Set Partitioning In Hierarchical Trees

林克帆 [1]

1 摘要

這篇專業論文深入探討了 SPIHT 算法在圖像壓縮領域的應用。SPIHT，全稱層次樹中的集合分割，是一種先進的技術。文章不僅介紹了這一算法的基本概念和運作原理，還展示了其硬件實現設計，包括離散小波變換、最大幅值分析和 SPIHT 處理階段。文章闡述了 SPIHT 如何利用小波技術高效壓縮圖像，並討論了其在靜態圖像壓縮中的卓越表現，特別是它能創建嵌入式位流的能力。在硬件設計方面，文章詳細介紹了系統架構及其在速度和效率上的優化策略。最後，通過將硬件加速設計與 SPIHT 的原始軟件版本相比較，強調了其在處理速度上的重大提升。

2 SPIHT 算法簡介

因爲電腦的記憶容量和運算限制，影像壓縮一直是相當重要的研究領域。目前有各式各樣的壓縮演算法被提出，試圖解決影像壓縮的限制。根據資訊理論中的編碼定理，當資料壓縮技術變得更有效率時，相對應它們的計算複雜性也會增加。然而 SPIHT 突破了這種限制，不僅在性能上能媲美最複雜的方法，還具有非常快的執行速度，並生成一種特殊的嵌入式位元流。SPIHT 是一種基於小波技術的圖像壓縮編碼工具。它的工作原理是先將圖像轉換成小波變換形式，然後傳輸有關小波係數的訊息。在解碼階段，解碼器利用接收到的信號重建小波，並進行逆向變換來恢復原始圖像。SPIHT 和它的前身——嵌入式零樹小波編碼器——在靜態圖像壓縮領域是重要的突破。它們相比向量量化、JPEG，以及結合量化的小波技術，提供了顯著更好的圖像質量。而且，SPIHT 不需要特別的訓練就能產生嵌入式的位元流，並在多個方面展現出優異的特性。

3 算法原理

離散小波變換（DWT）是一種特殊的圖像處理技術，它在單一維度上對圖像信號進行高低頻濾波。這個過程會生成一張由高頻和低頻部分組成的新圖像。接著，在另一個維度重複這個步驟，最終形成四個子帶：三個高頻和一個低頻子帶。爲了得到更深層次的小波變換，會在上一層的低頻子帶上再次進行橫向和縱向的變換。這個過程會根據需要重複多次。值得一提的是，每一步的變換都是可逆的，這意味著可以從經過小波變換的圖像中完整重建出原始圖像。圖 1 展示了舊金山的衛星圖片及其經過三層次小波變換處理後的樣子。

