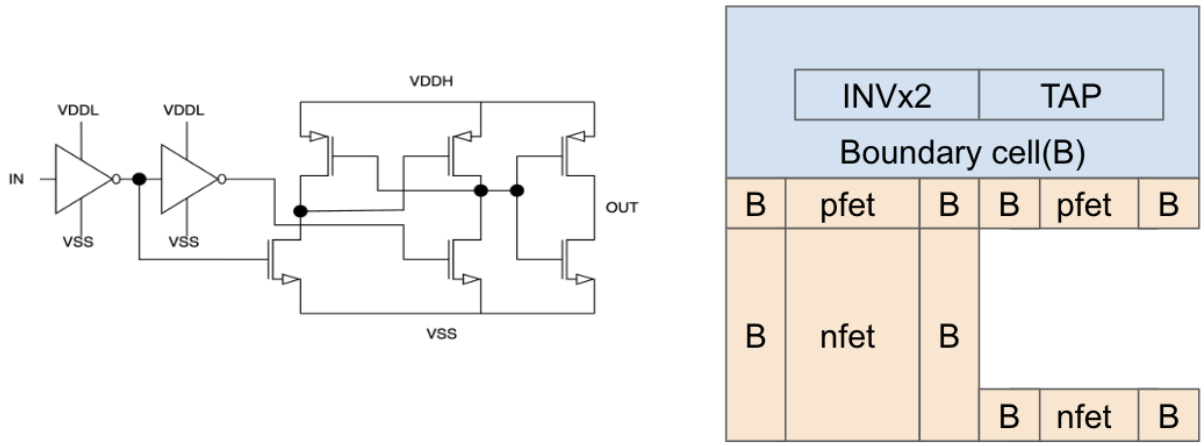


Figure 5.4: Level Shifter Schematic and Placement

flexibility of digital P&R tool, together with prebuilt analog devices, greatly reduced the iteration time from netlist to gds.

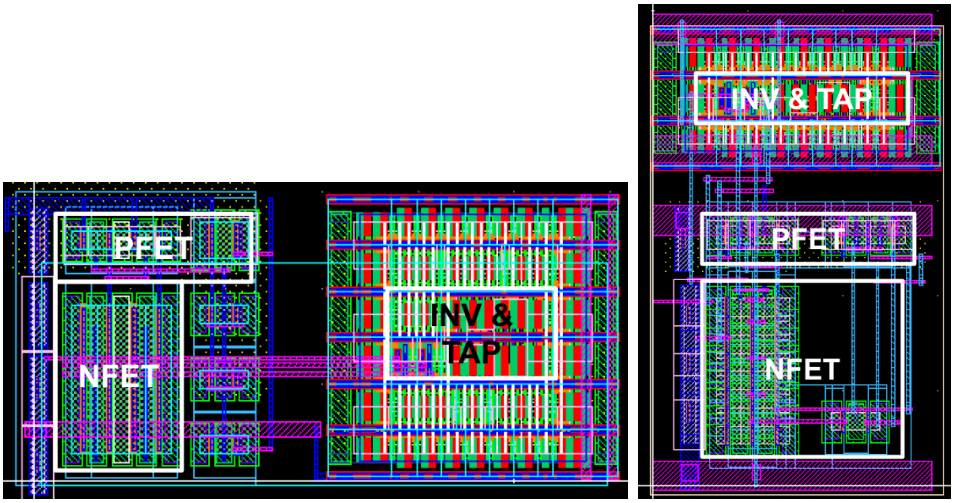
The pg grid follow the guidance of an experienced analog designer and post layout simulation verify the IR drop is acceptable without affecting the performance of the whole design.

5.4.1 Level Shifter

Level shifter(LS) in EIC is an essential small block to shift the low voltage level signal (0.7V) to high voltage level signal (1.7V).

The schematic and placement is shown in figure 5.4. It is composed with 2 core digital inveters and 6 EG devices that operates in high voltage domain. The placement is splited in two domains, minimium-length device/standard cells in blue area, and long-channel devices in the orange area, with boundary cells and dummy cells placed around the two areas.

The placement is compact with LEGO cells. Figure 5.5 shows the manual layout with LEGO automatic layout. To meet DRC rules, minimum distance between two areas need to be met. The manual layout has a size of 6.08 x 3.12 um and the LEGO layout has a size of 3.02 x 4.83 um, which is 25% less than the manual layout.



