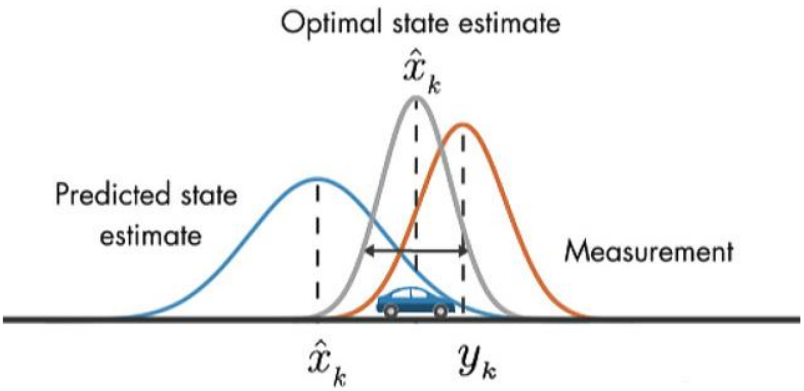
National Tsing Hua University
Department of Electrical Engineering
EE4292 IC Design Laboratory (積體電路設計實驗)
Fall 2022

4D Kalman filter
(Project Report)

Team Member : 108061272 林孟平 108061271 謝明翰
108060021 許庭崴

Abstract



Source: https://blog.csdn.net/young_gy/article/details/78177291

卡爾曼濾波器又稱為最佳線性濾波器，會根據目標變數在不同時間下的觀測值，考慮不同的機率分布，對此變數之未知狀況進行估計。在日常生活中，GPS導航系統是卡爾曼濾波器的其中一項重要應用，透過對行人或汽車進行移動預測，我們可以大幅修正觀測誤差所帶來的影響。因此，本次Project實作出卡爾曼濾波器希望能應用在汽車GPS導航應用上。除了物件的三維位置外，面對的方向 (水平旋轉角度) 也是決定導航策略極為重要的因素，因此，我們設計出之濾波器必須同時處理四個維度的物理量(X, Y, Z, Theta)。為了達成這個目標，我們使用四組相似的濾波器硬體，同時對於輸入之測資進行運算並同時輸出運算結果。

Functionality

電路總共包含四個維度的卡爾曼濾波器，若我們考慮其中一維度之運算，以預測物體X座標為例，我們主要需要控制位置、速度和noise(觀測產生的不確定性)等三個參數，因此在我們設計出卡爾曼濾波器中，需要包含以下核心功能:

(1) 利用前一時刻之運動狀態預測下一時刻之運動狀態，為預測值。

(2) 利用前一時刻之noise預測下一時刻之noise，為預測值。

(3) 由noise的觀測值(濾波器輸入)和預測值，計算出卡爾曼係數。

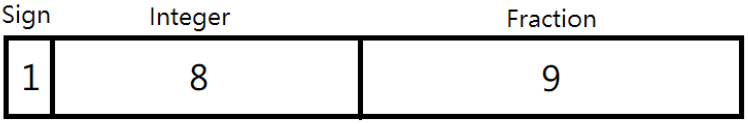
(4) 將卡爾曼係數作用於運動狀態觀測值(濾波器輸入)和預測值，計算出下一時刻修正後的運動狀態(濾波器輸出)

(5) 將卡爾曼係數作用於noise觀測值和預測值，計算出下一時刻修正後的noise

以上的操作包含大量的fixed points除法、矩陣乘加法、矩陣轉置以及反矩陣的求取。

Specification

我們預設的使用環境為台灣本島(394km * 140km)，若我們想像有一個衛星位於台灣的正上空(500km)，並假測衛星偵測出之數據經處理後，精確度可以達2^(-9)公里(~2m)，則我們需要處理的數據範圍將可以透過18bits的signed number進行描述:



根據此目標，我們最終實作出之硬體規格如下:

1. Throughput: 66.7M calculations/sec (@66.7MHz)
2. Area: P&R: 834427um^2 / synthesis: 417479um^2
3. Power: 69mW (P&R)

在這個規格下，若我們的導航系統每秒更新5次狀態，throughputs可供全台灣約1000萬輛的汽車作為導航使用，實用性可達到我們的目標，因此我們最後只需要用一組4D濾波器完成。

