Golub-Kahan SVD Hardware Implementation

R07943175 黃家翰, R07943176 高禎謙

***Abstract*—奇異值分解(Singular value decomposition, SVD) 廣泛被用於通訊、數值處理、人工智慧領域。尤其在數位通信上，以 SVD 作為預編碼及解碼是現行十分普遍的做法。為了提高通訊品質，高運算速度且低耗能的 SVD 運算器在現行通訊行動裝置是十分重要的。為了提升 SVD 的運算速率，增加其能源效率，使用硬體加速是必要的技術。本專案採用 Golub-Kahan SVD (GK-SVD)演算法，結合管線化之 CORDIC 電路，並以多個運算單元來進行平行化運算來進行優化，以此設計 4x4 SVD 之硬體電路架構。本專案所設計的 SVD 硬體電路以 130-nm CMOS 製程設計，面積為 1.30 mm^2，操作於 125 MHz 於 1.08V 之功耗為 184.6mW。可運算 4x4 複數矩陣之 SVD，並且能達到良好計算效率。在保持數據的一定精準度的情況下，相較於其他採用 GK-SVD 演算法作為設計主軸的晶片，其能源效率仍然高達 3 倍，且運算速度更是達到 5.17 倍。**

***Keywords*- Golub-Kahan SVD (GK-SVD), 雙對角化 (Bidigonalization), 管線化 CORDIC(Pipeline CORDIC)**

I. INTRODUCTION

奇異值分解(Singular Value Decomposition, SVD) 在線性代數重要的矩陣分解之一。當有一個 m × n 的複數矩陣 A，我們可以利用 SVD 的分解技術，將 A 拆成下式：

$$A = U\Sigma V^{*} \tag{1}$$

其中 U 是 m × m 的正交矩陣，V 是 n × n 的正交矩陣，分別是的 AA^T、A^TA 特徵向量所組成，兩者的元素可以為複數。而 Σ 是 m × n 的類對角矩陣對角線上的元素是特徵值，其值要是要為實數。

奇異值分解在許多領域上扮演十分重要的角色，像是數位通訊中[1]、或是在電腦視覺影像上的訊號處理[2]、或是在深度學習裡的網路壓縮中[3]都是不可或缺的。譬如說在現行數位通訊中廣泛使用的多輸入多輸出(Multiple-Input Multiple-Output, MIMO)，在得到通道(Channel)相關資訊的矩陣後，便可以對該矩陣進行 SVD 分解，所得到的奇異向量(Singular Vector)可以作為訊號的預編碼(Precoding)及解碼(Decoding)的用途，藉此以降低訊號受通道影響所造成的位元錯誤率(Bit Error Rate, BER)。

在現行通訊裝置中，SVD 的運算器是不可或缺的，為保持一定的資料傳輸速度及品質，求出矩陣的 SVD 的時間愈短與好，其運算速度更是要達到一定水準才能符合 IEEE 802.11 的規範。由於行動裝置的興起，在追求 SVD 的運算速度的同時，也需要把運算器的功率消耗納入考量。基於以上理由，本專案的目的便是在現有的演算法加以優化，來設計出一組高運算速度且有著良好的能源效率 SVD 運算加速器。

II. MIMO PRECODING SYSTEM BASED ON SVD

在考慮傳輸端的天線數為 N_t、接收端的天線數 N_r 的 MIMO 無線傳輸系統，其傳送訊號的模型可以由下面的式子來表示：

$$y = Hs + n \tag{2}$$

其中 s 為 N_t×1 的傳輸端所傳輸訊號複數向量，接收端所接收到的訊號向量為 N_r×1，其值也可為複數，天線之間的通道為 N_r×N_t 複數矩陣。若將通道矩陣以 SVD-based 的式來表示，(2)可以改寫成下式：

$$y = U\Sigma V^{*} + n \tag{3}$$

對角矩陣 Σ 中的元素為 H 的奇異值，U 跟 V 則是為 H 的奇異向量(singular vector)矩陣，矩陣裡面的向量相互證交。因此便可利用此特性來對訊號進行編碼：

$$\begin{aligned} s' &= Vs \\ y' &= U^{*}y \end{aligned} \tag{4}$$

最後在接收端所收到的訊號如下：

$$\begin{aligned} y' &= U^{*}U\Sigma V^{*}Vs + U^{*}n \\ &= \Sigma s + U^{*}n \end{aligned} \tag{5}$$

