國立臺灣大學工學院工程科學及海洋工程學系

碩士論文

Department of Engineering Science and Ocean Engineering

College of Engineering

National Taiwan University

Master Thesis

MOS 電流模式邏輯應用於數位電路之設計

Mos Current-mode Logic Applied to Digital Design

李允中

Yun-Chung Lee

指導教授：陳昭宏 博士

Advisor: Jau-Horng Chen, Ph.D.

中華民國 112 年 07 月

July 2023

doi:10.6342/NTU202303723

誌謝

在完成論文的開心之際，要感謝很多人。首先我要感謝陳昭宏老師，在我碩士階段給予了許多的幫助，因為整段設計流程的建立比較繁瑣，老師都會伸出援手或是給予一些建議，讓我能夠掌握問題的方向，並嘗試去解決。若是沒有老師的協助，是不會有這篇論文產生的，感謝之情真的溢於言表。

感謝口試委員陳怡然老師、陳彥廷老師和張瑞益老師，非常感謝他們在口試過程中給予的意見和指教，能夠讓這篇論文更加完整，且衍生出許多未來還可以繼續研究的方向。

我也十分感謝高效率電路及系統實驗室的同學、學弟妹和學長，在我遇到一些問題比較失落的時候都會給我鼓勵，並且了解我的問題，從旁給我一些意見，使我的研究之路並不孤單。

最後感謝我的家人，雖然沒有辦法給予我學術上的指導，但是總是默默的在我的研究道路上支持我，一路上雖然經歷了許多困難，但是有家人的支持跟鼓勵，我便不害怕，有了更多勇氣和力量去面對困難。在這邊也非常感謝所有曾經幫助過我的人。

doi:10.6342/NTU202303723

中文摘要

本文提出了一種用於實現 Current Mode Logic (CML)電路的數位設計流程。CML 電路是全差動電路，不能以目前商用 Computer-Aided Design (CAD)工具直接進行設計，必須開發一種有別以往的設計方法，以便使用標準 CAD 工具自動化設計 CML 電路。首先設計 CML 標準元件，並生成商業 CAD 工具所需的元件庫，在單端流程進行合成、佈局及繞線，讓商業 CAD 工具將 CML 電路視為標準的 CMOS 邏輯電路，同時開發了一種自動將單端電路轉換為全差動 CML 電路的演算法。在驗證完設計流程後，使用 UMC 的 0.18 微米製程設計了一個 100MHz 的 sigma-delta modulation digital-to-analog converter (DAC)。本研究相較於以前的 CML 設計自動化流程不同，本研究成功的解決晶片內金屬層密度和大規模 Design Rule Check (DRC)問題，並以此設計流程下線了一個實體晶片。

關鍵字：電流模式邏輯、差動電路、數位設計流程、高速、電腦輔助設計

doi:10.6342/NTU202303723

ABSTRACT

In this paper a digital design flow for implementing Current Mode Logic (CML) circuits is presented. CML circuits are fully differential and are not directly supported by commercial Computer-Aided Design (CAD) tools. A special design methodology must be developed such that design of CML circuits can be automated using standard CAD tools. CML standard cells are first designed and the required libraries required for commercial CAD tools are then generated. Synthesis and place & route (P&R) in a single-ended domain that tricks commercial CAD tools as treating CML circuits as standard CMOS logic circuits. An algorithm that automatically converts single-ended circuits into fully differential CML circuits is developed. After validation of the design flow, a 100-MHz sigma-delta modulation Digital-to-analog Converter (DAC) was taped out using UMC's 0.18um process. Unlike previous works on CML design automation, this work ended with a real fabricated chip, where lots of effort were put in to resolve density and large scale Design Rule Check (DRC) problems.

Keywords: current mode logic, differential circuit, digital design flow, high-speed, computer-aided design.

doi:10.6342/NTU202303723

目錄

doi:10.6342/NTU202303723

doi:10.6342/NTU202303723

圖目錄

doi:10.6342/NTU202303723

doi:10.6342/NTU202303723

表目錄

doi:10.6342/NTU202303723

第1章　緒論

1.1　研究背景與動機

電流模式邏輯（CML）[1]是一種用於高速邏輯設計替代傳統 CMOS 邏輯的方法，廣泛應用於高速有線通訊中。 CML 電路的工作原理是通過完全差動結構將恆定電流源重新導向，並使用一對互補負載的電壓降作為輸出。由於電晶體始終工作於飽和區，從而降低其電源切換雜訊，這使得 CML 逐漸被應用於加密電路中。

傳統 CMOS 邏輯的自動化設計已經非常成熟，然而卻沒有針對 CML 電路的商業自動化設計工具，因而本研究旨在使用傳統 CMOS 邏輯的自動化設計工具進行 CML 的設計。因為經過驗證的傳統設計工具，其優勢在於進行設計的正確性，不需要重新開發用於 CML 電路的 CAD 工具，僅需設計 CML 和 CMOS 邏輯電路之間轉換的工具，從而減少 CML 設計流程開發的成本和困難度。在[2]中，重點是整合傳統 CAD 工具和自製工具，實際上並沒有實現晶片。在本研究中，解決了實際製作晶片所需的缺失部分，希望可以設計出一套完整的設計流程，可以讓使用者只需要編寫硬體語言，跑完設計流程後，就可以轉換成 CML 架構的電路。讓高速電路的設計者能夠大幅降低他們花在 CML 電路實現上的時間跟精力。

doi:10.6342/NTU202303723

1.2 電流模式邏輯(Current Mode Logic, CML)

本論文選擇電流模式邏輯(Current Mode Logic, CML)作為高速電路的設計方式，該電路有許多優良的特性，讓它被廣泛得應用於高速有線通訊中。

首先利用一個簡單的(N)MOS 的差動對來介紹 CML 電路的操作原理，其示意圖如圖 1.1 所示，可以看到它是全差動的電路架構，利用其結構中的差動對將固定電流源(Iss)根據輸入的訊號做切換，切換過去的電流乘上該輸出端電阻，會產生對應的壓降，導致該輸出端之於另外一端為低準位，而另外一端因為沒有該電流產生的壓降，所以輸出的電壓就是電壓源的值，也就成為高準位，因此可以得知單端輸出的電壓擺幅(Voltage swing)，可以藉由設計所供應的固定電流源和負載的電阻值做控制。

由於 CML 電路是差動的電路結構，所有電晶體始終操作在飽和區，且任一單端所需的驅動電壓並不像 CMOS 這麼大，所以只需要較小的電壓就可以將電流源做切換，因此它可以在較小的電壓擺幅下運作，這樣準位切換的時間就可以縮短，速度上就可以有所提升。

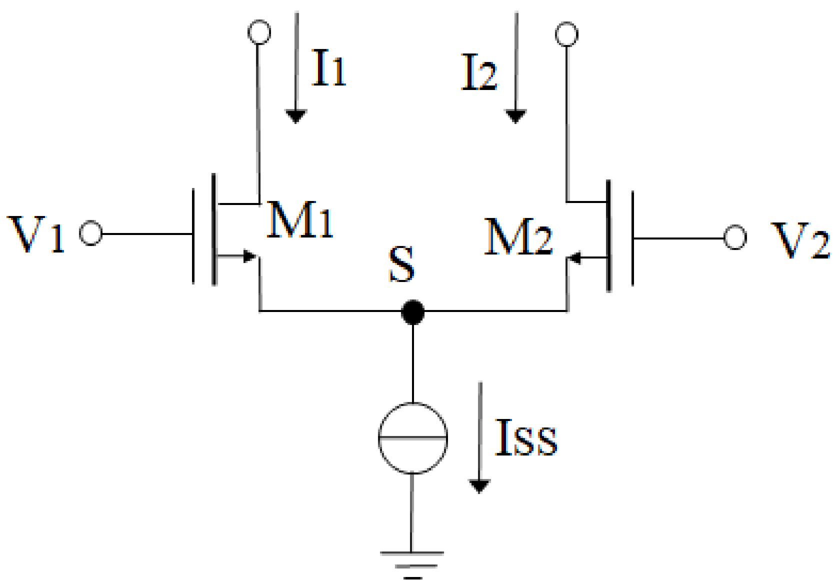


圖 1.1 CML 電路結構示意圖

doi:10.6342/NTU202303723

然後 CML 電路是藉由切換電流的方式得到輸出，所以切換的速度也明顯較 CMOS 邏輯快，且因為 CML 的電壓擺幅較小，所以消耗的動態功率也較低，基於上述原因，CML 成為許多人在高速電路設計上的選擇。

CML 在雜訊上的處理也有其優勢，因為它有一個固定的電流源，所以在 VDD 跟 VSS 之間固定有一個電流，當訊號更動時，電流產生變化的影響也就不大，因此對於電源和電路的運作上，所產生的雜訊也就少很多，訊號的傳遞也就更乾淨。不過也因為擁有固定的電流源，所以導致 CML 的功率跟操作頻率不像 CMOS 邏輯一樣成正比，因此在非高速運作的電路中，並無法發揮它的優勢。若是在非高速運行的時候，可以讓它維持在次臨限(subthreshold)如[3]，這樣可以使其運作維持極小的功率。

CML 電路在資安上面一樣有其發展性，在有加密的電路中，比起單端輸入輸出的 CMOS 邏輯，CML 要探到可用的訊號難度更高。因為 CML 本身是雙端的路徑，而且訊號進來的時間還不是同步，會在差動結構中做切換，因此對於完整訊號的掌握度也就更難，所以很適合運用在需要加密的電路。

doi:10.6342/NTU202303723

第 2 章 CML 設計方法

為了要能夠建立出一套完整的 CML 設計流程，首先介紹現今已經有的 CML 設計方式，並提出本文想要進行的改良方向，在已有的基礎上，補足現今設計方法的不足，進而提出改良的設計流程。

2.1 高速 CML 自動化設計方法

本文的目標是能夠設計出一套能應用在高速電路的 CML 設計流程，主要參考[2]中的設計流程，在該書中，提出了使用傳統 CMOS 邏輯設計工具進行自動化設計的方法，如圖 2.1。

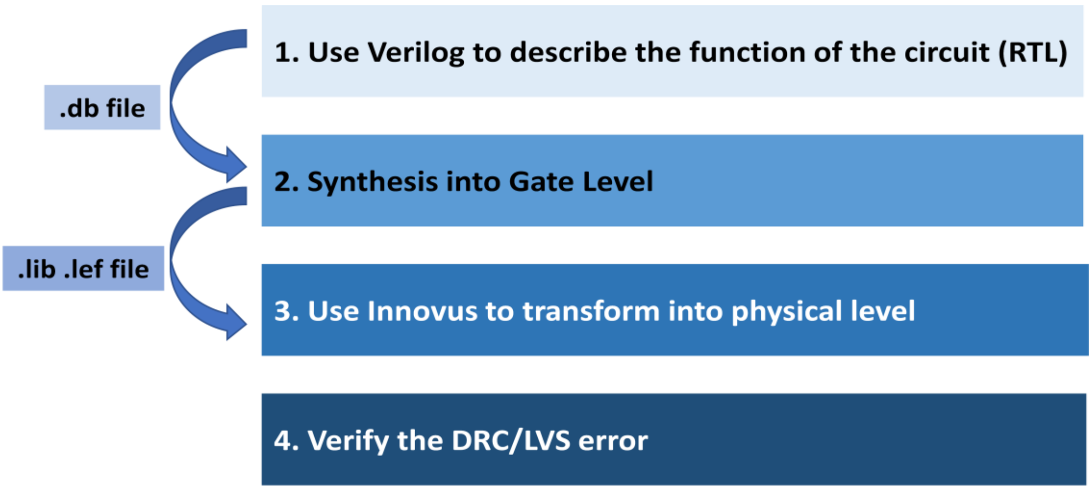


圖 2.1 CMOS 邏輯數位設計流程

其重點著重於整合傳統 CAD 工具和自製工具，因為若是要開發新的自動化設計工具，需要花費大量的時間跟成本，且無法確定該工具是否準確無誤，因此流程上主要是和 CMOS 設計流程相同，主要差別在標準元件的建立與差動結構的轉換方式，該流程主要分成三個階段，圖 2.2 為該設計之流程圖。

doi:10.6342/NTU202303723

圖 2.2 參考自[2]第 118 頁。

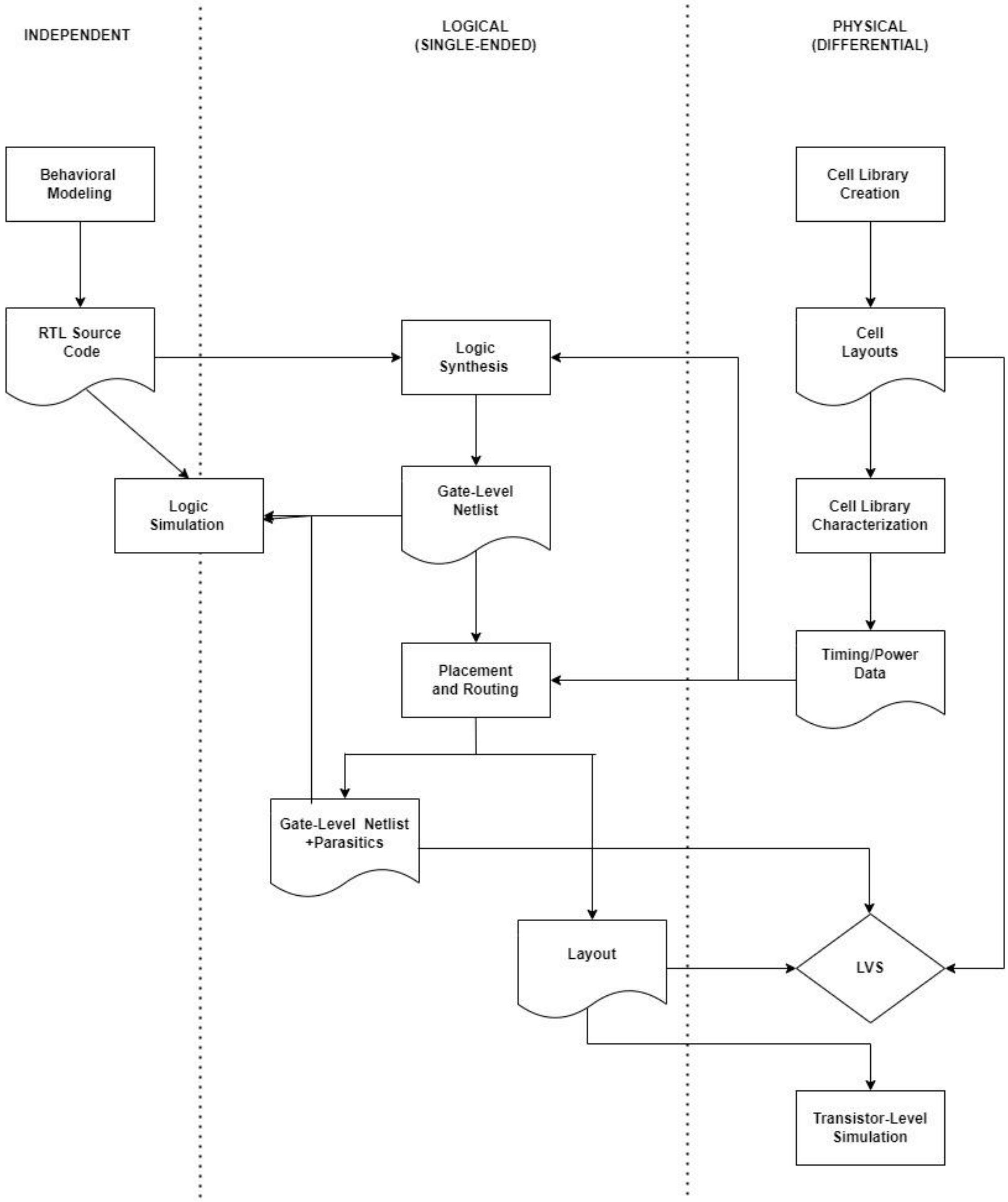


圖 2.2 高速 CML 自動化設計流程圖

該書雖然成功利用該設計流程轉換出高速 CML 電路，然而其內容並未清楚說明流程所使用的設計工具和實作方式，進行合成、佈局及繞線的步驟只是簡略帶過，且通篇未闡述其轉換成差動結構的方式，所以在流程的實現上就有其難度。再者，該流程在佈局及繞線的工具上是使用 Cadence Encounter，但是現今普遍使用的皆是 Cadence Innovus，所以在工具上的切換也有些許需要克服的問題。

doi:10.6342/NTU202303723

該書中雖然有成功地轉換出 CML 電路的範例，但是晶片規模並不大，推估可能是因為它並未進行 Clock tree synthesis(CTS)，所以沒有辦法設計出較大型的高速電路，且該流程並未實際下線實體晶片。若是要同時達到既高速又可以下線出具有一定規模的晶片，則如同[4][5]的設計方式，必須使用非自動方式去佈局和設計電路圖，雖然可以達到高速的電路表現，可是所花費的電路設計時間與成本也相對較高。因此我認為該流程必定存在實踐上的問題，否則應早被使用在高速 CML 電路的實現中，所以解決該流程的實踐問題，則為本研究的改良方向。

