國 立 交 通 大 學

電機學院 IC設計產業研發碩士班

碩 士 論 文

數位式脈波寬度調變控制電壓轉換電路

Digitally Controlled PWM for DC-DC Converter

研 究 生：林晏生
指導教授：陳科宏 博士

中 華 民 國 九 十 六 年 九 月

數位式脈波寬度調變控制電壓轉換電路

Digitally Controlled PWM for DC-DC Converter

研 究 生：林晏生　　Student：Yen-Sheng Lin

指導教授：陳科宏　　Advisor：Ke-Hong Chen

國 立 交 通 大 學

電機學院 IC 設計產業研發碩士班

碩 士 論 文

A Thesis

Submitted to College of Electrical and Computer Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Industrial Technology R & D Master Program on
IC Design

September 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年九月

數位式脈波寬度調變控制電壓轉換電路

學生：林晏生　　指導教授：陳科宏

國立交通大學電機學院產業研發碩士班

摘要

本論文研製一個以可規劃邏輯閘陣列(FPGA)為基礎之全數位控制脈波寬度調變控制電壓轉換電路，針對數位式脈波寬度調變控制電壓轉換器建立其數學等效模型，並探討電路特性、脈寬調變控制方式、與實現方法。控制架構以PID控制器之原理為設計基礎，包含比例控制器、積分控制器及微分控制器設計，以增加低頻增益及相位邊限，改善系統閉迴路的相對穩定度，使暫態響應較快，減少穩態誤差。模擬平台本文採用ModelSim及MATLAB/Simulink軟體工具進行系統整合模擬，驗證其功能正確性。在實驗方面，使用設計FPGA/CPLD 或嵌入式系統的Xilinx ISE 發展軟體及為希科技公司所推出的系統發展實驗板（ULINX_MB_XC3S250E_PQ208_V20A），利用硬體描述語言(Verilog)來完成數位電路系統之設計，並透過Buck 電壓轉換器之系統電路驗證所設計數位控制器之功能與性能。

Digitally Controlled PWM for DC-DC Converter

stuudent：Yen-Sheng Lin　　Advisors：Dr. Ke-Hong Chen

Industrial Technology R & D Master Program of
Electrical and Computer Engineering College
National Chiao Tung University

Abstract

The purpose of this thesis is to study and produce a full digitally controlled PWM for DC-DC converting circuit based on FPGA.This thesis is aimed at building a mathematically effective module equal to a Digitally Controlled PWM for DC-DC Converter; therefore, to explore the circuit quality, the pulse-width modulating fashion, and the way of realizing the converter. The controlling structure is based on the PID controller as its designing model, including the designing of the proportional controller, the integral controller, and the derivative controller, to enhance low frequency gain and phase margin, and to upgrade the relative stability at close loop system circuit, in order to expedite the transient response and minimize Steady state error. Simulating platform takes the ModelSim/Simlink software to simulate system integration to examine its function correctness. At the experiment, either FPGA/CPLD or inserting style system Xilinx ISE development software and ULINX_MB_XC3S250E_PQ208_V20A has been used, by the system circuit of the Buck voltage converter, to examine the function and quality of the designed digital controller.

誌謝

二年多的研究所求學過程即將結束，不論在學業及待人處事上，師長及實驗室同學都給予很多的建議及鼓勵，讓我受益良多。本論文得以順利完成，首先要感謝指導教授 陳科宏老師對學生的諄諄教誨，在研究期間不辭辛勞的悉心指導與啟發，在此深表誠摯的謝意。同時由衷的感謝口試委員，王清松博士及黃立人博士對本論文提供寶貴的意見與指正，使本論文更加完善。

此外我要感謝實驗室的同學們在課業與生活上的相互提攜與關懷，尤其是數位組的豐煜、國葆，在課業與研究上一起學習，以及在實驗上的協助與討論，讓我感受到深刻的情誼，並留下了許多一同走過的美好回憶。

最後我要感謝我的家人及所有關心我的人，你們的支持與無怨無悔的付出，讓我能一一克服難關，順利完成這段求學歷程，在此僅將這份榮耀與喜悅與你們分享。

目錄

圖錄

表錄

第1章 緒論

1.1 研究動機與目的

一般嵌入式系統中經常包括很多不同規格需求電源電路，傳統電源供應器的控制晶片主要以類比方式實現，其優點是成本低、電路實現容易、反應速度快，但缺點是通常只能提供特定功能，控制迴路補償不易調整，較不適用複雜電源系統整合等等。近年來，隨著消費性電子產品市場快速成長，電源的要求越來越高，其必須是省電、適用、高效能、高可靠。為了能提供這些功能，電源控制的數位化及可程式化將成為未來發展的趨勢，而在可控因素較多、即時反應速度更快、需要多個系統電源管理的高性能系統應用中，數位電源則具有更大的優勢，數位電源將可提供電源控制與系統應用進一步的整合，達到更佳的整體效能。

切換式直流至直流轉換器被廣泛應用在切換式直流供應器、直流馬達驅動器及各種消費性電子產品。電源轉換器最重要的功能是提供穩定的輸出電壓，因此對於切換式直流至直流轉換器的控制使其在輸入電壓與輸出負載變動的情況下，能夠調整使輸出電壓維持在一定的位準，實為研究的重要課題，也是本論文要實現的目標。

1.2 切換式直流至直流轉換器簡介

圖1-1為一降壓型直流電源轉換器，此為電壓控制型電源轉換器，即控制器只根據電源轉換器的輸出電壓來調變輸出方波的開關比率

(Duty)，在控制器方面我們是使用數位控制系統，因為數位式控制器具備多項優點，如可程式化、可以減小對製程參數、溫度參數變化造成的影響，當利用現成工具（EDA）可以使設計所需時間縮短..等等。

在外部電路方面，電感(L)、電容(C)及負載電阻(R)可以稱為LCR plant。這一個LCR plant的轉移函數就是控制器要補償的受控體。當輸出電壓經ADC與參考電壓比較產生一差值e(error)，將送至數位控制電路產生週期性的方波來控制功率電晶體(Power MOS)的開與關。當其中一個功率電晶體導通，另一個電晶體就會關閉。我們藉由控制功率電晶體開關的比率(Duty Ratio)，可以決定輸出電壓的大小。

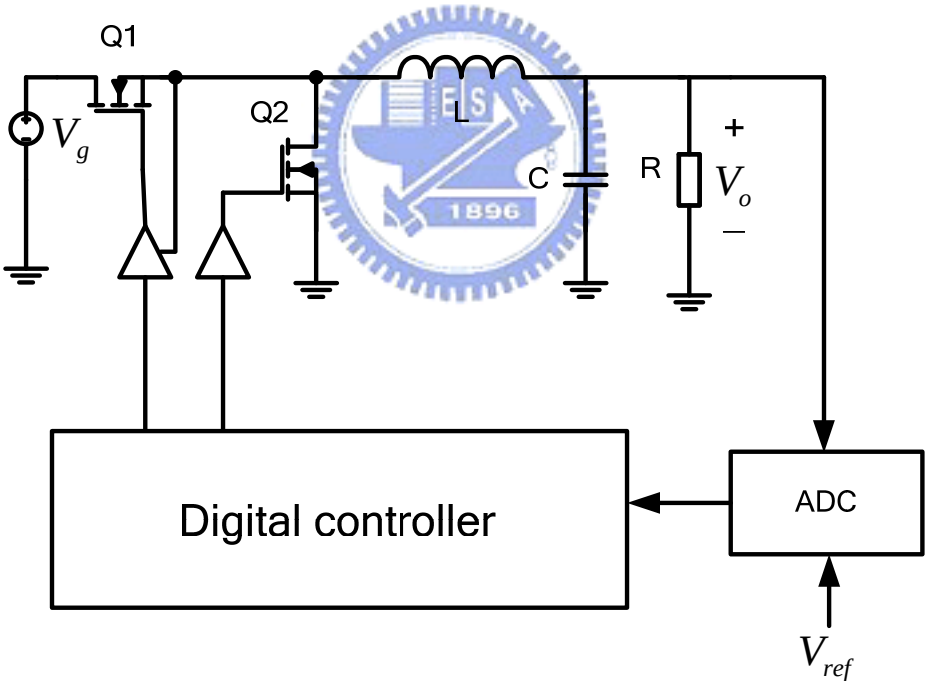


圖 1-1 降壓型直流電源轉換器

1.3 論文架構

本論文共分五個章節說明,第一章為概論,說明本論文之研究動機與目的及電源轉換器簡介。第二章將介紹數位脈波寬度調變控制器系統架構及運作方式。第三章為數位控制器介紹其中包含控制理論與設計方法。第四章為數位脈波寬度調變電路實現，利用硬體描述語言(Verilog)來設計。第五章為結論及未來方向，對整體的研究做總結並討論未來可能的研究方向。

第2章 數位脈波寬度調變控制電路之介紹

2.1 控制電路介紹

在本論文中，我們將介紹一個數位脈波寬度調變控制電路來控制一個切換式降壓型直流至直流轉換器電路。控制電路中包含了一些主要元件：類比至數位轉換器（ADC）、數位PID補償器（Digital PID Compensator ）、數位脈波寬度調變器(Digital Pulse-Width Modulator)、數位顫抖控制(Digital Dither)電路、盲時最佳化電路(Dead-Time Optimizer)。整個控制積體電路再加上直流至直流轉換器的方塊圖，如圖 2-1所示。

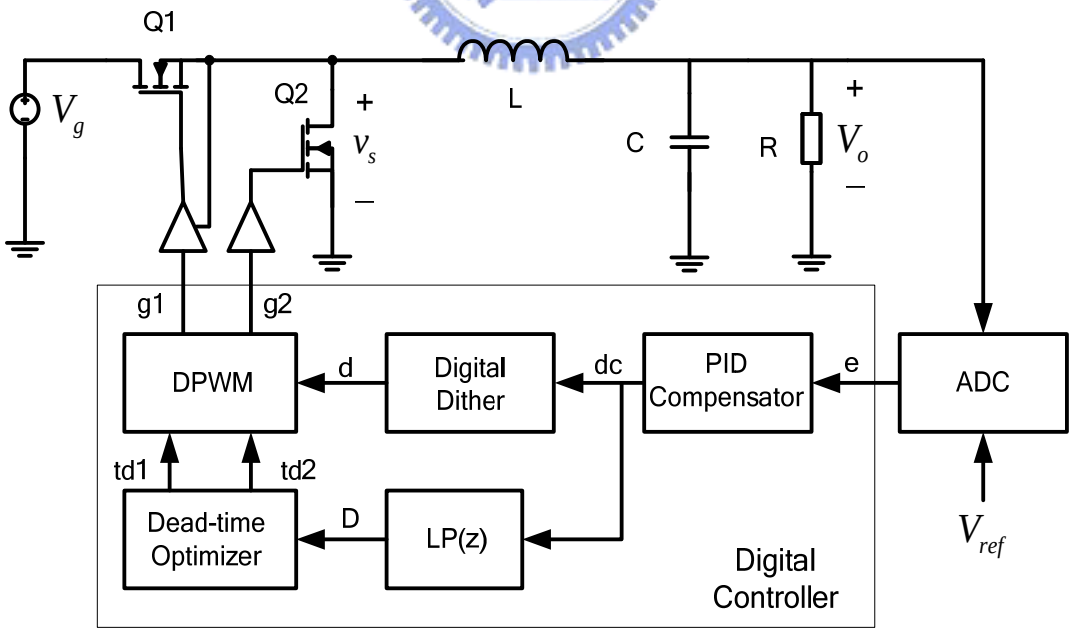


圖 2-1 數位式降壓型直流電源轉換器方塊圖

由圖 2-1知，數位脈波寬度調變控制積體電路與切換式降壓型直流至直流轉換器形成了一個閉迴路的回授控制系統，而控制電路的主要功能就是調整直流至直流轉換器的輸出電壓（V_o）。當切換式降壓型直流至直流轉換器的輸出電壓因為輸入電壓（V_i）或負載電流（I_o）的變動而改變時，此時，控制電路所送出的訊號（Duty Ratio）便會對轉換器做調整的動作，使得轉換器的輸出電壓又能夠穩定地保持在我們所要的參考電壓（V_{ref}）設定。整個數位脈波寬度調變控制電路的內部架構如圖 2-2 所示。

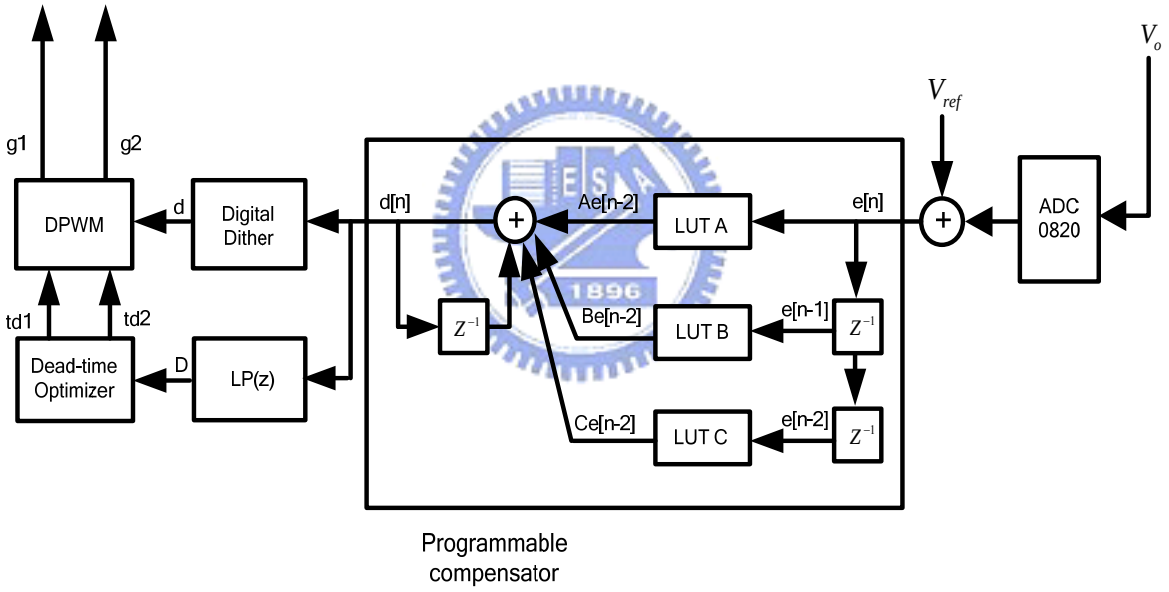


圖 2-2 數位式降壓型直流電源轉換器架構圖

首先，控制電路中的類比至數位轉換器（ADC）會對轉直流至直流轉換器的輸出電壓做取樣而得到一個取樣電壓（V_{sense}），一般的情況下，$V_{sense} = V_o$。

接著類比至數位轉換器會將取樣電壓會與送至類比至數位轉換器的參

考電壓（V_{ref}）做比較，其比較後的差值即為誤差信號（$e[n]$）。這個誤差訊號被送至數位補償器做運算，其運算的結果為責任週期命令（$d[n]$），將分別送至數位顫抖控制 (Digital Dither)、數位低通濾波器 $LP(z)$，其中數位顫抖控制將產生責任週期命令序列，而責任週期命令經過數位低通濾波器會產生責任週期命令的平均值D再送至盲時最佳化電路 (Dead-Time Optimizer)計算最佳dead-time資訊，數位顫抖控制及盲時最佳化電路輸出分別為 d、t_{d1}、t_{d2}，這些是數位脈波寬度調變器的輸入，調變器會根據輸入訊號的值，而送出二組頻率固定的脈波信號 $g1$、$g2$，其脈波寬度即是由 d、t_{d1}、t_{d2} 值所決定的。最後，產生的脈波寬度訊號會再送至切換式降壓型直流至直流轉換器的開關元件而做切換，使轉換器轉換不同的輸出電壓。經由這樣不斷的取樣、比較、補償，所以會使切換式降壓型直流至直流轉換器的輸出電壓穩定。

2.2 輸出電壓取樣電路

由圖2-1 所示，一開始類比至數位轉換器（ADC）要對轉換器的輸出電壓取樣，這在整個控制積體電路中是相當重要的一個電路。我們根據系統的要求再去選擇一顆適合的類比至數位轉換器。我們選擇了ADC0820 這顆類比至數位轉換器。

2.2.1 ADC0820 的操作模式

類比至數位轉換器(ADC0820)我們使用 Stand-Alone 這一種操作模式，在寫入-讀取模式下使用Stand-Alone 操作時，CS腳跟RD 腳可以

同時為〝0〞，當WR 腳由1 變0 時，ADC0820 開始轉換，大約800 ns 後資料就會送至輸出。Stand-Alone 模式時序圖如2-3 所示。

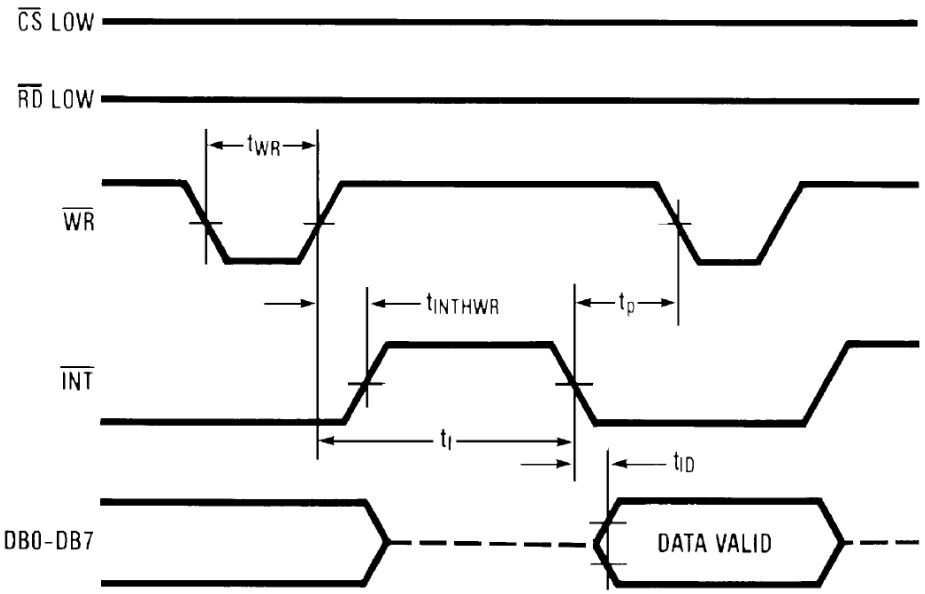


圖 2-3 ADC0820 寫入-讀取模式下 Stand-Alone 時序圖

2.3 數位補償器電路

數位脈波寬度調變控制電路中第二個部份的電路是數位補償器電路。當類比至數位轉換器取樣到直流至直流轉換器的輸出電壓（V_o）並與V_{ref} 比較得到一個誤差信號（error signal）值$e[n]$後，會把這誤差值送到此補償器。

本論文的補償器所要實現的控制理論是根據PID 控制器的觀念所設計，其所實現的關係式如下列式子所表示之：

$$d[n] = d[n-1] + ae[n] + be[n-1] + ce[n-2] \qquad (2\text{-}1)$$

其中$d[n]$代表的是補償器目前的責任週期（duty ratio）輸出值，$e[n]$則是代表目前的誤差信號（error signal）輸出值。$e[n\text{-}1]$、$d[n\text{-}1]$則為該信號前一次週期的誤差信號（error signal）及責任週期輸出值。$e[n\text{-}2]$則為該信號前二次週期的值，另外 a、b、c 均為常數。PID 控制器的觀念與詳細設計方法將會在下一章中說明。

由(2-1)式可知，設計一個補償器需要使用到數個加法器與乘法器，然而對於一個積體電路設計者而言，乘法器的使用是非常佔用晶片面積且又有速度上的考量，所以可以改用查表的設計的方式，來實現 PID控制器。

2.3.1 查表法（Look-up Table）介紹

查表法（look-up table）[1, 2, 3]，顧名思義就是利用查表的方式來代替許多複雜的運算。簡單來說，查表法是利用一張儲存資料的表，通常是設計者事先將計算好的數值結果存在記憶體中，這樣的方式可以用簡單的架構來完成乘法的動作。

我們以式(2-1)來說明，我們可以發現 $a\cdot e[n]$、$b\cdot e[n-1]$、$c\cdot e[n-2]$這三個是簡單的乘法運算，由於A/D 轉換器所輸出的e 值為很小的有限值，加上a、b、c 是常數，所以我們可以把它們的乘積事先運算出結果，然後再儲存成係數為a、b、c 的三張查表。此時，ADC所輸出的誤差信號 $e[n]$或 $e[n\text{-}i]$就可以當作是此張查表的位址。當每一個 $e[n]$值與其前 i 次週期的 $e[n\text{-}i]$值被送到補償器時，補償器的相對應a、b、c三張表之位址的內容就會馬上被查出來，因此就不需要再經過乘法的運算，不但

提高了效率也因為沒有使用到乘法器而大大的降低晶片面積。

利用查表法設計 PID控制器還有另一項優點，因為使用查表法保留了可調整、規劃的彈性，當應用在不同的電源轉換器時，不必重新設計控制電路，僅需調整表內數值即可完成。利用查表法實現的 PID控制器電路結構如圖2-4所示。

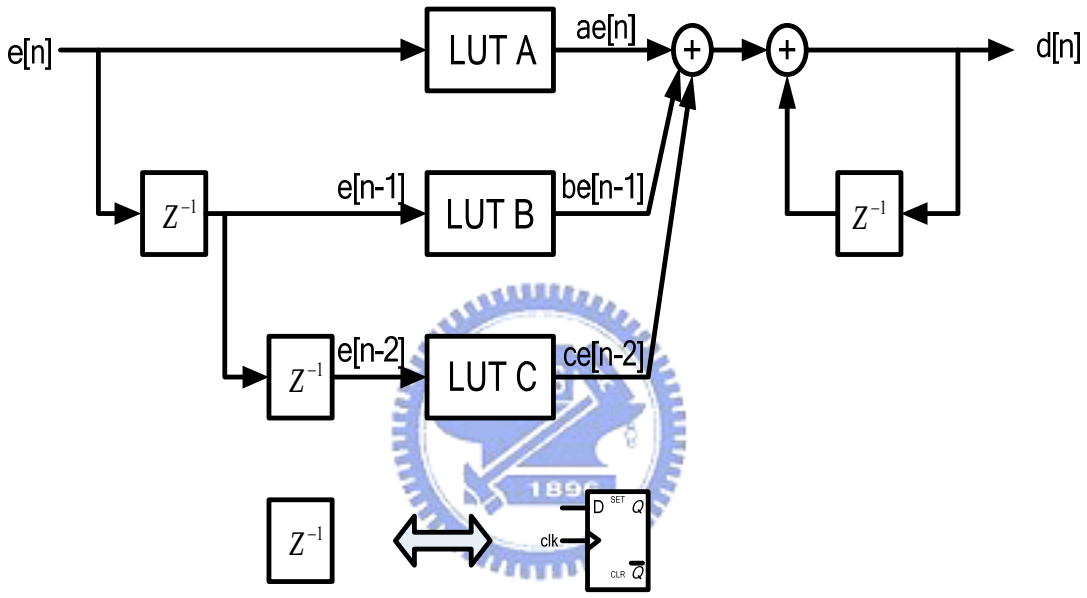


圖 2-4 查表法 PID 結構圖

2.4 數位脈波寬度調變器

目前大多數的交換式電源轉換器的控制電路都採用脈波寬度調變(PWM)方式控制，在整個系統中，數位脈波寬度條變器所提供的功能其實就是一個數位至類比的轉換器（D/A Converter）。因為它所輸出的是