我們可以看到，在經過化簡後，每個傳輸端天線到其相對應的接收端天線可視為個別獨立。有了 SVD-based 的編碼，在經過相關訊號的簡單處理後，便可以有效降低訊號傳遞的位元錯誤率。也因此在得到通道資訊後，若能以 SVD 將其分解，便大大有利於資料傳遞的品質。

III. SVD ALGORITHM

截至目前為止，過往的文獻對於 SVD 提供了許多種演算法。像是 Two-sided Jacobi 演算法[5]、Lanczos 演算法[6]、Divide and Conquer Bidiagonal 演算法[7]等等，其中以 Two-sided Jacobi 最為被廣泛使用。此演算法是以 Givens Rotation 為主，(6)(7)為其主要的矩陣運算式，其核心概念為針對矩陣中兩個元素(或是同一元素的實數跟虛數)，以其中一個做為軸心對另一個做旋轉，使得非軸心的元素的值為 0：

$$\theta = arctan(\frac{b}{a}) \tag{6}$$

$$\begin{bmatrix} x' \\ 0 \end{bmatrix} = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{7}$$

利用 Givens Rotation 便能來對將矩陣中所有非對角線的元素值進行旋轉以降低其值，而對所有非主軸的元素做一次 Givens Rotation 的行為稱為 sweep，來回多次 sweep 之後，非主軸的元素質便近乎為 0，就可以來藉此求出矩陣的 singular value，而 singular vector 可以從這過程中所使用的 Givens Rotation 的矩陣相乘得來。由於 Two-sided Jacobi 演算法結構比較容易進行平行運算，有許多文獻在其基底上進行改良，並以此為主軸設計晶片[8]。然而在 Two-sided Jacobi 演算法中，每次 sweep 所要進行 Givens Rotation 數量極多，代表極高的運算量，縱使可以平行運算的方式來加速，但所仍要負擔運算成本仍十分可觀。

為解決此問題，本作品採用 Golub-Kahan(GK-SVD) 演算法[9]作為硬體設計的 SVD 演算法，GK-SVD 演算法跟 Two-sided Jacobi 演算法一樣以 Givens Rotation 作為其主要運算，不過在做旋轉的元素上卻大不相同。GK-SVD 演算法主要可以分為兩階段(two-phase)來看，分別為雙對角化(Bidiagonalization)以及對角化(Diagonalization)。雙對角化的目的是將在對角線及上對角線以外的矩陣都按照順序以 Givens Rotation 矩陣來將其值轉到 0，同一時間也把對角線及上對角線上的值從複數轉為實數。而對角化則是採用類似 Two-sided Jacobi 的方式，對將矩陣中所有上對角線的元素值進行旋轉以降低其值，而將上對角線的元素來回旋轉一次的行為為 sweep，在多次 sweep 之後，上對角線的元素值也就便趨近於 0。GK-SVD 演算法的詳細細節都顯示在 Algorithm 1, 2。

雖然 GK-SVD 能進行平行運算的部分較於 Two-sided Jacobi 演算法要來的少，但是它在對角化時一次 sweep 所需要的 Givens Rotation 數量遠低於 Two-sided Jacobi 演算法，而且所需要處理的元素皆為實數，因此縱使 GK-SVD 在對角化的 sweep 數量比較多，且有雙對角化的步驟，但是在整體的運算數量上卻少於 Two-sided Jacobi 演算法。