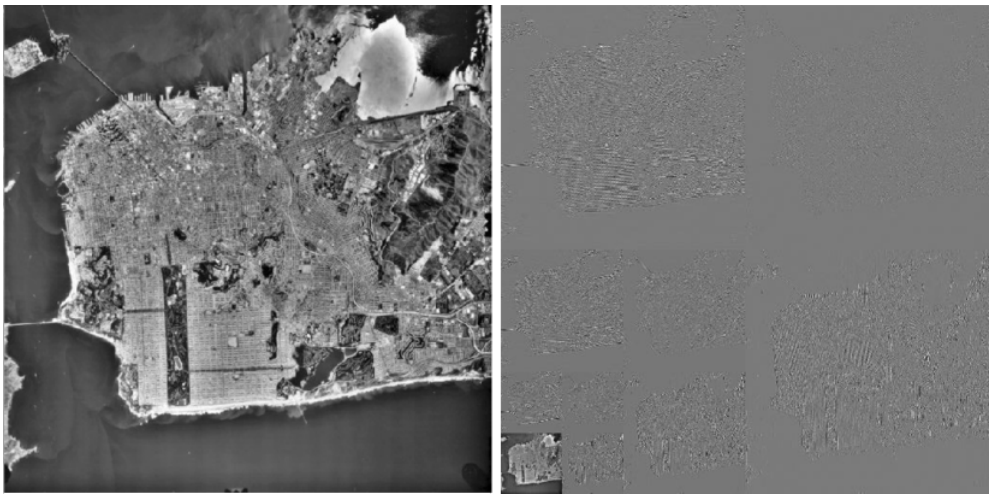

图 1 Three-level DWT

SPIHT 算法的第一步是對影像進行小波轉換，原始圖片會被分解爲不同尺度的子帶圖像。不一樣的尺度又可以被視作不同的層次，因此透過小波轉換得到的子代圖像會形成一種空間金字塔的結構。金字塔每個層次中的係數分別代表該層次特定區塊的頻率訊息以及鄰近區域的相關性。

在一般情況下，一張圖像中的大部分能量都集中在低頻部分。這意味著，當子帶金字塔的頂層向底層過渡時，圖像的方差會逐漸減小。此外，不同子帶之間存在著一定的空間

自相似性，這意味著，如果沿著金字塔保持相同的空間方向向下移動，那麼系數的排序將會更爲合理和有序。舉個例子，金字塔的最高層能夠發現大面積的低頻區域，這些區域會在較低層的相同空間位置被再次呈現。

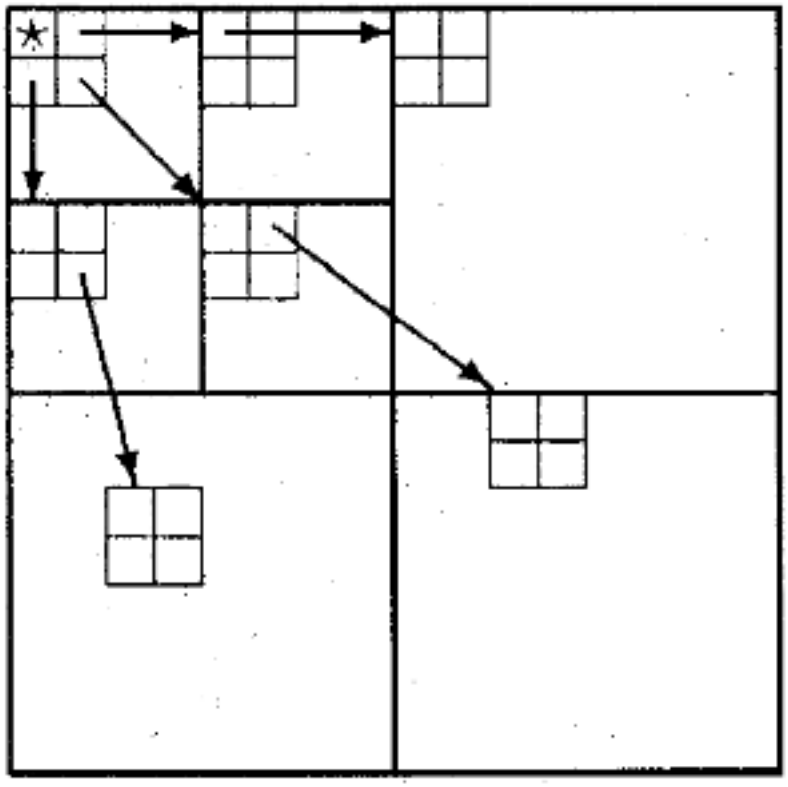

图 2 Spatial-orientation trees.