一個頻率固定寬度可依據 $d[n]$值而調變的信號（Duty Ratio）。這個固定頻率的脈波信號會輸出至切換式直流至直流轉換器的開關元件，利用不同的寬度調變，決定功率電晶體(Power MOS)導通與截止時間的長短，如此就可以調整我們所要的輸出電壓。

由於Duty Ratio 可以調整直流至直流轉換器的輸出電壓，因此數位脈波寬度調變器的解析度就格外顯得重要。假如數位脈波寬度調變器的解析度不夠高時，直流至直流轉換器的輸出電壓便會振盪，如圖2-5所示輸出電壓在穩態的情形下有一震盪現象而不會趨於穩定，這種現象稱limit cycle [4]。這是因為數位脈波寬度調變器所調整的輸出電壓增量值沒有落在相對應於前端ADC參考電壓（V_{ref}）的輸出電壓改變量（ΔV_o）的範圍內，因此就會造成輸出電壓值的振盪。解決的方式就是對應於 $d[n]$值最小位元（Least significant bit, LSB）的 V_o增量值必須小於ΔV_o。即

resolution(DPWM)> resolution(ADC)　　（2-2）

DPWM 所能調整的電壓變化量必須小於ADC的電壓取樣量。如此才能使輸出電壓穩定，如圖2-6所示。

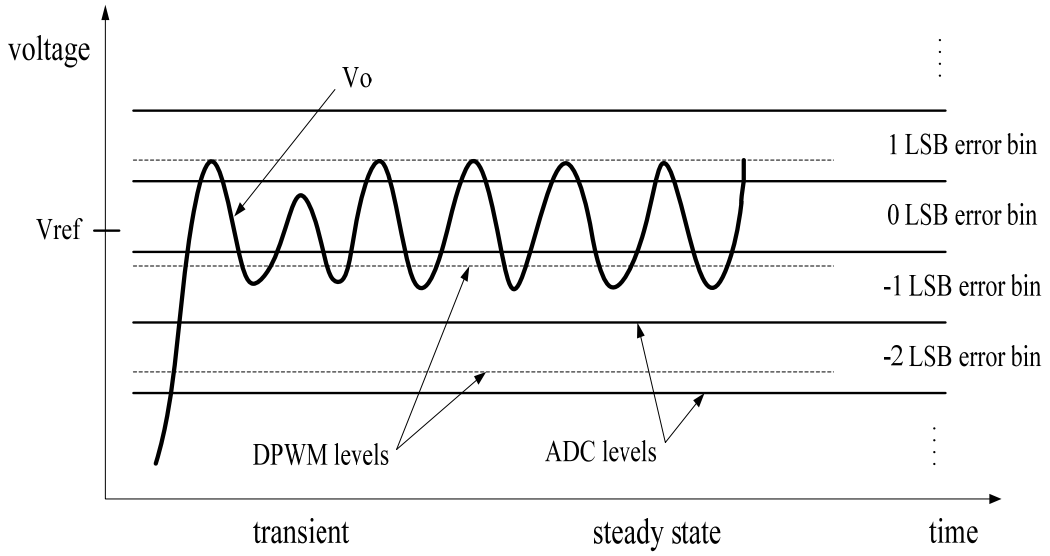


圖 2-5 resolution（DPWM）< resolution（ADC）波形圖

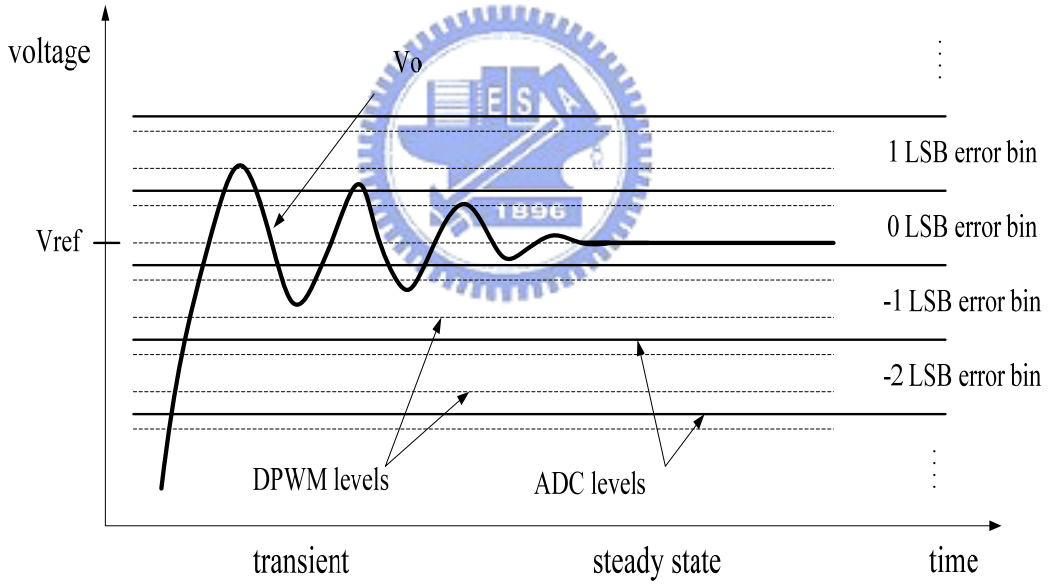


圖 2-6 resolution（DPWM）> resolution（ADC）波形圖

2.4.1 數位脈波寬度調變器之架構

一般來說 DPWM 的形式可分為三種[5]，第一種是計數型的 DPWM，它是利用除頻的方式將一個相當高的頻率經過計數器除頻來產

生開關比率(Duty Ratio)的方波，藉由計數器計數的值與命令週期(Duty command)做比較，來決定方波大小。圖 2-7 為計數型 DPWM 的架構圖，當 CLK 為正緣時，計數器就會加 1。當計數器狀態為 0 時開關比率(Duty Ratio)波形就會為 ON 的狀態，而計數器會跟比較器做比較。當比較器輸入 Duty Command 也就是開關比率(Duty Ratio)的輸入數值跟計數器狀態相等的時候，比較器將輸出 1 使 RS 正反器 Reset，讓 PWM output 呈現 OFF 的狀態，所以我們可以用比較器的輸入 Duty Command 來指定相對應開關比率(Duty Ratio)的方波輸出。在圖 2-7 中，當 n=3 時即可得到 8 bits 的解析度。Duty Command 為三個 bits ， CLK 所用的頻率是輸出頻率的 8 倍。圖 2-8 為 3-bits 的計數型數位脈波寬度調變產生器(DPWM)波形圖。

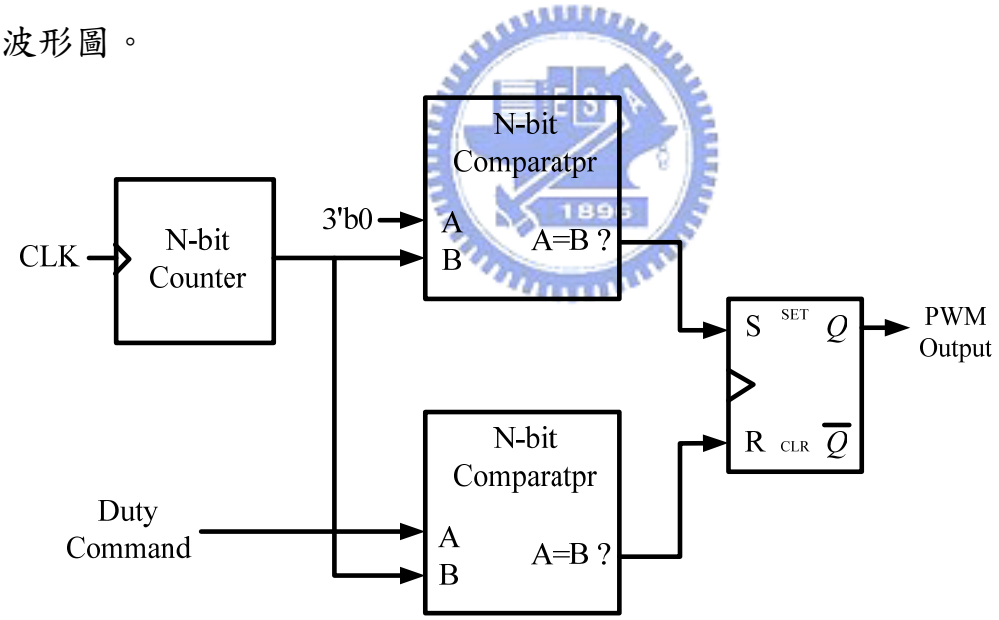


圖 2-7 3-bits 的計數型數位脈波寬度調變產生器(DPWM) 架構圖

計數型的 DPWM 需要相當高的頻率來除頻，若一個切換頻率 f_s=1MHz 要達到 8 bits 的解析度(Resolution)，系統頻率就需要 256 MHz 來做除頻。由此可知要將工作頻率在 f_s 的方波達到 n-bits 的解析度就需

要$2^n f_s$的除頻頻率。所以要增加 DPWM 的解析度可以用增加除頻頻率來達成，可是過高的除頻頻率將會造成極大的功率消耗。

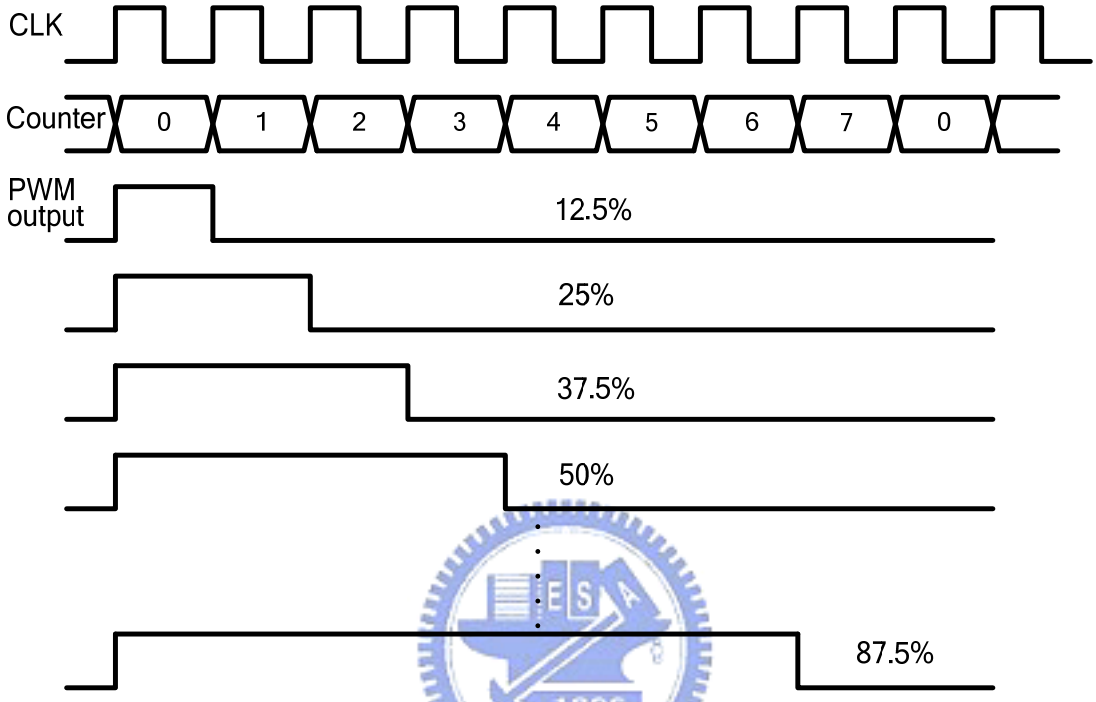


圖 2-8 計數型數位脈波寬度調變產生器(DPWM) 波形圖

第二種 DPWM 是採用 Delay Line 與多工器的架構來產生 DPWM 的開關比率(Duty Ratio)輸出。電路架構圖為圖 2-9，當 CLK 為正緣的時候開關比率(Duty Ratio)的方波輸出就為 ON，設定 RS 正反器的輸出為 1。另外 test 波形會輸入到 Delay-Line 中，並利用多工器去選取適當的重置(Reset)的時間將 RS 正反器 Reset，讓 PWM output 波型為 OFF。因此多工器輸入就可以決定開關比率(Duty Ratio)來得到相對應的方波輸出。Delay-line 形式的 DPWM 控制波形控制波形如圖 2-10。

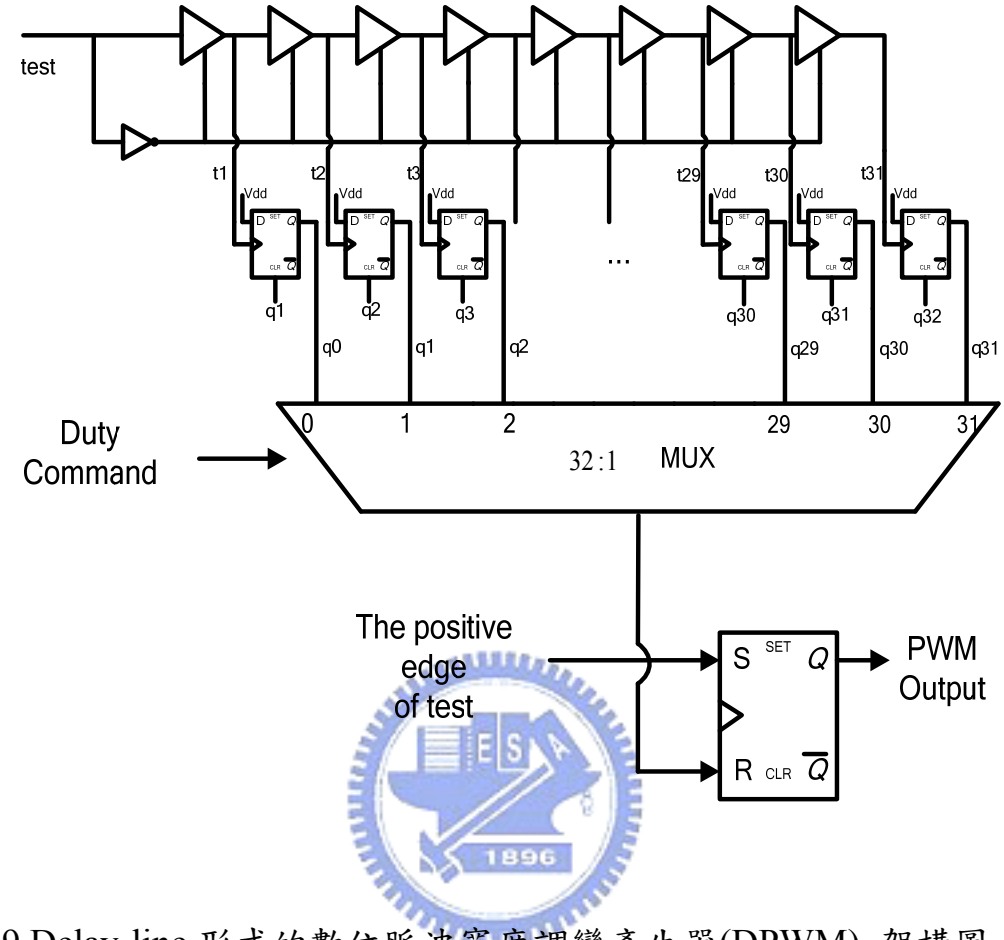


圖 2-9 Delay-line 形式的數位脈波寬度調變產生器(DPWM) 架構圖

Delay-line 形式 DPWM 雖然不需要太高的頻率就可以達到高解析度(Resolution)。但其高解析度的代價是需要多級的 Delay 元件及龐大 bits 的多工器來完成 Delay Line 解碼工作。例如要達到 5 bits($2^5 = 32$)的解析度，就需要 32 級 Delay-Cells 的 Delay-Line 架構與 32 bits 的多工器。相較於除頻式雖然節省功率消耗但是卻需要更多的硬體成本。

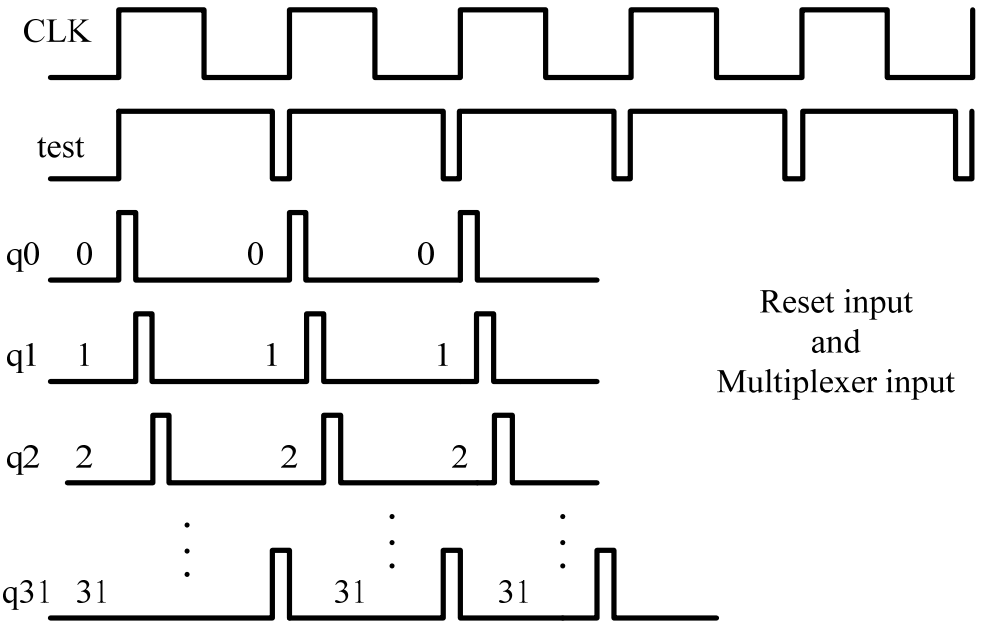


圖 2-10 Delay-line 形式的數位脈波寬度調變產生器(DPWM)控制波形

將上述兩種 DPWM 的優缺點做一個整理，如表 2-1。

Type	Fast-counter	Delay-line
優點	小面積	低時脈頻率 低電源消耗
缺點	解析度越高所需 計數頻率越高	解析度越高所需 晶片面積越大

表 2-1 二種 DPWM 的優缺點比較

第三種是混合式(Hybrid)的DWPM [3]，此種DPWM是將計數型及Delay-line型相結合，其擁有兩者的優點，可以使 DPWM在面積與功率消耗取得一個平衡。圖 2-11 為混合式的數位脈波寬度調變器的架構圖。

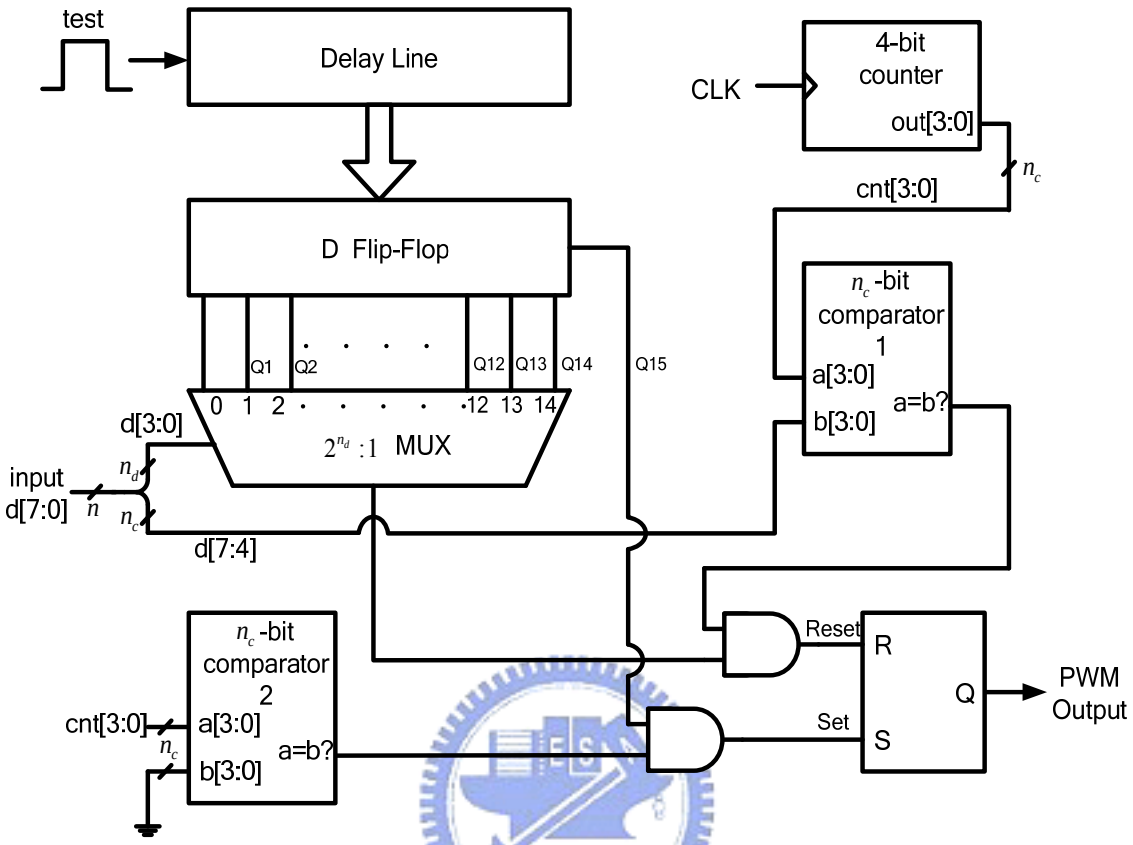


圖 2-11 混合式數位脈波寬度調變產生器(DPWM) 的架構圖

這一個8 bits混合式DPWM採用了4 bits計數型DPWM與4 bits Delay-line型的DPWM，其中包括了一個$2^{nd}\times1$多工器、16個D型正反器（$2^{nd}=16$）、4-bit 計數器（nc=4）、二個4-bit 比較器、二個AND 閘及一個RS 正反器。在此電路中用4 bits的計數器去將頻率除以為1/16。而用多工器去選取這1/16頻率的1/16的重置(Reset) RS暫存器的時間。工作原理說明如下，補償器的輸出（$d[n]$）分成兩部份輸入至數位脈波寬度調變器，較高的4-bit（d[7:4]）輸入至比較器，較低的4-bit（d[3:0]）輸入至多工器的選擇端。當電路啟動時，數位脈波寬度調變器的輸出（PWM output）先被RS 正反器設為〝1〞。計數器的輸出送至當4-bit 比

較器去與*d*[7:4]作比較，當兩數相同時，比較器輸出一個信號至RS 正反器R 端的AND 閘。而多工器亦會輸出信號至RS 正反器 R端的AND閘，假如這兩者信號皆為〝1〞時，RS 的R 端就會被設定為〝1〞。此時，RS 正反器處為重置狀態，PWM output 被設為〝0〞。所以當*d*[7:0]不同時，PWM output就會不同寬度的調變。控制波形如圖2-12，最下面的波形為開關比率(Duty Ratio)在35/256= 13.67 % 的時候的輸出波形。