Algorithm 1 Bidiagonalization

Require: Channel matrix Hr,Hi
Ensure: Bidiagonal matrix B.
for $k = 1$ to n **do**
  for $i_1 = 1$ to n **do**
    $angle = FindAngle(Hr_{i_1,k}, Hi_{i_1,k})$
    for $t = 1$ to n **do**
      $[Hr_{i_1,t}, Hi_{i_1,t}] = Rotation(Hr_{i_1,t}, Hi_{i_1,t}, angle)$
  for $i_2 = n-1$ to k **do**
    $angle = FindAngle(Hr_{i_2,k}, Hr_{i_2+1,k})$
    for $t = 1$ to n **do**
      $[Hr_{i_2,t}, Hr_{i_2+1,t}] = Rotation(Hr_{i_2,t}, Hr_{i_2+1,t}, angle)$
      $[Hi_{i_2,t}, Hi_{i_2+1,t}] = Rotation(Hi_{i_2,t}, Hi_{i_2+1,t}, angle)$
  for $j_1 = k+1$ to n **do**
    $angle = FindAngle(Hr_{k,j_1}, Hi_{k,j_1})$
    for $t = 1$ to n **do**
      $[Hr_{k,j_1}, Hi_{k,j_1}] = Rotation(Hr_{k,j_1}, Hi_{k,j_i}, angle)$
  for $j_2 = n-1$ to $k+1$ **do**
    $angle = FindAngle(Hr_{k,j_2}, Hr_{k,j_2+1})$
    for $t = 1$ to n **do**
      $[Hr_{t,j_2}, Hr_{t,j_2+1}] = Rotation(Hr_{t,j_2}, Hr_{t,j_2+1}, angle)$
      $[Hi_{t,j_2}, Hi_{t,j_2+1}] = Rotation(Hi_{t,j_2}, Hi_{t,j_2+1}, angle)$
$B = Hr$

Alg. 1: Bidiagonalization 演算法。

Algorithm 2 Diagonal Nullification

Require: Bidiagonal matrix B. Sweep Number m.
Ensure: Singular Values S.
for $k = 1$ to m **do**
  for $i = 1$ to $n-1$ **do**
    $angle = FindAngle(B_{i,i}, B_{i,i+1})$
    for $t = 1$ to n **do**
      $[B_{i,t}, B_{i+1,t}] = Rotation(B_{i,t}, B_{i+1,t}, angle)$
  for $j = 1$ to $n-1$ **do**
    $angle = FindAngle(B_{j,j}, B_{j+1,j})$
    for $t = 1$ to n **do**
      $[B_{j,t}, B_{j+1,t}] = Rotation(B_{j,t}, B_{j+1,t}, angle)$
$S = B$

Alg. 2: 透過多次 sweep 的 diagonal nullification 演算法。

IV. HARDWARE ARCHITECTURE DESIGN

上述所提 GK-SVD 演算法，是以 Givens Rotation 的複數或是向量間的旋轉為主要運算單元，因此在硬體設計中，Processing Element 主要是 Givens Rotation 的運算。在整體的設計中我們採用了記憶體導向(Memory-based)[10]的硬體架構，精準度為 18 位定點數。Fig. 1 為此次專案所使用的硬體架構圖。針對 4x4 複數矩陣，我們以 2 x 4 x 4 x 18 = 576 個 D flip-flop 儲存，總共所要存的矩陣為 3 個(一個拿來存通道矩陣，其餘兩個來存 singular vector 的矩陣)，且為加速運算，我們共放置了 4 個 Processing Element，兩個主要負責 singular value 的運算，其他兩個則是負責計算 singular vector。Rotatiion Angle Registers 則是用來儲存旋轉時所用的角度，以便之後在旋轉時取用。

A. Processing Elements

Processing Unit (PE)為因應不同的運算有四種不同的設定模式，分別為 Complex to Real, Complex Rotation, Real Nullification, Related Rotation，不同的運算所針對的矩陣元素對象也不相同，以下為四種模式的功能介紹：

- Complex to Real: 將指定元素的值從複數轉到實數，也就是說該元素的虛數值將會被轉到 0。1 個 PE 一次可以同時接收處理兩個複數元素。
- Complex Rotation: 對指定元素做指定角度的複數平面旋轉。1 個 PE 一次可以同時接收處理兩個複數元素。
- Real Nullification: 對於一組(兩個)指定實數元素，將其中一個做為軸心，將另一個元素轉到 0。1 個 PE 一次僅能接收處理一組實數元素。
- Related Rotation: 對於兩個指定元素做指定角度的旋轉。1 個 PE 一次僅能接收處理一組複數元素。