作者將這個金字塔結構稱爲空間方向樹。正如圖 2 所示，空間方向樹是透過在逐步分裂四個子帶的方式構建的。樹中的每個節點都對應一個像素，並由該像素的坐標所識別。每個節點的直接後代（子節點）則對應金字塔下一層更細的相同空間方向的像素。這棵樹被設計成每個節點要麼沒有後代(即葉子節點)，要麼有四個後代，這四個後代總是組成一組 2 x 2 的相鄰像素。在圖 2 中，箭頭表示從父節點指向它的四個子節點。位於金字塔最高層的像素是整棵樹的根部，也是以 2 x 2 相鄰像素的形式分組的。然而，這些像素的後代分支規則有所不同，每組中有一個（在圖 2 中以星號標示的）是沒有後代的。

在 SPIHT 編碼過程從初始化階段開始。首先算法會計算出需要多少位元（bits）'n' 來表示最大的小波係數的絕對值，$n = \left\lfloor \log_2(\max_{(i,j)} |c_{i,j}|) \right\rfloor$，這個數字將成爲圖像表示深度的基

準。接著，它會建立兩個主要的列表：一個是空白的重要像素列表 (LSP)，這個列表隨後會加入那些重要係數的座標；另一個是不重要像素列表 (LIP)，裡面包含了所有還未被判定爲重要的係數的座標。此外，它還會創建一個不重要集合列表 (LIS)，裡面填入了一些類型 A 的條目，這些條目代表了在小波域中相關聯的一組係數。

完成初始化之後，算法進入了排序階段，這一階段它會仔細處理 LIP 和 LIS。對於 LIP 裡的每個係數，算法會根據當前的閾值來評估它們的重要性。那些被認爲重要的係數會被移動到 LSP，同時它們的符號也會被記錄下來。而對於 LIS 而言，處理起來更爲複雜，因爲它有自己的層次結構。LIS 裡類型 A 的條目代表了那些在空間上相關的係數集。算法會檢查這些集合的整體重要性。如果一個集合被認爲重要，它的每個成員都會被單獨檢查：重要的成員會轉移到 LSP，並且它們的符號會被輸出，而其餘的則會被轉移到 LIP 以便進一步檢查。如果評估過後，集合中還有未處理的係數，它就會被轉變成類型 B 的條目，進行更深入的處理。在 LIS 中，類型 B 的條目會以不同的方式處理：需要輸出整個集合的重要性，如果重要，就會將其成員升級到 LIP，作爲新的類型 A 條目進行進一步分析。

最後階段是細化過程。在這個階段，算法會通過迭代 LSP 中的重要係數（不包括上次排序階段處理過的）來提升圖像的細節表現。它會輸出每個係數二進制表示中的下一位重要位元，從而爲圖像增加更多細節層次。這種逐步細化的方式使得圖像能夠從壓縮的數據流中達到所需的還原精度。

透過這些處理，SPIHT 演算法能夠利用自然圖像的特性，比如那些均勻的亮度區域或對比區域間的邊緣，來有效編碼圖像資訊，這是通過優先處理的小波係數來達成的。透過採用階層結構，它可以確保在壓縮流程的早期就捕捉到重要的視覺細節，從而在壓縮後能夠提供出高品質圖像的重建和漸進式傳輸能力。

4 硬體設計

這篇論文的硬體系統設計分爲三大部分：小波變換、最大幅值的計算，以及固定順序的 SPIHT 編碼。這三個階段分別在三塊 Virtex 晶片上運行。將每個階段安置在不同的晶片上，使得系統能夠同時處理多張圖像，提高了工作效率。數據會透過共享記憶體從一個階

段傳送到下一個階段。

當一個階段的處理完畢後，系統會切換到交叉開關模式，使下一個晶片能夠讀取到已處理的數據。這種設計允許同時對不同圖像進行編碼，系統整體的處理速度將取決於最慢的階段，而整體的延遲時間則是三個階段的總和。圖 3 中展示了系統的整體架構。