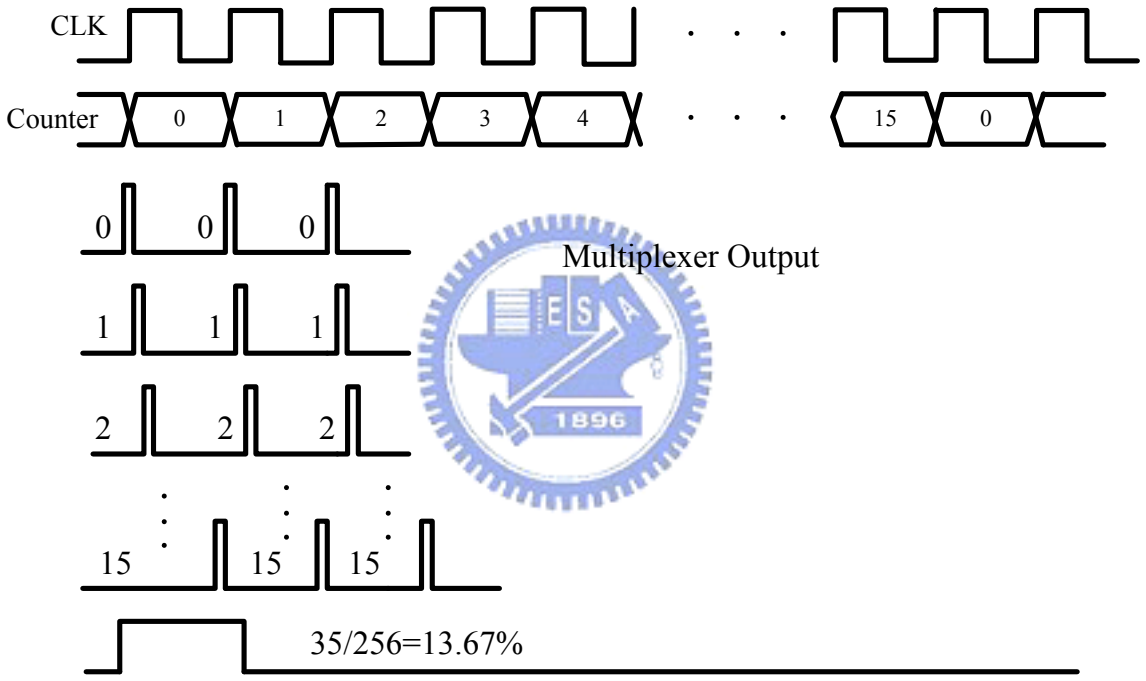


圖 2-12 混合式數位脈波寬度調變產生器(DPWM) 控制波形

2.5 數位顫抖控制(Digital Dither)設計

數位顫抖控制是一種增加DPWM解析度的方法[4]，與類比顫抖控制相比較為容易產生及控制，數位顫抖控制是藉由在數個開關週期內LSB變化來改變開關比率(Duty Ratio)，使得平均的開關比率(Duty Ratio)可以落在二相鄰位準之間。而〝平均〞動作是由外部的LC濾波器實現。

數位顫抖控制動作說明如圖 2-13，D_{c1} 與 D_{c2} 分別表示二相鄰DPWM產生的開關比率(Duty Ratio)位準，其中 $D_{c2}=D_{c1}+LSB$，若每一次開關週期 D_{c1} 與 D_{c2} 輪流產生，則平均的開關比率(Duty Ratio)可表示成: $(D_{c1}+D_{c2})/2=D_{c1}+(1/2)LSB$。由此可知，藉由二個開關週期的平均，可以得到一個介於 D_{c1} 與 D_{c2} 二者中間(1/2)*LSB*的位準，相當於增加了DPWM一位元有效的解析度。利用相同的方法，在四個開關週期中可產生(1/4)*LSB*及(3/4)*LSB*位準如圖 2-14，相當於增加二位元DPWM有效解析度。

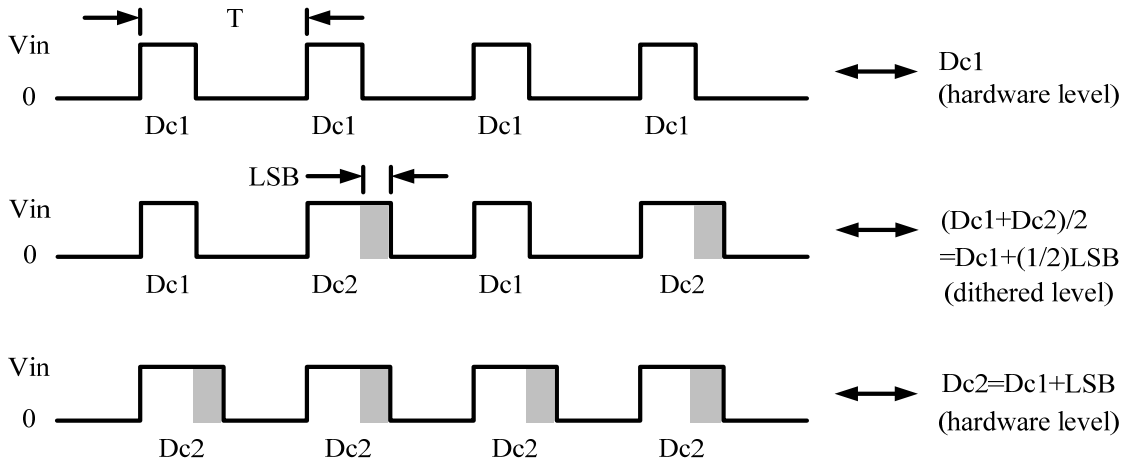


圖 2-13 1-bit dither

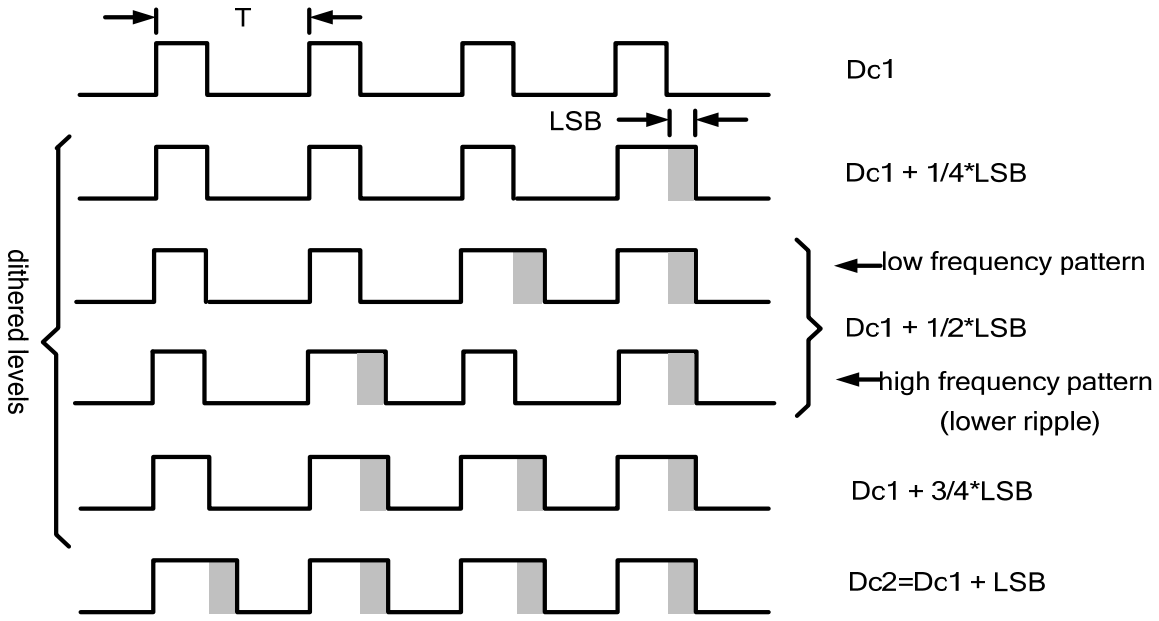


圖 2-14 2-bit dither

因此，在$2^{N_{dith}}$開關週期中可得到有效的DPWM解析度:

$$N_{dpwm,eff} = N_{dpwm} + N_{dith} \qquad (2\text{-}3)$$

其中N_{dpwm}為DPWM解析度，N_{dith}為數位顫抖控制位元數。

雖然數位顫抖控制可以增加DPWM有效解析度，但由於數位顫抖平均效果是藉由外部*LC*濾波器實現，所以在數位顫抖過程中會在外部*LC*濾波器上產生額外的AC漣波效應，因此數位顫抖控制位元數必須有所限制。

另外，在實現數位顫抖控制時，有些不同的數位顫抖控制序列會產生相同平均DC位準，如圖 2-14中(1/2)*LSB*位準可以由二種不同順序的形式實現:$\{D_{c1}, D_{c1}, D_{c2}, D_{c2}\}$或$\{D_{c1}, D_{c2}, D_{c1}, D_{c2}\}$，後面的形式有較高的頻

率，在外部低通濾波器的特性下，可降低輸出電壓的漣波，相較之下是比較建議使用的。

以3-bit數位顫抖控制為例，其序列表如表2-2，其中〝1〞表示在開關比率(Duty Ratio)加入一*LSB*，表2-2 表示一般直觀的數位顫抖控制序列[5]，表2-3 是針對產生最小漣波加以變形的數位顫抖控制序列，二表比較結果可發現表2-3 產生漣波效應較小，以1/8即{0,0,0,0,0,0,0,1}為例，在表2-3 是最差的狀況卻等於表2-2 最好的狀況，所以我們將採用表2-3 的數位顫抖控制序列設計。

Sequence Average	Dither Sequence	Ripple
0	0 0 0 0 0 0 0 0	none
1/8	0 0 0 0 0 0 0 1	lowest
2/8	0 0 0 0 0 0 1 1	
3/8	0 0 0 0 0 1 1 1	
4/8	0 0 0 0 1 1 1 1	highest
5/8	0 0 0 1 1 1 1 1	
6/8	0 0 1 1 1 1 1 1	
7/8	0 1 1 1 1 1 1 1	lowest

表 2-2 一般的數位顫抖控制序列

Sequence Average	Dither Sequence								Ripple
0	0	0	0	0	0	0	0	0	none
1/8	0	0	0	0	0	0	0	1	highest
2/8	0	0	0	1	0	0	0	1	
3/8	0	0	1	0	0	1	0	1	
4/8	0	1	0	1	0	1	0	1	lowest
5/8	0	1	0	1	1	0	1	1	
6/8	0	1	1	1	0	1	1	1	
7/8	0	1	1	1	1	1	1	1	highest

表 2-3 最小漣波數位顫抖控制序列

2.5.1 數位顫抖控制(Digital Dither)電路實現

圖 2-15為一數位顫抖控制電路內部架構,可以產生最小漣波的數位顫抖控制如表2-3，其內部電路包含一 $N_{dith}-bit$ 之計數器、$2^{N_{diyh}}\times 2^{N_{dith}}$ 容量的表格及一個加法器，其中表格儲存$2^{N_{dith}}$個數位顫抖控制序列，每個序列有$2^{N_{dith}}$位元長度。

PID補償器輸出分成二個部份輸入至數位顫抖控制電路，較低的N_{dith} LSB's位元決定數位顫抖控制序列，再依據計數器的輸出值選擇適當的*LSB*與高位元N_{dpwm} MSB's相加產生Dc'送至DPWM。

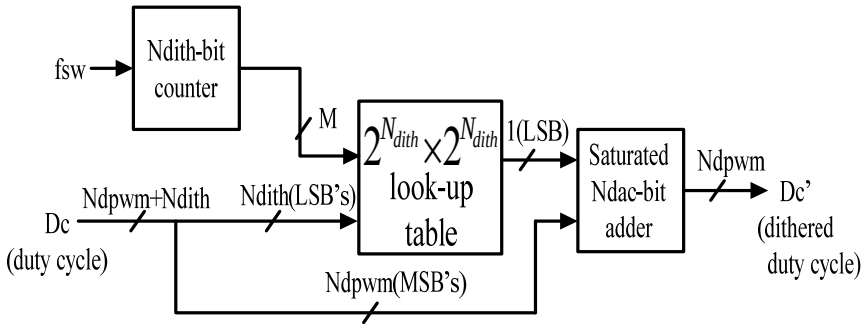


圖 2-15 數位顫抖控制電路內部架構

2.6 盲時(Dead-time)最佳化實現

本論文使用之dead-time最佳化方法[6]，最大的特色是執行dead-time最佳化過程中，不需要任何額外的感測器去感測外部電路信號，只需觀察責任週期命令(duty cycle command)平均值D之變化來調整dead-time大小使系統達到最大的效率。此方法尤其適用數位控制電路，因為其不需要增加任何類比元件或改變驅動電路，只須少量的數位電路即可完成。dead-time最佳化在數位控制器內部之結構如圖 2-16。

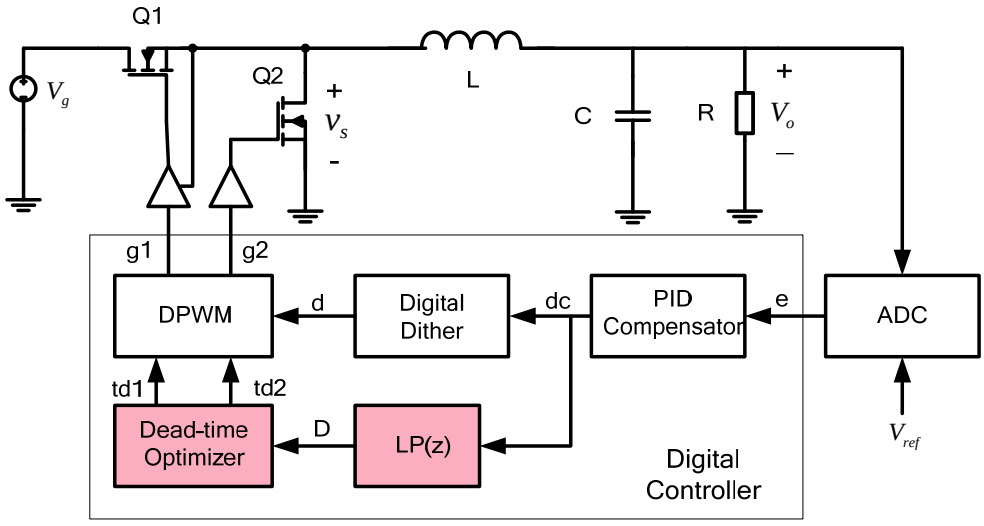


圖 2-16 dead-time最佳化在數位控制器內部之結構

2.6.1 Dead-time最佳化理論

在考量電壓轉換器最大效能$\eta(t_{d1},t_{d2})$我們可以得到下列式子:

$$\underset{t_{d1},t_{d2}}{Max}\ \eta(t_{d1},t_{d2})|_{V_{out}=V_{ref}} \Rightarrow t_{d1opt},t_{d2opt} \quad (2\text{-}4)$$

同時在最佳的dead-time情況下可以的到最小的責任週期命令(duty cycle command)平均值D即:

$$\underset{t_{d1},t_{d2}}{Min}\ D(t_{d1},t_{d2})|_{V_{out}=V_{ref}} \Rightarrow t_{d1opt},t_{d2opt} \quad (2\text{-}5)$$

原理說明如下，在圖 2-17中分別表示切換點電壓(switch node voltage)、閘極輸出波形在（a)最佳dead-time、(b)過長dead-time、(c)不足dead-time情況下之圖形，責任週期命令(duty cycle command)平均值D必須增加(〝+〞區域)去補償dead-time不佳所造成的損耗(〝—〞區域)。

當dead-time t_{d1}太長(圖 2-17 (b))，會導致body diode效應產生一負電壓於v_s上，如此將使平均切換點電壓(switch node voltage)<v_s>下降，為維持電路要求即$V_o = V_{ref}$，責任週期命令(duty cycle command)平均值D將會增加去補償平均切換點電壓(switch node voltage)<v_s>的損耗。

在圖 2-17 (b)中平均切換點電壓(switch node voltage)<v_s>損耗可表示成:

$$\Delta V_{s^-} = -V_D \frac{t_d}{T_s} \quad (2\text{-}6)$$

其中:T_s是開關週期，V_D是body diode的電壓降。

同理，不足dead-time情況時，因為閘極驅動信號的overlap，將會造成$v_s(t)$的損耗，使平均切換點電壓(switch node voltage)<v_s>下降，責任週期命令(duty cycle command)平均值D增加。在圖 2-21 (c)中平均切換點電壓(switch node voltage)<v_s>損耗可表示成:

$$\Delta V_{s^-} = -\frac{R_{on1}V_g}{R_{on1}+R_{on2}}\frac{t_{d2}}{T_s} \qquad (2\text{-}7)$$

其中: V_g 為輸入電壓，T_s 是開關週期，R_{on1}、R_{on2} 分別為為 Q_1 及 Q_2 之 on-resistances。

由以上分析可知，在非理想的情況下(即dead-time非最佳化時)，責任週期命令(duty cycle command)平均值D必須增加以補償平均切換點電壓<v_s>造成之損耗。由此特性可知最佳的dead-time值將發生在責任週期命令(duty cycle command)平均值D最小時。我們將根據此特性，利用演算法來求出最佳的dead-time值，相關說明於之後的章節介紹。

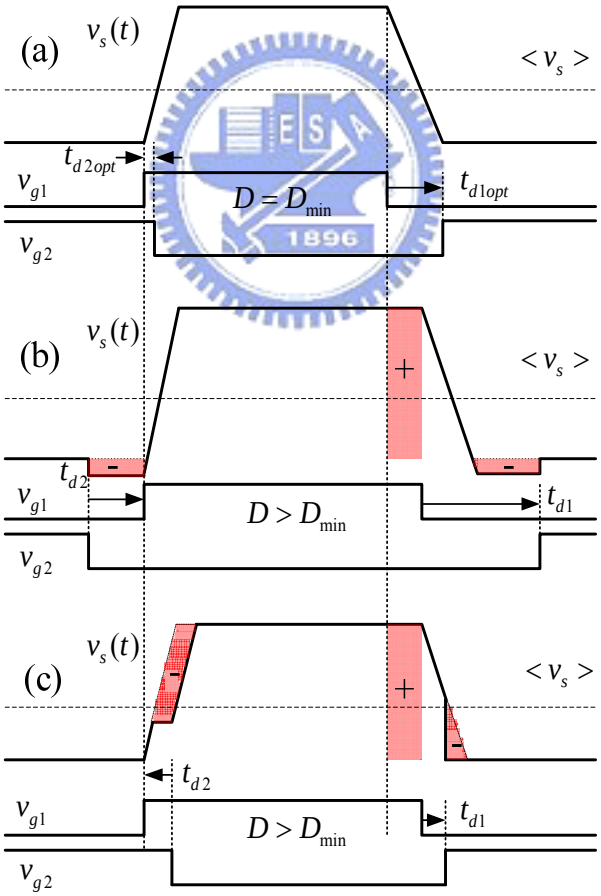


圖 2-17 $v_s(t)$、dead-time與責任週期命令關係圖

2.6.2 低通濾波器 LP(z)

低通濾波器的功用是要計算出責任週期命令(duty cycle command)平均值D，其表示式如下:

$$D[n] = (1-\alpha)D[n-1] + \alpha \cdot d[n] \qquad (2\text{-}8)$$

將(2.8)式取z轉換可得轉移函數:

$$LP(z) = \frac{D}{d} = \frac{\alpha \cdot z}{z-(1-\alpha)} \qquad (2\text{-}9)$$

再將(2-8)式整理可得

$$D[n] = D[n-1] + \alpha \cdot d[n] - \alpha \cdot D[n-1] \qquad (2\text{-}10)$$

取α為$1/2^p$，p為一正整數，則(2-10)式乘法計算可由移位運算所取代，實現方塊圖如下:

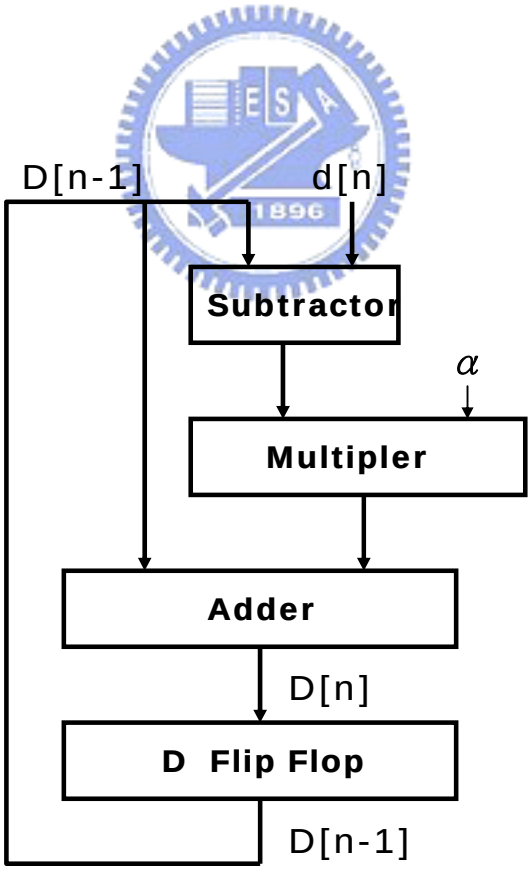


圖 2-18 低通濾波器實現方塊圖

2.6.3 Dead-time最佳化演算法

DPWM共有三個輸入:責任週期命令(d)、dead-time時間命令t_{d1}及t_{d2}；二個輸出:控制信號$g1$及$g2$，送給功率電晶體(Power MOS) Q1及Q2，其操作示意圖如圖 2-19 所示:

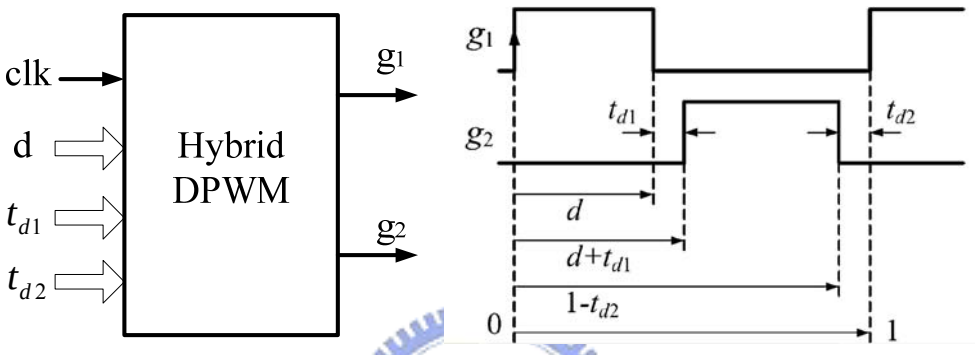


圖 2-19 dead-time控制示意圖

$g2$於$d+t_{d1}$時啟動，在$1-t_{d2}$時重置，t_{d1}及t_{d2}由dead-time最佳化演算法求得。

依據前面介紹的觀念，我們可以利用dead-time最佳化演算法，從最小的穩態責任週期命令D中，尋找最佳的dead-time時間來獲得系統最好的效能。演算法流程圖如圖 2-20 所示，在輸入電壓及負載固定的條件下執行dead-time的尋找，一開始由一觸發信號啟動，t_d初始值設為$t_{d\max}$，在演算法過程中t_d將減一Δt_d後再比較D的數值大小，在t_d減少的過程中，D值亦逐漸變小，直到發現一旦D值變大此時即結束演算法。

結束演算法後，將產生的最佳化dead-time數值t_{dopt}送至DPWM，此演算法將執行二次，第一次先求出t_{d1}，第二次再求出t_{d2}。模擬波形範例如圖 2-21。