(a) (b)

Figure 5.5: LS Layout comparison of (a) manual layout with size of 6.08 x 3.12 (b) LEGO-based layout with size of 3.02 x 4.83

Figure 5.6 shows the post-layout simulation of LS. The simulation shows the functionality of level up the input from 0.72V to output 1.87V with delay of 700ps.

5.4.2 Comparator in High Voltage Domain

The comparator in EIC is operating under high voltage domain, thus, all the devices, including the two output inverters are using EG devices. The schematic and layout are shown in figure 5.7.

Symmetry constraint is an essential layout constraint in the comparator. The symmetric nets includes input differential signal VIN/VIP, interconnection ON/OP, FN/FP and output differential signal VON/VOP. Table 5.2 shows the mismatch of the parasitic C + CC extracted between the differential nets.

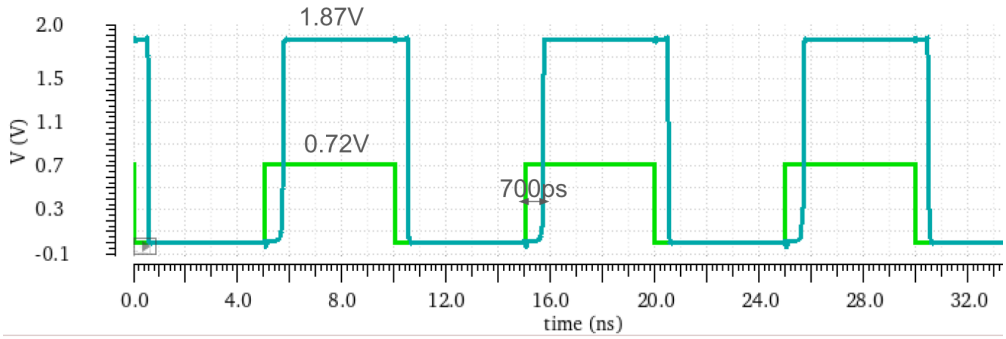
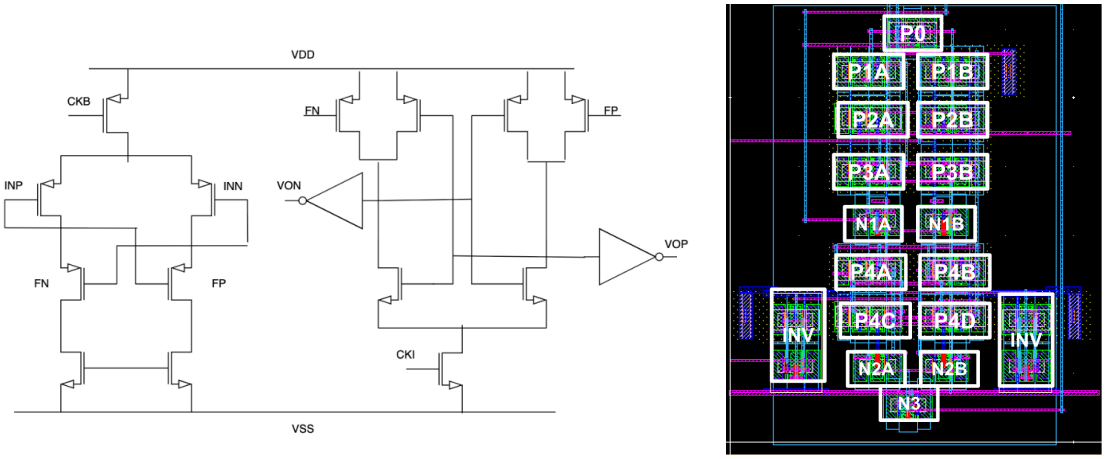


Figure 5.6: Level Shifter Simulation Results



(a) (b)

Figure 5.7: Comparator (a) schematic (b) layout

	iteration2
Sym net pair	C+CC mismatch
VIN/VIP	0
ON/OP	0.01
FN/FP	0.01
VON/VOP	0.06

Table 5.2: The parasitic mismatch of symmetry nets in comparator

5.4.3 Supply Independent Biasing Circuit

The bias circuit takes digital configuration in low voltage domain, level up the signal with LS from previous section and configure the bias circuit to provide bias voltage to other blocks. The floorplan is devided into 4 areas: (1) standard digital cells on the top left, (2) LS array on the bottom left, (3) extra EG devices on the top middle, and (4) bias circuit on the right. Matching constraints of 3 devices are given by the analog designer to match with the unit cell in TCL. As described in section 5.2.4, the PG ratio needs to fall under 3.8 to reduce IR drop. With parallel power strips, the pg length to width ratio is $14/4 = 3.5$, which meet the PG routing constraint.