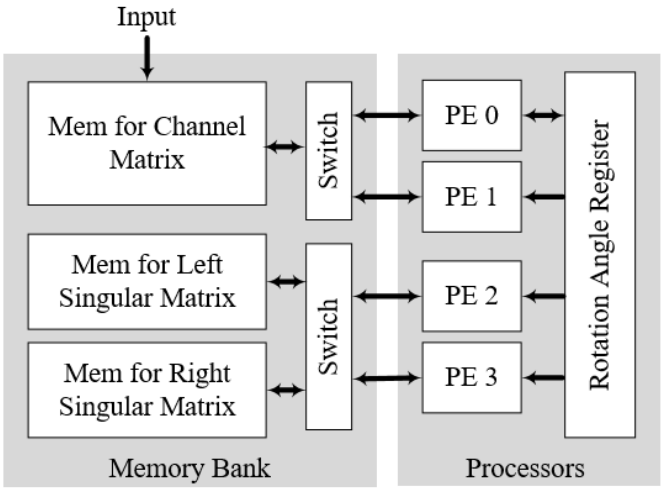

Fig. 1: 本設計之電路模型架構，主要分為 Memory Bank。

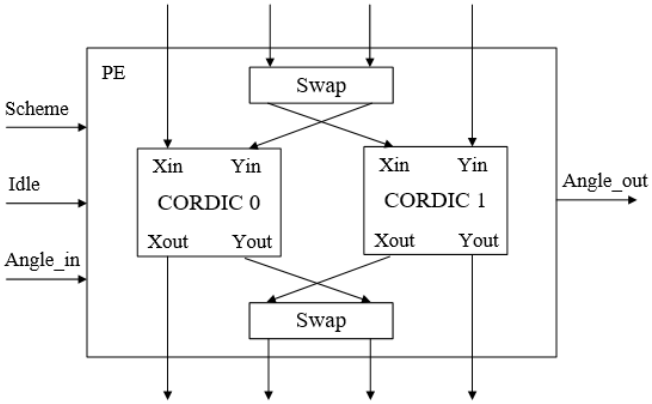

Fig. 2: Processing element (PE) 之內部架構，主要由兩個 CORDIC 旋轉單元組成。

	PE 0			PE 1		
	Swap	CORDIC 0	CORDIC 1	Swap	CORDIC 0	CORDIC 1
Complex to Real	0	Vectoring	Rotation	0	Vectoring	Rotation
Complex Rotation	0	Rotation	Rotation	0	Rotation	Rotation
Real Nullification	1	Vectoring	Rotation	1	Vectoring	Rotation
Related Rotation	1	Rotation	Rotation	1	Rotation	Rotation

Table 1: PE 在不同輸入 scheme 下之運作模式。

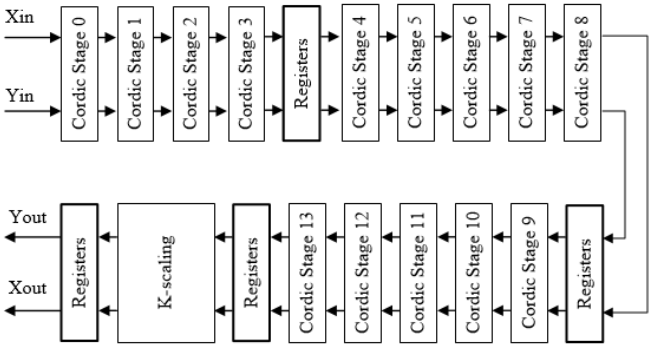

Fig. 3: CORDIC 管線化架構。14 級 CORDIC 被分為 4 級管線。

架構設計，包含兩個 CORDIC，皆可以操作在 Vector Mode 跟 Rotation Mode，以及兩個交換(swap)的 switch。而 Table 1 為不同的 PE 模式所對應的 swap 控制以及 CORDIC 模式。從中可以看出並不是所有 PE 都需要用到這四種操作模式，像是負責 singular vector 計算的 PE 2 跟 PE 3，只會使用到 Complex Rotation 及 Related Rotation，在特定的元素旋轉安排下，PE 1 也是只會用到上述的兩種模式，因此在 CORDIC 中便不會用到 Vector Mode，便可簡化電路，省去不需要運算資源。