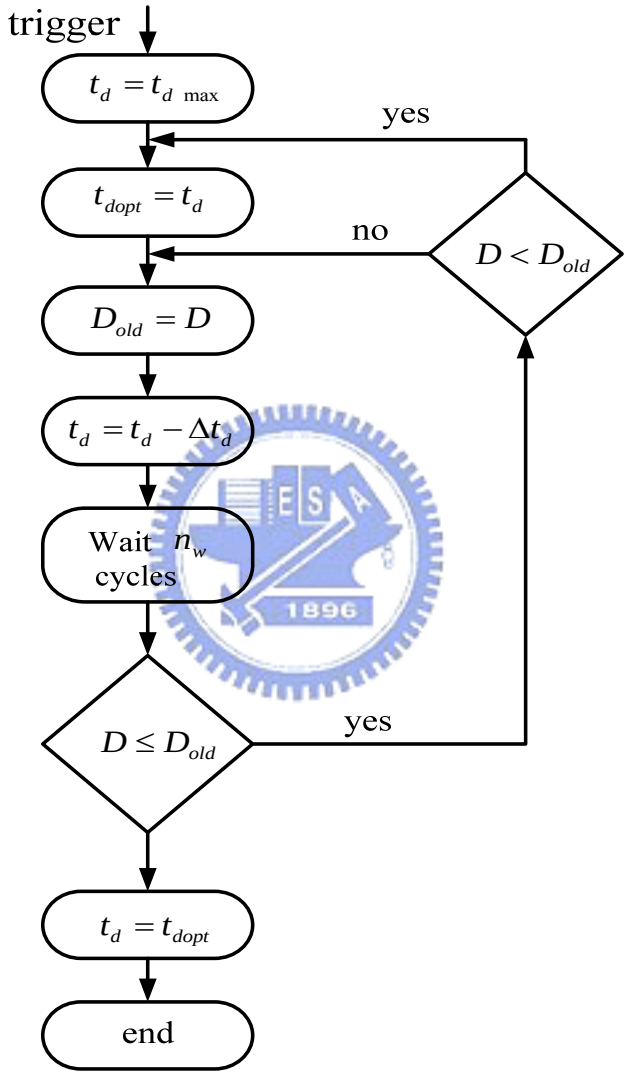


圖 2-20 dead-time最佳化演算法流程圖

第3章數位控制器設計與系統模擬

3.1 數位控制器介紹

數位控制器可以利用電腦、微處理器、數位訊號處理器（DSP）等來實現，與類比的控制器相比較，使用數位控制器最大的優點，就是數位控制器可以藉由改變程式容易地修改，而類比的控制器一旦建立之後，便很難改變其元件。

一般數位控制器的Z-轉移函數寫成(4.1)式的形式[7]。

$$D(z)=\frac{b_0+b_1z^{-1}+\cdots+b_mz^{-m}}{a_0+a_1z^{-1}+\cdots a_nz^{-n}} \tag{3-1}$$

其中n 和m 為正整數，若輸出並未搶先在輸入之前發生，則轉移函數D(z)稱為實際可實現。此表示D(z)的級數展開不能有任何z的正冪次。在(4.1)式的D(z)項中，若$b_0\neq 0$，則 $a_0\neq 0$。若D(z)表示為

$$D(z)=\frac{b_0+b_1z^{1}+\cdots+b_mz^{m}}{a_0+a_1z^{1}+\cdots a_nz^{n}} \tag{3-2}$$

則實際可實現性的條件為$n\geq m$。

3.1.1 Z轉換

對於一離散或數位資料的線性系統及線性差分方程式，適合使用Z 轉換[7]。

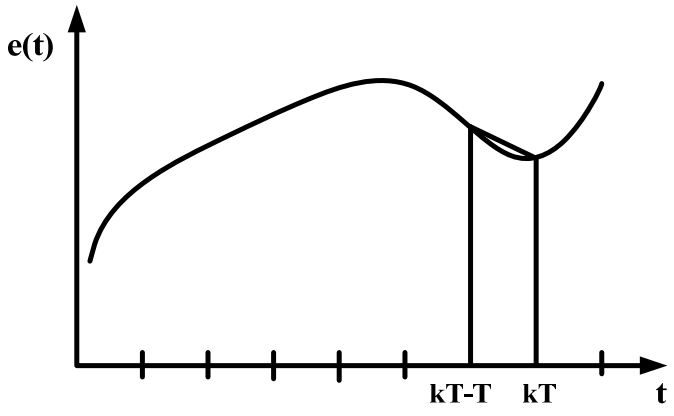
考慮一序列y(k)，k=0，1，2，…，其中y(k)代表一連串的數字或事件。則y(k)的z 轉換定義為

Y(Z)=y(k)的z轉換$= Z[y(k)] = \sum_{k=0}^{\infty} y(k)z^{-1}$ （3-3）

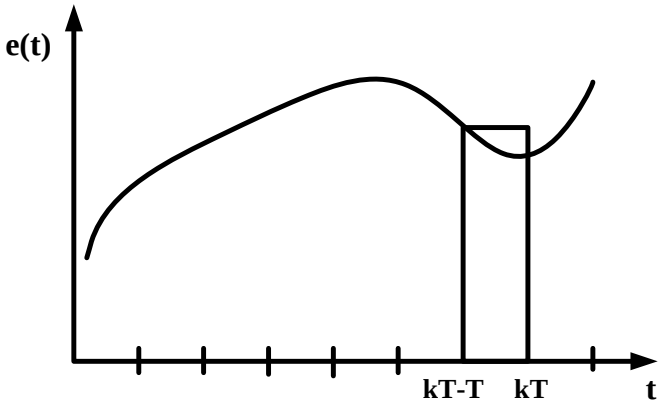
其中z 是具有實部和虛部的複變數。

3.1.2 數值積分等效法

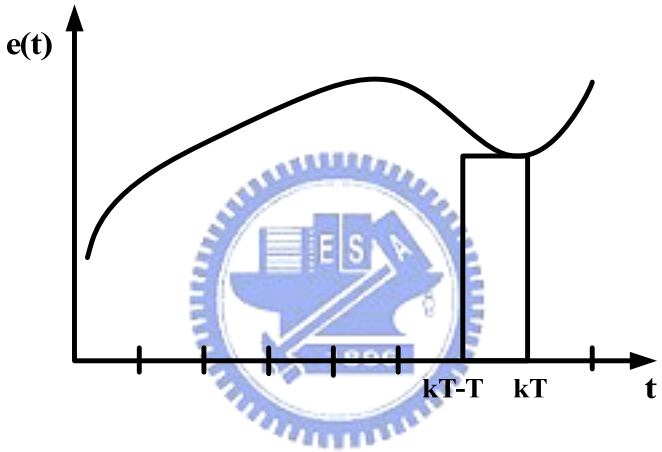
一般的控制系統轉移函數(Transfer Function)，大多是用來描述連續(continuous)訊號的系統。如果系統要以數位方式來呈現就必須要將轉移函數改寫成離散(discrete)訊號的系統來做分析。根據A/D轉換器取樣(Sample)的頻率，離散的轉移函數也許不會完全近似連續的轉移函數。如果取樣頻率愈高，我們所得到離散轉移函數就會近似於連續的控制系統。本節主要探討如何將連續的轉移函數利用數學等效的方式得到離散的轉移函數，以方便後續章節的轉移函數補償與控制器設計。以下介紹三種數值積分等效方法，如圖 3-1所表示。



(a) Tustin Rule



(b) Forward Rectangular Rule



(c)Backward Rectangular Rule

圖 3-1 三種面積近似的方法從 kT 到 kT+T. 近似的方法有(a) Tustin rule (b) Forward rectangular rule (c) Backward rectangular rule.

1.Tustin方法:

Tustin方法或稱雙線性近似(bilinear approximation)是將函數$e(t)$以下的區域以一系列的梯型來近似，如圖 3-1(a)所示，在二個取樣值間，以直線來近似$e(t)$。假設

$$\frac{U(s)}{E(s)} = D(s) = \frac{1}{s} \tag{3-4}$$

其為積分，故

$$u(kT)=\int_0^{kT-T} e(t)dt+\int_{kT-T}^{kT} e(t)dt \tag{3-5}$$

其可被改寫為

$u(kT)=u(kT-T)+e(t)$ 在一個T時間下的面積

為簡便起見，將$u(kT)$寫成$u(k)$，$u(kT\text{-}T)$寫成$u(k\text{-}1)$，如此我們可將(4.4)式改寫為

$$u(k)=u(k-1)+\frac{T}{2}[e(k-1)+e(k)] \tag{3-6}$$

對(3-6)式取其z-轉換可得

$$\frac{U(s)}{E(s)}=\frac{T}{2}\left(\frac{1+z^{-1}}{1-z^{-1}}\right)=\frac{T}{2}\left(\frac{z+1}{z-1}\right) \tag{3-7}$$

即 $s=\frac{2}{T}\left(\frac{z-1}{z+1}\right)$，將s代入任意$D(s)$中即可求得基於梯型積分近似之離散函數$D(z)$。

2.Forward-rectangular方法:

這一個表示在圖 3-1 (b) Forward Rectangular 方法，是將取樣的兩點面積取前一點的大小為長，取樣時間為寬所得到的面積近似成長方形。如公式:

$$u(k)=u(k-1)+Te(k) \tag{3-8}$$

對　　式取其z-轉換可得

$$\frac{U(s)}{E(s)}=\frac{T}{1-z^{-1}}=\frac{Tz}{z-1} \tag{3-9}$$

2.Backward-rectangular方法:

而 Backward Rectangular 方法在 圖 3-1(c). 他將長方形的長從後面

一點看過來，$e(t)$在 $t=Kt$ 時的積分近似為

$$u(k)=u(k-1)+Te(k-1) \tag{3-10}$$

對　式取其z-轉換可得

$$\frac{U(s)}{E(s)}=\frac{Tz^{-1}}{1-z^{-1}}=\frac{T}{z-1} \tag{3-11}$$

以上可以做一個總結，s-domain 到 z-domain 的映射關係在表 3-1

Method	Approximation
Forward Rule	$s \leftarrow \frac{z-1}{T}$
Backward Rule	$s \leftarrow \frac{z-1}{Tz}$
Bilinear Rule	$s \leftarrow \frac{2}{T}\frac{z-1}{z+1}$

表 3-1 三種近似方法的互相映射關係在從 s-domain 到 z-domain

同樣的我們也可以將這幾種方法的 z-domain 到 s-domain 的映射。這一個用 z 取代 s 的的關係，我們整理在表 3-2

Method	Approximation
Forward Rule	$z=1+Ts$
Backward Rule	$z=\frac{1}{1-Ts}$
Bilinear Rule	$z=\frac{1+Ts/2}{1-Ts/2}$

表 3-2 z-domain 到 s-domain 的取代表示法整理

3.1.3 離散系統穩定性分析

對於一個離散系統而言，要判斷該系統是否穩定，我們可以從它特性方程式的根落在z-平面上的位置去判定[7]。

現假設有一個離散資料系統，其系統轉換函數如圖 3-2所示。

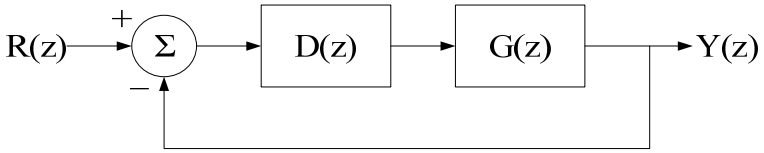


圖 3-2 閉迴路離散資料系統方塊圖

由圖3-3 知，該系統的閉迴路轉移函數為：

$$\frac{Y(z)}{R(z)}=\frac{D(z)G(z)}{1+D(z)G(z)} \tag{3-12}$$

為了討論此一閉迴路系統的特性，我們必須算出離散特性方程式(4.10式)的根。

$$1+D(z)G(z)=0 \tag{3-13}$$

當特性方程式的所有根均落於z平面的單位圓 |z|=1 之內，如圖 3-3所示，則我們可以說該系統為穩定。

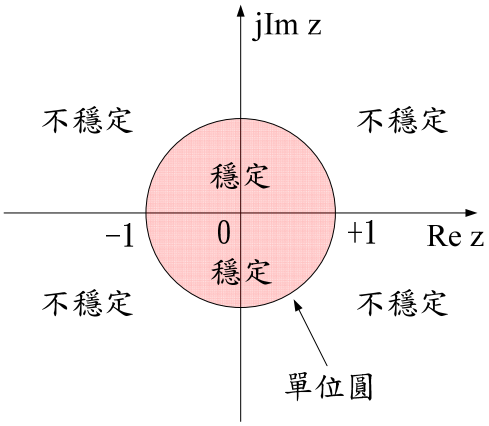


圖 3-3 z-平面

3.2 數位PID控制器設計

3.2.1 數位PID控制器介紹

PID控制器的歷史非常悠久，它的構思首見於1922年Minorsky的文獻[8]，而PID控制器的原型出現在Callender等人1936年的文獻上[9]。著名的Ziegler and Nichols則在1942年提出PID控制器的調整法[10]，歷經半世紀，它一直是現今產業界裡應用廣泛、實用性相當高的控制器。

PID 控制器是由比例增益、積分、微分三個動作組合而成的控制器，其結合PI 和PD 控制器的優點，能改善系統閉迴路的相對穩定度，使暫態響應較快，減少穩態誤差。在連續時間的PID 控制法如下式

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{d}{dt} e(t) \tag{3-14}$$

其中，$u(t)$:控制訊號

$e(t)$:誤差訊號

K_p:比例增益

K_i:積分增益

K_d:微分增益

而類比PID控制器轉移函數可以在s-domain下表示為

$$D(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s \tag{3-15}$$

為了得到數位式PID控制器，我們用梯型面積之和近似積分項，並以二不同時間點誤差的差值來近似微分項，T_s為取樣時間(Sampling Time)，則可推導出離散時間PID控制器，推導過程如下：

$$u(k) = K_p e(k) + K_i T_s \left(\frac{e(0)+e(1)}{2} + \frac{e(1)+e(2)}{2} + ... + \frac{e(k-1)+e(k)}{2} \right) + K_d \frac{e(k)+e(k-1)}{T_s}$$

$$= K_p e(k) + K_i T_s \sum_{h=1}^{k} \frac{e(h)+e(h-1)}{2} + \frac{K_d}{T_s}[e(k)+e(k-1)] \quad (3\text{-}16)$$

定義 $f(h) = \frac{e(h)+e(h-1)}{2}$ ， $f(0) = 0$

將上式做z轉換，可得

$$Z[\sum_{h=1}^{k} \frac{e(h)+e(h-1)}{2}] = Z[\sum_{h=1}^{k} f(h)] = \frac{1}{1-z^{-1}}[F(z) - f(0)] = \frac{1}{1-z^{-1}} F(z) \quad (3\text{-}17)$$

其中 $F(z) = Z[f(h)] = \frac{1+z^{-1}}{2} E(z)$

因此 $Z[\sum_{h=1}^{k} \frac{e(h)+e(h-1)}{2}] = \frac{1+z^{-1}}{2(1-z^{-1})} E(z)$

將 式做z轉換可得

$$U(z) = [K_p + \frac{1}{2} K_i \cdot T_s \cdot \frac{1+z^{-1}}{1-z^{-1}} + \frac{K_d}{T_s}(1-z^{-1})] E(z)$$

$$= [K_p - \frac{1}{2} K_i + \frac{K_i \cdot T_s}{1-z^{-1}} + \frac{K_d}{T_s}(1-z^{-1})] E(z)$$

所以 $\frac{U(z)}{E(z)} = \left(K_p - \frac{1}{2} K_i \right) + \frac{K_i \cdot T_s}{1-z^{-1}} + \frac{K_d}{T_s}(1-z^{-1})$

$$= K_P + \frac{K_I}{1-z^{-1}} + K_D(1-z^{-1}) \quad (3\text{-}18)$$

其中，$K_P = K_p - \frac{1}{2} K_i$ ，為數位PID控制器比例增益。

$K_I = K_i \cdot T_s$ ，為數位PID控制器積分增益。

$K_D = \frac{K_d}{T_s}$ ，為數位PID控制器微分增益。

由上式可知，將連續時間之PID控制器離散化，其數位PID控制器之參數會隨著取樣時間改變而改變。若是應用於數位控制上，則PID控制器的差分方程表示式如下:

$$u(k)=u(k-1)+K_P[e(k)-e(k-1)]+K_I\cdot e(k)+K_D[e(k)-2e(k-1)+e(k-2)] \quad (3\text{-}19)$$

上式整理可得

$$u(k)=u(k-1)+(K_P+K_I+K_D)e(k)+(-K_P-2K_D)e(k-1)+K_De(k-2) \quad (3\text{-}20)$$

我們定義:

A= $K_P+K_I+K_D$

B= $-K_P-2K_D$

C= K_D

所以可寫成(3-21)式:

$$u(k)=u(k-1)+A\cdot e(k)+B\cdot e(k-1)+C\cdot e(k-2) \quad (3\text{-}21)$$

當 K_P 、 K_I 、 K_D 設計決定後，A、B、C 便可決定。

由式（3-18）可看出，數位控制器是以乘積來表示，因而必須用到乘法器。由於數位控制器中的A、B、C為已知的固定係數，所以可以建立一個查表法將乘法的動作簡化為查表、加法，如此可以大幅提高運算速度，並節省面積。

3.2.2 Ziegler-Nichols 調整法

在1942年Ziegler和Nichols提出一種實用的PID控制經驗公式，可以容易的計算出PID控制器的相關參數，其應用說明如下，圖 3-4為具有控制器之閉迴路控制系統，令控制器為比例積分微分控制器(PID)，連續時間的PID控制式如下:

$$G_c(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{d}{dt}e(t)$$
$$= K_p \left[e(t) + \frac{1}{T_i} \int e(t)dt + T_d \frac{d}{dt}e(t) \right] \quad (3\text{-}22)$$

其中: $K_i = \frac{K_p}{T_i}, K_d = K_p \cdot T_d$

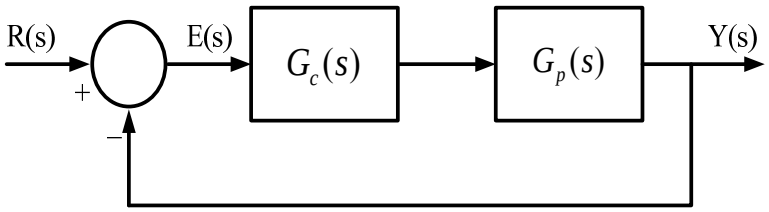


圖 3-4 具有控制器之閉迴路控制系統

Ziegler-Nichols補償法求控制器係數步驟為:

1. 先設$G_c(s)=K_p$，將K_p調整至系統達臨界穩定，此時K_p值即為K_{p0}，而震盪的週期記為T_0。
2. 將步驟1計算所得代入表3-3中，即可設計出PID控制器的參數

控制器類型	參數值		
	K_p	T_i	T_d
P	$K_p = 0.5K_{p0}$		
PI	$K_p = 0.5K_{p0}$	$T_i = 0.83T_0$	
PID	$K_p = 0.6K_{p0}$	$T_i = 0.5T_0$	$T_d = 0.125T_0$

表 3-3 Ziegler-Nichols 調整公式

3.2.3 數位PID控制器參數設計

對於我們所設計的數位脈波寬度調變器中所含的數位PID補償器，它的型式如下列式子所表示：

$$d[n] = d[n-1] + a \cdot e[n] + b \cdot e[n-1] + c \cdot e[n-2] \tag{3-23}$$

只要系數a、b、c 求出來，就可以用查表法來實現該補償器。

首先我們先考慮由數位脈波寬度調變控制積體電路與切換式直流至直流轉換器所組成的一個離散時間閉迴路系統[3]，如圖 3-5所示。

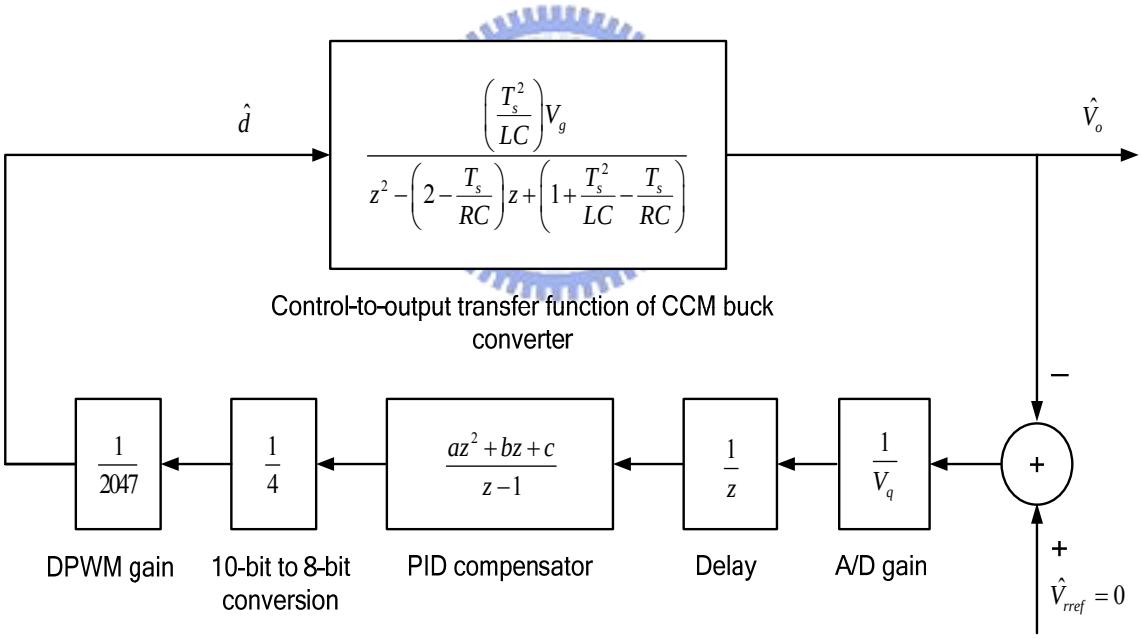


圖 3-5 離散時間閉迴路系統

其中PID 補償器是由（4-20）式取Z 轉換而得到的。

$$D(z) = D(z)z^{-1} + aE(z) + bE(z)z^{-1} + cE(z)z^{-2} \tag{3-24}$$

把　式整理移項：

$$D(z)(1-z^{-1}) = E(z)(a+bz^{-1}+cz^{-2})$$

$$\frac{D(z)}{E(z)} = \frac{a+bz^{-1}+cz^{-2}}{1-z^{-1}} = \frac{az^2+bz+c}{z^2-z} = \frac{az^2+bz+c}{z-1}\frac{1}{z} \tag{3-25}$$