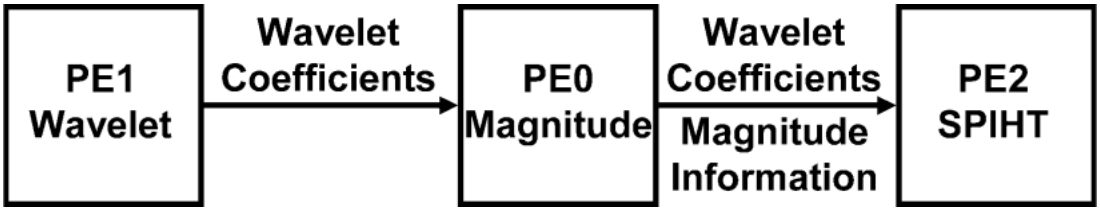

图 3 Illustration of a folded architecture

5 DWT Phase

在開發離散小波變換（DWT）架構的過程中，作者選擇了摺疊方法而非之前的平行處理方法。儘管平行處理架構在節省記憶體頻寬方面有所優勢，但它們需要更多的資源和複雜的排程算法。而且由於每個後續小波層級的大小只有前一層的四分之一，節省效果並不明顯。例如，在一個七層級的 DWT 中，前四層的計算時間僅占第一層的 2%。

作者選擇的摺疊架構一次只處理一個小波層級的單一維度。像素從一個記憶體埠水平讀取後直接輸出到另一個埠，同時將圖像沿 45 度對角線進行反轉。這種反轉允許在下一階段計算小波的垂直維度，從而有效地將圖像恢復到原始方向。在每個小波層級中，圖像在每個維度上都被縮小一半，並且這一過程會反覆進行。

在這個系統設計中，考慮到每個小波係數佔用 16 位元，而記憶體埠的寬度爲 64 位元，因此在每個時鐘週期內最多可以傳輸四行像素。圖 4 說明了離散小波轉換階段的架構。這是因爲一個 64 位元寬的記憶體埠可以在一個時鐘週期內同時處理四個 16 位元的係數。因此，對於 N×N 大小的圖像，最底層的小波層級可在 N/4 的時鐘周期內完成計算，每個後續層級的計算時間則是前一層的四分之一。理論上，如果小波層級無限增加，整個圖像的處理時間將是 $\frac{3}{4} \cdot N^2$ 時鐘周期。