B. Pipeline Cordic Design

本次所設計的 CORDIC 架構總共有 14 層旋轉運算，為了讓 CORDIC 在每個週期中都可以處理新的元素運算，以提升 CORDIC 的運算量，因此這 14 層旋轉的運算架構是以串聯的方式所連接起來。然而 14 層旋轉運算的 critical path 的 delay 極高，可能致使週期極大。為了在維持每個週期的運算量下降低電路 critical path 的長度，我們在這 14 層旋轉中以管線化(pipeline)插入 4 層 D flip-flop。Fig. 3 顯示 4 層 D flip-flop 所插入的詳細位置，可以看出是近乎平均分配的方式插入在這 14 層旋轉及 K scaling 的組合邏輯電路朱，以期能最大化提升操作頻率。

C. Schedule of Processing Element

根據 IV-A 中所描述，由於單一 PE 在單一時間所能安排的運算量有限，是故如何安排 PE 去處理相對應的元素便顯得十分重要。由於在 GK-SVD 中兩個階段所遇到的狀況大不相同，因而分兩段處理。在這邊處理 Channel Matrix 的 PE 0 與 PE 1 作為代表，來探討 PE 的運算安排，而 PE 2 跟 PE 3 基本上操作的運算跟前兩個相似，只是對應的矩陣有些差別而已。

- 雙對角化(Bidiagonalization):

  在 Fig. 4 中顯示了在雙對角化過程的 PE 運算過程。在圖中可以看到

- 對角化(Diagonalization):

  在 Fig.5 中則是顯示對角化的過程，由於在對角化期間，在同一時間需要做運算的元素通常只有一組 2x2 的區域，而且往往需要等其運算完畢才能算下一組 2x2 的區域，因此會在中間插閒置週期(idle cycle)來等待，像是在 Fig.5 中的 cycle 3~4 及 7~8，不過每次在 sweep 的過程中仍有機會遇到可以平行運算的週期，像是 Fig.5 中的 cycle 9~12，在這邊便不會插入閒置週期，而是插入新的元素來運算，如此便可以在每次 sweep 中省下 4 個週期。

V. IMPLIMENTATION RESULT AND COMPARISON

本次專案在將對角化的 sweep 次數限制為 8 的情況下實現上述的 4x4 SVD 硬體電路架構。在這架構設計下，對於每個 4x4 的矩陣，總共需要 280 個週期來運算，前 16 個週期是用來接收欲計算之矩陣，緊接著所接的 68 個週期是進行雙對角化用途的，再來是 160 個週期是做矩陣對

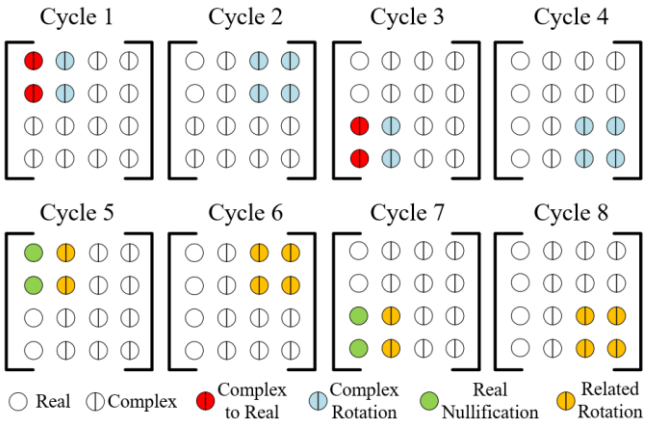

Fig. 4: PE 進行雙對角化的操作順序

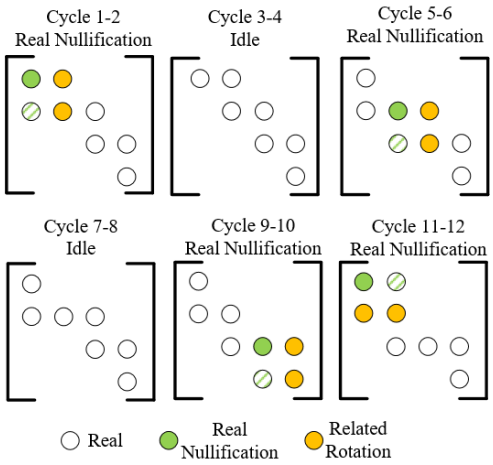

Fig. 5: PE 進行對角化消去的操作順序

角化的運算，最後 36 個週期是輸出運算結果。接著以 TSMC 130 nm 製程合成。合成後的電路之操作頻率為 125MHz，面積約為 1.296mm^2。並進行以下測試。