2.2 低功耗 CML 自動化設計方法

在[3]中主要是利用近臨界(near-threshold)的操作原理，應用在其標準元件的設計上，並建立一套建立在傳統自動化傳統 CAD 工具上，可產生低功耗 CML 電路的設計方式。該流程主要分成三個部分，圖 2.3 為該設計之流程圖。

從該流程圖可看出其基本的自動化設計流程，和高速的設計方式雷同，但有別於[2]的部分，該流程對於設計流程的細節描述較為詳盡。在不同的階段都有將標註其使用的 CAD 工具，並且較具體的描述在流程中的細部程序。像是在建立元件庫(library)的過程，就有說明特徵萃取(characterization)使用 Synopsys Siliconsmart；在合成過程時使用 Cadence RTL Compiler，並詳述單端傳換成差動結構的步驟，最後的實體設計則使用 Cadence SOC Encounter。

該流程雖有實際設計下線晶片，且電路為較具規模的微控器(microcontroller)，然而其設計方式為了要達到低功耗的目的，只適用於實現較為低速的電路，和本研究欲達成高速 CML 電路的設計目標不符合，但是其在流程中所描述的流程細節和自動化工具的選用，也成為本研究後續改良 CML 設計流程的參考方向。

doi:10.6342/NTU202303723

圖 2.3 參考自[3]第 89 頁。

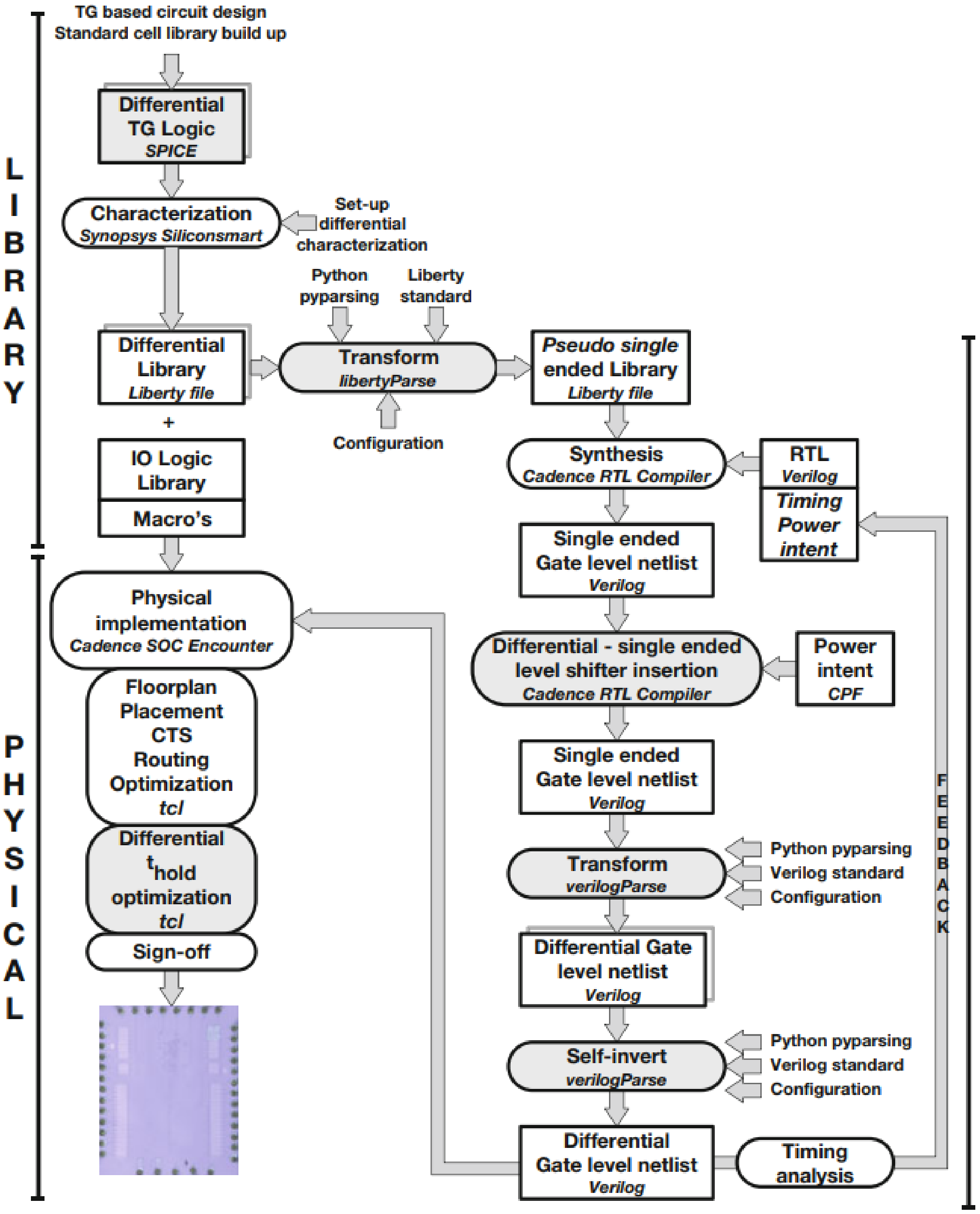


圖 2.3 低功耗 CML 自動化設計流程圖

doi:10.6342/NTU202303723

2.3 設計理念

有鑒於上述的設計方法，可以歸納出現今 CML 設計電路的方式，主要往三大優勢發展：高速、設計時間短和晶片可實現性，然而上述的設計方式，基本上都只能兼具圖 2.4 的其中兩項優勢，無法同時並存。

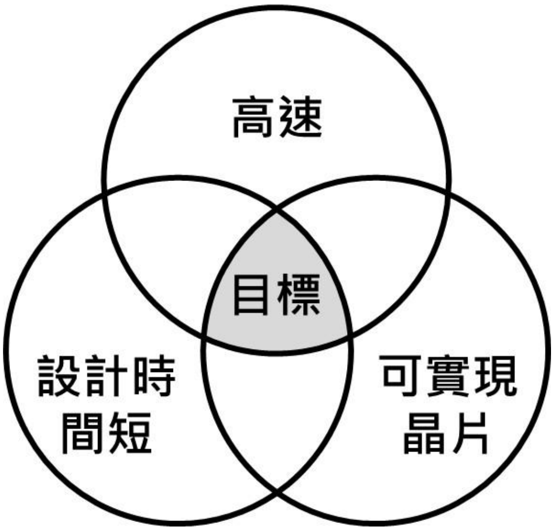


圖 2.4 CML 設計電路優勢關係圖

因此，本研究決定以[2]的高速 CML 設計架構為基礎，並參考[3]所提及的自動化設計工具的選用，實現其設計流程，待成功轉換出 CML 電路後，再解決實際製作晶片缺失的部分，以進行晶片的下線。本研究希望可以設計出一套完整的設計流程，從而讓使用者只需編寫硬體程式語言，跑完設計流程後，便能轉換為可實現的 CML 架構電路。高速電路的設計者能夠大幅降低其花在 CML 電路實現上的時間跟精力，使該設計流程同時發揮出高速、設計時間短和可實現晶片的優勢。

doi:10.6342/NTU202303723

第 3 章 設計流程之改良及實現

商業 CAD 工具在 CMOS 邏輯和 CML 之間的主要區別在於單端輸入/輸出和完全差動輸入/輸出。從元件庫(library)特徵萃取(characterization)開始，商業工具如 Synopsys Siliconsmart 和 Cadence Liberate 只支援單端邏輯元件。合成工具如 Synopsys Design Compiler 和 Cadence GENUS 也同樣僅支援符合某些形式的單端邏輯元件。佈局與繞線工具如 Synopsys IC Compiler 和 Cadence Innovus 可以接受全差動的電路結構，並將所有導線視為單端，但這會降低 CML 電路作為差動電路的優勢。在本研究中，將 CMOS 邏輯替換為 CML，使用了三個主要步驟，即元件庫準備、單端流程和差動流程。設計流程如圖 3.1 所示。

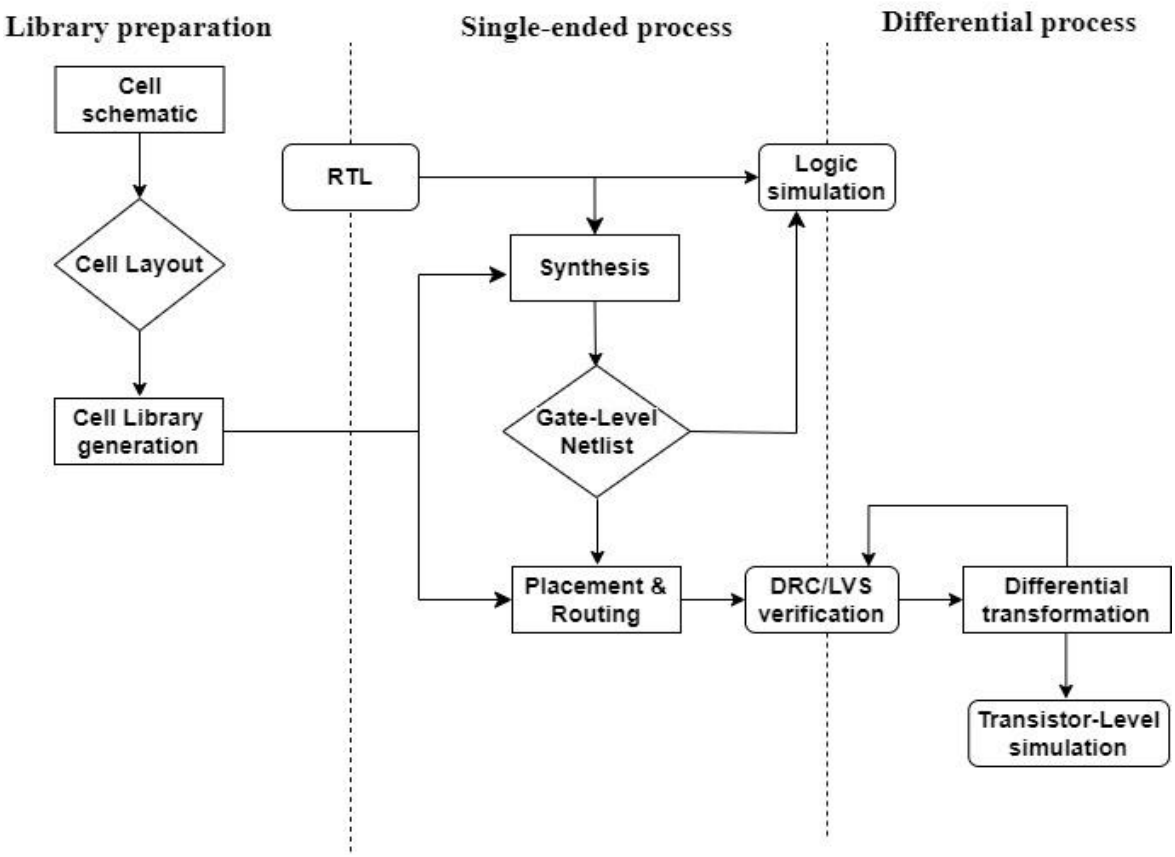


圖 3.1 設計總流程圖

doi:10.6342/NTU202303723

3.1 元件庫準備

數位設計的基礎是標準元件庫。對於 CML 電路而言，標準元件庫並不容易獲得，然而電路設計的最終性能將在很大程度上取決於元件庫，因而標準元件庫有其進行設計的必要性。合成之前，需要先行設計標準元件，而後對每個元件進行特徵萃取，最後將元件庫生成合成工具所需的時序庫(timing library)。

3.1.1 標準元件原理圖設計和佈局

本研究使用的標準元件，參考[4][5][6]中所示的電路圖(schematic)設計。元件採用 UMC 的 0.18um CMOS 製程，製程中使用了 6 層金屬層(ME)。標準元件只利用了 ME1，以便 ME2 ~ ME6 用於繞線。本研究共建立了 10 個標準元件與相關反轉後元件，圖 3.2 以常用邏輯閘的電路圖作為範例。

doi:10.6342/NTU202303723

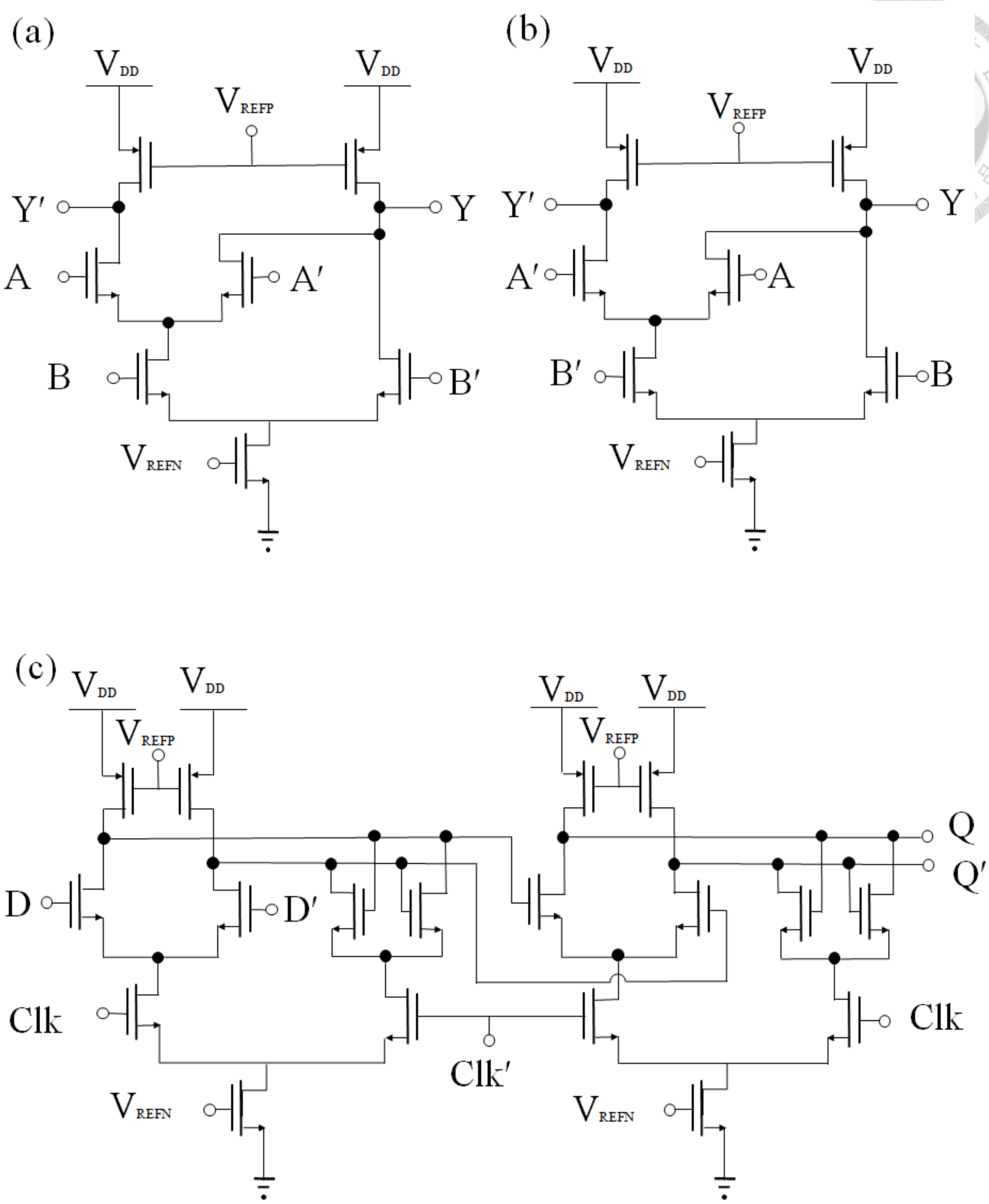


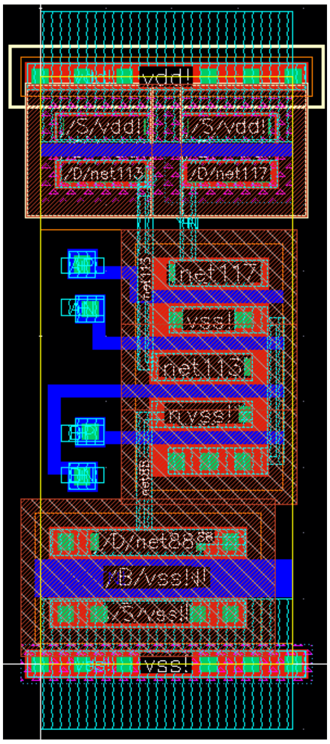
圖 3.2 常用 CML 邏輯閘的電路圖

(a) AND 邏輯閘 (b) OR 邏輯閘 (c) D Flipflop 邏輯閘

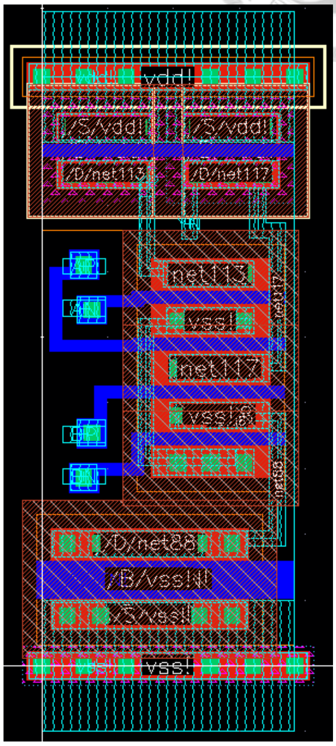
doi:10.6342/NTU202303723

完成標準元件的電路圖設計後，接著需要進行佈局設計，圖 3.3 以常用邏輯閘的佈局設計圖作為範例。

(a)