因為其他的值皆為已知：

$V_g = 5V$，$\frac{T_s^2}{LC} = 2.128\times10^{-3}$　$\frac{T_s}{RC} = 7.88\times10^{-3}$　所以Control to output 轉移函數為:

$$Gvd(z) = \frac{\hat{Vo}}{\hat{d}} = \frac{(\frac{Ts^2}{LC})Vg}{Z^2-(2-\frac{Ts}{RC})Z+(1+\frac{Ts^2}{LC}-\frac{Ts}{RC})} \tag{3-26}$$

$$= \frac{0.0109}{Z^2-1.992Z+0.984}$$

利用 Ziegler-Nichols調整法，令$K_I = K_D$=0，即設控制器為K，整個切換式降壓型直流至直流轉換器的閉迴路轉移函數為：

$$T(z) = K\cdot\frac{1}{4}\cdot\frac{1}{2047}\cdot Gvd(z)\cdot\frac{1}{Vq}\cdot\frac{1}{Z}$$

$$= K\cdot\frac{1}{4}\cdot\frac{1}{2047}\cdot\frac{0.0109}{Z^2-1.992\cdot Z+0.984}\cdot\frac{1}{19.5mV}\cdot\frac{1}{Z} \tag{3-27}$$

$$= K\cdot\frac{3.5\cdot10^{-3}}{Z^3-1.992\cdot Z^2+0.984\cdot Z}$$

利用根軌跡觀念找出系統臨界穩定之K值(即系統的閉迴路極點位於z平面的單位元上)，此時的K=K_m，三個參數分別是

$$K_P = 0.6K_m = 1.75 \tag{3-28}$$

$$K_I = \frac{K_p\omega_m}{\pi} = 0.25 \tag{3-29}$$

$$K_D = \frac{K_p\pi}{4\omega_m} = 30 \tag{3-30}$$

其中ω_m為系統的震盪頻率，可由極點位於單位圓上的角度得到。

最後可得:

a=$K_P + K_I + K_D$=32

b=$-K_P - 2K_D$=-62

c=K_D=30

我們可以使用MATLAB計算各項參數，MATLAB Code及根軌跡圖如圖圖 3-5及圖 3-5。

```
num=[3.5e-3]
den=[1 -1.992 0.984 0]
g=tf(num,den,-1)
rlocus(g)
[k,pole]=rlocfind(g)
k=2.9
kp=0.6*k
wm=angle(pole(1)/1e6)
ki=kp*wm/pi
kd=kp*pi/(4*wm)
```

圖 3-6 MATLAB Code

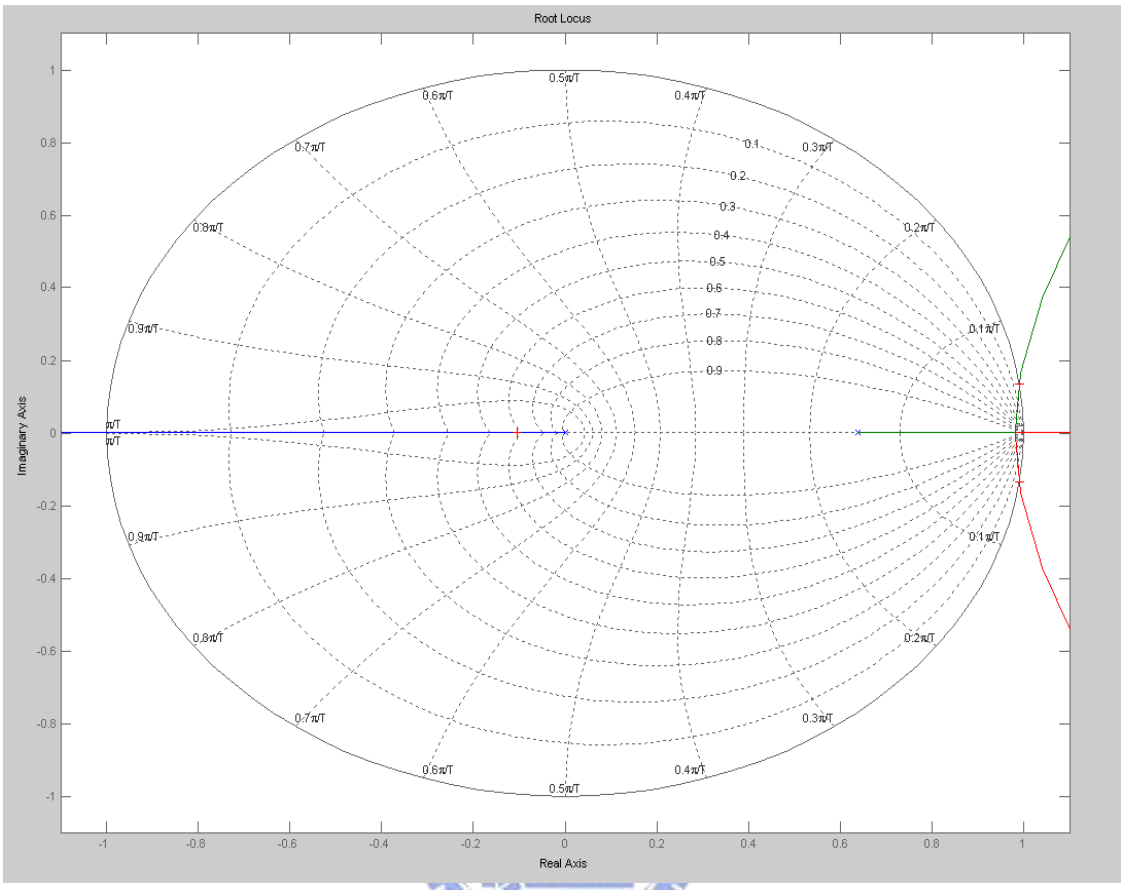


圖 3-7 根軌跡圖

3.3 系統模擬

本論文我們使用模擬軟體MATLAB中的Simulink來進行系統模擬，並同時觀察輸出結果是否穩定及符合我們所設計的要求。

Simulink是一種對動態系統進行模型化、模擬和分析的軟體。它支援連續、離散或是二者混合使用在線性及非線性系統。Simulink提供了圖形化的使用者介面(GUI)，只要使用時點選和拖曳滑鼠就可以操作及建立方塊圖模型。此外Simulink也提供了許多現成的方塊圖庫可共使

用，如電源系統的模擬(SimPowerSystems)，圖3-8 即為用Simulink建立的切換式降壓型直流至直流轉換器電路。

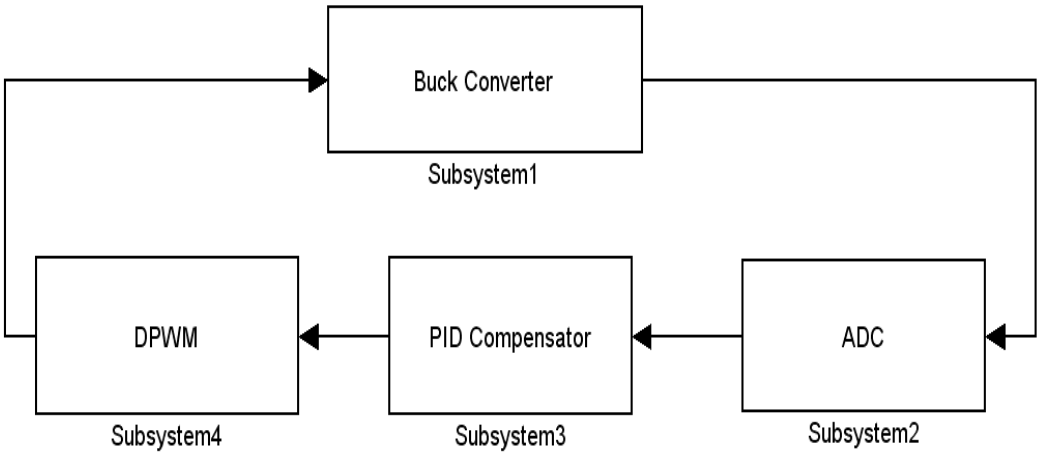


圖 3-8 Simulink之切換式降壓型直流至直流轉換器電路

圖3-8中可分成四大區塊，依序為Buck converter、ADC、PID compensator及DPWM，其各區塊之功能介紹如下:

圖3-9為Buck converter之內部架構圖，模擬5V降至2.7V之直流電源轉換器，我們將量測電感電流、輸出電壓及輸出電流並用Scope功能來觀察電壓及電流的變化。

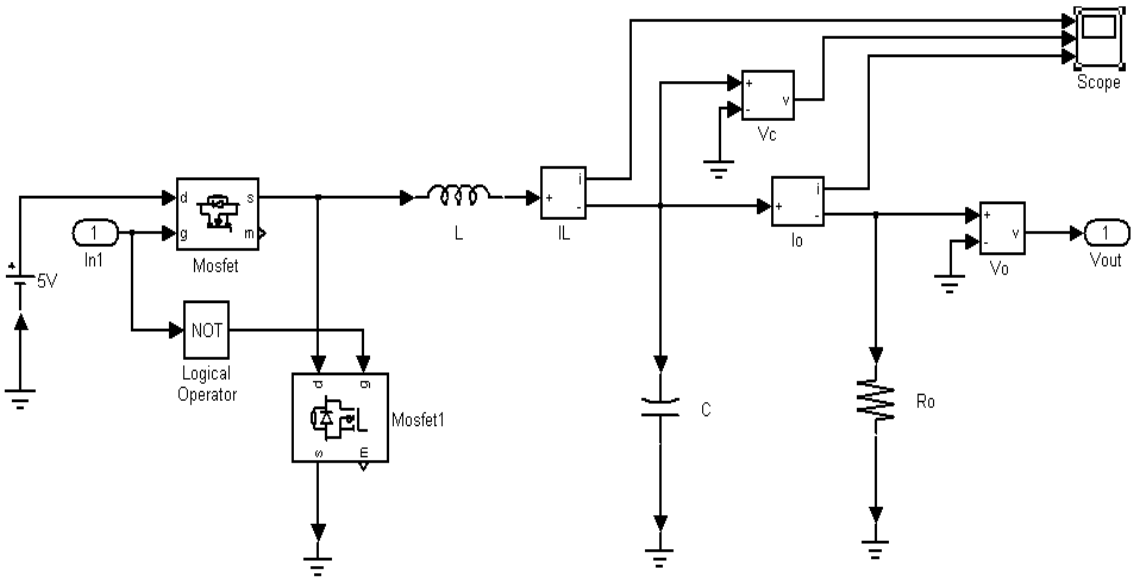


圖 3-9 Simulink之Buck converter之內部架構圖

圖3-10為ADC，輸出電壓會與參考電壓相比，產生的誤差值經過取樣並量化，即可獲得轉換後的數位誤差信號。

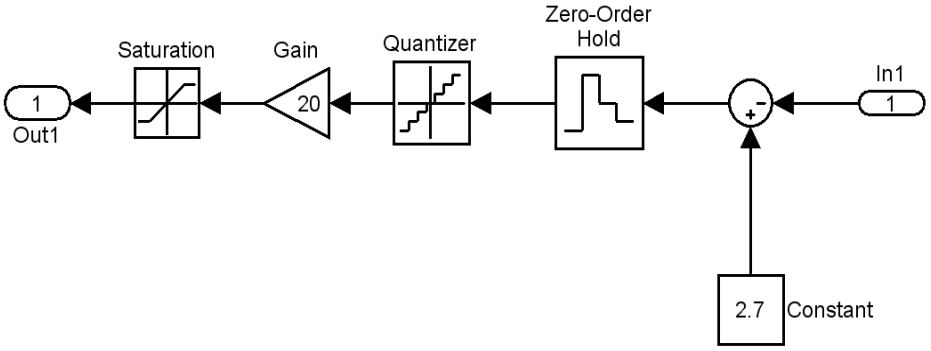


圖 3-10 Simulink之ADC

圖3-11為PID compensator區塊，其作用及執行(3-23)式，$d[n-1]$是前一周期之責任週期命令(duty cycle command)，$e[n]$為目前週期的數位誤

差信號，$e[n-1]$為前一週期的數位誤差信號，$e[n-2]$為前二週期的數位誤差信號，在此區塊中我們會代入之前利用 Ziegler-Nichols 調整法算出的 PID 控制器參數，最後這些資料經處理並相加後即可得到新的 $d[n]$責任週期命令值，並送至DPWM。

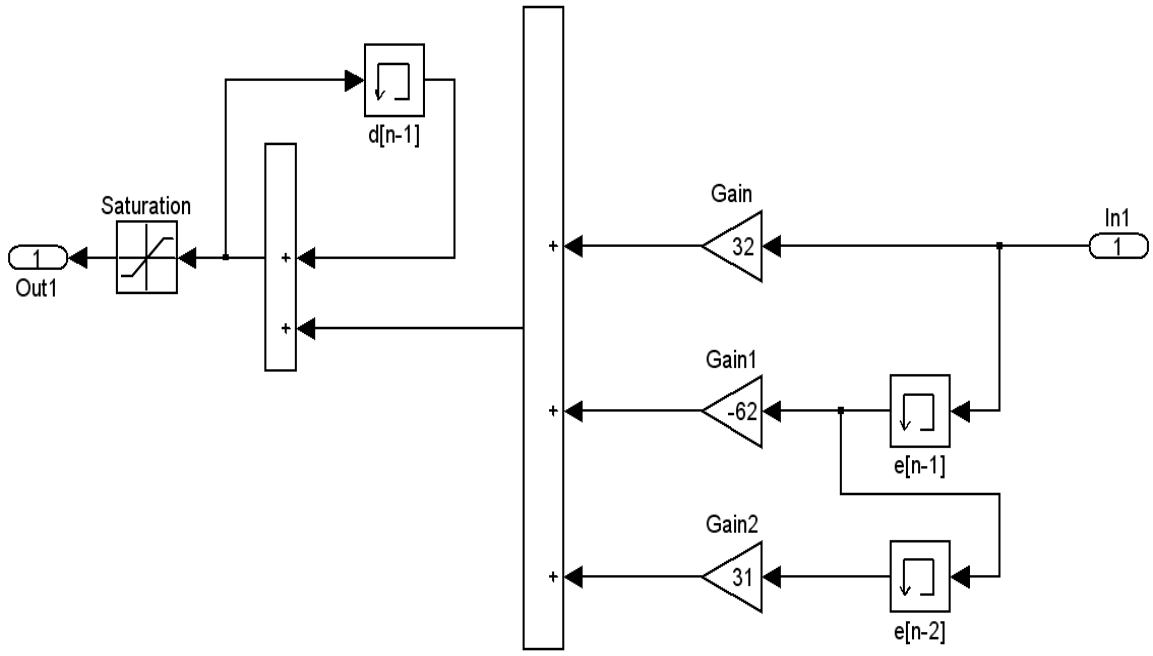


圖 3-11 Simulink之PID compensator區塊圖

圖3-12為DPWM區塊，此區塊是接收來自PID compensator的責任週期命令值 $d[n]$，並依DPWM解析位元數輸出一相對應之責任週期，進而控制開關切換。

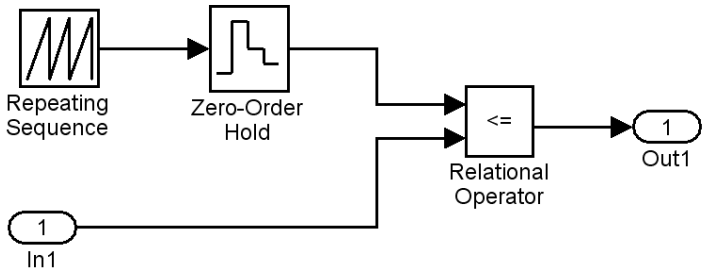


圖 3-12 Simulink之DPWM區塊圖

圖3-13為由啟動至穩定及改變負載時之模擬圖形，由上而下依序是電感電流、輸出電壓及輸出電流波形。

由圖 3-13中可以看出由啟動至穩定所需時間約0.002秒。

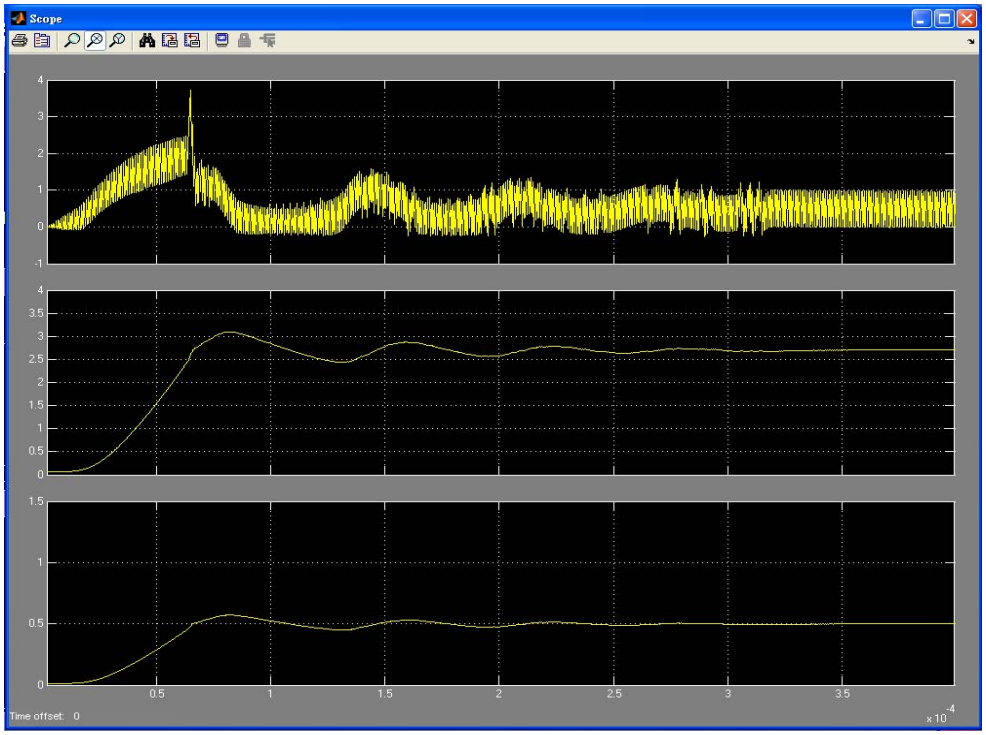


圖 3-13由啟動至穩定之模擬圖形(電感電流、輸出電壓及輸出電流)

第4章 數位脈波寬度調變控制電壓轉換電路硬體實現

本章主要在利用硬體去實現整個數位脈波寬度調變控制電路。使用的硬體描述語言(HDL)為Verilog 硬體描述語言。撰寫程式的工作環境為使用Xilinx ISE 發展軟體。這是一套由Xilinx 公司所推出用來設計FPGA/CPLD 或嵌入式系統的發展軟體，其操作畫面如圖4-1 所示。另外在硬體功能方面亦使用了為希科技公司所推出的系統發展實驗板（ULINX_MB_XC3S250E_PQ208_V20A），如圖4-2 所示。

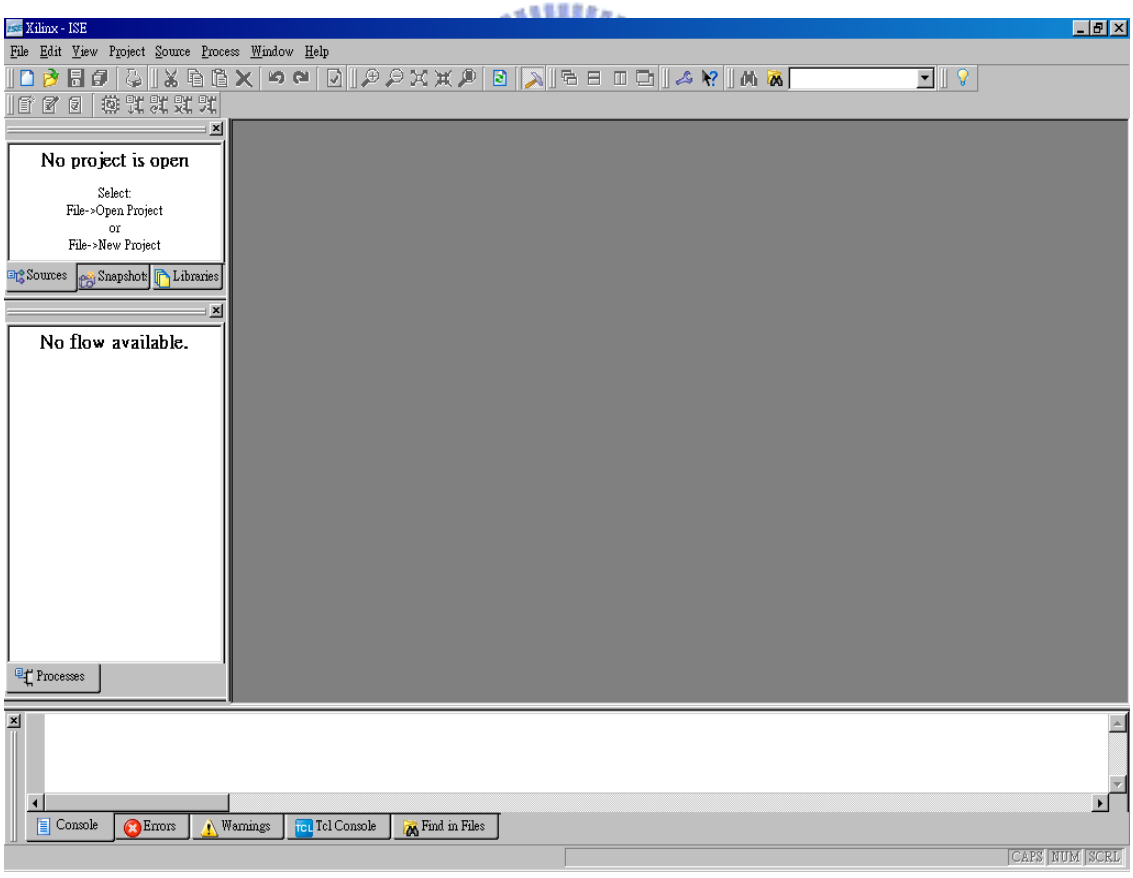


圖 4-1 Xilinx發展環境

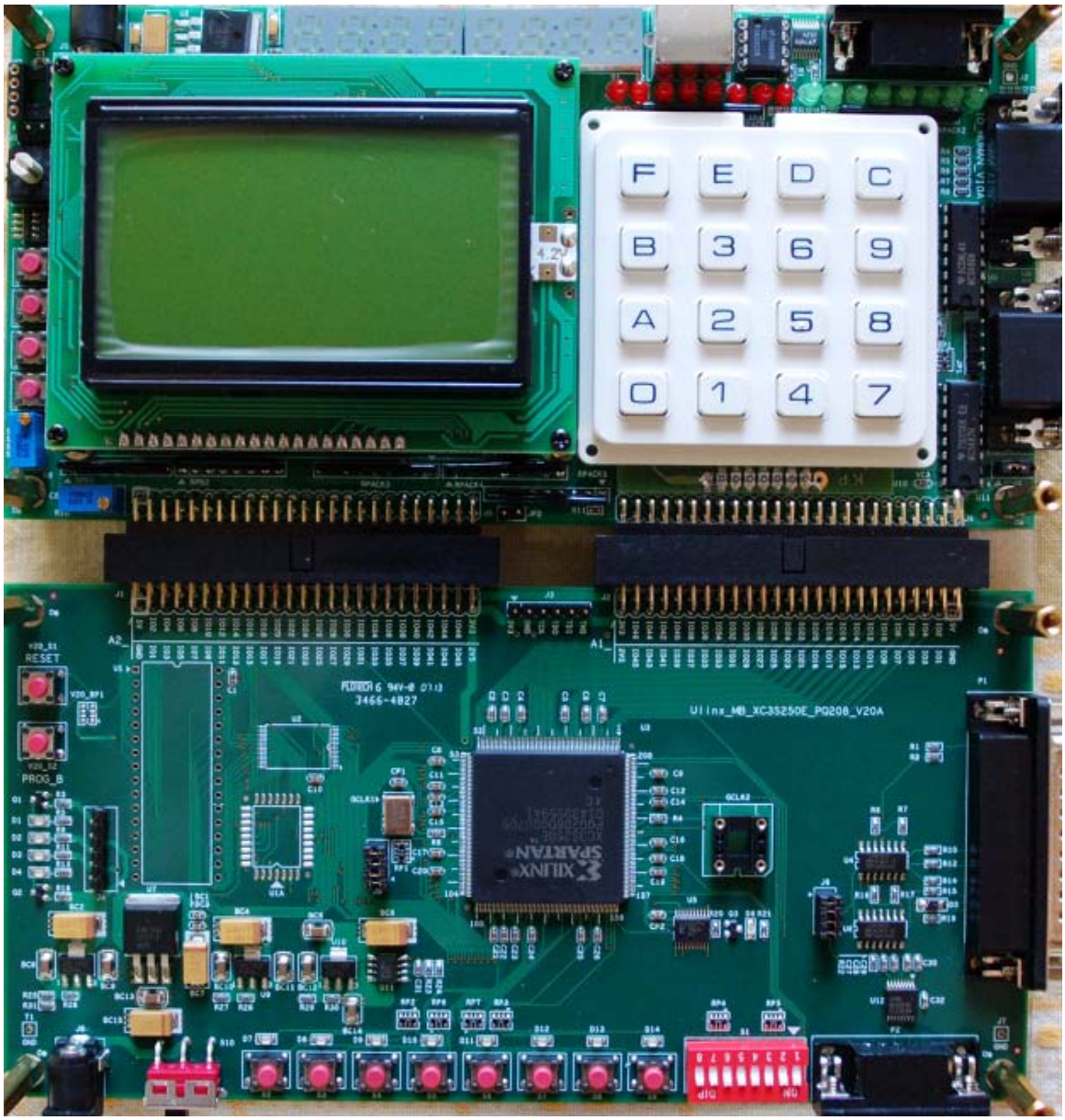


圖 4-2 Xilinx系統發展實驗板

4.1 Verilog 硬體描述語言簡介

Verilog 是一個標準化的硬體描述語言，它被廣泛地使用在積體電路設計與數位系統設計方面。Verilog 是第一個能支持混合各種設計層

次的專用語言，在同一個模組中可以有不同層次的表示法共同存在。所以設計者可以在同一模組中混合使用邏輯閘層次模型（Gate Level Model）、暫存器轉移層次模型（Register Transfer Level Model）以及行為模型（Behavioral Model）等三種不同的層次的表示法來描述所設計的電路。