5.4.4 TIA

The placements are designed to minimize the high speed signal routing length to reduce the parasitic and increase the signal speed. The mim cap are placed and overlap with other devices to reduce area. Due to the flexibility of floorplan and placement, 3 major floorplan are applied on the TIA top level block, as shown in figure 5.9. The first iteration has a floorplan constraint of height ¡ 55um. Major high speed signal paths are labeled in figure (a). The green circle shows the location of ubump. In the second iteration, the floorplan is limited to have a height less than 40um and high speed path is designed to be horizontally.

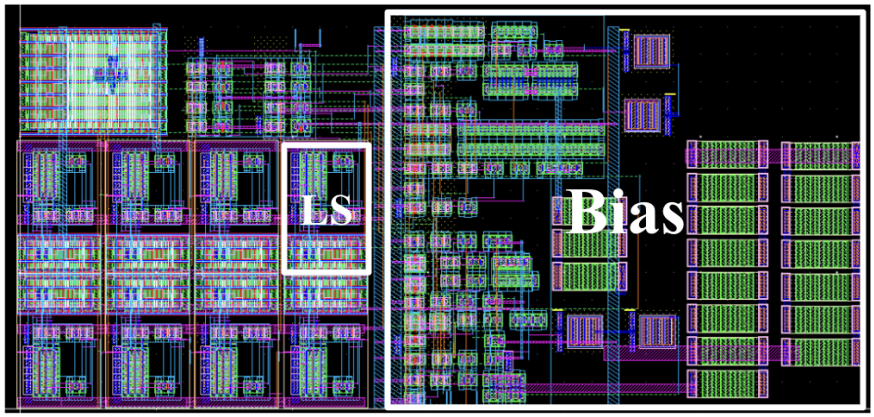


Figure 5.8: BIAS and LS layout

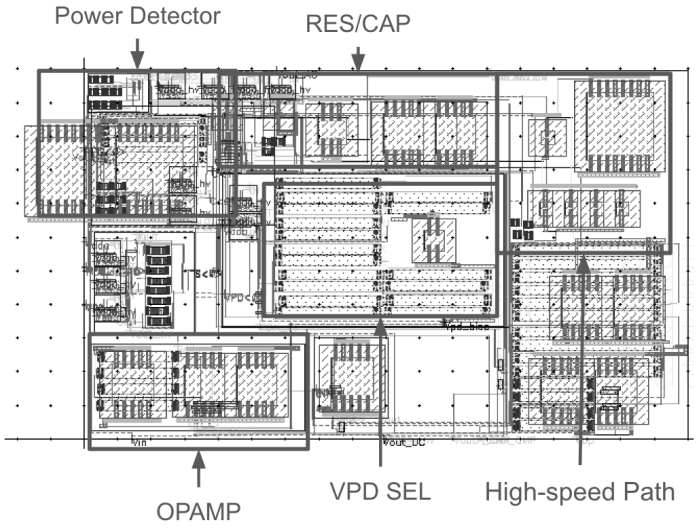
The last floorplan as a further detailed floorplan constraint of width less than 105um and output pins placed on the top of the floorplan. Each layout iteration, including placement modifications, 87 nets of routing, PG connection and DRC/LVS check, takes approximate 3 hours, which greatly increased the optimization efficiency.

5.4.5 C2C

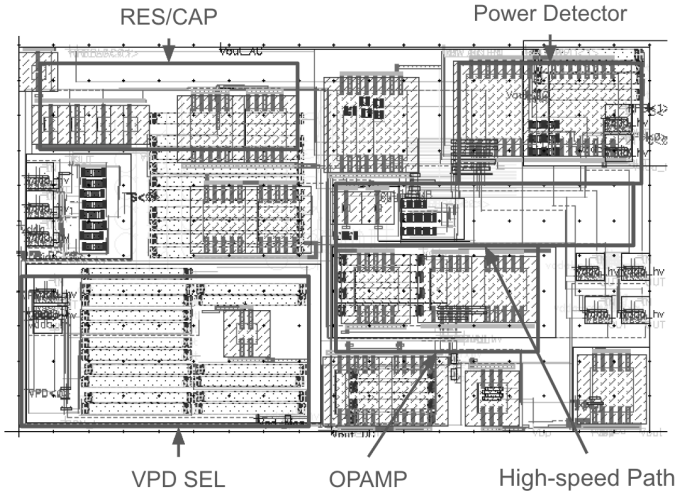
The C2C layout is shown in figure 5.11. The clock signal CKIN to CKOUT signal path left to right, thus the symmetry is on the x-asis. For best compactness, the resistors are placed on the top and bottom side with symmetry placement. All the devices are placed in the middle with filler, tap cells and boundary cells.