(b)



(c)

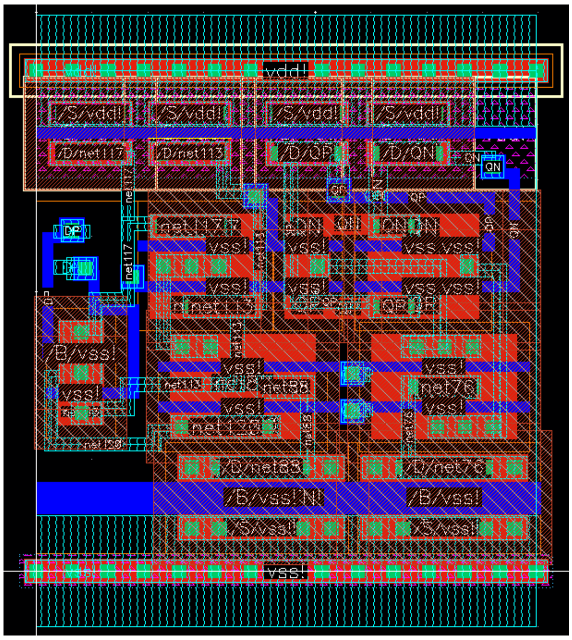


圖 3.3 常用 CML 邏輯閘的佈局圖

(a) AND 邏輯閘 (b) OR 邏輯閘 (c) D Flipflop 邏輯閘

doi:10.6342/NTU202303723

建立完基礎的標準元件後，發現後續 CTS 的緩衝器(buffer)具有推力問題：電路較大時緩衝器推力不夠，因此必須將其推力增強。我參照[2]裡面的方式，將其尺寸橫向拓寬，已增強其推力，如圖 3.4 所示：

(a) (b) (c)

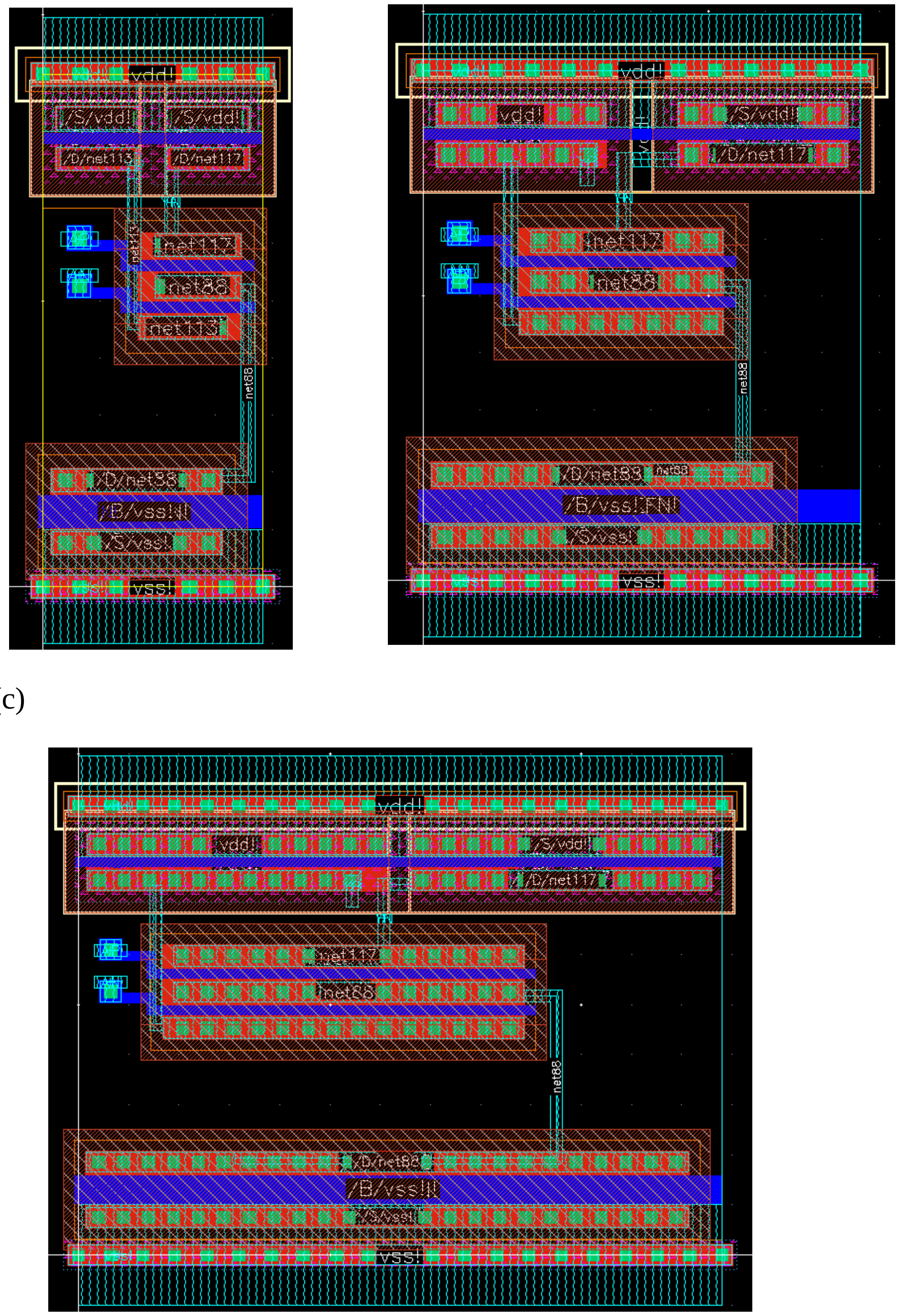


圖 3.4 不同推力緩衝器的佈局。

(a) BUFX1 (b) BUFX2 (c) BUFX4

doi:10.6342/NTU202303723

本工作中用於 CML 元件庫生成過程中創建的標準元件如表 3.1 所示：

表 3.1 CML 標準元件總表

Gate(Input number)	Drive strengths
INV	1 , 2 , 4 , 8
BUF	1 , 2 , 4 , 8
AND	1
NAND	1
OR	1
NOR	1
XOR	1
XNOR	1
DFF	1
DL	1

3.1.2 元件庫生成

流程如圖 3.5 所示，此步驟基本上是將生成出來的寄生萃取(PEX)，經由格式上的調整後，交給 Synopsys Siliconsmart 去產生對應的時序庫，以進行後續合成和佈局的流程。

doi:10.6342/NTU202303723

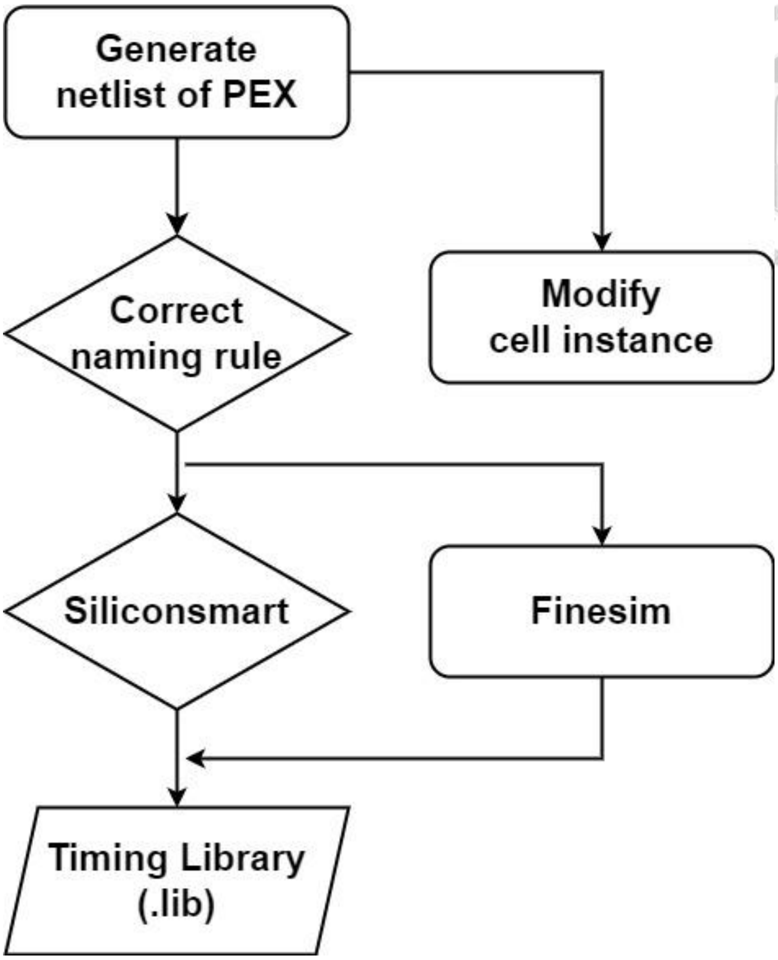


圖 3.5 時序庫的生成流程

於此步驟，Synopsys Siliconsmart 產生出的時序庫細節將事先匯出成一個資料表，以便在後續的流程中可清楚時序庫中的標準元件資料，進而去進行後續設計。表 3.2 以 AND 邏輯閘的資料表作為參考：

doi:10.6342/NTU202303723

表 3.2 AND 邏輯閘資料表

PIN CAPACITANCE (pf)

Pin	Type	Capacitance (pf)
A	input	0.0041
B	input	0.0047

LEAKAGE POWER

When Condition	Related Supply Pin	Power (nW)
default	VDD	352914.7320

DELAY AND OUTPUT TRANSITION TIME

Input Pin	Output	When Condition	Tin (ns)	Out Load (pf)	Delay (ns)	Tout (ns)
A(LH)	Y(LH)	default	1.0492	0.2098	0.4942	0.9696
A(HL)	Y(HL)	default	1.0492	0.2098	0.6412	1.4583
B(HL)	Y(HL)	default	1.0492	0.2098	0.5951	1.3928
B(LH)	Y(LH)	default	1.0492	0.2098	0.5660	0.9898

DYNAMIC ENERGY

Input Pin	Related Supply Pin	When Condition	Tin (ns)	Output	Out Load (pf)	Energy (pJ)
A	none	default	1.0492	Y(LH)	0.2098	0.1345
B	none	default	1.0492	Y(HL)	0.2098	-0.2135
A	none	default	1.0492	Y(HL)	0.2098	-0.2126
B	none	default	1.0492	Y(LH)	0.2098	0.1270
B(HL)	none	default	1.0492	n/a	n/a	-0.0025
A(LH)	none	default	1.0492	n/a	n/a	0.0014
B(LH)	none	default	1.0492	n/a	n/a	-0.0036
A(HL)	none	default	1.0492	n/a	n/a	-0.0011

doi:10.6342/NTU202303723

上述步驟產生的檔案是用於佈局及繞線的工具，因此需另外經由 Synopsys Library Compiler 轉換成資料庫(.db)檔案，供給 Synopsys Design Compiler 做合成，步驟如表 3.3：

表 3.3 db 生成步驟

將 lib 轉換成 db 的步驟	(1)Open the Library compiler. (2)Load the .lib file from the previous step (read_lib) (3)Convert the file into a .db format (write_lib)

此外，另需從佈局中生成一個資料庫交換格式（Library Exchange Format, LEF）文件用於佈局與繞線，該文件提供了進行元件間繞線的針腳位置和金屬層的資訊。雖然 CML 元件的設計是差動的，但仍需生成單端資料庫以便商業設計工具使用，因為商業設計工具對應的設計電路對象皆為單端結構。

再者，需注意電容表的部分。因本研究中並未生成佈局繞線所需使用的電容表，因此我參考[2]裡對於建立線間電容的計算公式去估算電容值，大約是單端線的三分之一，並將這樣的電容值設定在 LEF 文件裡，讓工具去做讀取。

3.2 單端流程

商業 CAD 工具只支援具有單端針腳的傳統 CMOS 邏輯元件，如[7]所述。本研究的目標之一是盡可能利用傳統的數位設計流程操作，單端流程基本上與[8]中描述的傳統 CMOS 數位設計流程相同。首先，使用 Synopsys Design Compiler 對 Register transfer level (RTL)進行合成，生成邏輯閘層級 netlist (gate-level netlist)，並使用 Cadence Innovus 對上述的 netlist 進行佈局與繞線。

doi:10.6342/NTU202303723

3.2.1 合成

合成的步驟基本上與數位設計流程相同，在進行合成前，需先將時序庫(.lib)檔案經由 Library compiler 轉換成(.db)，並供給 Synopsys Design Compiler 使用，

較需注意的部分是電路的觸發訊號，只能是同步訊號，否則在後續佈局和繞線的過程中會在 CTS 的流程裡衍生出錯誤，

3.2.2 佈局及繞線

此階段遇到較多問題，為了使電路功能不受影響，佈局及繞線的每個階段皆花費許多時間解決 DRC 問題。圖 3.6 是佈局及繞線的基本流程，在這些流程中都有做些許調整，以讓整個設計流程可以正確運行。

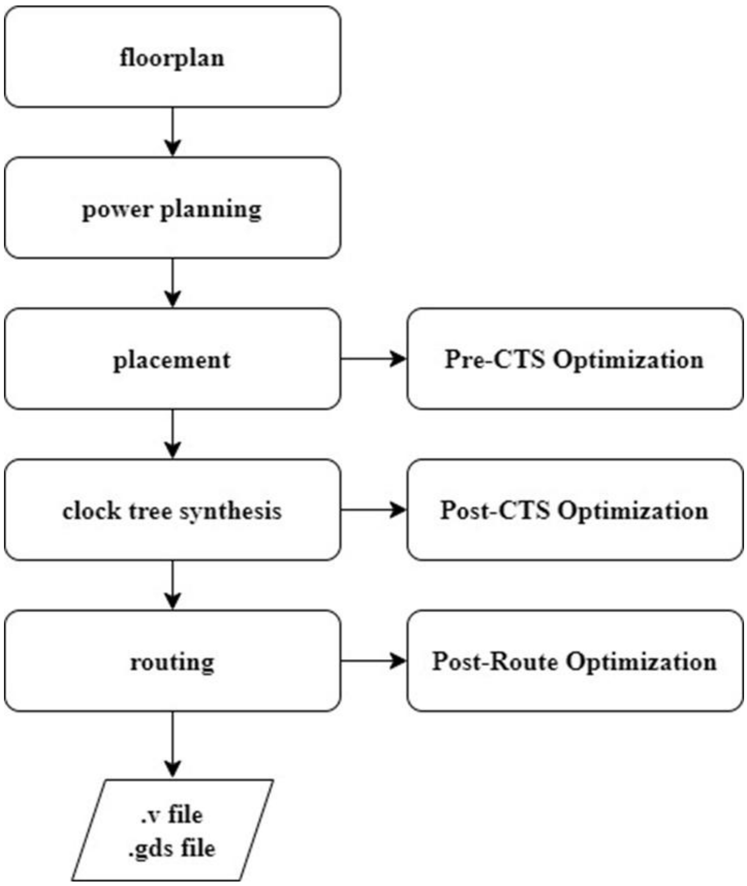


圖 3.6 佈局及繞線流程

doi:10.6342/NTU202303723

匯入合成後的 Verilog 文件和 Synopsys Design Constraints (SDC) 文件後，需將先前在元件庫生成出的時序庫(.lib)和 LEF，匯入 Cadence Innovus 裡進行佈局。