Verilog 有下列有幾項特色：

- 提供多種不同的設計方法。
- 不同的製程特性也不用考量。
- 可以用來設計各式各樣的數位電路。
- 一般的邏輯合成工具普遍支援Verilog。
- 許多的製造廠商也都有提供Verilog 的函數庫。

4.2 A/D 轉換器

我們所使用的ADC型號為ADC0820，其參考電壓由第11腳（$V_{ref}(-)$）與第12腳（$V_{ref}(+)$）所決定。所以參考電壓如可以由下列式子表示之：

$V_{ref} = V_{ref}(+) - V_{ref}(-)$

因此，ADC0820 的每一位元的解析度計算公式如下列所表示之：

$$LSB = \frac{V_{ref}}{256}$$

由上面公式知道，當我們輸入$V_{ref}(+)$=5V、$V_{ref}(-)$=0V 時，這時候ADC0820 每位元（bit）之解析度即為:

$$LSB = \frac{5-0}{256} = 0.0195mV$$

當ADC0820 的第一腳（V_{in}）取樣到5V 電壓時，ADC0820的輸出資料（DB0~DB7）即為〝11111111〞。

由於我們要對直流至直流轉換器做電壓的取樣，而該轉換器的輸出電壓規格可為0V~5V，所以符合我們ADC0820 的規格。另外，根據我們目標所設計的控制積體電路規格，我們重新規劃了ADC0820電壓取樣的特性圖，如圖4-3 所示。

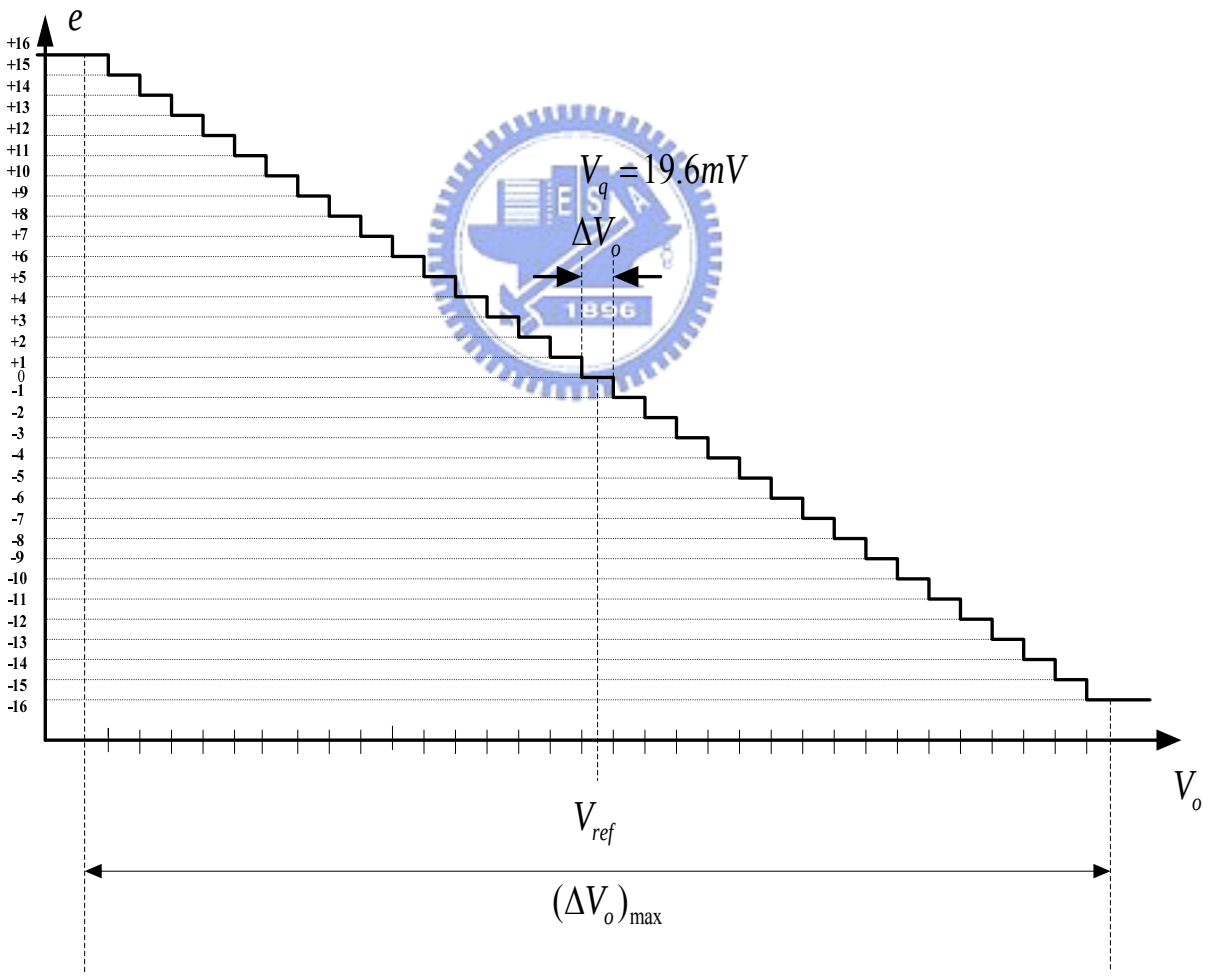


圖 4-3ADC0820電壓取樣的特性圖

表中的V_q即為ADC0820 的解析度，$V_q = \frac{5}{256} = 19.6mV$。

由上表可知，當切換式直流至直流的輸出電壓大於或小於V_{ref}的值時，ADC0820 會輸出誤差信號（$e[n]$）值（+16~-16）。

ADC0820我們操作在WR-RD模式下的stand-alone的工作模式，所以ADC0820的控制接腳如圖4-4所示。

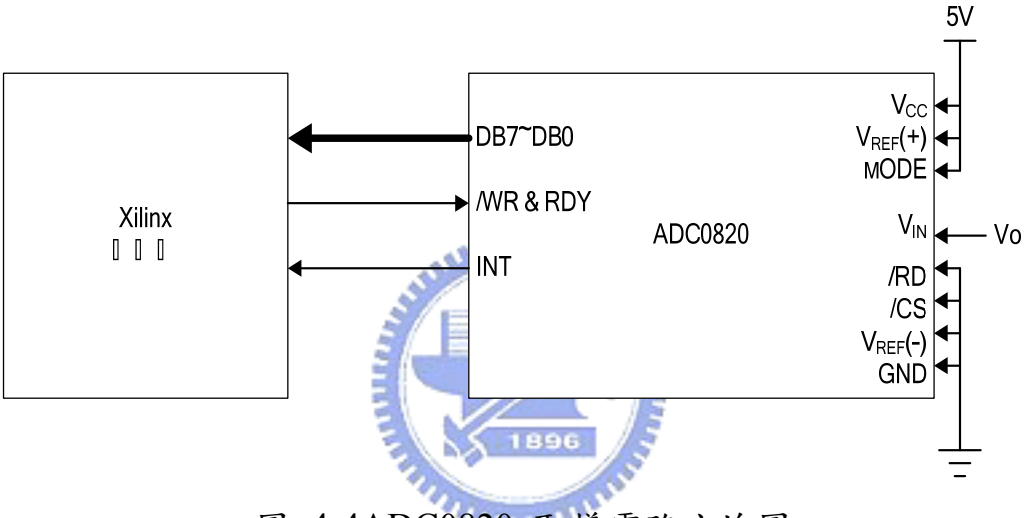


圖 4-4ADC0820 取樣電路方塊圖

4.2.1 Verilog 描述A/D 轉換器控制信號

圖4-5 即為控制ADC0820的Verilog 描述程式，以下為我們使用Verilog 去描述ADC0820 的控制信號，使ADC0820 能正常動作。

```
module adc0820(clk,reset,intin,datain,countin,e,wrout);
    input clk,reset;

    input [7:0]datain;
```

```
input [7:0]countin;
input intin;

output [8:0]e;
output wrout;

reg wrout;

reg [8:0]e;
reg [3:0]adccount;
reg [1:0]count_div4;

assign count_divact = (countin==0)? 1 : 0;

always@(negedge reset or posedge count_divact)
begin
	if(!reset)
	count_div4 <= 0;

	else
	begin
		if(count_div4 == 2'd3)
		count_div4 <= 0;
		else
		count_div4 <= count_div4 + 1;
	end
end

assign adc_act = (count_div4==0) & (~intin);

always@(negedge reset or posedge clk)
begin
	if(!reset)
	adccount <= 0;

	else
```

```
        begin
            if(adc_act == 1)
            adccount <= adccount + 1;
            else
            adccount <= 0;
        end
    end

    always@(negedge reset or posedge clk)
        begin
            if(!reset)
            wrout <= 0;

            else
            begin
                if((adccount>132) && (adccount<288))
                wrout <= 1;
                else
                wrout <= 0;
            end
         end

always@(negedge reset or posedge clk)
    begin
        if(!reset)
        e<= 0;

        else
         begin
            if(adccount==24)
            begin
             if(datain > 8'd154)
                e<=9'b111110000;
             else if(datain < 8'd122)
                e<=9'b000010000;
                else
```

```
                    e<=(9'd138) + {1'b1,(~datain)} + 1;
                end
            end
        end

endmodule
```

圖 4-5 ADC0820控制時序電路Verilog程式

4.3 數位補償器

我們利用查表法來實現數位脈波寬度調變器中的補償器。在3.2.2章節中，我們已經求得了a、b 及c 的值，現在我們就利用這些值來建立一個查表。我們所需求的誤差信號（$e[n]$）其值為+16~-16，其中代表負數的二進位值，我們以2補數法表示之。我們取a=32、b=-62、c=30建立此查表。其中負數的值皆使用2補數的做法。表4-1到表4-3即為所建立的查表。

e[n]	a=32	a*e[n]
+16	32	+512= 001000000000
+15	32	+480= 000111100000
+14	32	+448= 000111000000
+13	32	+416= 000110100000
+12	32	+384= 000110000000
+11	32	+352= 000101100000
+10	32	+320= 000101000000
+9	32	+288=000100100000
+8	32	+256=000100000000
+7	32	+224=000011100000
+6	32	+192=000011000000
+5	32	+160=000010100000
+4	32	+128=000010000000
+3	32	+96=000001100000
+2	32	+64=000001011110
+1	32	+32=000000100000
0	32	+00=000000000000
-1	32	-32=111111100000
-2	32	-64=111111000000
-3	32	-96=111110100000
-4	32	-128=111110000000
-5	32	-160=111101100000
-6	32	-192=111101000000
-7	32	-224=111100100000
-8	32	-256=111100000000
-9	32	-288=111011100000
-10	32	-320=111011000000
-11	32	-352=111010100000
-12	32	-384=111010000000
-13	32	-416=111001100000
-14	32	-448=111001000000
-15	32	-480=111000100000
-16	32	-512=111000000000

表 4-1 a=32之查表

e[n]	b=-62	b*e[n]
+16	-62	-992=110000100000
+15	-62	-930=110001011110
+14	-62	-868=110010011100
+13	-62	-806=110011011010
+12	-62	-744=110100011000
+11	-62	-682=110101010110
+10	-62	-620=110110010100
+9	-62	-558=110111010010
+8	-62	-496=111000010000
+7	-62	-434=111001001110
+6	-62	-372=111010001100
+5	-62	-310=111011001010
+4	-62	-248=111100001000
+3	-62	-186=111101000110
+2	-62	-124=111110000001
+1	-62	-62=111111000010
0	-62	+00=000000000000
-1	-62	+62=000000111110
-2	-62	+124=000001111111
-3	-62	+186=000010111010
-4	-62	+248=000011111000
-5	-62	+310=000100110110
-6	-62	+372=000101110100
-7	-62	+434=000110110010
-8	-62	+496=000111110000
-9	-62	+558=001000101110
-10	-62	+620=001001101100
-11	-62	+682=001010101010
-12	-62	+744=001011101000
-13	-62	+806=001100100110
-14	-62	+868=001101100100
-15	-62	+930=001110100010
-16	-62	+992=001111100000

表 4-2 b=-62之查表

e[n]	c=30	c*e[n]
+16	30	+480=000111100000
+15	30	+450=000111000010
+14	30	+420=000110100100
+13	30	+390=000110000110
+12	30	+360=000101101000
+11	30	+330=000101001010
+10	30	+300=000100101100
+9	30	+270=000100001110
+8	30	+240=000011110000
+7	30	+210=000011010010
+6	30	+180=000010110100
+5	30	+150=000010010110
+4	30	+120=000001111000
+3	30	+90=000001011010
+2	30	+60=000000111100
+1	30	+30=000000011110
0	30	+00=000000000000
-1	30	-30=111111100010
-2	30	-60=111111000100
-3	30	-90=111110100110
-4	30	-120=111110001000
-5	30	-150=111101101010
-6	30	-180=111101001100
-7	30	-210=111100101110
-8	30	-240=111100010000
-9	30	-270=111011110010
-10	30	-300=111011010100
-11	30	-330=111010110110
-12	30	-360=111010011000
-13	30	-390=111001111010
-14	30	-420=111001011100
-15	30	-450=111000111110
-16	30	-480=111000100000

表 4-3 c=30之查表

4.3.1 Verilog 描述數位補償器

以下為我們使用Verilog 去描述數位補償器的實現方式。由之前建立之查表，將這些數值帶入硬體描述語言中。圖4-6 即為數位補償器的

Verilog 描述程式。

```
module pid_lut(adc_clk,reset,e,e_n1,e_n2,dutycom);
input adc_clk,reset;
input [8:0]e;
output [8:0]e_n1,e_n2;
output [10:0]dutycom;

reg [8:0]e_n1,e_n2;        //e(n-1)
reg [11:0]d_n1;        //d(n-1)
reg [11:0]lut_mam1[0:32];    //TABLE_1
reg [11:0]lut_mam2[0:32];    //TABLE_2
reg [11:0]lut_mam3[0:32];    //TABLE_3
reg [11:0]ae,be,ce;
reg [10:0]dutycom;
wire[8:0]e;
wire [11:0]d;

assign d=d_n1+ae+be+ce;

always@(d)
begin
 if(d[11]==0)
    begin
     if(d[11:0]>11'b01110000100)
          dutycom<=11'b11111010110;
     else
        dutycom<=d[10:0]+11'b10001010001 ;
    end
 else
    begin
     if((~d[11:0]+1'b1)>11'b10000000000)
        dutycom<=11'b00001010001;
     else
        dutycom<=d[10:0]+11'b10001010001 ;
    end
end
always@(posedge adc_clk or negedge reset)
begin
if (~reset)
begin
e_n1<=0;
e_n2<=0;
d_n1<=0;
end
```

```
else
begin
e_n1<=e;
e_n2<=e_n1;
d_n1<=d;
end
end
always@(e)
begin
case(e[5:0])
6'b110000: ae=lut_mam1[0];
6'b110001: ae=lut_mam1[1];
6'b110010: ae=lut_mam1[2];
6'b110011: ae=lut_mam1[3];
6'b110100: ae=lut_mam1[4];
6'b110101: ae=lut_mam1[5];
6'b110110: ae=lut_mam1[6];
6'b110111: ae=lut_mam1[7];
6'b111000: ae=lut_mam1[8];
6'b111001: ae=lut_mam1[9];
6'b111010: ae=lut_mam1[10];
6'b111011: ae=lut_mam1[11];
6'b111100: ae=lut_mam1[12];
6'b111101: ae=lut_mam1[13];
6'b111110: ae=lut_mam1[14];
6'b111111: ae=lut_mam1[15];
6'b000000: ae=lut_mam1[16];
6'b000001: ae=lut_mam1[17];
6'b000010: ae=lut_mam1[18];
6'b000011: ae=lut_mam1[19];
6'b000100: ae=lut_mam1[20];
6'b000101: ae=lut_mam1[21];
6'b000110: ae=lut_mam1[22];
6'b000111: ae=lut_mam1[23];
6'b001000: ae=lut_mam1[24];
6'b001001: ae=lut_mam1[25];
6'b001010: ae=lut_mam1[26];
6'b001011: ae=lut_mam1[27];
6'b001100: ae=lut_mam1[28];
6'b001101: ae=lut_mam1[29];
6'b001110: ae=lut_mam1[30];
6'b001111: ae=lut_mam1[31];
6'b010000: ae=lut_mam1[32];
endcase
end
always@(e_n1)
```

```
begin
case(e_n1[5:0])
6'b110000: be=lut_mam2[0];
6'b110001: be=lut_mam2[1];
6'b110010: be=lut_mam2[2];
6'b110011: be=lut_mam2[3];
6'b110100: be=lut_mam2[4];
6'b110101: be=lut_mam2[5];
6'b110110: be=lut_mam2[6];
6'b110111: be=lut_mam2[7];
6'b111000: be=lut_mam2[8];
6'b111001: be=lut_mam2[9];
6'b111010: be=lut_mam2[10];
6'b111011: be=lut_mam2[11];
6'b111100: be=lut_mam2[12];
6'b111101: be=lut_mam2[13];
6'b111110: be=lut_mam2[14];
6'b111111: be=lut_mam2[15];
6'b000000: be=lut_mam2[16];
6'b000001: be=lut_mam2[17];
6'b000010: be=lut_mam2[18];
6'b000011: be=lut_mam2[19];
6'b000100: be=lut_mam2[20];
6'b000101: be=lut_mam2[21];
6'b000110: be=lut_mam2[22];
6'b000111: be=lut_mam2[23];
6'b001000: be=lut_mam2[24];
6'b001001: be=lut_mam2[25];
6'b001010: be=lut_mam2[26];
6'b001011: be=lut_mam2[27];
6'b001100: be=lut_mam2[28];
6'b001101: be=lut_mam2[29];
6'b001110: be=lut_mam2[30];
6'b001111: be=lut_mam2[31];
6'b010000: be=lut_mam2[32];
endcase
end
always@(e_n2)
begin
case(e_n2[5:0])
6'b110000: ce=lut_mam3[0];
6'b110001: ce=lut_mam3[1];
6'b110010: ce=lut_mam3[2];
6'b110011: ce=lut_mam3[3];
6'b110100: ce=lut_mam3[4];
6'b110101: ce=lut_mam3[5];
```

```
6'b110110: ce=lut_mam3[6];
6'b110111: ce=lut_mam3[7];
6'b111000: ce=lut_mam3[8];
6'b111001: ce=lut_mam3[9];
6'b111010: ce=lut_mam3[10];
6'b111011: ce=lut_mam3[11];
6'b111100: ce=lut_mam3[12];
6'b111101: ce=lut_mam3[13];
6'b111110: ce=lut_mam3[14];
6'b111111: ce=lut_mam3[15];
6'b000000: ce=lut_mam3[16];
6'b000001: ce=lut_mam3[17];
6'b000010: ce=lut_mam3[18];
6'b000011: ce=lut_mam3[19];
6'b000100: ce=lut_mam3[20];
6'b000101: ce=lut_mam3[21];
6'b000110: ce=lut_mam3[22];
6'b000111: ce=lut_mam3[23];
6'b001000: ce=lut_mam3[24];
6'b001001: ce=lut_mam3[25];
6'b001010: ce=lut_mam3[26];
6'b001011: ce=lut_mam3[27];
6'b001100: ce=lut_mam3[28];
6'b001101: ce=lut_mam3[29];
6'b001110: ce=lut_mam3[30];
6'b001111: ce=lut_mam3[31];
6'b010000: ce=lut_mam3[32];
endcase
end
endmodule
```

圖 4-6 補償器 Verilog 描述程式

圖4-7為PID之輸入與輸出的模擬結果。

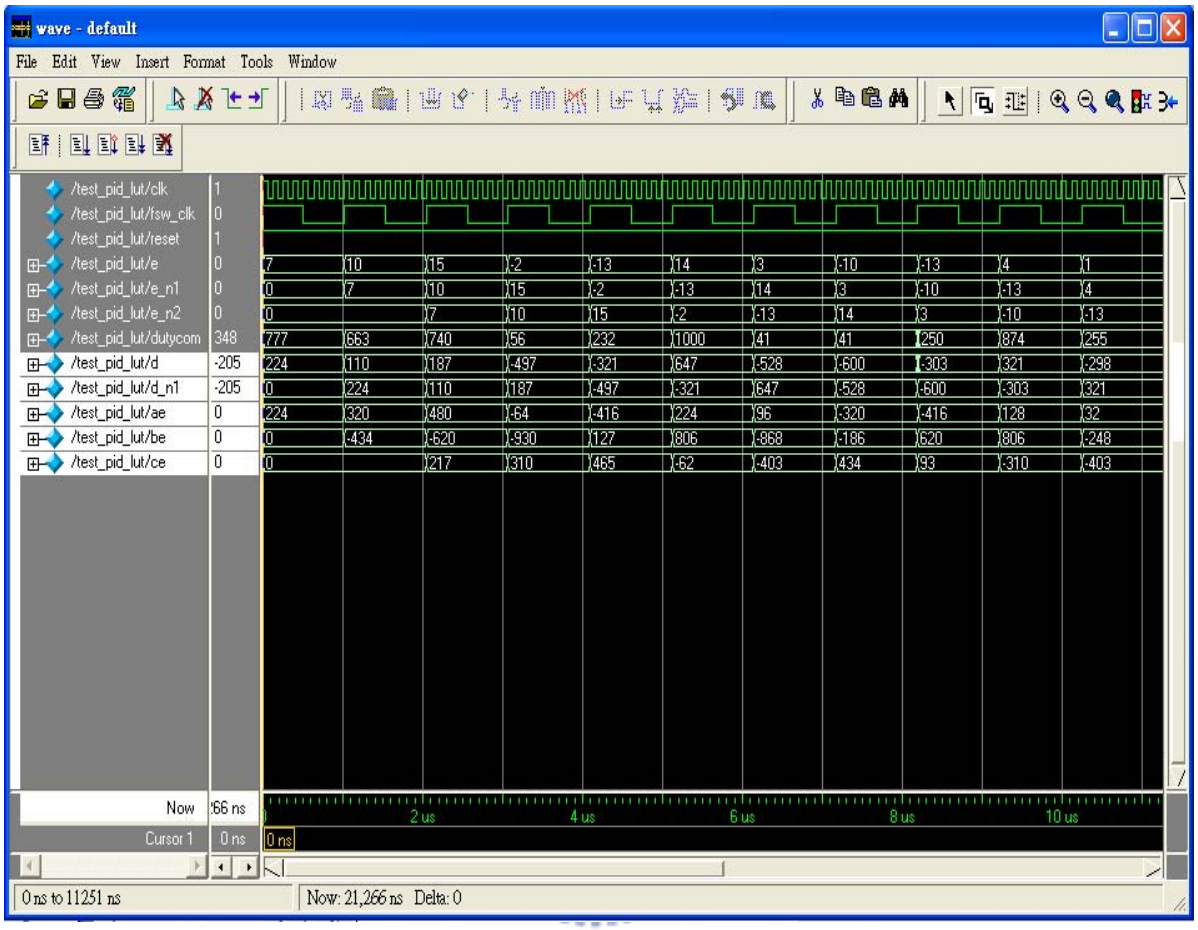


圖 4-7 PID輸入與輸出模擬圖

4.4 數位脈波寬度調變器

我們要設計一個數位脈波寬度調變器，其電路方塊如圖4-8所示，其中包含一個4-bit計數器、二個4-bit比較器、16個D型正反器、16×12的多工器、二個AND閘及一個RS正反器。