5.5 Summary

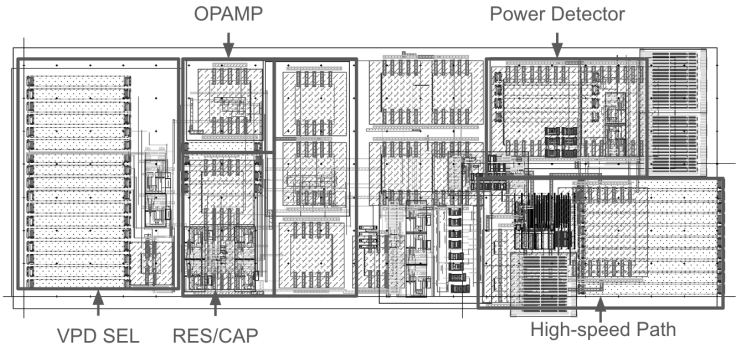
This chapter demonstrated the LEGO-cell design methodology and demonstrates its application to EIC. In contrast to the earlier AIB case study, which was restricted to minimum-length devices and a single power domain, the present work extends the methodology to accommodate high-voltage devices, long-channel transistors, diodes, and multiple power domains. These extensions illustrate the methodology's applicability to a broader class of



(a)



(b)



(c)

Figure 5.9: TIA layout, with annotated components over 3 iterations

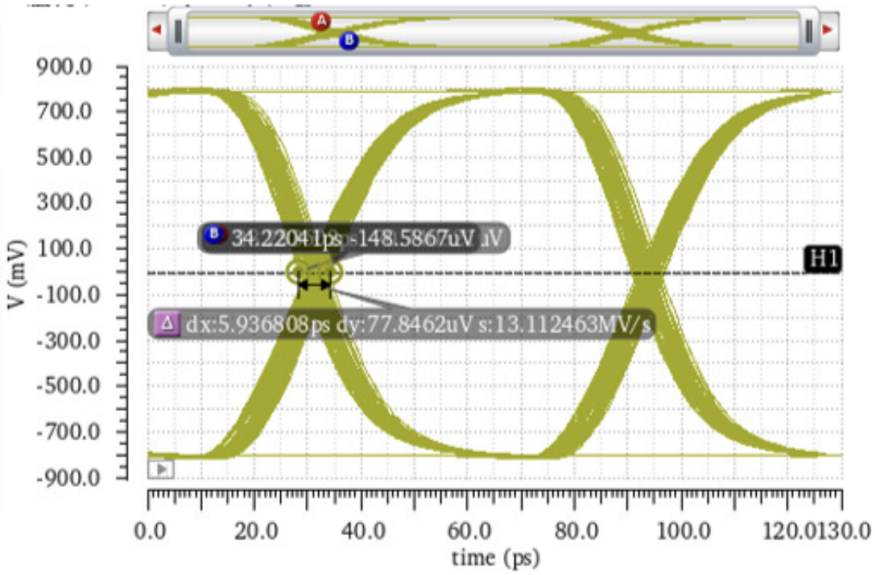


Figure 5.10: TIA TX Eye Diagram

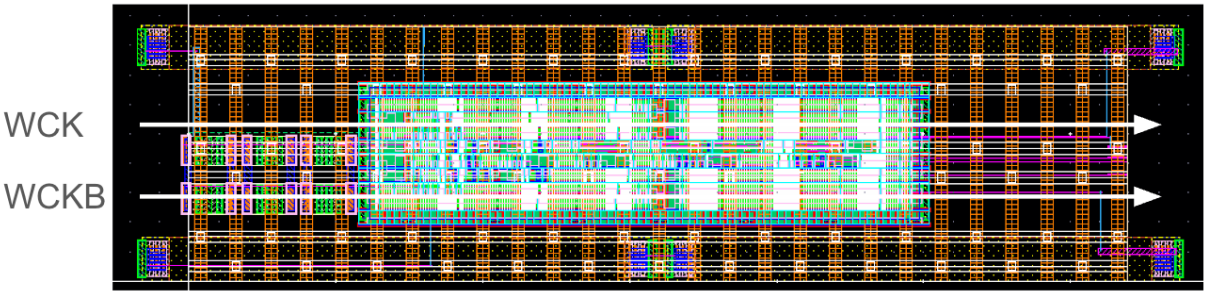


Figure 5.11: C2C layout, the WCK and WCKB paths are symmetry along the x axis

Matching Nets	C+CC Mismatch
CKIN/CKINB	0.01
a/b	0.09
c/d	0.03
e/f	0.02
ckout/ckoutb	0.35*