接著進行佈局規劃(floorplan)，此步驟裡需要注意晶片大小的決定。使用 CML 架構設計的晶片主要是希望能運行在高速電路中，因而基本上較高速的訊號例如時脈的輸入端，會希望能夠固定設立在晶片的上下方向，以便長時脈樹(clock tree)時較為順暢。因此，決定晶片的長寬比的時建議使用長方形，使晶片具有較高的面積使用率，至於晶片的利用率則不建議設太高，因為之後還需轉換成雙端，預留一些空間會較為保險。

在電源規劃(power planning)階段，因為 CML 結構比一般 CMOS 結構多了兩個偏壓(bias)的輸入，因而在加入電源環(power ring)時，除了 VDD 跟 VSS 之外，需多加入 VREFP 跟 VREFN 的偏壓，最後生成四條電源環在電路四周。

完成好電源環的設置後，接著要加入電源供應走線(power stripe)，在設置之前，應先考量電路模擬出來的電流值再進行走線設計。因為 CML 的電流是恆定的，在電流值的確認上並不困難，且在一開始電源環的設定就已經使用了所有的金屬層，原則上所有金屬層皆可做為電源供應走線，然而上層的金屬層較為建議使用，越上層的金屬層單位可承受的電流越大，因而使用上層金屬能以較少的走線達成分配電流的目的。本研究皆盡可能以最上層的金屬層(ME6)作為電源供應走線的設置，以增加後續繞線的空間使用。

doi:10.6342/NTU202303723

在輸入、輸出針腳的設置上，並不採用自動放置的方式，而是手動放置針腳位置。原因是 CML 電路是差動對的設計，希望它們路徑的時序比較接近，所以在針腳的設置上需要設計過，此點對於高速的電路是有其必要性的。

佈局(placement)階段與數位流程沒有太大差別，可直接交由 Innovus 處理，再以 Pre-CTS Optimization 確認進行 CTS 之前時序的分析是沒有問題的。完成佈局後就進到 CTS 的階段。CTS 的運行原理是當時脈間的時序相差過大時，需要補上緩衝器，以使時脈到達每個元件的時間差不多。前面提及時脈緩衝器(clock buffer)的推力不夠，因而本研究設計了推力 X1、X2、X4 和 X8 的緩衝器跟反相器(inverter)，在執行 CTS 之前，需要宣告這些緩衝器和反相器可使用於 CTS，再進行 CTS 的程序。宣告這些緩衝器跟反相器的程式碼如表 3.4。

表 3.4 宣告時脈緩衝器跟反相器的程式碼

加入不同緩衝器尺寸的程式碼	set_ccopt_property buffer_cells {I_BUFX1 I_BUFX2 I_BUFX4 I_BUFX8} set_ccopt_property inverter_cells {I_INVX1 I_INVX2 I_INVX4 I_INVX8}

接著進行 Post-CTS Optimization 以確認在完成 CTS 後時序的分析是沒有問題的，最後進行繞線步驟。本文的標準元件使用的是 ME1，因此在繞線的過程中需要限制繞線僅能使用 ME2 到 ME6 的金屬層以避免產生 DRC 錯誤。

doi:10.6342/NTU202303723

此外，較為重要的限制是，如果在同一層發生轉向的繞線，經過差動轉換後會產生短路，如示意圖 3.7 中紅點所演示的，為了避免此問題產生，繞線的當下會多下限制，讓每一層金屬層都只走同一方向來避免差動轉換後的電路問題。

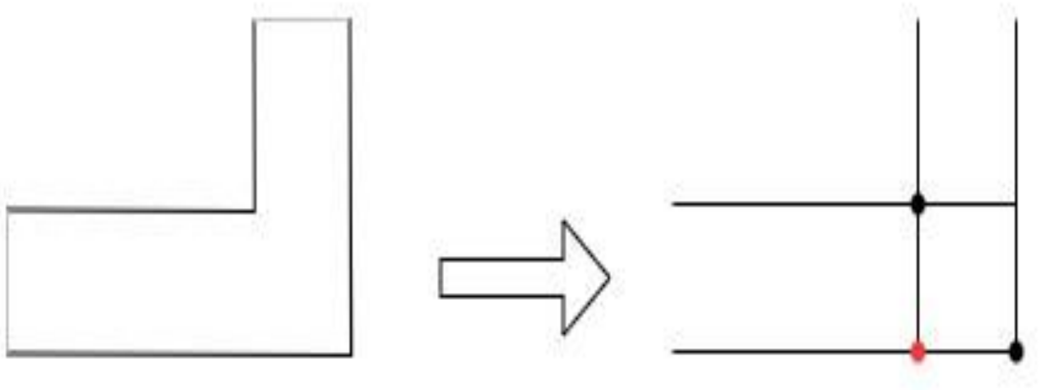


圖 3.7 繞線轉換錯誤示意圖

經過差動轉換後的繞線以 6 位元解碼器為例，可從圖 3.8 看到同一層的金屬層都是同一方向，並不會發生前述轉向的問題，若需進行轉向，則會切換成另一金屬層去作切換。接著進行 Post-route Optimization，確認在進行繞線之後，時序的分析是沒有問題的。

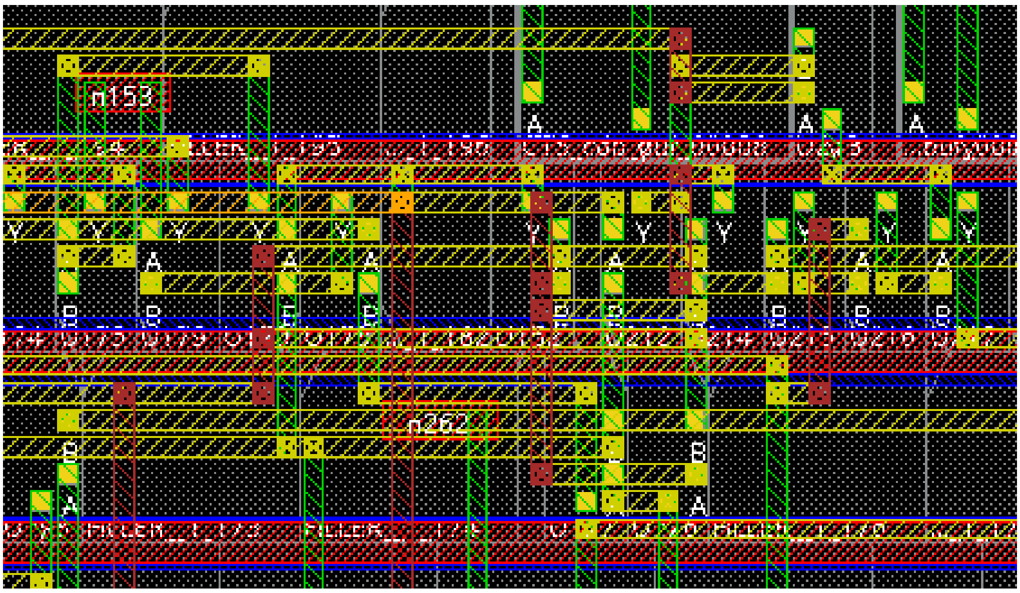


圖 3.8 6 位元解碼器單向繞線佈局圖

doi:10.6342/NTU202303723

完成佈局與繞線的流程後，在 Innovus 中可進行初步的 DRC 和 Layout Versus Schematic (LVS)檢驗。然而，在 Innovus 中使用的元件不是真正的 CML 元件，而是用於讓 Innovus 接受的單端元件，且元件之間的繞線是如[2]中所述的單端粗線，並未轉換成雙端的細線。最後將單端佈局導出為 Graphical Data System (GDS)文件，並在 Cadence Virtuoso 中進一步處理。同時生成了一個經過繞線的單端 Verilog netlist，其中包括時序樹合成後附加的時脈緩衝器，可於 Cadence Virtuoso Schematic Entry 中使用。

3.3 差動流程

CML 的差動流程將單端佈局和 netlist 轉換為差動佈局和對應的 netlist，並通過傳統的全客製化(full-custom)設計流程進一步驗證。首先，將生成的佈局 GDS 文件導入到 Cadence Virtuoso 中，需要轉換成真正的差動 CML 元件替換單端 CML 元件，並且如[9]中所述，需要將單端繞線的線路替換為差動線路。與單端元件不同，佈局流程中翻轉的差動元件可能導致繞線線路的反轉，因此，必須替換元件以確保邏輯正確性。

為了在 Cadence Virtuoso 中生成電路圖，導入了繞線後的單端 post-route netlist。與導入的佈局一樣，會將單端元件替換為差動元件，而之前提到的為確保邏輯正確性而替換的翻轉元件也在電路圖中進行了替換，表 3.5 以 AND 邏輯閘來舉例：

doi:10.6342/NTU202303723

表 3.5 AND 邏輯閘反轉後的邏輯真值表

A、B 皆不反轉		
A	B	output
0	0	0
0	1	0
1	0	0
1	1	1

A 反轉、B 不反轉		
!A	B	Output
0	0	0
0	1	1
1	0	0
1	1	0

A 不反轉、B 反轉		
A	!B	output
0	0	0
0	1	0
1	0	1
1	1	0

A、B 皆反轉		
!A	!B	Output
0	0	1
0	1	0
1	0	0
1	1	0

從中可看到 AND 邏輯閘會因為在繞線中發生翻轉，導致邏輯也產生反轉。因為每一個輸入都可能出現反轉、不反轉兩種結果，所以若有 n 個輸入(此例 n=2)，則會產生 2 的 n 次方種結果，進而導致 AND 邏輯閘所產生的輸出有四種可能性。但若仔細觀察，可發現當 A、B 皆反轉時，其實就跟 NOR 邏輯閘的輸出是一模一樣的，便可將 A、B 皆反轉的 AND 邏輯用 NOR 邏輯代替，歸納完所有可能性後，便能將轉換對應的結果寫入演算法裡，如表 3.6 所示。

表 3.6 AND 邏輯閘轉換後對應結果

AND2X1		
A 反轉	B 反轉	替代邏輯閘
X	X	AND2X1
V	X	AND2BX1
X	V	AND2CX1
V	V	NOR2X1

doi:10.6342/NTU202303723

為了方便在 Cadence Virtuoso 中處理佈局圖和電路圖，使用 Cadence SKILL 語言編寫了程式碼，SKILL 程式碼是供 virtuoso 執行的程式碼[19]，SKILL 能夠直接讓 virtuoso 執行自定義的操作方式，可以控制編程，較容易去了解流程出現錯誤的部分。還可以建立使用介面(Gui)，使用者操作該流程時更為方便。且為避免減少工具間轉換可能產生的錯誤，使用 SKILL 進行演算法的編寫，讓轉換過程都在 virtuoso 內進行。

理論上，通過 Cadence Innovus 導出對應的 netlist 和佈局圖，在執行 SKILL 程式碼後，如果演算法沒有錯誤，應該呈現正確的對應結果。接著，使用 Mentor Graphics Calibre 對差動佈局和電路圖進行 DRC 和 LVS 的驗證。此設計流程中進行了多次迭代，讓使用於 Cadence Innovus 的 LEF 文件能夠於此步驟中產生沒有 DRC 錯誤的差動佈局圖。完成這一步的差動流程後，便可產生出佈局圖以下線、生產晶片，使用 Cadence SKILL 實現差動轉換演算法的詳細步驟如圖 3.9 所示。

doi:10.6342/NTU202303723

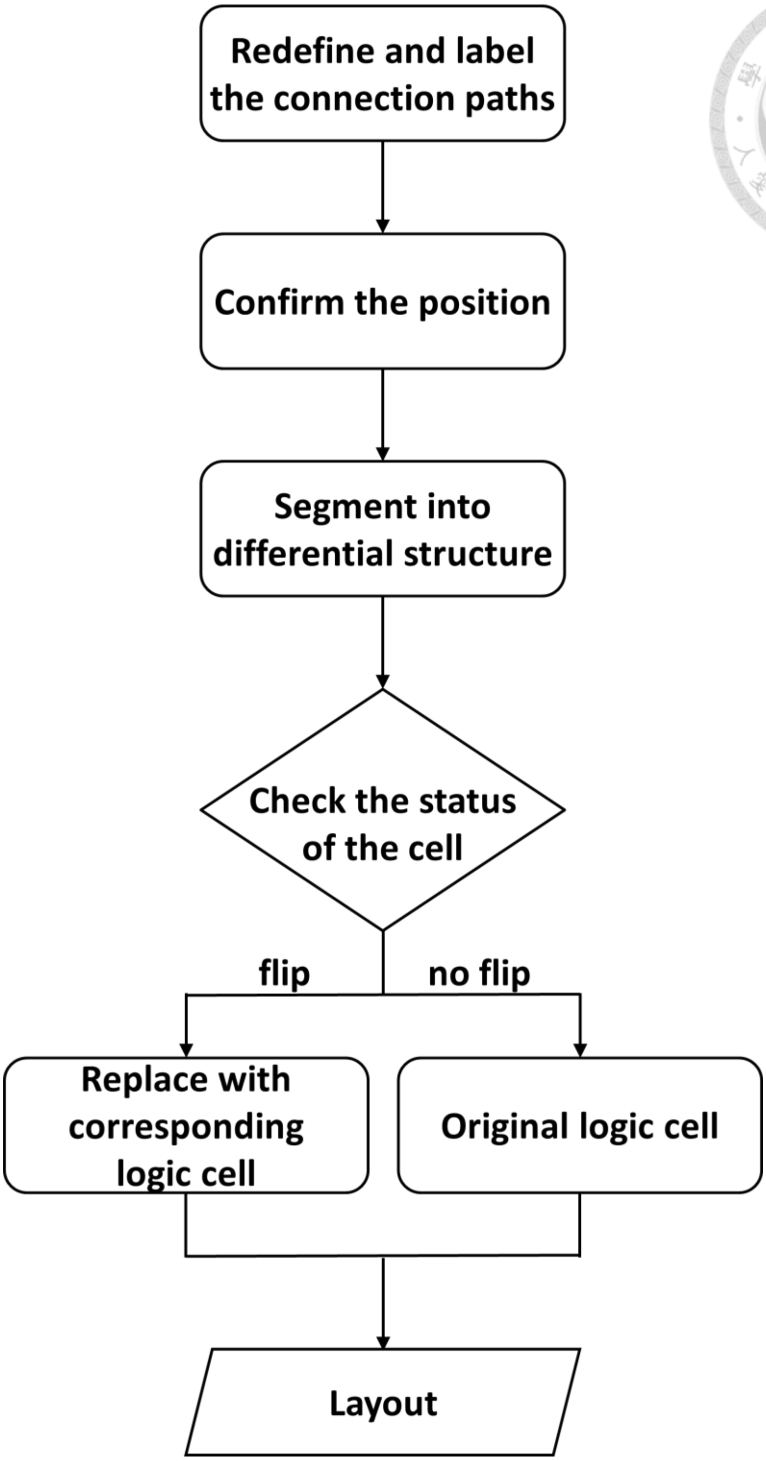


圖 3.9 差動轉換演算法

doi:10.6342/NTU202303723

3.4 驗證模擬

基本上整個 CML 設計流程的驗證模擬會分成四個階段完成：1. RTL 編譯後的模擬 2. 合成後的模擬、3. pre-layout 模擬和 4. post-layout 模擬，如同下面的流程圖 3.10 所示：