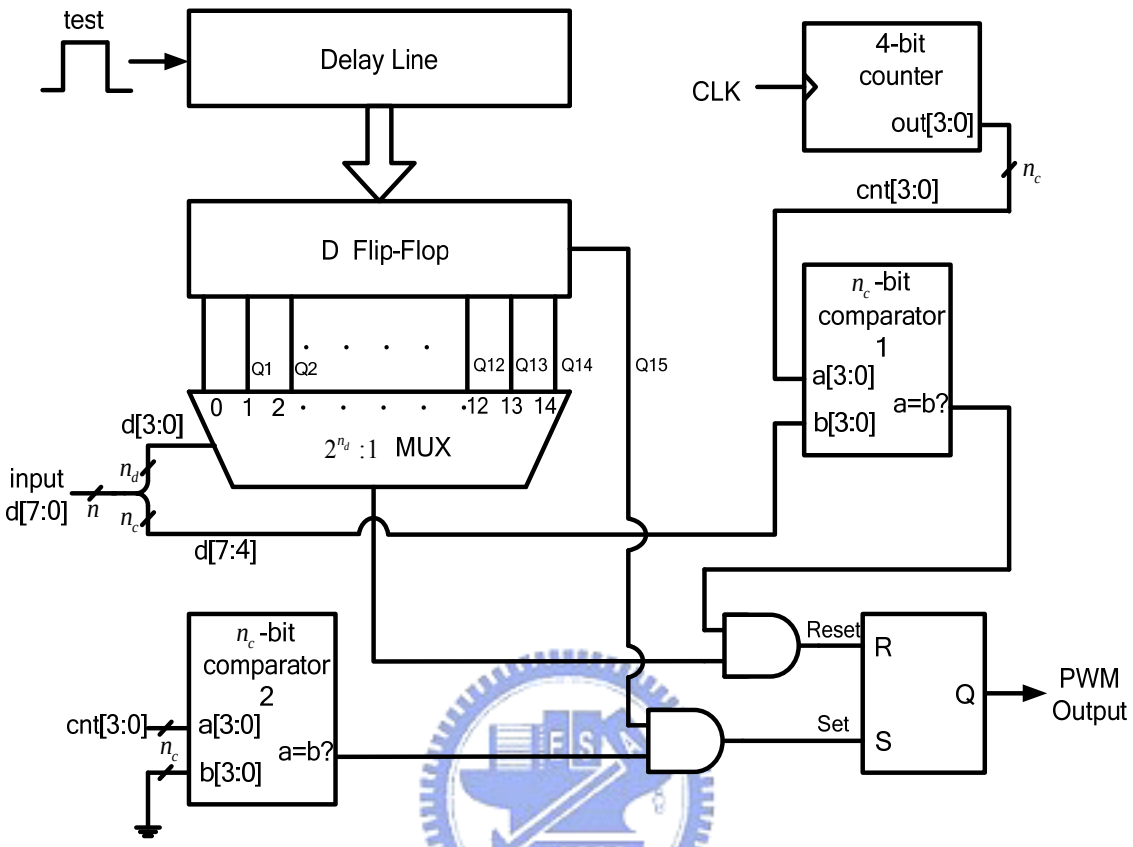


圖 4-8 混合式(Hybrid)DWPM架構圖

由圖可知，此混合式(Hybrid)DWPM中4 bits Delay-line型的DPWM與一個4-bit 計數型DPWM合成，讓這個數位脈波寬度調變器可以提供8-bit的解析度。

4.4.1 Verilog 描述數位數位脈波寬度調變器

圖4-9所示，為我們使用Verilog 去描述數位數位脈波寬度調變器的控制信號及功能。

```
`include "delayline.v"
`include "counter.v"
module hydpwm(clk, ,reset,duty_dith ,pwmout);

	input clk,,reset;
	input [7:0]duty_dith;
	output pwmout;

	reg compareout;
	reg pwmffsr_s;
	reg pwmffsr_r;
	reg cc;
	reg [7:0]duty_dith_reg;
	wire pwmqbar,pwm;
	wire [7:0]duty_dith;
	wire [3:0]countout;
	wire [16:0]delay;
	wire [14:0]q;
	wire q15;

	counter u1(clk,reset,countout);
	delayline delayline1(reset,delayin,delay,q,q15,delayreset);

	assign delayin = test;
	assign delayreset =0;

	//-- SET RS_Flip-Flop at counter=0 ----
	always@(negedge reset or countout or duty_dith[6:0] or q15)
	begin
		if(~reset | (countout[3:0]==3'b000)&(q15==1)&(cc==0))
		begin
			pwmffsr_s<=1;
			duty_dith_reg[7:0] <= duty_dith[7:0];
		end
		else
			pwmffsr_s<=0;
	end

	//-- Counter Compare ------------------
	always@(countout or duty_dith_reg[7:4])
	begin
		if(countout[3:0]>=duty_dith_reg[7:4])
			compareout <= 1;
		else
			compareout <= 0;
	end
```

```
    //----Counter Check (set=1 one times in switch cycle)
    always@(negedge reset or negedge fsw_clk or negedge pwmffsr_s)
        begin
          if(~reset | ~fsw_clk)
          cc<=0;
          else if(~pwmffsr_s)
          cc<=1;
        end

    //-- Delayline Multiplexer ------------
    always@(duty_dith_reg[3:0] or q or delay or compareout)
    begin
    case(duty_dith_reg[3:0])
          4'b0000 : pwmffsr_r <= delay[0] & compareout;
          4'b0001 : pwmffsr_r <= q[1] & compareout;
          4'b0010 : pwmffsr_r <= q[2] & compareout;
          4'b0011 : pwmffsr_r <= q[3] & compareout;
          4'b0100 : pwmffsr_r <= q[4] & compareout;
          4'b0101 : pwmffsr_r <= q[5] & compareout;
          4'b0110 : pwmffsr_r <= q[6] & compareout;
          4'b0111 : pwmffsr_r <= q[7] & compareout;
          4'b1000 : pwmffsr_r <= q[8] & compareout;
          4'b1001 : pwmffsr_r <= q[9] & compareout;
          4'b1010 : pwmffsr_r <= q[10] & compareout;
          4'b1011 : pwmffsr_r <= q[11] & compareout;
          4'b1100 : pwmffsr_r <= q[12] & compareout;
          4'b1101 : pwmffsr_r <= q[13] & compareout;
          4'b1110 : pwmffsr_r <= q[14] & compareout;
          4'b1111 : pwmffsr_r <= q15 & compareout;

    endcase
    end

    //-- SR Flip-Flop with non-clock ------
    nor norqbar(pwmqbar,pwmffsr_s,pwm);
    nor norq(pwm,pwmffsr_r,pwmqbar);

Endmodule


module counter(clk,reset,countout);
    input clk,reset;
    output [3:0]countout;
    reg [3:0]countout;
```

```
	always@(posedge clk or negedge reset)
	begin
		if(!reset)
		begin
			countout <= 0;
		end
		else
		begin
			countout <= countout + 1;
		end
	end
endmodule


module delayelement(delayelementout,delayelementin);
	input delayelementin;
	output delayelementout;

	nor #4 nor1(delayelementout,delayelementin);
endmodule
`include "delayelement.v"
module delaycell(delayin,delayout,delayreset);
	input delayin,delayreset;
	output delayout;

	delayelement delayelement0(delayelementout,delayin);
	nor nor1(delayout,delayelementout,delayreset);

endmodule

`include "delaycell.v"
`include "d_ff.v"
module delayline(reset,delayin,delay,q,q15,delayreset);
	input reset,delayin,delayreset;
	output [14:0]q;
	output q15;
	output [16:0]delay;
	reg q15;
	wire [14:0]q;
	wire [16:0]delay;

	assign delay[0]=delayin;
	delaycell delaycell01(delay[0],delay[1],delayreset);
	delaycell delaycell02(delay[1],delay[2],delayreset);
	delaycell delaycell03(delay[2],delay[3],delayreset);
	delaycell delaycell04(delay[3],delay[4],delayreset);
```

```
    delaycell delaycell05(delay[4],delay[5],delayreset);
    delaycell delaycell06(delay[5],delay[6],delayreset);
    delaycell delaycell07(delay[6],delay[7],delayreset);
    delaycell delaycell08(delay[7],delay[8],delayreset);
    delaycell delaycell9(delay[8],delay[9],delayreset);
    delaycell delaycell10(delay[9],delay[10],delayreset);
    delaycell delaycell11(delay[10],delay[11],delayreset);
    delaycell delaycell12(delay[11],delay[12],delayreset);
    delaycell delaycell13(delay[12],delay[13],delayreset);
    delaycell delaycell14(delay[13],delay[14],delayreset);
    delaycell delaycell15(delay[14],delay[15],delayreset);
    delaycell delaycell16(delay[15],delay[16],delayreset);


    d_ff d1(delay[1],reset,q[1],q[0]);
    d_ff d2(delay[2],reset,q[2],q[1]);
    d_ff d3(delay[3],reset,q[3],q[2]);
    d_ff d4(delay[4],reset,q[4],q[3]);
    d_ff d5(delay[5],reset,q[5],q[4]);
    d_ff d6(delay[6],reset,q[6],q[5]);
    d_ff d7(delay[7],reset,q[7],q[6]);
    d_ff d8(delay[8],reset,q[8],q[7]);
    d_ff d9(delay[9],reset,q[9],q[8]);
    d_ff d10(delay[10],reset,q[10],q[9]);
    d_ff d11(delay[11],reset,q[11],q[10]);
    d_ff d12(delay[12],reset,q[12],q[11]);
    d_ff d13(delay[13],reset,q[13],q[12]);
    d_ff d14(delay[14],reset,q[14],q[13]);
    d_ff d15(delay[15],reset,q15,q[14]);

    always@(posedge delay[16] or negedge reset or posedge q[0])
    begin
       if (q[0])
         q15<=0;
       else if((delay[16])|(~reset))
         q15<=1;
    end

endmodule
```

圖 4-9 數位脈波寬度調變器Verilog 描述程式

圖4-10為DPWM之輸入與輸出的模擬結果，我們將duty_dith分別輸入00111010、00001110、11111100、01000000、00000100時所輸出的脈波寬度輸出。

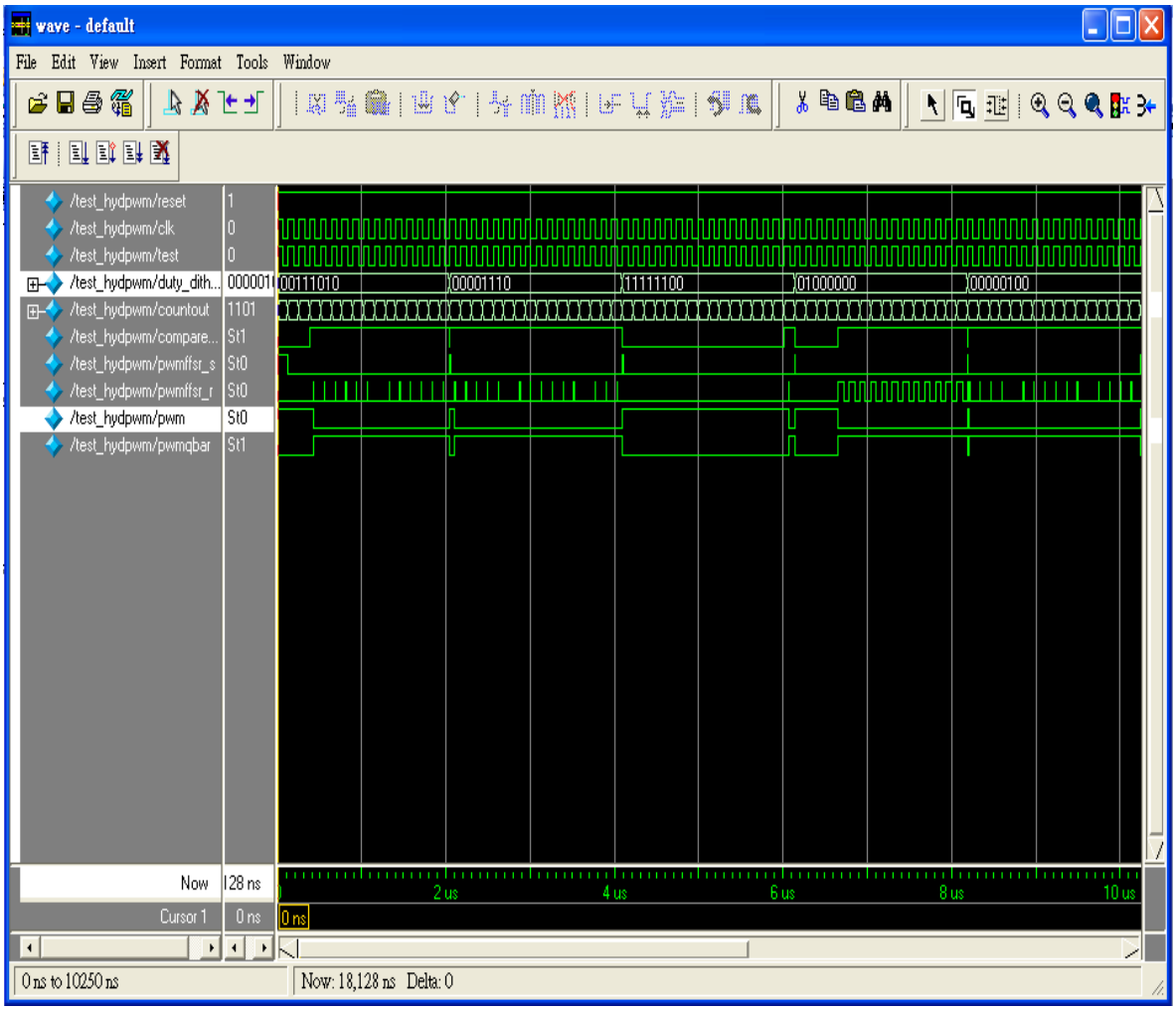


圖 4-10 DPWM 輸入與輸出模擬圖

4.5 數位顫抖控制電路

在式子(2-2)中我們知道 resolution（DPWM）> resolution（A/D）才可以避免Limit-Cycling 現象產生，因此DPWM的解析度必須大於A/D轉換器的解析度，所以我們必須增加DPWM的解析度，我們可以利用數位顫抖控制設計(Digital Dither)增加DPWM的有效解析度，由原始的DPWM硬體8-bits多增加3-bits，等效成為11-bits的DPWM。

4.5.1 Verilog 描述數位數位顫抖控制電路

圖4-11為我們使用Verilog 去描述數位顫抖控制電路。

```
module dither(clk,fsw_clk,reset,dutycom,duty_dith);
      input clk,fsw_clk,reset;
      input [10:0]dutycom;
      output [7:0]duty_dith;
      reg [10:0]dutycom1;
      reg dith_bit;
      reg [2:0]countout;
      reg[7:0]mam[0:7];
      wire[7:0]dith_word,duty_dith;
      wire[10:0]dutycom;

      initial
      begin
            mam[3'b000]=8'b00000000;
            mam[3'b001]=8'b00000001;
            mam[3'b010]=8'b00010001;
            mam[3'b011]=8'b00100101;
            mam[3'b100]=8'b01010101;
            mam[3'b101]=8'b01011011;
            mam[3'b110]=8'b01110111;
            mam[3'b111]=8'b01111111;
      end
always@(posedge fsw_clk or negedge reset)
begin
      if (~reset)
         begin
```

```
        dutycom1<=11'b00000000000;
        countout<=0;
           end
        else if(dutycom!=dutycom1)
           begin
              countout<=0;
              dutycom1<=dutycom;
           end
        else if(fsw_clk)
        countout<=countout+1;
end

assign dith_word=mam[dutycom[2:0]];
assign duty_dith=dutycom[10:3]+dith_bit;

always@(countout)
case(countout)
        3'b000: dith_bit=dith_word[7];
        3'b001: dith_bit=dith_word[6];
        3'b010: dith_bit=dith_word[5];
        3'b011: dith_bit=dith_word[4];
        3'b100: dith_bit=dith_word[3];
        3'b101: dith_bit=dith_word[2];
        3'b110: dith_bit=dith_word[1];
        3'b111: dith_bit=dith_word[0];
endcase
endmodule
```

圖 4-11 數位數位顫抖控制電路 Verilog 描述程式

圖4-12為Digital Dither之輸入與輸出的模擬結果。

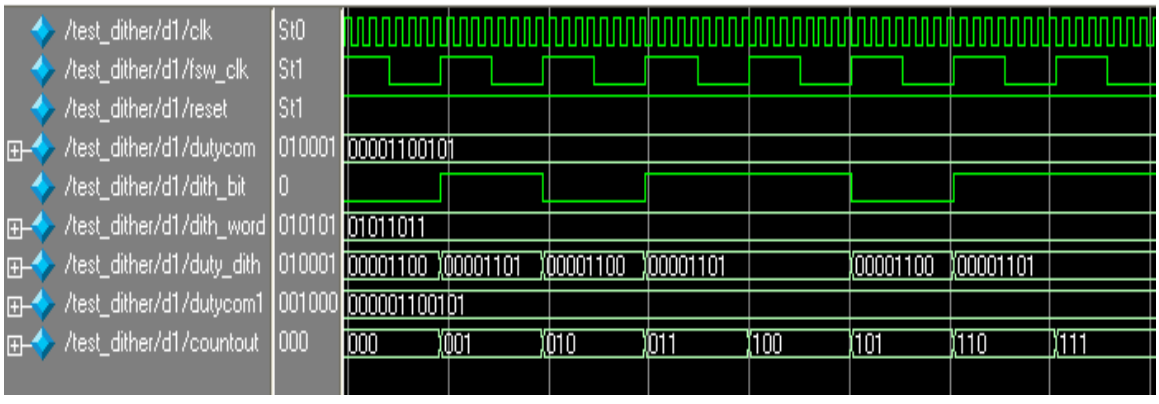


圖 4-12 Digital Dither 之輸入與輸出的模擬波形

4.6 低通濾波器 LP(z)之實現

我們可由圖 2-18低通濾波器實現方塊圖為依據，完成Verilog程式。

4.6.1 Verilog描述低通濾波器

圖4-13為我們使用Verilog 去描述低通濾波器功能。

```
module LP(adc_clk,reset,dutycom,D);
      input adc_clk,reset;
      input[10:0]dutycom;

      output[10:0]D;

      reg[10:0]D1,b;
      wire[10:0]dutycom,D,a;

      always@(negedge reset)
      begin
            if(!reset)
       D1=11'b00000000000;
      End

      assign a=dutycom-D1;

    always@(a)
          begin
            b=a>>7;
          end
        assign D=D1+b;
      always@(posedge adc_clk)
      begin
          D1=D;
      end

endmodule
```

圖 4-13 低通濾波器Verilog 描述程式

圖4-14為低通濾波器LP(z)之輸入與輸出的模擬結果。

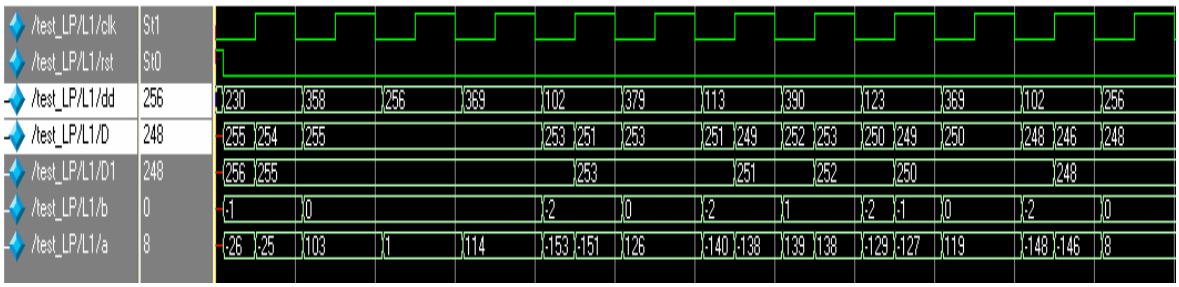


圖 4-14 低通濾波器 LP(z)之輸入與輸出的模擬

4.7 盲時(Dead-time)最佳化實現

藉由3.6.6章節介紹的Dead-time最佳化演算法，我們可以從最小的穩態責任週期命令D中，尋找最佳的dead-time時間。在程式撰寫方面，我們可以利用Finite State Machine的概念將演算法加以實現。

4.7.1 Verilog 描述盲時(Dead-time)最佳化電路

圖4-13為我們使用Verilog 去描述盲時(Dead-time)最佳化電路。

```
module dead_time_opt(adc_clk,reset,D,f,k,td1,td2,CS);
input adc_clk,reset;
input[10:0]D;
input [8:0]f;
output[7:0]td1,td2;
output[5:0]k;
output[2:0]CS;

wire [8:0]f;
wire[10:0]D;

reg[5:0]k;
reg[2:0]CS,NS;
reg[7:0]td1,td2,tdopt;
reg[10:0]Dold;
reg[7:0]c;
```

```
parameter
T0=3'b000,T1=3'b001,T2=3'b010,T3=3'b011,T4=3'b100,T5=3'b101,T6=3'b110;

always@(posedge adc_clk or negedge reset)    //Current State Register
begin
  if (~reset)
     begin
     CS<=T0;
     Dold<=11'b11111111111;
     end
  else                  CS<=NS;
end

always@(D,CS)    //Next State Logic
case(CS)
T0:NS=T1;
T1:NS=T2;
T2:NS=T3;
T3:NS=T4;
T4:
     begin
     if (c==8'd200) NS=T5;
     else NS=T4;
     end
T5:
     begin
     if (D>Dold) NS=T6;
     else if (D<Dold) NS=T1;
     else if (D==Dold) NS=T2;
     end
T6:NS=T0;
endcase

always@(posedge adc_clk or negedge reset)    //Output Logic
begin
       if (~reset)
         begin
           k<=6'd2;
           c<=200;
           td1<=8'b00110010;
           td2<=8'b00110010;
         end

       else if (f==0)
       case(CS)
```

```
        T0:
              begin
              td1<=8'b00110010;
              c<=0;
              end
        T1:tdopt<=td1;
        T2:Dold<=D;
        T3:td1<=td1-8'b00000100;
        T4:c<=c+1;
        T5:c<=0;
        T6:
            begin
              td1<=tdopt;
              k<=6'd1;
            end
      endcase

      else if (k==1)
      case(CS)
        T0:td2<=8'b00110010;
        T1:tdopt<=td2;
        T2:Dold<=D;
        T3:td2<=td2-8'b00000100;
        T4:c<=c+1;
        T5:c<=0;
        T6:
            begin
              td2<=tdopt;
              k<=6'd2;
            end
      endcase
  end
endmodule
```