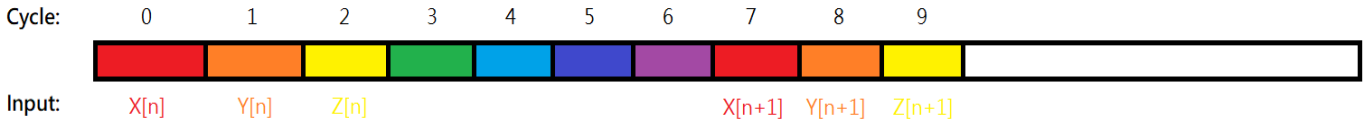
由於卡爾曼濾波器牽扯到除法運算，或許此運算所需要之時間會較長，因此我們共想到直接除法、長除法兩種實現方式，為了比較兩者之優缺點，我們將兩者實際單獨合成後，再將結果作比對，以下是合成結果:

	Timing	Area
Straight division	11.3	~5200
Long division	1~10	6300~12700

最後我們採用直接除法來實作。以下是我們考量的原因:

(1) 計算速度已足夠: 雖然使用長除法可降低Timing(Critical path轉變成矩陣運算中的乘法)，但目前的throughput已經可以達到我們的要求，因此壓低Timing是較為次要的目標。
(2) 長除法需要的面積較大: 雖然面積與長除法中的pipeline的階數有關，但根據我們的觀察結果，整個除法所需要的面積通常都會大於直接除法。
(3) 長除法導致更多的cycle數: 由於卡爾曼濾波器的運算需要X[n-1]的資訊去計算X[n]，若使用長除法搭配pipeline，會導致所需的cycle數更大，進而延後下一個iteration開始的時間。

除此之外，我們實作濾波器時曾考慮是否要在時間上提高平行度(例如: 同時處理X[n]和X[n+1])，但由於卡爾曼濾波器牽扯到遞迴計算，因此對於同一個物理量而言，在前一時間的計算完成前也無法進行下一時間的計算。因此，考量到無法透過加大硬體降低1個iteration所需之時間，我們最終的濾波器未實現時間上的平行度。另外，我們的濾波器會以多工的方式處理輸入的資料，假設我們共處理7台汽車的資料，且計算採7 stage pipeline完成，若我們希望對1筆汽車的資料有最高的Throughput，則概念圖如下:



一筆資料共需7個cycle(105ns)完成，雖然我們未做到時間上的平行度，但速度上足以支持我們的target application。

Implementation

1. 使用Python實現卡爾曼濾波器的軟體演算法，包含所有硬體中的quantization steps，並做fixed point quality(誤差)分析，最後輸出四個維度分別的測試資料以供RTL驗證使用。
2. 根據此演算法先行定義好各個module的 I/O ports，然後分別進行RTL架構以及sub-module(矩陣乘法)的設計。
3. 完成testbench，使用軟體產生出的golden file進行驗證。
4. 功能正確性達成後，完成電路pipeline，並觀察合成結果。

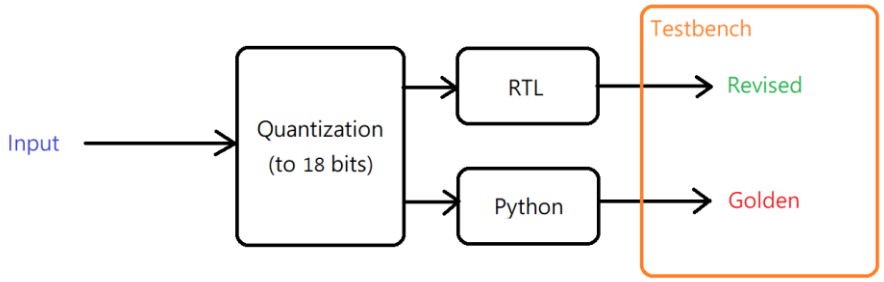
5. 進行電路效能的優化，包含進行直接除法和長除法實用性的討論，最後決定以面積為優先考量點，使用直接除法完成。

6. 進行P&R，由於Interim presentation中報告的spec未預留P&R所需之timing，因此將spec放鬆到一定程度後才完成第一次的後端流程。