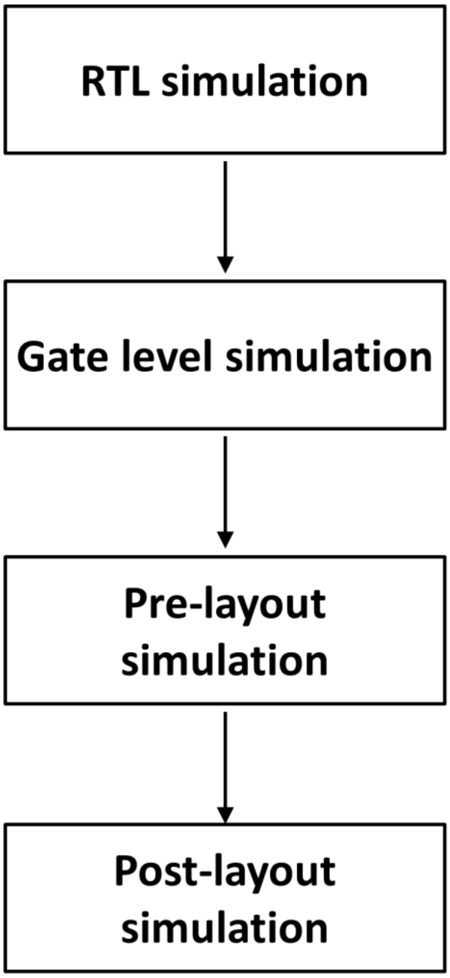


圖 3.10 驗證模擬流程圖

完成硬體程式語言編譯之後，首先會針對設計電路的功能性正確與否以 test bench (TB)進行驗證，並且生成波型作為可供後續模擬參考的結果。

電路經過合成後會產生對應的 Standard Delay Format (.sdf)及合成好的網路連線表，同樣搭配原先寫好的 TB 加以驗證，觀察實際轉換成 CML 的標準元件

doi:10.6342/NTU202303723

後，是否會導致結果發生問題，若是於此步驟發生問題則可能是在先前硬體程式語言的編寫上具有瑕疵，抑或是時序太近而需要進行修正。

接著，轉換成雙端後需做 pre-layout 的模擬。前面經過 Cadence Innovus 佈局的處理之後，基本上已是加入環境變因的模擬，此時的模擬結果已和晶片最終模擬結果十分貼近。CML pre-layout 的模擬步驟和 CMOS 並無太大區別，得到模擬結果後，可將結果和原先用 TB 驗證的結果比較，觀察經過單端轉換成雙端後，最後的結果是否會受到影響，若有，則需檢查演算法的運算是否出現問題，抑或是前述的繞線密度太過密集從而導致轉換不出正確的結果。

最後是 post-layout 的模擬，此步驟和 CMOS 的流程同樣沒有太大不同，主要是因為這邊的電路通常轉換過來比較大，因而若要做 post-layout 的模擬，所花費的時間會相對較長。然而為避免衍生出轉換後的錯誤，仍然會進行此模擬步驟驗證，以觀察 post-layout 與 pre-layout 模擬結果的異同，而更貼近實際晶片在現實運作中的狀況。

doi:10.6342/NTU202303723

第 4 章 結果與討論

4.1 設計流程結果

在構建完整的設計流程後，本文使用一個簡單的 6 位元解碼器電路進行驗證。如圖 4.1 所示，左側可以看到進行單端流程後獲得的佈局，從右側放大的佈局中可以看到繞線是正常的。

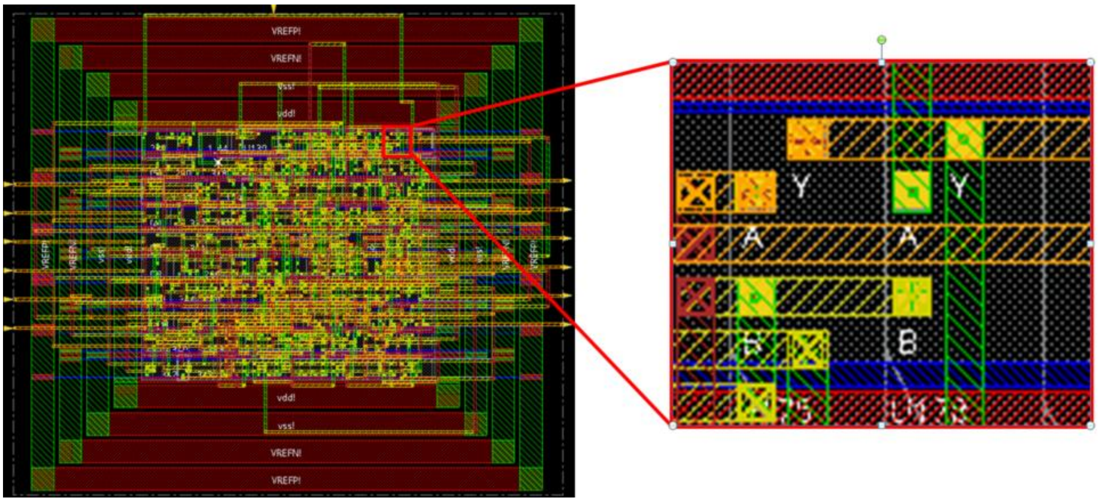


圖 4.1 單端電路結果

確認電路在此設計流程下有成功產出後，就針對它是否有 DRC 的錯誤進行驗證，同時跑了它的面積分析，可以從圖 4.2 看到它並沒有出現任何的錯誤。另外比較需要注意的一點就是，雖然在設計上應該是要盡可能的讓晶片面積縮小，這樣可以節省電路的成本，但是在後續流程還要進行轉換成 CML 的步驟，單端轉換成雙端會稍微多花費一些繞線的面積，所以在這邊先預留了一些面積，讓之後的流程有較多的空間可以利用。

doi:10.6342/NTU202303723

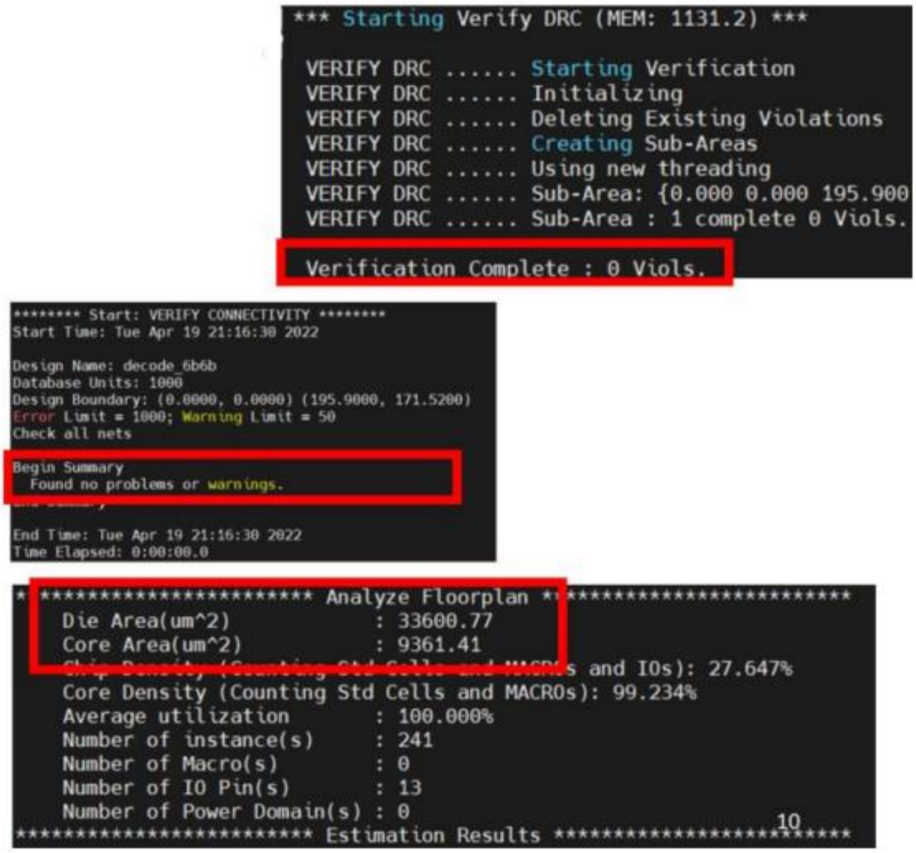


圖 4.2 單端電路驗證結果

圖 4.3 中為在執行差動流程後的解碼器佈局，從右側放大的佈局中可以看到，原來的單端繞線已成功轉換為差動繞線。

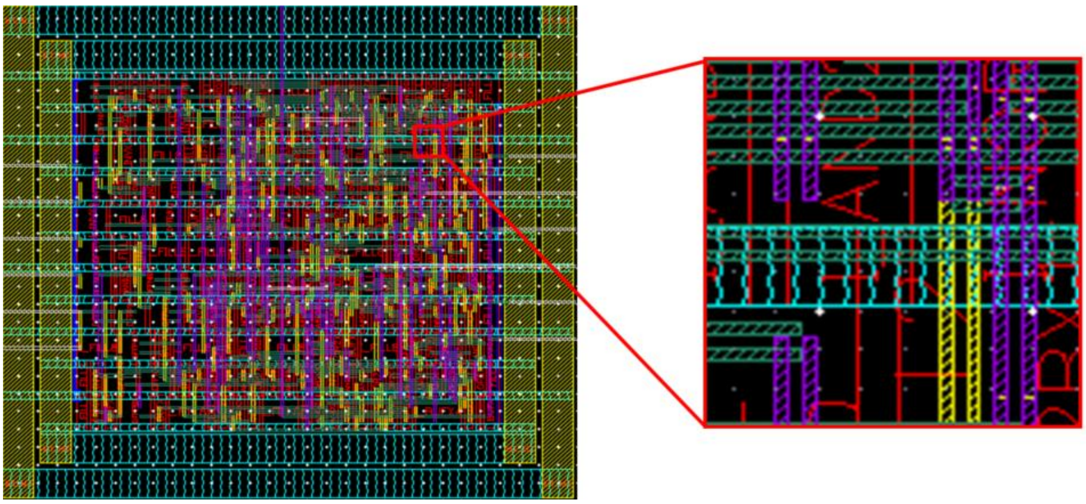


圖 4.3 差動電路結果

doi:10.6342/NTU202303723

除此之外，線路已經連接在一起，經過驗證之後發現沒有 DRC 跟 LVS 錯誤，如圖 4.4 所示。因此可以得出結論，儘管在反覆迭代中重新設計元件，並修改單端設計規則給商業 CAD 工具使用，但這樣的轉換對於小型電路而言並不會產生錯誤。

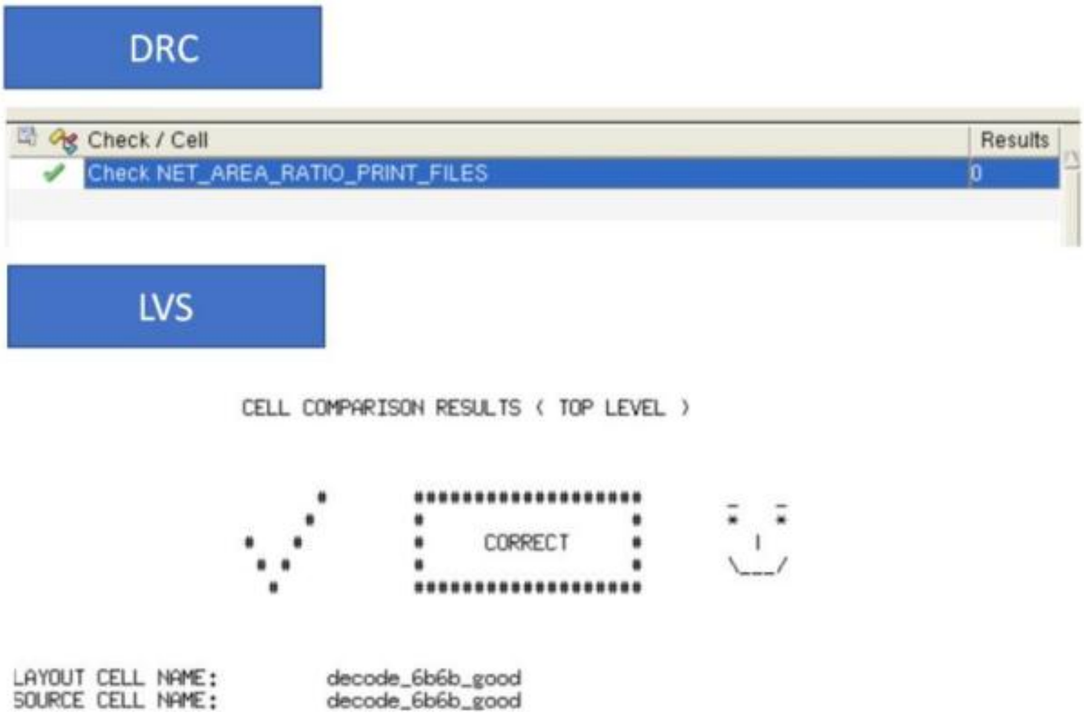


圖 4.4 差動電路驗證結果

4.2 CML 和 CMOS 電路表現比較

建立了完整的設計流程後，接著以建立好 UMC 0.18um 製程的 CML 元件庫去跟 UMC 0.18um 製程的 CMOS 元件庫去做電路表現的比較，去驗證使用 CML 的架構去做設計之後，對於電路的表現確實有所提升。

首先這邊可以先比較兩者元件庫的標準元件去做比對，UMC 0.18um 製程的 CMOS 邏輯元件列表如表 4.1，因為先前建立 CML 的標準元件，只有優先建立電路設計較為常見的標準元件，所以和 CML 和 CMOS 的標準元件表相比，可以觀察出 CMOS 擁有驅動能力較大的 AND、OR 閘，且 NAND 跟 NOR 都有三個輸入的邏輯閘，也有在條件運算中很常使用的 MUX。

doi:10.6342/NTU202303723

表 4.1 CMOS 標準元件總表

Gate(Input number)	Drive strengths
INV	1 , 2 , 4
BUF	1 , 2 , 4
AND	1 , 2
NAND(2)	1
NAND(3)	1
OR	1 , 2
NOR(2)	1
NOR(3)	1
XOR	1
XNOR	1
DFF	1
DL	1
CLKBUF	1
MUX	1

這邊一樣使用 6 位解碼器電路進行電路表現的驗證，為了單純進行元件庫邏輯的比較，所以在這邊下的合成指令，基本上只有給它時脈的時間，就進行編譯。使用上述 CMOS 跟 CML 的元件庫合成出的電路，兩者可以達到的最高速如下表 4.2 所示：

表 4.2 不同 library 的速度比較

Library type	Timing(GHz)
UMC0.18um CMOS	1.09(GHz)
UMC0.18um CML	2.00(GHz)

doi:10.6342/NTU202303723

可以從表中得知本文建立的CML的標準元件，確實可以發揮出CML電路在高速電路上的優勢，明顯比CMOS的標準元件建構出的電路更為快速，這還是在建立的標準元件還沒有很齊全的情況下，基本上現有的CML元件除了反相器跟緩衝器以外，其它的驅動能力都只有1而已，光是這樣的CML 元件庫，就已經可以達到1.83倍的速度提升，若是可以加上條件判定常用的MUX，或是建立驅動能力強的邏輯閘，以及增加輸入數目較多的邏輯閘，相信應該可以再讓電路的速度往上提升。

確認建立的CML元件庫的速度確實比CMOS更快之後，就進而去比較在同一個時間運行下，電路的面積跟所消耗的功率，測試建立的設計流程所設計出來的CML電路是否符合其電路特性，以作進一步的校正，下表4.3為同速度下，用CMOS跟CML元件庫生成的電路表現比較的結果。

表4.3 同速度下不同library的電路表現比較

Library type	Timing	Area	Power
UMC0.18um CMOS	1.09(GHz)	2072	1.0906(mW)
UMC0.18um CML	1.09(GHz)	5058	49.7801(mW)

上表面積比較呈現的結果並不意外，因為CML的邏輯閘是全差動的結構，在標準元件的面積上，本身就較CMOS邏輯大，而且單端的繞線轉換成雙端繞線的面積也會有所增長，在繞線的限制上也是如此，因為在前面的章節有說明，為了避免在轉換成雙端的流程中，發生短路的狀況，加了限制讓同一層的金屬只能以同一個方向進行繞線，所以繞線的方式會導致沒有辦法走一些較為短的路徑，因此會產生額外的面積消耗。

至於功耗的部分，CML電路產生的功耗明顯大於CMOS邏輯電路，根據[13]裡面的研究結果可以知道，目前採用的這些標準元件基本上扇出(fanout)的數目都不多，CML跟CMOS邏輯的功耗比較和操作頻率的關係就如同下圖4.5所示，

doi:10.6342/NTU202303723

因為 CML 擁有一個穩定的電流源，並不會隨著操作頻率的增加而增加，但是 CMOS 的平均供應電流，會隨著操作頻率的增加而提升，所以當操作頻率還不夠高頻的時候，CMOS 所對應產生的功耗其實是會比 CML 小的，這也就是 CML 會被使用在高頻電路的原因。[13]的研究結果大約要在大於 4GHZ 的操作頻率下，才會在功耗上比 CMOS 小，因此在 1GHZ 左右的操作頻率自然 CMOS 的功耗會小於 CML。