A. MIMO Channel Reconstruction and Sumulation

在隨機生成三份複數 4x4 通道的矩陣測資(A, B, C)的情況下，以此電路跑完 SVD 後的運算結果，我們定義其 SQNR 值以評估測次矩陣在經過 SVD 後所能還原成原矩陣的程度如下:

$$SQNR = 20log(\frac{\|H\|}{\|H - U\Sigma V^*\|}) \quad (8)$$

其中 H 為輸入之 channel matrix，U、Σ、V 分別為輸出之 left singular vectors、singular values、right singular vectors。Table 2 中可以看出，在採用 18 位定點數儲存資料，經由此電路計算出其 SVD 結果具有良好的精準度，在上述之 SQNR 定義下皆超過 45 dB。

Testbench	SQNR [dB]
Channel A	53.1230
Channel B	46.4868
Channel C	54.4338

Table 2: 不同 channel matrix 在本電路上測試之 SQNR 值。

我們再根據上述 SVD 所運算之結果作為訊號傳遞的預編碼跟解碼，並以 MATLAB 模擬來傳輸資料在不同訊噪比的情況下，以 256 QAM 來傳輸資料 1000000 組資料，並且以 MMSE filter 來處理解碼後之訊號。Fig. 6 為其結果，可以從圖中看出，在訊噪比大於 16 dB 的情況下，其位元錯誤率便可達到 10^{-3} 左右。

B. Comparison

Table 3 為本設計與其他 SVD 架構之比較。需要注意的是本設計之設計流程只到電路合成，而其餘架構為 layout 數據。在 Table3 中可以看到本設計相較於採用 Two-sided Jacobi 演算法[8]仍有段差距。但相較於採用相同 GK-SVD 演算法的[11]，本設計有較快的計算時間，進而提高了其運算速度。另外由於製程的差異，在比較能源效率上(power efficiency)和面積效率上(area efficiency)時應將各數據標準化至相同的製程進行比較。在 Table 3 中能源效率和面積效率的數據皆為標準化至 130nm 製程的結果。可以看到本設計受制於面積較大的因素，area efficiency 較其他三者為低，但能源效率則因運算速度較佳的緣故而比[11]要好。整體來說，本設計與[11]所採用相同的演算法，但能源效率為其 3 倍，且運算速度更是達到 5.17 倍，其原因是在於[11]有較長的計算時間與較小的面積，而本次設計則以較大的面積來換取較高的運算速度。