Table 5.3: C+CC mismatch of matching nets of C2C. *ckoub has extra load, thus the mismatch is relatively larger

	pre	post
Duty cycle	49.6	50.6
Power	135uW	371mW

Table 5.4: C2C pre and post simulation comparison

mixed-signal and multi-domain circuits. Experimental evaluations confirm the effectiveness of the approach, showing that it enables efficient post-layout optimization iterations and expedites the achievement of DRC and LVS closure.

Chapter 6

CONCLUSION AND FUTURE DIRECTIONS

6.1 Thesis Summary

This dissertation discusses the automated analog layout methodologies for mixed-signal SoC designs. In this theis, we focus on the methodology of generating high-performance AMS layout. Two main methodologies, template-based layout generator and analog standard cell-based digital approaches are discussed in the thesis.

The bottom-up template-based layout generator and batch-mode verification for optimization are reported. The templates generate layouts for the designated architecture with design and layout parameters. With the templates for each blocks, the verification flow runs the post-layout simulation and explore the solution map with the parameters. The generation time for each sub-block's layout is within minutes, which enables fast optimization. The transmitter design in TSMC 28nm shows the skew of matching data paths reduced by 60% .

Then, the LEGO methodology builds a process-portable analog standard cell library where each "LEGO cell" is a fully verified, fixed-dimension layout block with predefined pin positions. These cells can be tiled and abutted to form larger analog/mixed-signal systems. While less customizable at the per-device level, this approach offers exceptional reusability, predictable performance, and rapid integration.

Two interface designs, Advanced Interface Bus (AIB) and Electronic Integrated Circuit (EIC), were realized using the LEGO cell library:

- AIB: Constructed from LEGO cells for transmit/receive slices, clock trees, and termination circuits, enabling uniform lane structures and tight skew control.

- EIC: Assembled from LEGO cells for level shifters, comparators, bias circuits, power

detection, opamps, allows straightforward adaptation to varying data rates and configurations. In both cases, the modular, tile-based assembly reduced design turnaround while preserving critical matching and symmetry.

6.2 Future Directions

Although the LEGO methodology significantly accelerates the creation and integration of analog layouts, further improvements can be realized through analog cell timing modeling. By developing standardized timing characterization for each analog standard cell, designers could perform system-level timing analysis in a manner similar to digital flows. This would enable mixed-signal co-design, automated interface verification, and early performance validation without relying solely on post-layout simulations.

Integrating timing models into the LEGO cell library would bridge the gap between physical layout and high-level timing closure, making the methodology not only layout-accurate but also timing-aware from the earliest stages of design.

BIBLIOGRAPHY

[1] Z. Wang et al., "An Output Bandwidth Optimized 200-Gb/s PAM-4 100-Gb/s NRZ Transmitter With 5-Tap FFE in 28-nm CMOS," IEEE J. Solid-State Circuits, vol. 57, no. 1, pp. 21–31, Jan. 2022, doi: 10.1109/JSSC.2021.3109562.

[2] H. Chen, M. Liu, X. Tang, K. Zhu, N. Sun, and D. Z. Pan, "Challenges and opportunities toward fully automated analog layout design," J. Semicond., vol. 41, no. 11, p. 111407, Nov. 2020, doi: 10.1088/1674-4926/41/11/111407.

[3] Seo M J, Roh Y J, Chang D J, et al. A reusable code-based SAR ADC design with CDAC compiler and synthesizable analog building blocks. IEEE Trans Circuits Syst II, 2018, 65(12), 1904

[4] T. Zhou et al., "Abuttable Analog Cell Library and Automatic AMS Layout," in Proceedings of the 2025 International Symposium on Physical Design, Austin TX USA: ACM, Mar. 2025, pp. 191–199. doi: 10.1145/3698364.3705352.