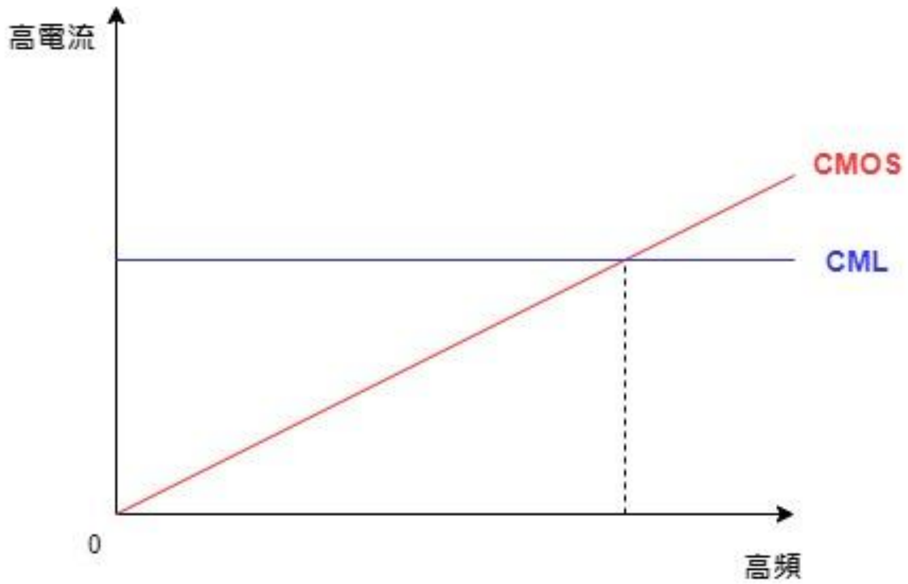


圖 4.5 CML 跟 CMOS 邏輯的功耗比較和操作頻率的關係圖

4.3 設計流程最佳化

為了最佳化設計流程，首先嘗試了擴展元件庫的方法。由於 DAC 涉及大量的加法操作，雖然在[10]裡有使用 CML 架構去設計全加器(full adder)，但是涉及三階的標準元件會有許多實現上的困難，所以先針對加法器裡面會運用到的邏輯閘進行擴充，加法器由標準元件組成的電路圖如圖 4.6，可以看到加法器是由 XOR、XNOR 和 AND 邏輯閘所組成。

doi:10.6342/NTU202303723

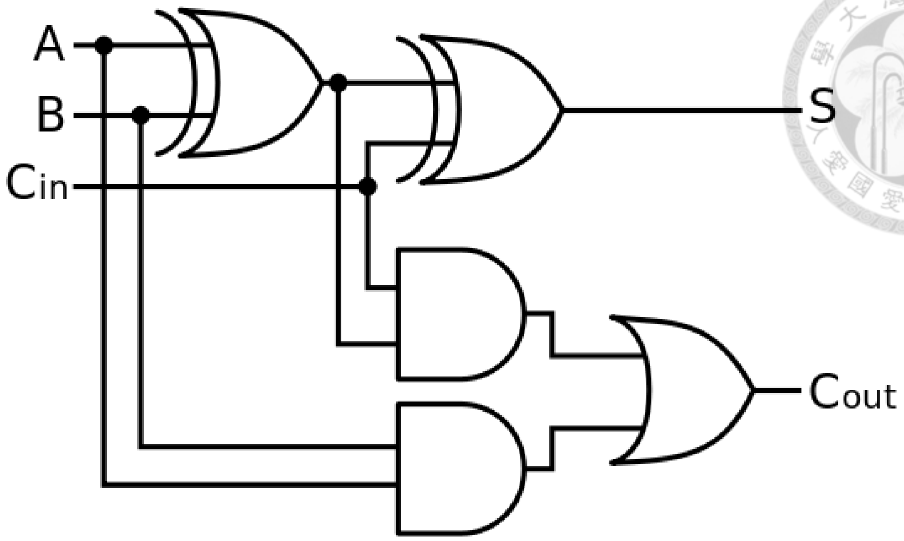


圖 4.6 加法器電路圖

因為原先的標準元件庫已經有 AND 邏輯閘了，所以只需要再補上 XNOR 和 XOR，就可以組合成一個加法器的電路，因此決定優先以 XNOR 和 XOR 進行擴充。本研究參考了[11] [12]來做設計，圖 4.7 為 XOR 的電路圖，XNOR 的電路圖和 XOR 相同，所以這邊就以 XOR 的電路圖表示，接著畫出 XOR 和 XNOR 的佈局圖，如圖 4.8 所示，可以看到它們結構基本上一模一樣，只差在輸入的腳位相反而已。完成電路圖跟佈局圖之後就按照前面產生元件庫的方式，將新建立的標準元件加進舊有的元件庫中，包含 db、lib 和 lef 檔案，都需要做更新。

doi:10.6342/NTU202303723

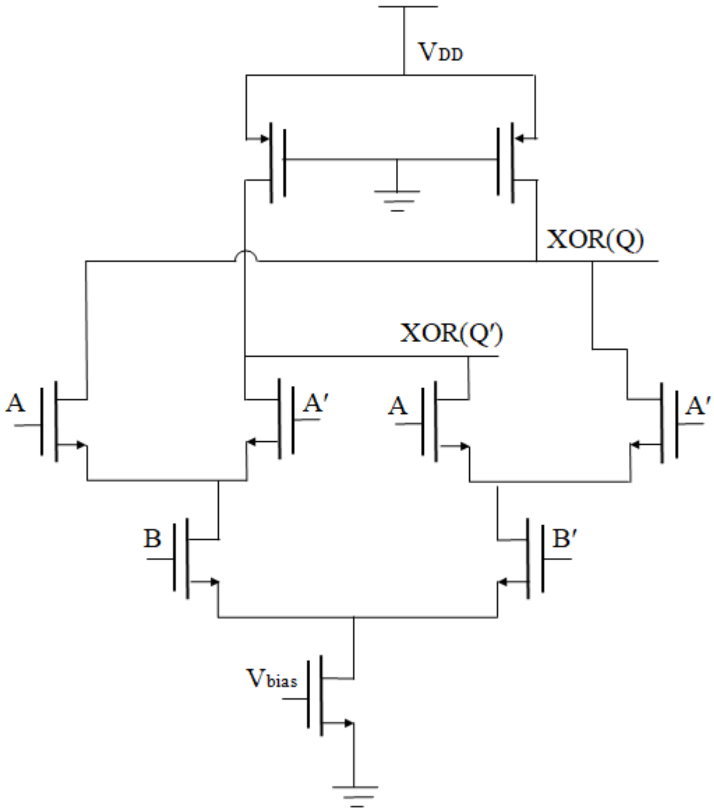


圖 4.7 XOR 邏輯閘電路圖

(a) **(b)**

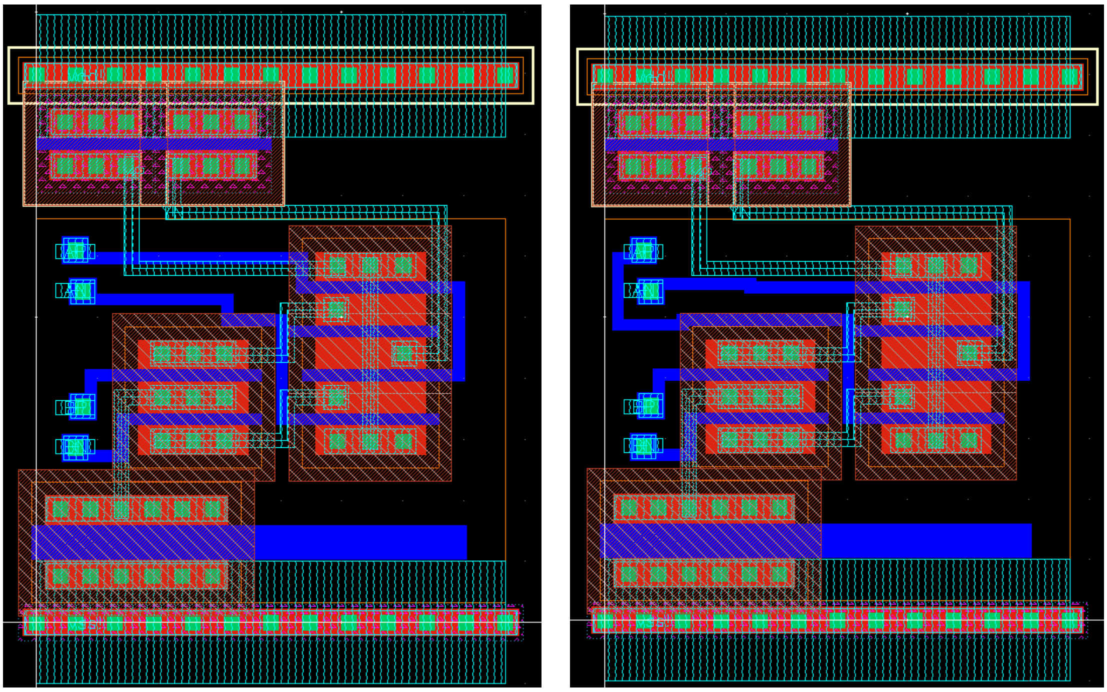


圖 4.8 XOR 和 XNOR 邏輯閘佈局圖

(a)XOR 邏輯閘(b)XNOR 邏輯閘

doi:10.6342/NTU202303723

在建立好新的元件庫之後，接著要去計算在差動流程轉換的時候，可能會遇到的反轉可能性，表 4.4 和表 4.5 為考量 XOR 和 XNOR 反轉可能性後做的整理。

表 4.4 XOR 邏輯閘轉換後的邏輯真值表及對應結果

A、B 皆不反轉		
A	B	output
0	0	0
0	1	1
1	0	1
1	1	0

A 反轉、B 不反轉		
!A	B	output
0	0	1
0	1	0
1	0	0
1	1	1

A 不反轉、B 反轉		
A	!B	output
0	0	1
0	1	0
1	0	0
1	1	1

A、B 皆反轉		
!A	!B	output
0	0	0
0	1	1
1	0	1
1	1	0

XOR2X1		
A 反轉	B 反轉	替代邏輯閘
X	X	XOR2X1
V	X	XNOR2X1
X	V	XNOR2X1
V	V	XOR2X1

doi:10.6342/NTU202303723

表 4.5 XNOR 邏輯閘轉換後的邏輯真值表及對應結果

A、B 皆不反轉		
A	B	output
0	0	1
0	1	0
1	0	0
1	1	1

A 反轉、B 不反轉		
!A	B	output
0	0	0
0	1	1
1	0	1
1	1	0

A 不反轉、B 反轉		
A	!B	output
0	0	0
0	1	1
1	0	1
1	1	0

A、B 皆反轉		
!A	!B	output
0	0	1
0	1	0
1	0	0
1	1	1

XNOR2X1		
A 反轉	B 反轉	替代邏輯閘
X	X	XNOR2X1
V	X	XOR2X1
X	V	XOR2X1
V	V	XNOR2X1

可以從兩者的反轉可能性看出，雖然兩個輸入的邏輯閘應該會有四種可能的反轉結果，不過經過整理後可以發現其實只會有兩種對應的結果，所以其實只要生成 XOR2X1 跟 XNOR2X1 就可以包含所有的轉換結果了，因此只要在原先的演算法裡面加上這兩個標準元件的反轉及結果，搭配前面生成的檔案，新的元件庫就產生了。

doi:10.6342/NTU202303723

從表 4.6 中可以觀察到，在合成邏輯中，加入 XOR 跟 XNOR 的新元件庫並沒有展示出任何時序的改進，這可能是因為無法限制新增元件的在關鍵路徑的使用量，可以從圖 4.9 中看到整個關鍵路徑中，其實只用到一個 XOR，所以對於時序的最佳化並沒有太大的幫助。

Point	Incr	Path
clock clk (rise edge)	0.62	0.62
clock network delay (ideal)	0.00	0.62
ang_reg[12]/CK (I_DFFPBX1)	0.00	0.62 r
ang_reg[12]/Q (I_DFFPBX1)	0.10	0.73 f
U5270/Y (I_INVX2)	0.02	0.75 r
U5271/Y (I_NAND2X1)	0.06	0.80 f
U5272/Y (I_AND2CX1)	0.06	0.86 f
U5273/Y (I_AND2CX1)	0.04	0.90 r
U4665/Y (I_INVX1)	0.06	0.95 f
U5279/Y (I_AND2X1)	0.05	1.00 f
U5179/Y (I_AND2X1)	0.04	1.04 f
U5131/Y (I_AND2CX1)	0.04	1.08 r
U5130/Y (I_AND2X1)	0.03	1.10 r
U5126/Y (I_AND2CX1)	0.03	1.13 r
U4858/Y (I_AND2CX1)	0.04	1.18 f
U4835/Y (I_INVX2)	0.04	1.21 r
U5010/Y (I_INVX4)	0.07	1.28 f
U6339/Y (I_XOR2CX1)	0.08	1.36 f
U6340/Y (I_NAND2X1)	0.00	1.36 r
U5135/Y (I_AND2X1)	0.05	1.41 r
U5241/Y (I_OR2BX1)	0.08	1.48 f
U5240/Y (I_AND2CX1)	0.09	1.57 f
U5065/Y (I_OR2BX1)	-0.01	1.56 r
U4949/Y (I_NOR2X1)	0.06	1.61 f
U4897/Y (I_OR2BX1)	0.10	1.71 f
U4833/Y (I_INVX1)	0.01	1.73 r
U6950/Y (I_NOR2X1)	0.04	1.76 f
U6953/Y (I_OR2X1)	0.06	1.82 f
U4957/Y (I_NOR2X1)	0.02	1.84 r
U6959/Y (I_AND2CX1)	0.02	1.86 r
datr_reg[16]/D (I_DFFPX1)	0.00	1.86 r
data arrival time		1.86

圖 4.9 新元件庫時序報告

然而，從面積和功耗方面來看卻有顯著的進展，面積漸少了約 37%、功耗則是減少了約 33%，由於有更多可使用邏輯閘的選擇，整體電路仍然可以從減少標準元件利用率中受益，雖然在關鍵路徑上並沒有太大的改善，但是因為整體使用的標準元件減少了，要消耗的功率和花費面積自然也就減少。

doi:10.6342/NTU202303723

表 4.6 新舊元件庫電路表現

	Timing	Area	Power
original library	800(MHz)	165873	1.6729(W)
New library (with XNOR,XOR)	800(MHz)	103896	1.1299(W)

經由上述的最佳化之後，總結出如果要達到最佳化時序的效果，有兩個比較具體的做法，一個是利用已經有的標準元件去拼成可用的運算邏輯，像是直接將AND、XNOR 和 XOR 邏輯閘拼湊成一個加法器，然後讓該加法器成為建構電路的標準元件做利用，則應該可以讓最佳化的部分更有效的利用於關鍵路徑；另一個則是將設計好的標準元件做最佳化，可以歸納出對於時序影響最大的即是 D Flip-Flop (DFF)，所以會針對它做結構和參數的調整，希望能夠調整它的性能，進而去改善整體電路的表現。

另外可以注意的就是合成完到繞線之後所造成的速度大幅降低，如表 4.7 結果所示，雖然可以知道限制繞線路徑多少會造成速度的損失，不過我也懷疑當初在設立時脈緩衝器的時候並沒有設計好，導致在做 CTS 之後，速度就大幅的降低，所以重新設計時脈緩衝器也是提升 CML 電路速度重要的一環。

表 4.7 Sigma-Delta modulation DAC 在不同階段可達到的運作速度

	合成完	繞線完
Sigma-Delta modulation DAC	800(MHZ)	100(MHZ)

doi:10.6342/NTU202303723

4.4 設計流程驗證

為了去驗證本文的設計流程是正確無誤的，且可以應用在規模較大的晶片設計上也不會出現錯誤，於是使用了上述的設計流程設計了一個 100MHz 的 Sigma-Delta modulation DAC 下線製成晶片，以驗證本研究的設計流程為正確無誤的，圖 4.10 為設計該晶片的架構：