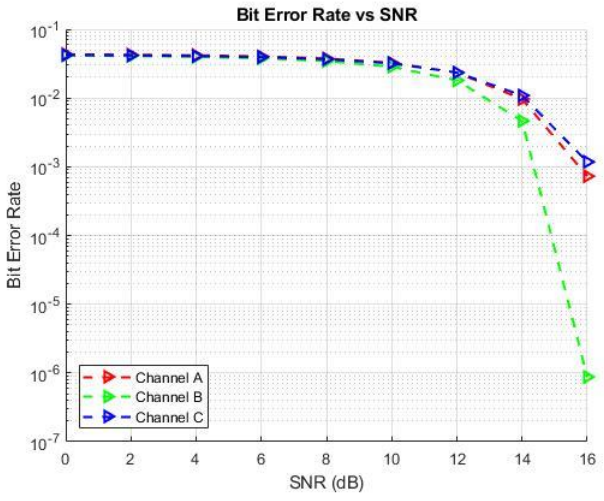

Fig. 6: 以 256QAM 對本設計之電路測試之 MMSE 結果。可以看到自 12dB 起 BER 有明顯的下降。

	napSVD[8]	MDU1[11]	MDU2[11]	Ours
Process	90	180	180	130
Implementation	layout	tape-out	tape-out	synthesis
SVD Algorithm	2-sided Jacobi	GK	GK	GK
Matrix Dimension	4x4	4x4	4x4	4x4
Data Representation	Fixed-point 13 bits	Fixed-point 16 bits	Fixed-point 16 bits	Fixed-point 18 bits
Area [mm^2]	1.34	0.41	0.37	1.30
Frequency [MHz]	752	133	272	125
Computation Cycle	48	1539	4306	280
Throghuput [kMat/s]	15666.7	86.4	63.2	446.4
Core Voltage [V]	1.0	1.8	1.8	1.08
Power [mW]	595	105	155	184.6
Norm. Area Eff. [MMats/s/mm^2]	2.80	0.50	0.40	0.34
Norm. Power Eff. [Mat/μJ]	11.27	0.75	0.83	2.42

Table 3: 本設計架構與其他 SVD 架構之比較。其中 power efficeincy 與 area efficiency 為換算至 130nm 製程後的結果。

VI. CONCLUSIONSON

本專案採用 GK-SVD 演算法來實作奇異值分解(Singular value decomposition, SVD) 的電路，藉由向量的旋轉消去虛部與非對角線的矩陣值，並同時透過反向的旋轉產生 singular vector。由於此演算法在計算旋轉時需要同時取複數值或是兩個矩陣值，因此在硬體設計上兩者可藉由平行化同時處理。並將 CORDIC 旋轉單元分為 4 級管線，除了配合矩陣大小進行 4 個週期的消去過程外，同時也縮短 CORDIC 的 critical path 並提高運算速度。將本設計以 130nm 製程合成後，運作於 125 MHz 並達到約 50dB 的精準度，然而面積較大。在整體電路表現上雖不及 state-of-the-art[4]，但相較於採用相同演算法的[5]，其在計算時間和能源效率上有著不錯的表現。

REFERENCES

[1] Y. Hwang, K. Chen and C. Wu, "A high throughput unified SVD/QRD precoder design for MIMO OFDM systems,“ DSP , 2015

[2] H. S. Prasantha, H. L. Shashidhara and K. N. B. Murthy, "Image Compression Using SVD," ICCIMA, 2007

[3] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, “Exploiting linear structure within convolutional networks for efficient evaluation,” NIPS , 2014

[4] T.-D. Chiueh, P.-Y. Tsai, and I.-W. Lai, Baseband Receiver Design for Wireless MIMO-OFDM Communications, 2nd edition, Wiley-IEEE Press, pp. 239-244, 2012

[5] M. W. Berry, D. Mezher, B. Philippe, A. Sameh, "Parallel algorithms for the singular value decomposition. In: Kontoghiorghes," E.J. (Ed.),Handbook of Parallel Computing and Statistics. Chapman & Hall/CRC Press, London/Boca Raton, pp123-127, 2005

[6] H. D. Simon and H. Zha, "Low-rank matrix approximation using the Lanczos bidiagonalization process with applications," SIAM J. Sci. Comput., 21, pp. 2257–2274. 2000

[7] M. Gu, and S.C. Eisenstat, “A divide and conquer algorithm for the bidiagonal SVD,” SIAM J. Matrix Anal. Appl. 16, pp.79-92, Jan 1995

[8] D. Guenther, R. Leupers and G. Ascheid, "A Scalable, Multimode SVD Precoding ASIC Based on the Cyclic Jacobi Method,“ TCAS I, vol. 63, no. 8, pp. 1283-1294, Aug. 2016

[9] L. Hogben, Handbook of Linear Algebra. Boca Raton, FL, USA: CRC Press, 2nd ed., Ch.45, 2013

[10] C. Wu and P. Tsai, "An SVD Processor Based on Golub–Reinsch Algorithm for MIMO Precoding With Adjustable Precision," TCAS I: Regular Papers, vol. 66, no. 7, pp. 2572-2583, July 2019.

[11] C. Studer, P. Blosch, P. Friedli, and A. Burg,“Matrix decomposition architecture for MIMO systems: Design and implementation trade-offs,” in Conf. Rec. Fourty-First Asilomar ACSSC, Nov. 2007, pp. 1986–1990