7. 將電路再進行優化後，完成第二次的後端流程。

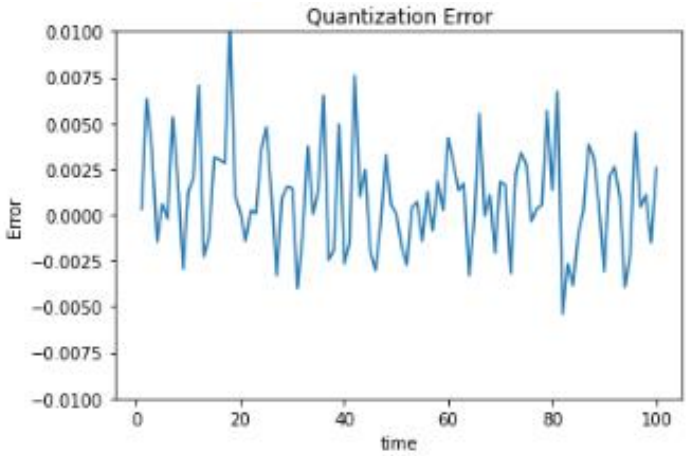
Verification

在這個project當中，我們使用Python作為RTL的behavior model以及產生golden file的工具。為了讓硬體和軟體的運算結果能更接近，Python不僅包含所有濾波器的運算，也同時包含硬體中所有的quantization steps，驗證流程大致為: 將一組訊號作quantization至可供硬體計算後，同時輸入RTL與Python，並在Testbench中比對兩者計算的結果，如下圖:



特別的是，我們使用Python中的Random函式隨機產生輸入測試資料(noise)，因此輸入的訊號有較大的隨機性。

除此之外，由於經quantization後會產生計算誤差，為了檢驗quantization後的硬體計算結果與軟體(無quantization steps)不會有太大的差距，我們也將兩者的計算結果作比對，如下圖:



可以看到以18bits fixed point number的形式，quantization的誤差最大值大約為0.01km(10m)，這樣的誤差在汽車高速行進的狀況下，我們認為是可以被接受的。

Placement and Routing

由於時間上的問題，我們在P&R使用的策略與Lab10~12相似，以能順利完成整個後端流程為主要目標。由於我們之前以測試過synthesis需要之最低Timing，因此後來在synthesis階段就已經預留Global Route需要的空間，到了P&R階段，我們能使用相近的Timing Constraint下去完成。也因為使用的策略與Lab使用的相似，因此Core area使用效率較低(~50%)。未來若有機會嘗試更嚴格的constraint，或許能降低P&R core area與Timing，並將Chip performance提高一定的程度。

Conclusion

目前已有大量關於卡爾曼濾波器的研究與實作，但大部分是側重在數學理論或是軟體實現，以硬體方式實現濾波器的方法較少見。這個Project較為創新的點在於，我們以硬體實現出的濾波器以「4D GPS導航」為目標，並且將電路的Spec以此應用方向做適當的調整，使得整個濾波器能支援位於台灣本島上空的同步衛星，速度方面也得以應付全台灣汽車的導航應用。

除此之外，由於計算精準度是卡爾曼濾波器中很重要的特性，因此我們也在Verification階段說明了：濾波器透過硬體加速器實現所造成誤差是可以被接受的。

最後，我們在這個Project當中也對於硬體資源的取捨做了一些探討，包括是否實現時間平行度以及選擇濾波器中的除法實現方法。透過這些探討，我們也了解到：同一種計算操作在硬體實現上有非常多的可行性，透過瞭解自己的target application所需要的資源與討論各種方法的優缺點，才能從中找出最合適的硬體解決方案。

Contribution of members

林孟平: Python、Verification及RTL架構優化、合成、Interim/Final Presentation、Proposal、Final Report撰寫

謝明翰: Placement and Routing、Post-Simulation與其他後端工作

許庭崴: RTL submodule撰寫

Reference

[1] Kalman filter Tutorial https://www.kalmanfilter.net/alphabeta.html

[2] Use of a Kalman filter to improve real-time video stream image processing https://www.cs.cmu.edu/~motionplanning/papers/sbp_papers/kalman/video_kalman.pdf

[3] Kalman filter – Machine Learning TV https://www.youtube.com/watch?v=LioOvUZ1MiM&ab_channel=MachineLearningTV