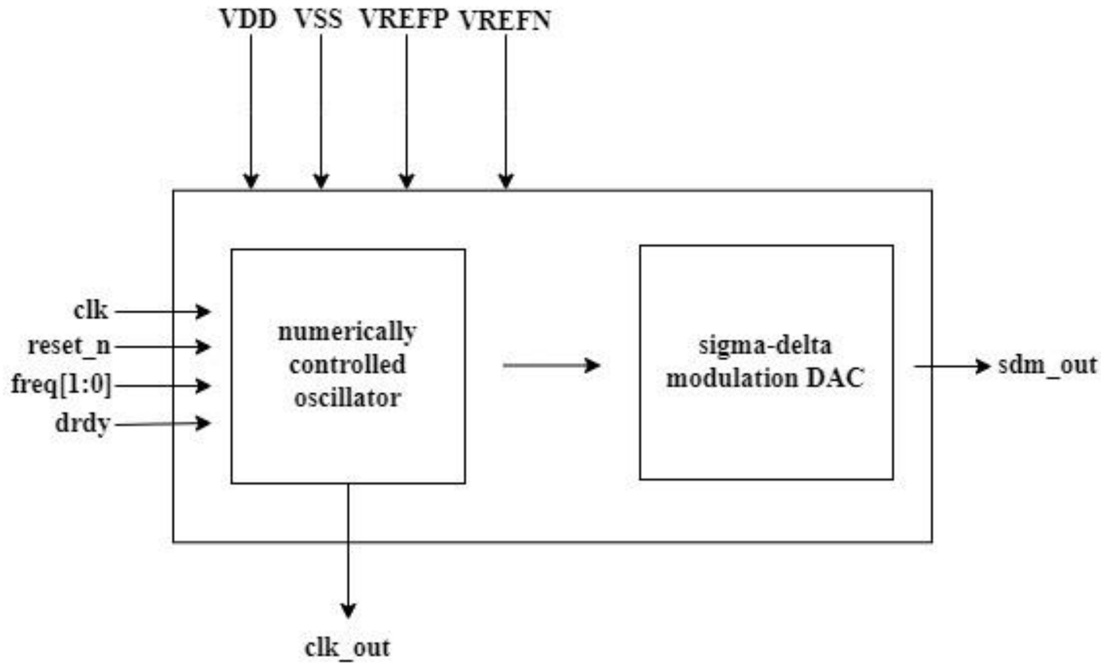


圖 4.10 晶片架構

4.4.1 晶片架構及原理

本研究的晶片架構主要分成兩個部分：第一個部分是 numerically controlled oscillator (NCO)，架構如[14]；第二個則是 Sigma-Delta modulation DAC，架構如[15]，輸入端基本上控制的晶片訊號就是時脈(clk)和頻率(freq)，搭配一個重置訊號(reset_n)，而 drdy 只是接上一個除頻器，在 clk_out 端輸出，用來測試時脈訊號是否正常運作，然後最終經過 DAC 處理出來的 1bit 訊號，會從 sdm_out 端輸出。表 4.8 為本晶片設計的規格：

doi:10.6342/NTU202303723

表 4.8 晶片規格

供應電壓	VDD	1.8V
偏置電壓	VREFP	980mV
	VREFP	267mV
clk	swing	600mV
	offset	1.5V

首先，晶片前半段的部分是一個 numerically controlled oscillator (NCO)，NCO 是一個數位信號產生器，它的用途為建立同步的離散訊號的波型表示，通常是正弦曲線，所以時常會和輸出端的 DAC 結合使用。NCO 主要分成兩個部分，phase accumulator (PA)和 phase-to-amplitude converter (PAC)，phase accumulator 會在時脈觸發時，依據所給的頻率做相位的累加，會形成鋸齒狀的不連續波型，再經由 phase-to-amplitude converter (PAC)，也就是一個 Look Up Table (LUT)，並利用截斷的 bits 去查表，將波行轉換成正弦波型，輸出給 DAC 做使用。由於 NCO 是由頻率和相位去做調節的，所以本研究的晶片設置了四種頻率去做控制，其架構可參考圖 4.11。

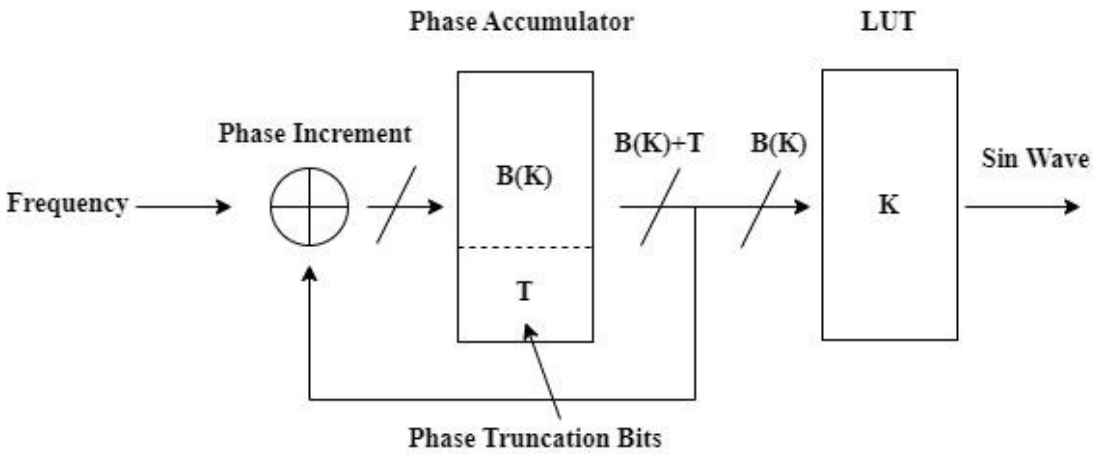


圖 4.11 NCO 電路架構

doi:10.6342/NTU202303723

後半段則是此晶片的主軸，也就是 Sigma-Delta modulation DAC，它主要是由累加器和比較器所組成，本晶片設計的為三階的 DAC，所以會有三組，首先接收從 NCO 傳進來的正弦訊號，然後每一組會搭配一個係數去做調整，然後一層一層的去做疊加，直到最上層的 MSB 溢出的時候，從 sdm_out 輸出。圖 4.12 為 Sigma-Delta modulation DAC 的電路架構。

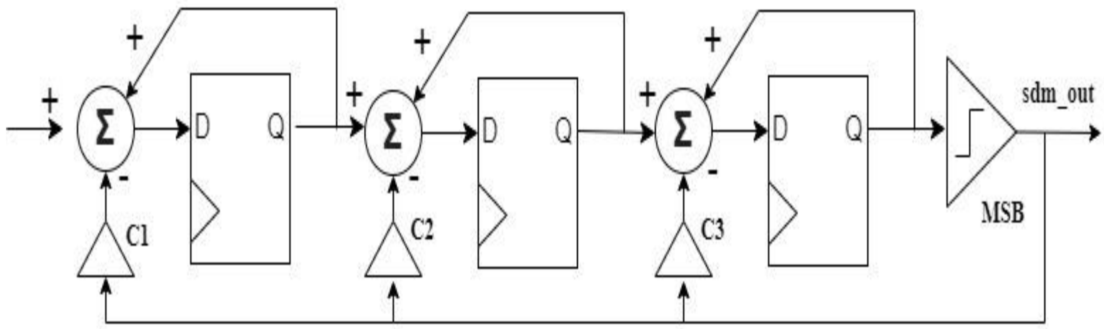


圖 4.12 Sigma-Delta modulation DAC 電路架構

接著使用硬體程式語言去實現這樣的電路架構，主要是以計數器(clk_out)，去觸發相位跟位置資料的更新，再去 LUT 中查表，然後再輸出成 17bits 的正弦暫存器，最後再送到三階的 DAC 裡頭進行累加，最後從 sdm_out 輸出，整顆晶片的電路架構圖如圖 4.13 所示，可以看到上半部的結構是 NCO，匯出正弦訊號之後，就連接到 Sigma-Delta modulation DAC。

doi:10.6342/NTU202303723

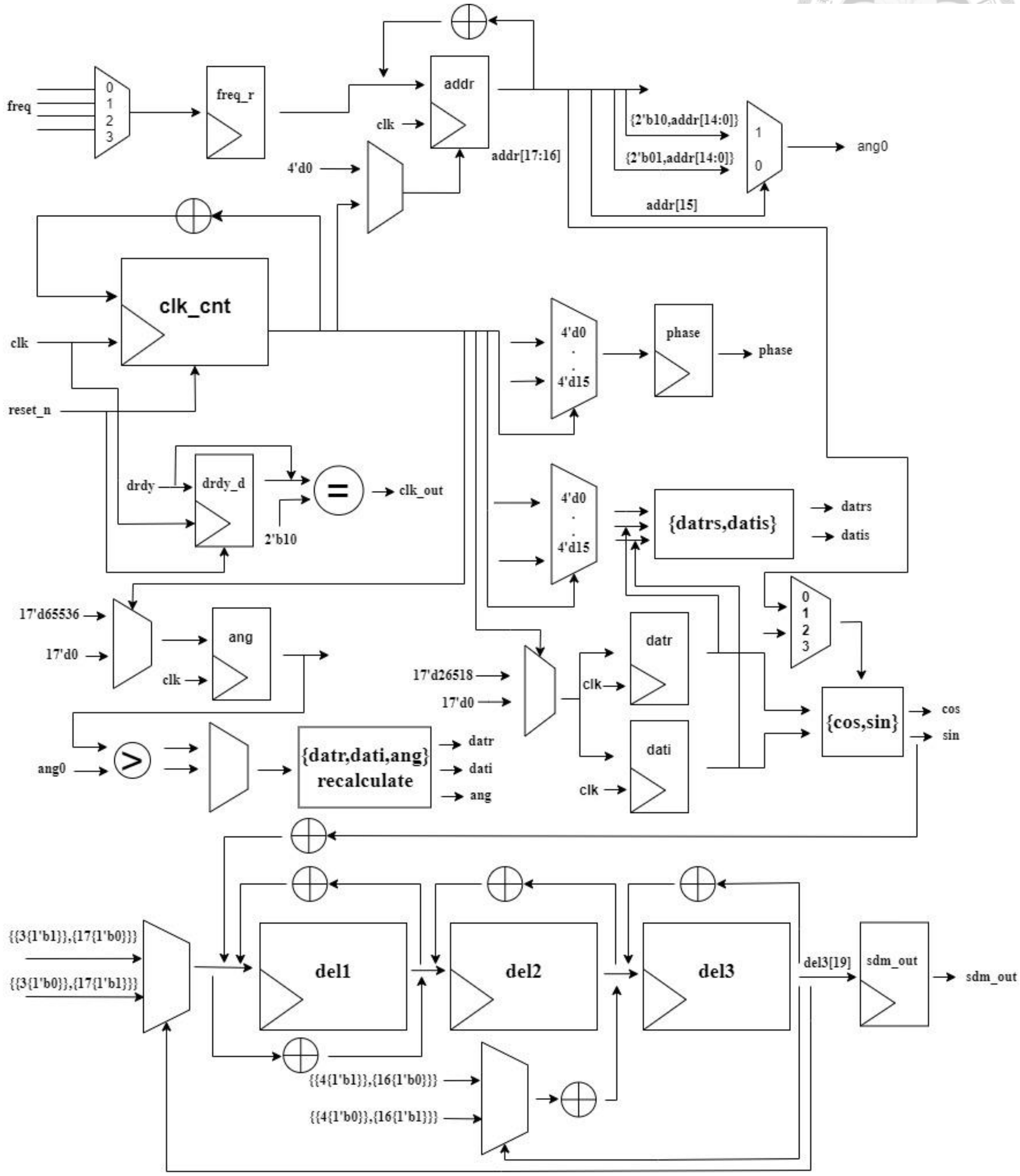


圖 4.13 晶片的電路架構圖

doi:10.6342/NTU202303723

4.4.2 晶片的佈局和實現

完成 Sigma-Delta modulation DAC 在硬體程式語言上的設計之後，就使用 CML 的設計流程去做實現，雖然設計流程經過檢驗之後，已經順利可以運行了，原先因為測試電路的大小並沒有這麼大，所以在跑佈局跟繞線的流程中，並沒有出現到的問題在這時候就浮現出來了，但是在實際較大型的晶片實現上，還是遇到許多問題。

首先，在做晶片尺寸的設定上就需要做調整，必須要預留比較多的空間給後續的繞線，因為晶片被限制只能同層單向繞線，所以在較為複雜的晶片會容易出現真的無法完成沒有 DRC 錯誤的繞線，這個時候就還要把晶片的利用率再降低，以給予它更多的繞線空間。若是在最後做繞線的時候，產生金屬層短路的問題數不多的話，在不希望降低電路表現的前提下，也可以將發生短路的地方單獨選取，然後把發生錯誤的地方刪掉再重新繞線，有機會就可以重新繞出沒有問題的繞線。

在做轉換的時候，會偶發性的出現 via 和電源供應走線之間的 DRC 錯誤，這是因為在前面做繞線和佈局的時候，自動化設計工具認為電路是單端的架構，所以在 Cadence Innovus 裡做 DRC 的驗證的時候，這個 via 還沒有轉換成可以雙端的兩個小 via，所以沒有報錯，但是在後面差動階段做轉換之後，新的小 via 會比原先還要更靠近旁邊的電源供應走線，所以有的時候就會報錯，如同下面的示意圖 4.14 所示，但是這個發生的機率比較小，最多就會發生個位數的錯誤，所以其實可以用手動去做修正，基本上只要把電源供應走線上，跟小 via 發生錯誤的 via 刪掉，就不會報錯了。

另外因為不想要太多的電源供應走線，容易影響到繞線，所以在晶片的上下都加上了兩對的 VDD 跟 VSS 的 PAD，這樣可以讓分流較為簡單，不用太多走線，對於繞線的影響也會減小。

doi:10.6342/NTU202303723

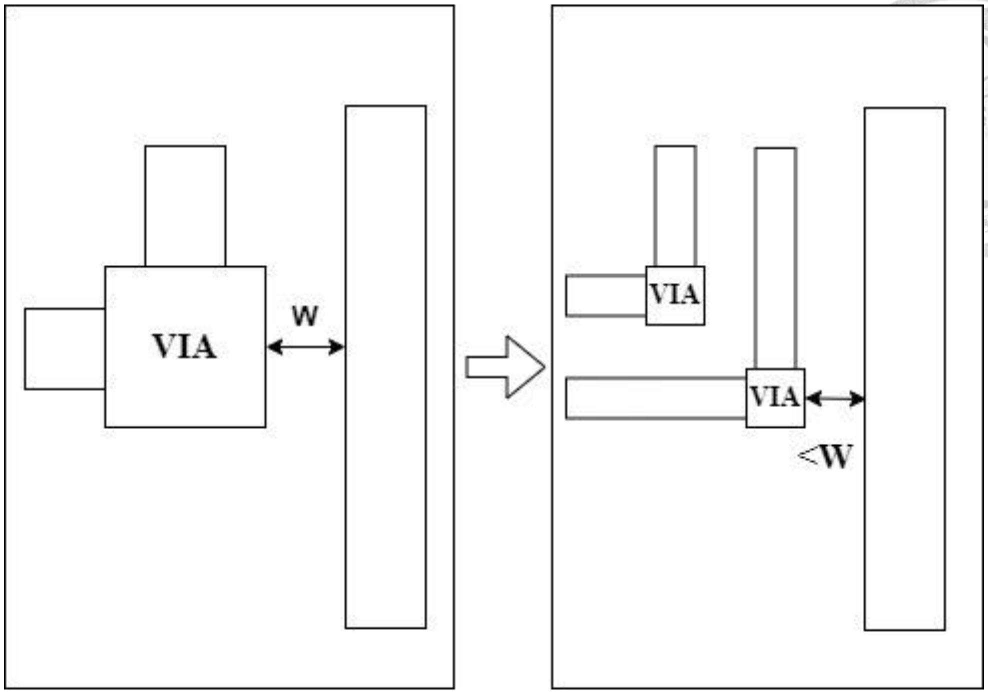


圖 4.14 差動轉換後 via 衍生錯誤之示意圖

比較令人頭痛的是密度(density)的問題，本文主要採用三種方式去解決：

第一種方式，把 6 層的金屬層平均的分配成電源環，為了方便繞線，環的方向會設定和繞線方向相同，所以金屬層 1、3、5 都是水平方向，而金屬層 2、4、6 則是垂直方向，這樣的方式可以讓每一層的金屬更多的分布在晶片裡，是解決密度問題的主要方式。

第二種方式則是在 Cadence Innovus 做完繞線之後，可以選取金屬填充(metal fill)的功能，它會在剩餘空閒的地方填入不同層的金屬，一樣可以得到補足密度的功能，但是因為原本的佈局繞線其實就已經沒有太多空間了，所以對於提升密度的能力，並沒有太大的幫助。