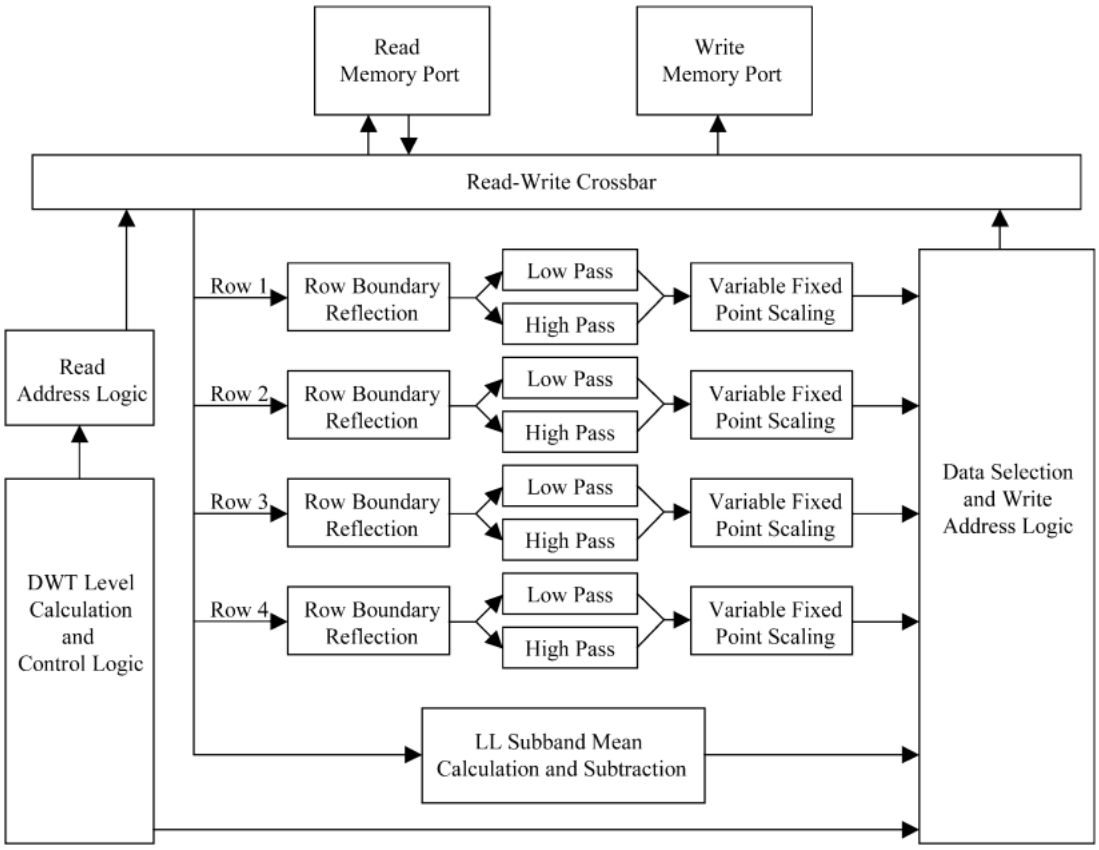

图 4 DWT Architecture

相比之下，其他設計用來同時處理多個小波層級的平行架構通常需要超過一個時鐘周期來處理每個圖像。這種局限性通常來自於可用資源的限制，特別是在現場可編程閘陣列（FPGA）中的運算資源有侷限性。儘管 FPGA 技術非常先進，但它們依舊無法同時處理多行以提高效率。因此，這些平行架構的運行時間往往超出摺疊架構設定的每像素 3/4 時鐘周期的基準。

6 Maximum Magnitude Phase

在 SPIHT 圖像壓縮算法的最大幅值階段，採取了幾項重要步驟來組織和評估圖像的小波係數。這個階段的關鍵在於計算每一組小波轉換後的子節點中，哪個有最大的幅值。這不僅包括追蹤當前最大的幅值，還要記錄下每 16 個子係數的閾值和正負符號。其中，一個關鍵的過程是按照一種稱爲 Morton 掃描的方式，重新排列這些小波係數。

爲了精確找出每個節點下係數的最大幅值，系統會以深度優先的方式進行搜尋，這確

保在檢查新的係數時，它的所有子係數都已經處理過，從而實時更新最大係數值。在這個過程中，每次都會將新出現的係數與當前最大值比較，並在需要時進行更新。這個過程非常高效，因爲在這個幅值階段，系統使用了32位寬的記憶體端口，一次可以讀取半塊的數據。

系統中的空間方向樹是通過一種特定的狀態機制來導航的，它在遍歷樹的過程中，下降時讀取半塊數據，上升時讀取另一半。這樣做確保了在機制向上移動時，計算每個塊的最大幅值所需的所有數據都已準備就緒。此外，系統會把每一層的最新四個塊存儲起來，以便父節點可以訪問它的所有16個子係數。

深度優先搜索順序的一個顯著優勢是它與每個級別內的 Morton 掃描排序自然對齊，儘管在不同層次之間會有所交錯。通過將每一層的數據分配到不同的記憶體區域，然後從最高層到最低層依次訪問，有效地實現了 Morton 掃描排序。此過程還通過同時讀取兩個像素，只需對圖像進行一次掃描，大大簡化了操作，使每個像素的處理時間只需半個時鐘週期。如此高效的運作確保了這一幅值階段的速度比小波轉換階段還要快，從而保持了整體系統的高效吞吐。

7 SPIHT Phase

在 SPIHT 圖像壓縮算法的最後階段，即 SPIHT 編碼階段，系統根據之前階段的數據，同時進行 SPIHT 編碼。這裡，從最高小波層次的係數塊開始，一直到最低層次進行讀取。在從記憶體加載數據的同時，數據的表示形式會從變動定點表示轉換爲每一層次小波轉換的共同定點表示形式。當每個塊都調整爲統一的數字表示後，系統會使用SPIHT的並行版本來計算每個塊將對每個位平面提供哪些信息。