[5] E. Chang et al., "BAG2: A process-portable framework for generator-based AMS circuit design," in 2018 IEEE Custom Integrated Circuits Conference (CICC), San Diego, CA: IEEE, Apr. 2018, pp. 1–8. doi: 10.1109/CICC.2018.8357061.

[6] T. Dhar et al., "ALIGN: A System for Automating Analog Layout." arXiv, Aug. 24, 2020. Accessed: Mar. 21, 2024. [Online]. Available: http://arxiv.org/abs/2008.10682

[7] H. Chen et al., "MAGICAL: An Open- Source Fully Automated Analog IC Layout System from Netlist to GDSII," IEEE Des. Test, vol. 38, no. 2, pp. 19–26, Apr. 2021, doi: 10.1109/MDAT.2020.3024153.

[8] "Low Power Double Data Rate 4 (LPDDR4) specification", https:// www.jedec.org.sites.default/files/docs/JESD209-4.pdf

[9] "UCIe Specification 1.0," https://uciexpress.org/team-3.

[10] Yongsuk Chio, Gyunam Jeon, and Yong-Bi Kim, "Transceiver design for LVSTL signal interface with a low power on-chip self calibration scheme," INTEGRATION, the VLSI journal, vol. 63, pp. 148-159, Sep. 2018..

[11] Chang-Kyo Lee et al., "Dual-Loop Two-Step ZQ Calibration for Dynamic Voltage-Frequency Scaling in LPDDR4 SDRAM," IEEE J. Solid-State Circuits, vol. 53, pp. 2906-2916, Oct. 2018.

[12] Hae-Kang Jung et al., "A 4.35Gb/s/pin LPDDR4 I/O Interface with Multi-VOH Level, Equalization Scheme, and Duty-Training Circuit for Mobile Applications" in Symp. VLSI Circuits Dig., Jun. 2015, pp. 184-185.

[13] F. Silveira, D. Flandre, and P. G. A. Jespers, "A g/sub m//I/sub D/ based methodology for the design of CMOS analog circuits and its application to the synthesis of a silicon-on-insulator micropower OTA," IEEE Journal of Solid-State Circuits, vol. 31, no. 9, pp. 1314–1319, Sep. 1996, doi: 10.1109/4.535416.

[14] J. P. A. van der Wagt, G. G. Chu, and C. L. Conrad, "A layout structure for matching many integrated resistors," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 51, no. 1, pp. 186–190, Jan. 2004, doi: 10.1109/TCSI.2003.821303.

[15] J. Pliva et al., "Design of a custom standard-cell library for mixed-signal applications in 28 nm CMOS," in 2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM), Donostia, San Sebastian, Spain: IEEE, May 2017, pp. 1–6. doi: 10.1109/ECMSM.2017.7945867.

[16] R. O. Topaloglu, "Design with FinFETs: Design rules, patterns, and variability," in 2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA: IEEE, Nov. 2013, pp. 569–571. doi: 10.1109/ICCAD.2013.6691172.

[17] G. Cijan and T. Tuma, "MODELING AND SIMULATION OF MOS TRANSISTOR MISMATCH," 2007.

[18] S. . Gondi and B. . Razavi, "Equalization and Clock and Data Recovery Techniques for 10-Gb/s CMOS Serial-Link Receivers," IEEE J. Solid-State Circuits, vol. 42, no. 9, pp. 1999–2011, Sep. 2007, doi: 10.1109/JSSC.2007.903076.

[19] David Kehlet, *Accelerating Innovation Through a Standard Chiplet Interface: The Advanced Interface Bus (AIB)*, Intel Programmable Solutions Group, White Paper, 2025. Available: https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/accelerating-innovation-through-aib-whitepaper.pdf

[20] Advance Interface Bus (AIB) Specification, avalible at https://github.com/chipsalliance/AIB-specification/blob/master/AIB_Specification%201_2.pdf

[21] J.-H. Kang et al., "A 24-Gb/s/Pin 8-Gb GDDR6 With a Half-Rate Daisy-Chain-Based Clocking Architecture and I/O Circuitry for Low-Noise Operation," IEEE Journal of Solid-State Circuits, vol. 57, no. 1, pp. 212–223, Jan. 2022, doi: 10.1109/JSSC.2021.3114205.

[22] J. Lee, "A 20-Gb/s Adaptive Equalizer in 0.13-$muhboxm$CMOS Technology," IEEE J. Solid-State Circuits, vol. 41, no. 9, pp. 2058–2066, Sep. 2006, doi: 10.1109/JSSC.2006.880629.