最後一種方式則是，在 PAD 放置的時候盡量讓它們之間的距離接近 DRC 可以接受的最小距離，這樣所形成的晶片面積也會縮小，相對起來不同層金屬在該晶片的比例也就會增加，進而達到提升密度的效果。

doi:10.6342/NTU202303723

原先在差動轉換的過程是快速的，時間的花費主要落在後面檢驗所有被標明的部分是否完全轉換完的過程中，會花費較久的時間。檢驗演算法的執行過程後，發現其實原先的設計就算檢驗完，下一次的檢驗還是會把全部標明的部分拿來檢驗，所以將其修正成會過濾掉已檢驗的項目後再做檢驗，花費的時間就明顯減少很多了。

經過排除掉上述的問題之後，就成功使用建立的 CML 設計流程設計出了一個 100MHz 的 Sigma-Delta modulation DAC，並且通過 DRC 跟 LVS 的驗證，最終的佈局圖如圖 4.15 所示，圖中的輸入和輸出都是差動的結構：

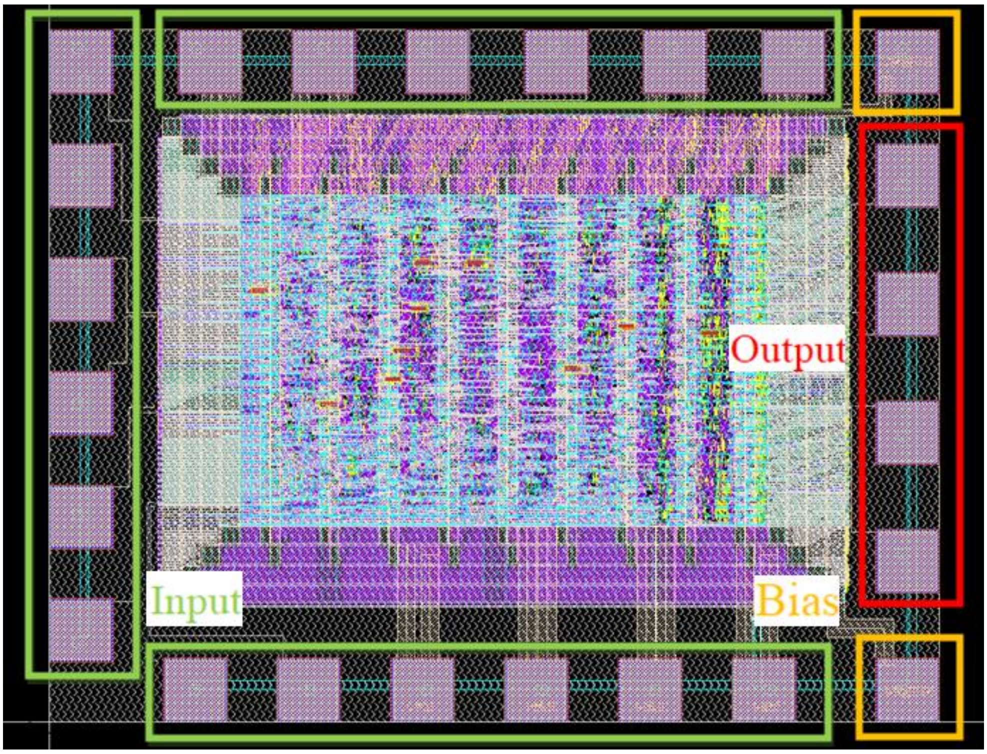


圖 4.15 Sigma-delta modulation DAC 的佈局圖

在驗證正確性後，使用 UMC 的 0.18um 的製程下線，製作了一顆 100MHz 的 Sigma-Delta modulation DAC 的晶片，它的晶片大小為 0.9mm*0.7mm。可以在圖 4.16 中看到已生產出的晶片照片。

doi:10.6342/NTU202303723

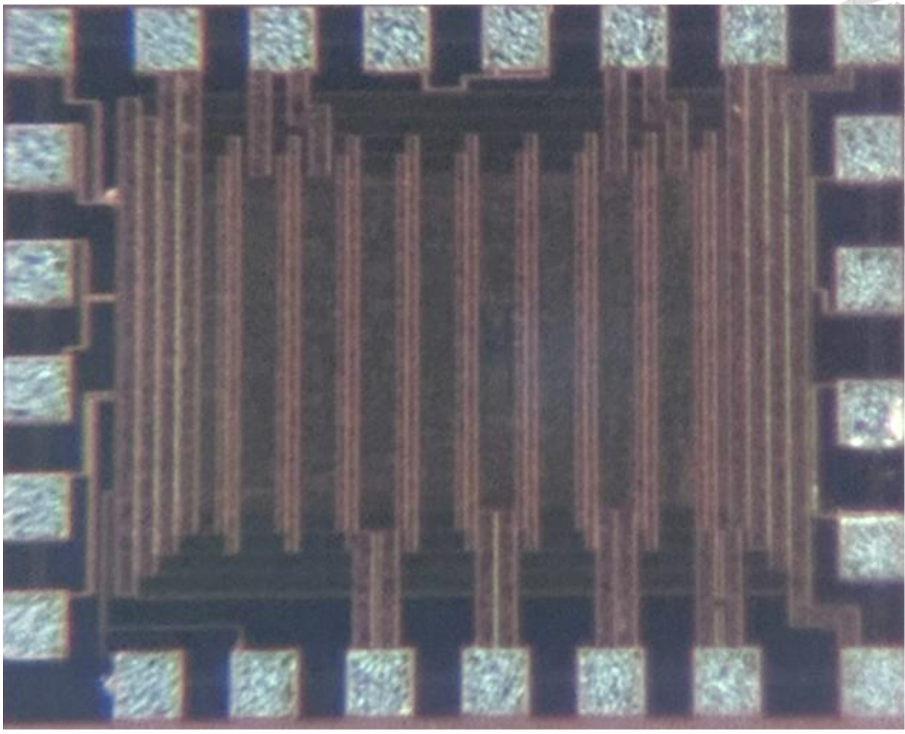


圖 4.16 Sigma-delta modulation DAC 的實體晶片圖

4.4.3 驗證與模擬

將 DAC 轉換為差動結構之後，希望驗證電路的功能和性能是否受到影響。因此本文將使用 Verdi 得到的單端模擬結果與使用 Virtuoso 得到的差動配置後佈局模擬結果進行對比。從圖 4.17 和圖 4.18 中，得到了相同的結果，確認了該設計流程的正確性，但是可以從差動模擬結果發現它會有一些漣波(ripple)的產生，說明了標準元件的設計上還可以進行更進一步的最佳化，然後 DFF 和時脈緩衝器的元件參數也需要進一步調整，這兩者對於時序的影響是非常可觀的。

doi:10.6342/NTU202303723

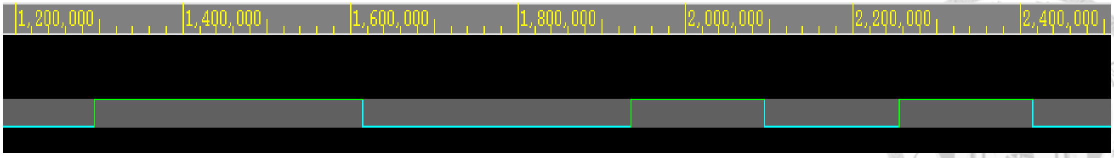


圖 4.17 單端模擬結果

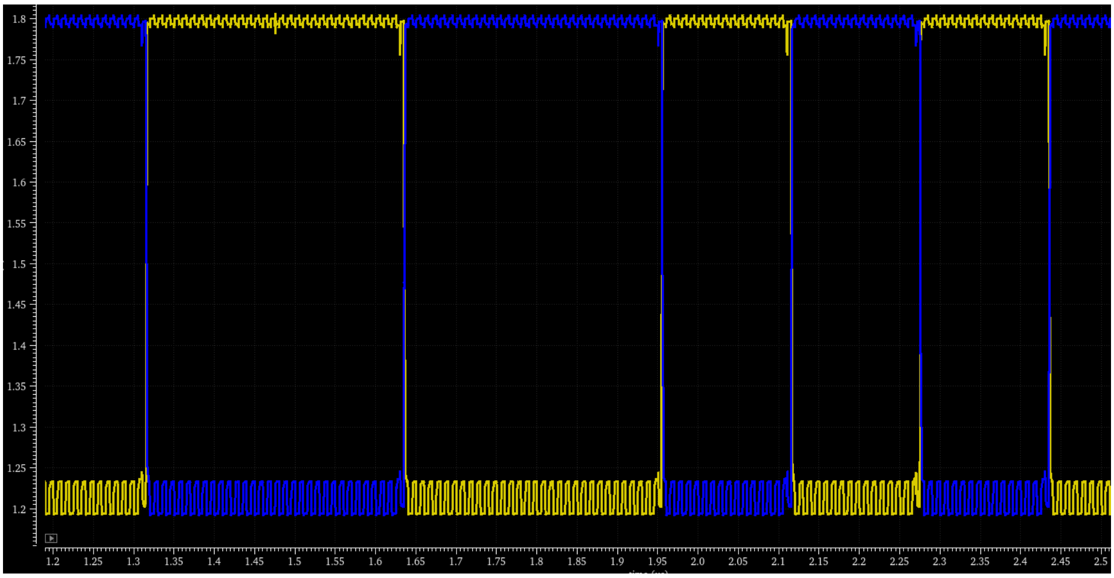


圖 4.18 差動模擬結果

doi:10.6342/NTU202303723

第 5 章 結論與未來展望

5.1 結論

本文成功開發了一個功能完整的 CML 設計流程，其中包括驗證其將硬體程式語言準確轉換為 CML 佈局的能力，並成功地使用該流程下線且生產出一顆晶片。有了這樣的設計流程，設計 CML 電路的設計者只需專注於編寫 RTL 程式碼和開發高性能的 CML 元件，這可以大大減少 CML 電路設計所需的時間和精力，並提供一種全新的 CML 設計方式。

5.2 未來展望

本研究雖然建立了一個可行的 CML 設計流程，並確保其流程並無問題，但是在流程的不同階段都存在著可以改進的部分，使產生出來的 CML 電路可以有更好的電路表現。未來可以針對這些方向去做更進一步的最佳化，下面是可以改良的方向和方法。

(1) 元件庫最佳化

因為 CML 電路的優勢主要是在高頻電路的實現上，目前實現出來的電路速度其實還有很大的進步空間，若是要提升電路速度，元件庫的最佳化是一個很重要的方向，DFF 的表現跟電路特性，其實還可以在藉由調整它的元件參數去做最佳化，且前面提及的時脈緩衝器，可以針對它的 DC 去做調整，讓 CTS 的流程不會讓電路時序上有所犧牲，還有將邏輯閘組成較大的運算元件，也有機會有效的提升電路的速度，在完成以上的最佳化之後，可以再進一步去嘗試製程的提升，應該會對於電路的表現有所助益。

(2) 模擬方式的簡化

因為在模擬的方式上，跑 pre-layout 跟 post-layout 模擬花費的時間成本實在有點高，所以希望未來可以將前面跑完單端流程時，因為 innovus 已經加入環境

doi:10.6342/NTU202303723

變因的條件去做模擬，所以希望可以比較做完佈局繞線之後跟 pre-layout 跟 post-layout 的模擬結果，讓電路設計者可以使用在單端流程模擬的結果去評估最終的電路特性。

(3) 建立 IO PAD

在設置 PAD 跟解決密度的問題上，花費了需多力氣跟時間，所以若是可以建立一個通用的 IO PAD，在裡面也同時做密度的配置，相信對於實體下線晶片也會有很大的幫助。

(4) 統一 CAD 工具

因為設計流程中，交錯使用了許多 CAD 工具，所以在流程的通順跟使用的複雜度其實並沒有這麼友善，所以希望未來能夠趨向使用同一間公司的 CAD 工具,這樣在工具上的相容性一定會更好,最終希望能夠將裡面的流程都連結起來,用一套指令就可以從頭執行到尾，讓使用者在操作上更加便利。

doi:10.6342/NTU202303723

參考文獻

[1] M. Yamashina, H. Yamada, "An MOS current mode logic (MCML) circuit for low-power subGHz processors," *in IEICE Transactions, E75-C*, October 1992.

[2] S. Badel, C. Baltaci, A. Cevrero and Y. Leblebici, "Design automation for differential MOS current-mode logic circuits," *Springer International Publishing*, 2019.

[3] H. Reyserhove and W. Dehaene, "Efficient Design of Variation-Resilient Ultra-Low Energy Digital Processors," *Springer*, 2019.

[4] Z-V Chia, S-Y-S Ng and M. Je, "Current mode logic circuits for 10-bit 5 GHz high speed digital to analog converter," *International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering 7, no. 9*, 2013.

[5] Y. Ling, W. Ni, Y. Shi and F. F. Dai, "A 10-bit 2GHz current-steering CMOS D/A converter," *in 2007 IEEE International Symposium on Circuits and Systems, pp. 737-740*, 2007.

[6] F. Cannillo, C. Toumazou, and T. S. Lande, "Bulk-drain connected load for subthreshold MOS current-mode logic," *Electronics Letters 43, no. 12*, 2007.

[7] J. Musicer and J. Rabaey, "MOS Current Mode Logic for Low Power, Low Noise CORDIC Computation in Mixed-Signal Environments," *in Proc. International Symposium on Low Power Electronics and Design (ISLPED), pp. 102 - 107*, 2000.

[8] M. Mizuno, M. Yamashina, K. Furuta, H. Igura, H. Abiko, K. Okabe, A. Ono, and H. Yamada, "A GHz MOS Adaptive Pipeline Technique Using MOS Current-Mode Logic, " *IEEE J. Solid-State Circuits, vol. 31, no. 6, pp. 784 - 791*, 1996.

doi:10.6342/NTU202303723

[9] S. Badel, I. Hatirnaz and Y. Leblebici, "A semi-automated design of a MOS current-mode logic standard cell library from generic components," *in IEEE Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)* , 2005.

[10] S. Badel, I. Hatirnaz and Y. Leblebici and E. J. Brauer, "Implementation of Structured ASIC Fabric Using Via-Programmable Differential MCML Cells," *in 2006 IFIP International Conference on Very Large Scale Integration, pp. 234-238*, 2006.

[11] I. Hatirnaz, S. Badel and Y. Leblebici, "Towards a unified top-down design flow for fully differential logic blocks with improved speed and noise immunity," *Research in Microelectronics and Electronics, 2005 PhD, vol. 1, pp. 89-92*, 2005.

[12] S. Badel, E. Güleyüpoğlu, Ö. Inaç, A.P. Martinez, P. Vietti, F. K. Gürkaynak, and Y. Leblebici, "A generic standard cell design methodology for differential circuit styles," *in Proceedings of the conference on Design, automation and test in Europ*e, pp. 843-848, 2008.

[13] S.R.P. Sinha and N. Tiwari, "Design and Analysis of Dynamic Current Mode Full Adder with reduced Power and Delay," *International Journal of Science and Research (IJSR)*, 2015.

[14] M. Fouad, H.H. Amer, A.H. Madian and M.B. Abdelhalim, "Current mode logic testing of XOR/XNOR circuit： a case study," *Circuits and Systems, Scientific Research Publishing, vol.4, pp. 364-368,* August 2013.

[15] H. Chen, G. Guo, Q. Lai, Y. Zhang, J. Han and Y. Yan, "0.3–4.4 GHz wideband CMOS frequency divide-by-1.5 with optimized CML-XOR gate," *IEICE Electronics Express 14, no. 12*, 2017.

doi:10.6342/NTU202303723

[16] Z. Toprak and Y. Leblebici, "Low-power current mode logic for improved DPA-resistance in embedded systems," *in 2005 IEEE International Symposium on Circuits and Systems, pp. 1059-1062*, 2005.

[17] R.R. Gujjula, C. Perumal, P. Kodali and V.R. Bodapati, "Design and analysis of dual-mode numerically controlled oscillators based controlled oscillator frequency modulation," *International Journal of Electrical and Computer Engineering 12, no. 5*, 2022.

[18] P. Ju, K. Suyama, P.F. Ferguson and W. Lee, "A 22-kHz multibit switched-capacitor sigma-delta D/A converter with 92 dB dynamic range," *in IEEE Journal of Solid-State Circuits, vol. 30, no. 12, pp. 1316-1325*, Dec. 1995.

[19] Cadence, SKILL Language User Guide (v06.70), January 2007

doi:10.6342/NTU202303723