這些信息會先進行分組和計數，然後加入到每個位平面的三個不同的變數FIFO（先進先出隊列）中。這些變數 FIFO 接收的數據大小不一，範圍在 0 到 37 位之間。變數 FIFO 將這些塊數據安排成規則的32位字節，方便記憶體的存取。此外，系統還特別設計了防止算法在任一變數FIFO過滿時進行暫停的機制。一個動態的調度器會持續選擇最滿的FIFO進行數據寫入記憶體。SPIHT 編碼階段的整體流程圖可以在圖 5 中找到。

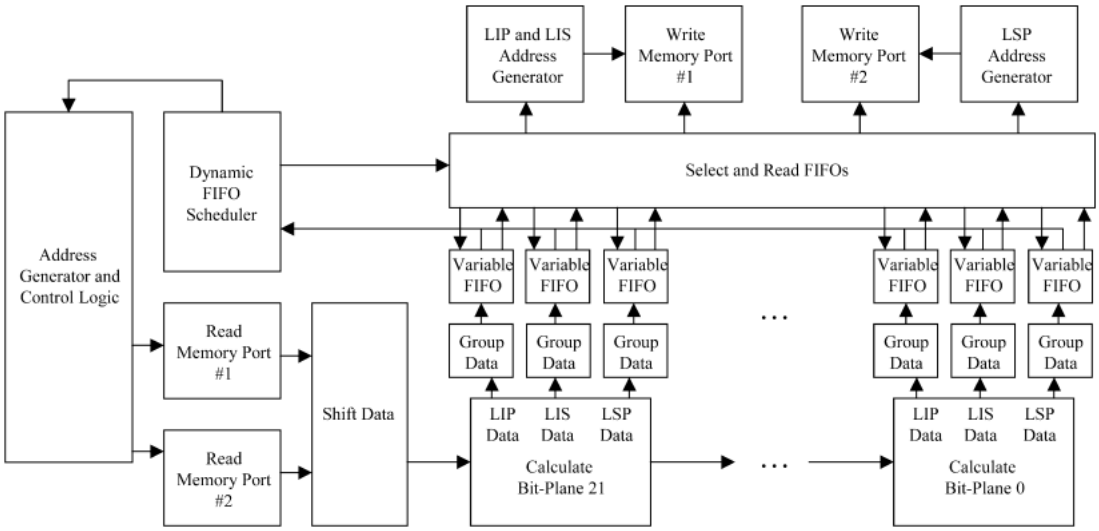

图 5 SPIHT coding phase block diagram

8 Results

表 1 對比了硬體加速設計與 SPIHT 官網上提供的原版軟體版本的 SPIHT 。這個比較沒有考慮算術編碼，因爲目前的硬件版本還沒有對最終的位流進行算術編碼。在對 NASA 提供的樣本圖像進行測試時，作者發現算術編碼對整體壓縮效果提升不大，因此決定不採用算術編碼。使用了 IBM RS/6000 Model 270 工作站進行比較，並結合了標準圖像壓縮基準圖像和 NASA 官網的衛星圖像進行測試。在不計算硬碟讀取時間的情況下，軟體版本的 SPIHT 平均需要 1.101 秒來壓縮一張 512×512 的圖像。而在硬體版本中，小波階段只需 2.48 毫秒即可完成計算，這使得 SPIHT 引擎的速度提高了 443 倍。另外通過對小波階段進行並行處理，有可能進一步提高 SPIHT 引擎的運行效率。

雖然這是在不考慮資料傳輸時間的情況下所能達到的加速比例，但這種設計也可以用來加速 CPU 上的 SPIHT。在這樣的系統中，數據的讀取和寫入時間也是必須考慮的因素。透過 PCI 總線與主處理器連接，寫入圖像需要 13 毫秒，讀取最終的數據流則需要 20.75 毫秒。即使加上數據傳輸的延遲，總體來說，SPIHT 的硬體版本仍然實現了 31.4 倍的速度提升。