圖 4-15 盲時(Dead-time)最佳化電路 Verilog 描述程式

圖4-16為盲時(Dead-time)最佳化電路之輸入與輸出的模擬結果。

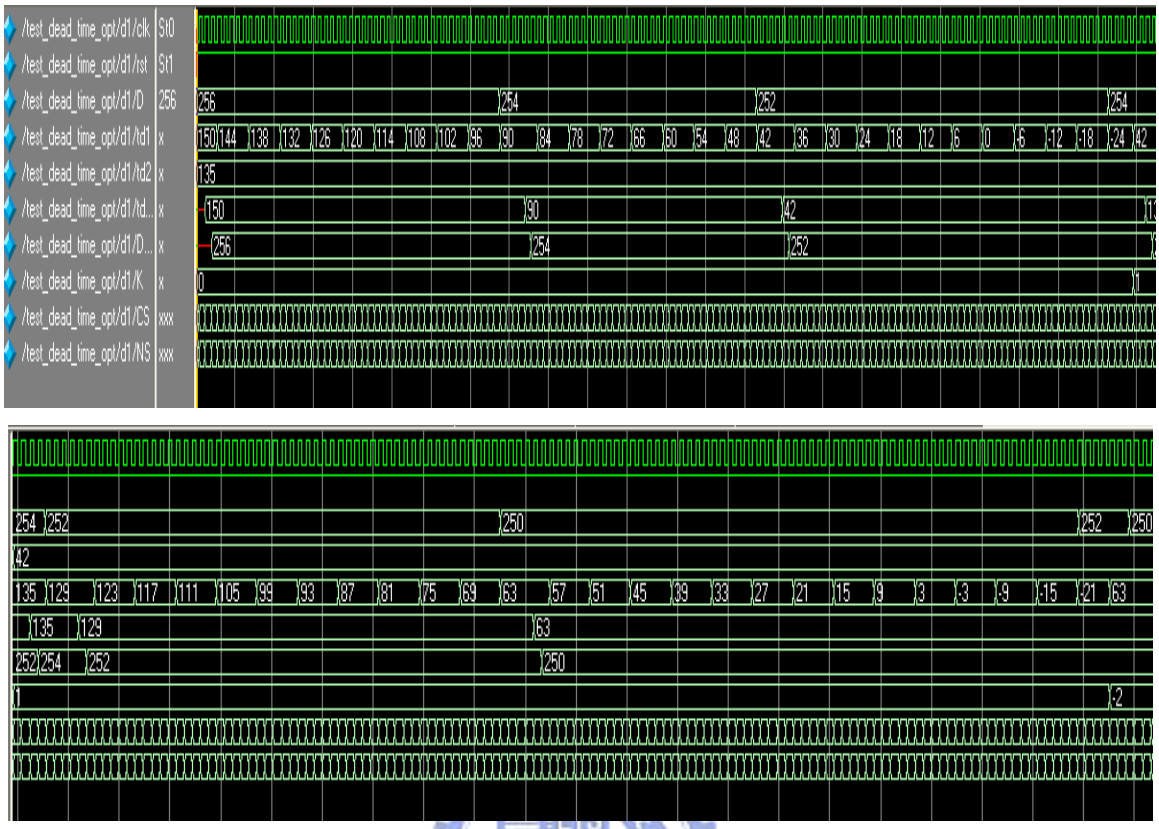


圖 4-16 盲時(Dead-time)最佳化電路之輸入與輸出的模擬波形

4.8 實驗結果

圖4-17為實驗系統示意圖，我們使用FPGA實驗版及設計之外部電路將ADC、數位PID、DPWM、Digital Dither、Dead-time最佳化電路及Power Stage做系統整合驗證，使用規格如下:

- 系統操作頻率（取樣頻率）：300KHz。
- 輸入電壓（V_i）：5V~10V
- 輸出電壓（V_o）： 2.7 V。

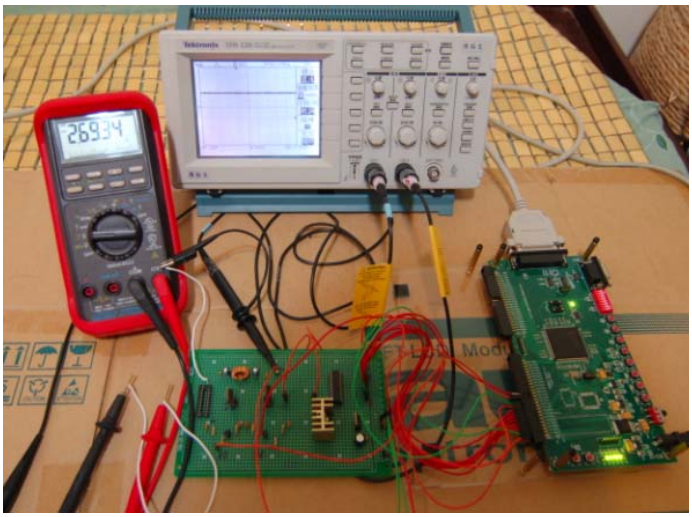


圖 4-17 實驗系統接線示意圖

圖4.18為輸出電壓及PWM輸出波形，由示波器可以看出輸出電壓可以穩壓在2.7V。

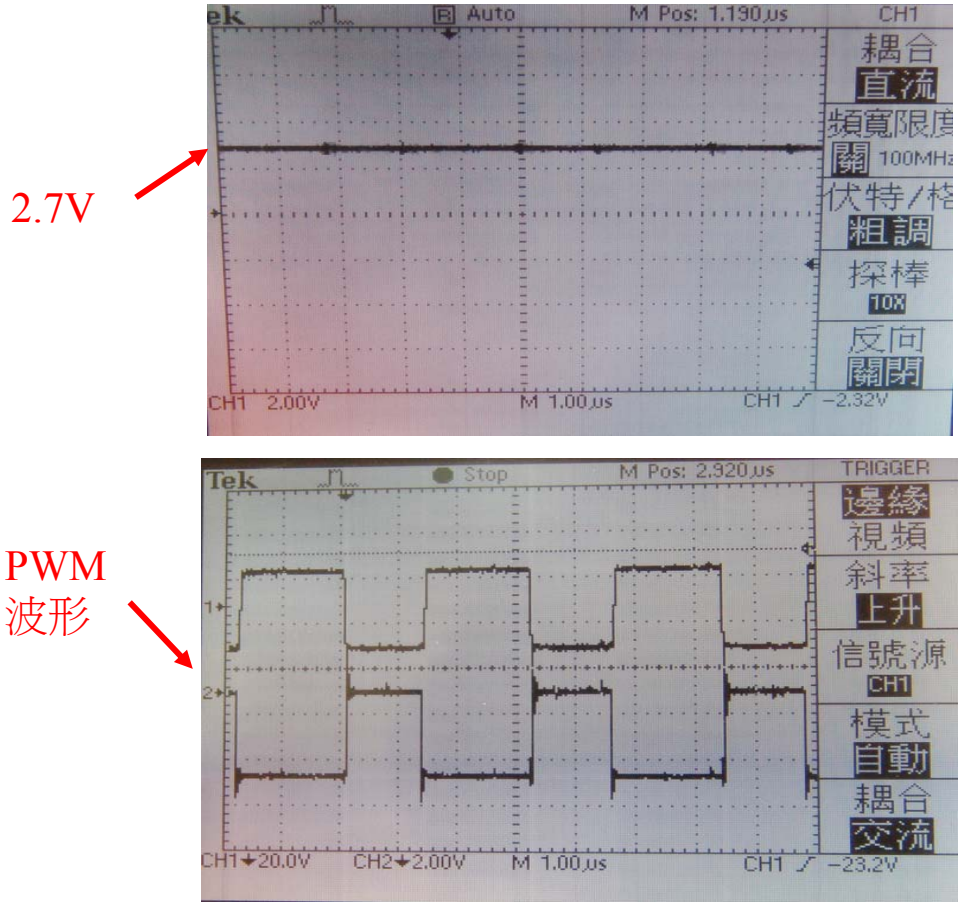


圖 4-18 輸出電壓及PWM波形

圖4.19為採用一般電路實現時的dead-time波形，而圖4.20為加入盲時(Dead-time)最佳化電路所實現之修正dead-time效果，由二圖可以明顯發現，加入盲時(Dead-time)最佳化電路後可以得到最佳的dead-time，有效提升系統效率。

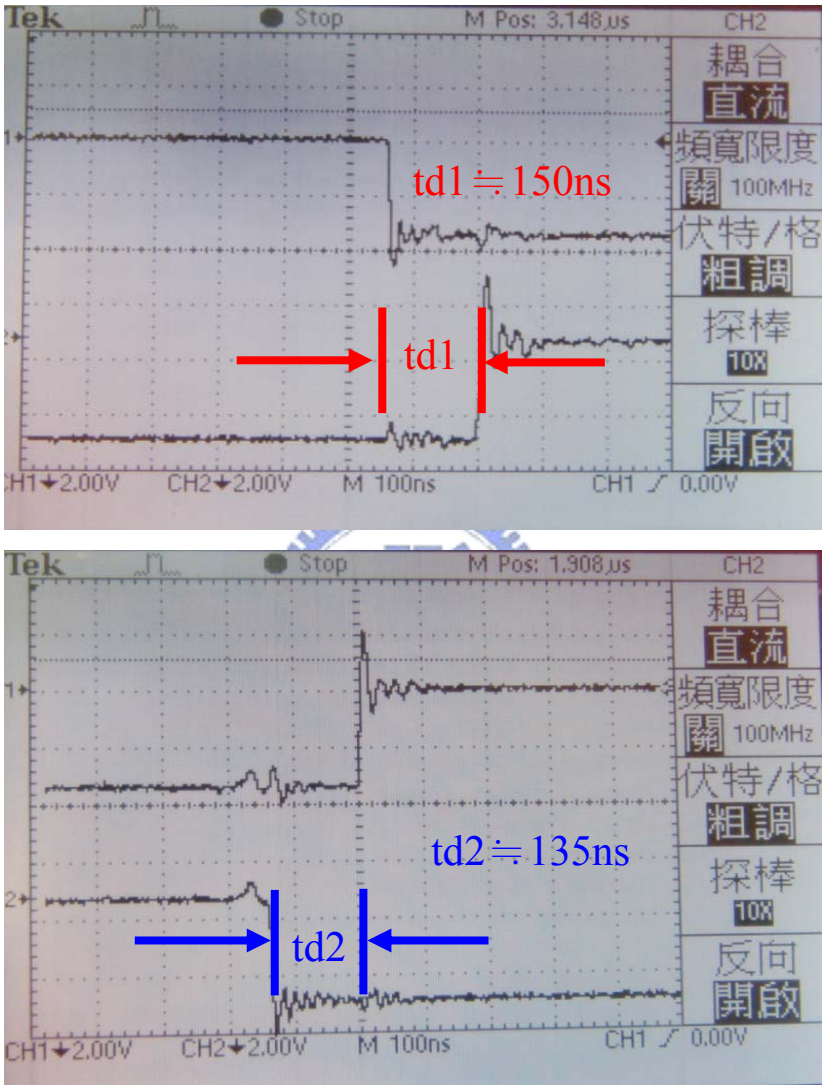


圖 4-19一般無Dead-time最佳化之電路

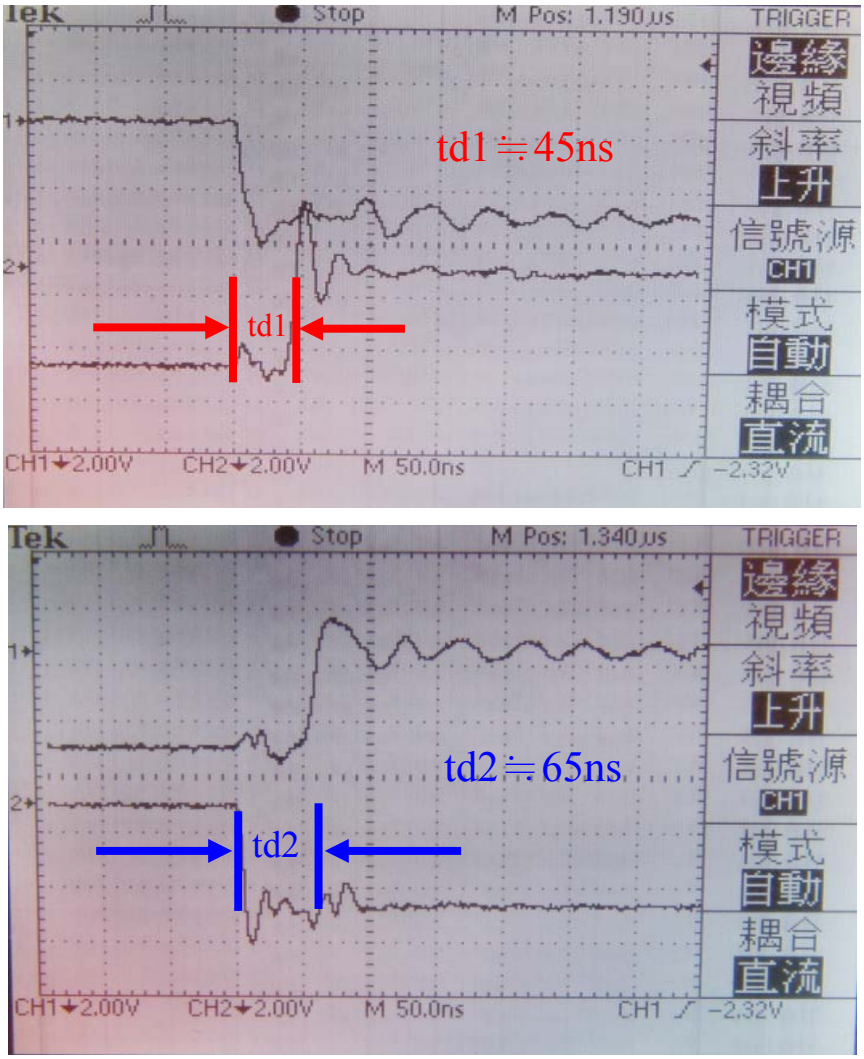


圖 4-20 使用 dead-time 最佳化之 dead-time 輸出波形

第5章結論與未來研究方向

5.1 結論

在本論文中，我們以數位設計方式為基礎，分析並實現一切換式降壓型直流至直流轉換器。該電路主要的功能為取樣直流至直流轉換器的輸出電壓（V_o），並與我們所設定的參考電壓（V_{ref}）作比較，如果兩者電壓有誤差的話，則取樣電路部份會輸出一個誤差信號（$e[n]$），而數位補償器此時就會根據此$e[n]$值去對應到相對的三個表位址而查到所需要補償的數值且輸出$d[n]$值，再經過數位顫抖電路及數位脈波寬度調變器，送出一個頻率固定的方波控制切換式降壓型直流至直流轉換器之開關元件的導通與不導通之間的比例，藉此方式來達到使輸出電壓穩壓的目標。最後我們整理出本論文實現之數位電源轉換器其特色及優點:

1. 使用 Hybrid(混合式)DPWM，使得在面積與功耗中取得平衡。
2. 以查表法(Look-up Table)實現數位PID補償器，保留了可調整的彈性，方便之後設計新的電路架構。
3. 使用數位顫抖控制電路(Digital Dither)，有效提升 DPWM 解析度。
4. 利用盲時最佳化電路(Dead-Time Optimizer)尋找最佳 dead-time，使系統整體效能提升。

整個電路由 MATLAB 及 Verilog 模擬並用 FPGA 進行實驗驗證，實驗結果證實我們所設計的數位脈波寬度調變控制電路，其功能皆能符合我們的要求。

5.2 未來方向

本論文中所設計的電路架構為單相位的切換式降壓型直流至直流轉換器，未來可進一步以多相位並聯之架構來實現，不但能增進系統的穩定性、降低所需系統頻率且對於負載較大的系統更能發揮其作用。

在設計PID控制器方面，最重要的是相關參數的計算，當採用Ziegler-Nichols 調整法來計算時，因為其為經驗公式，所得的解並非最佳解，所以未來在設計PID參數時，可以利用如NCD、類神經網路、基因演算法、Fuzzy理論..等等方法來搜尋PID參數的最佳解，使系統能夠有更佳的暫態響應。

數位式電源轉換器主要的優點是容易重新設計，我們可以根據不同的電源電路規格做調整與轉換，容易的設計出其他不同類型的電壓轉換器，或更進一步將不同的系統加以整合，所以未來的目標將是以數位控制技術的優勢來達到更大整合度、更小尺寸、更高效率、更高安全可靠性、更快瞬時響應、更高靈活性，以及更低的成本。

參考文獻

[1] A. Prodic. D. Maksimovic, and R. W. Erickson "Design and Implementation of a Digital PWM Controller for a High-Frequency Switch DC-DC Power Converters," in Proc. *IEEE Conference* 2001.

[2] A.Prodic, D.Maksimovic, “Design of a digital PID regulator based on look-up tables for control of high-frequency DC-DC converters,” *IEEE COMPEL*, 2002, pp.18-22.

[3] B. J. Patella, A. Prodic, A. Zirger, and D. Maksimovic, "High-frequency digital PWM controller IC for DC–DC converters, " *IEEE Trans. Power Electron.*, vol. 18, no. 1, pp. 438–446, Jan. 2003.

[4] A. V. Peterchev, S. R. Sanders, "Quantization resolution and limit cycling in digitally controlled PWM converters, " *IEEE Trans. on Power Electronics*, Vol.18, No.1, January 2003, pp.301-308.

[5] L. Peng, X. Kong, Y. Kang, and J. Chen, “A novel PWM technique in digital control and its application to an improved DC/DC converter,” in *Proc. IEEE Power Electron. Spec. Conf.,* 2001.

[6] V. Yousefzadeh, S. Member, D. Maksimovic "Sensorless Optimization of Dead Times in DC–DC Converters With Synchronous Rectifiers" *IEEE Trans. Power Electron.*, vol. 32, no. 5, pp. 134-139, Apr. 2003.

[7] B. C. Kuo "Automatic Control System" seventh edition,1997.

[8] N. Minorsky, "Directional Stability of Automatically Steered Bodies," *J. Amer. Soc. Naval Engineers*, 42(2),pp.280-309, 1922

[9] A. Callender,D.R. Hartree and A.Porter, "Time-Lag in a Control System," *Phil.Trans. Roy. Soc. London*,235(756),PP.415-444, 1936

[10] J.G. Ziegler and N.B. Nichols, "Optimum Settings for Automatic Controllers, " *Trans. ASMS*, 64, PP.759-768,1942

[11] Angel V. Peterchev, "Digital Pulse-Width Modulation Control in Power Electronic Circuits: Theory and Applications", *GRADUATE DIVISION of the UNIVERSITY OF CALIFORNIA, BERKELEY*

[12] R. F. Foley, R. C. Kavanagh, " An Area-Efficient Digital Pulse Width Modulation Architecture Suitable for FPGA Implementation" *IEEE Applied Power Electronic Conference*,2005,pp.480-484.

[13] Datasheet:ADC0820 8-bit High Speed uP compatible A/D Converter with Track/Hold Function.

[14] R. Erickson, D.Maksimovic, "Fundamentals of Power Electronics, " 2nd Edition, Kluwer 2000.

[15] G. F. Franklin, J. D. Powell and M. Workman, "Digital Control of Dynamic Systems, " Addison-Wesley,third Edition, 1998.

[16] T. W. Martin and S. S. Ang. "Digital control of switching converters, " In Proc. *IEEE Internat. Symp. on Indust. Electron.*, volume 2, pages 480–4, 1995.

[17] A. Syed, E. Ahmed, and D. Maksimovic, "Digital PWM controller with feed-forward compensation, " in Proc. *IEEE APEC'04*, 2004, pp. 60–66.

[18] A. Prodic and D. Maksimovic, "Digital PWM Controller and Current Estimator for a Low-Power Switching Converter," *IEEE Computers in Power Electronic* 2000, pp. 123-128.

[19] G. F. Franklin, J. D. Powell Abbas Emami-Naeini, " Feedback Control of Dynamic System,"4nd. Edition, 2002.

[20] H.Peng,A.Prodit, Dr. Maksimovid,"Modeling of Quantization Effects in Digitally Controlled DC-DC Converters," *IEEE Transations on Power Electronics*, vol. 18, pp. 438-446,January 2003

[21] L. G. John, Y. Hung, R.M. NELMS, "Digital Controllers Design for Buck and Boost Converters Using Root Locus " ,*IEEE Conference* Vol

2, pp. 1864-1869,2003

[22] V. Yousefzadeh, Toru Takayama, D. Maksimovic "Hybrid DPWM wiyh Digital Delay-Locked Loop " *IEEE COMPEL*, july, 2006

[23] O. Trescases, G. Wei, Wai Tung Ng, "A Low-Power DC-DC Converter with Digital Spread Spectrum for Reduced EMI" *37th IEEE Power Electronics Specialists Conference*, june, 2006.

[24] Albert M. Wu, Jinwen Xiao, Dejan Markovic, Seth R. Sanders, "Digital PWM Control: Application in Voltage Regulation Modules", *Annual IEEE of Power Electronics Specialists Conference,* VOL. pp.77-83 July 1999,

[25] M. Morris Mano, "DIGITAL DESIGN" 3rd Edition, 2002

[26] J. Xiao, A. V. Peterchev, J. Zhang, and S. R. Sanders. A 4-*μ*a quiescent-current dual-mode digitally controlled buck converter IC for cellular phone applications. *IEEE J. Solid-State Circ.*, 39(12):2342–2348, December 2004.

[27] Y. Qiu, M. Xu, K. Yao, J. Sun, and F. C. Lee. The multi-frequency small-signal model for buck and multiphase interleaving buck converters. In *Proc. IEEE Applied Power Electron. Conf.,* pages 392–398, 2005.

[28] Matlab 系列叢書，"視覺化建模環境 Simulink 入門與進階"，鈦思科技股份有限公司編著，2001

[29] 張智星，MATLAB 程式設計與應用，清蔚科技出版，2001

[30] 鄭錦聰，莊鎮嘉，MATLAB 進階(含 Simulink)，全華科技圖書，1997

[31] 李宜達，麥焜燦，MATLAB 在工程上的應用，全華科技圖書，2002

[32] 俞克維，控制系統分析與設計 使用 MATLAB，新文京圖書，2004

[33] 林灶生，劉紹漢，Verilog FPGA 晶片設計，全華科技圖書，2004

[34] 簡弘倫，Verilog 晶片設計 使用 ModelSim，文魁資訊出版，2005

[35] 黃英叡，張銓淵， Verilog 硬體描述語言，全華科技圖書，2003

[36] Samir Palnitkar，Verilog 硬體描述語言，全華科技圖書，1999

[37] 鄭信源，Verilog 硬體描述語言數位電路，儒林圖書，2004

[38] 林振華，數位積體電路設計，全華科技圖書，2001

[39] 黃漢邦，自動控制系統，東華書局，2001