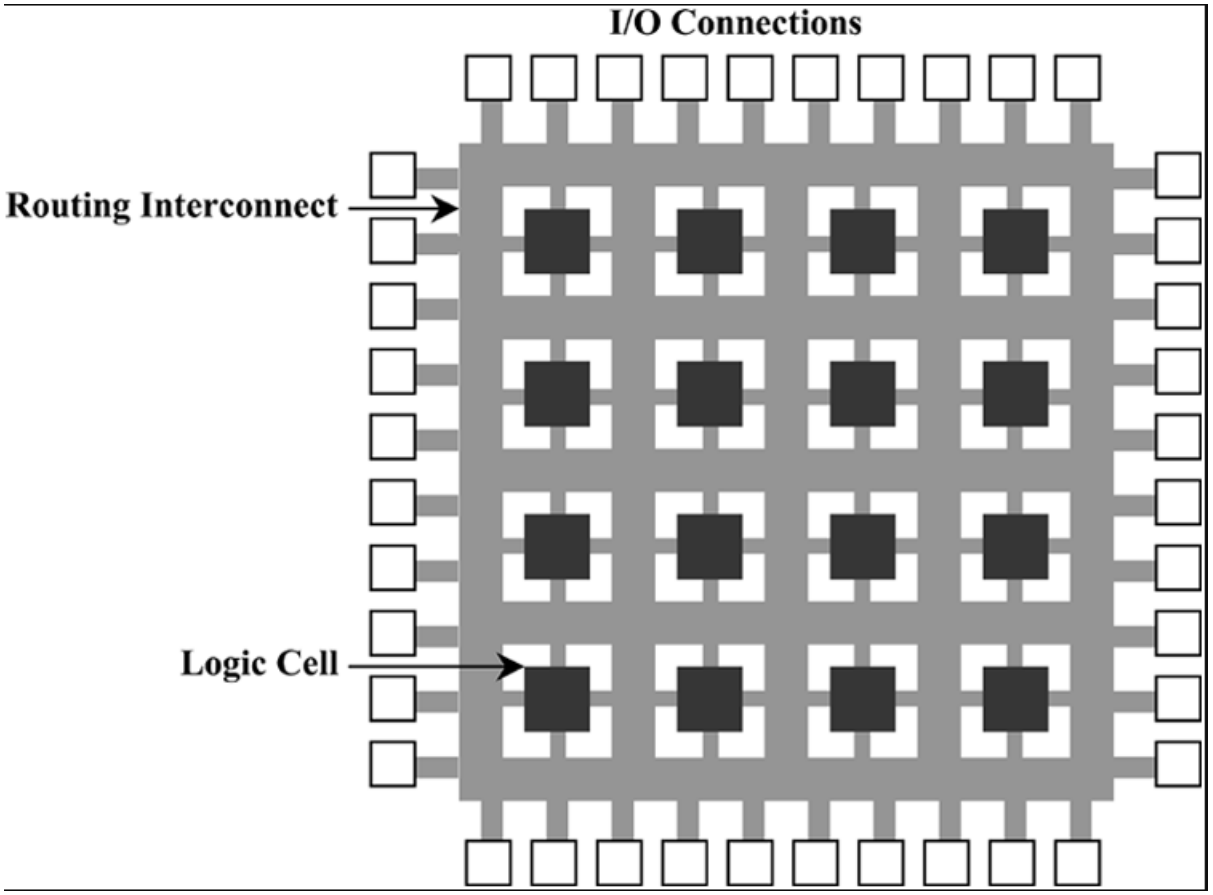

图 6 Typical FPGA structure

9 Conclusion

在本文中，成功展示了一種在可重構平台上的有效圖像壓縮方法。通過分析算法各個部分處理的數據範圍，作者發現創建特定的記憶體結構（比如變數定點處理）是非常有益的。這種方法不僅可以最大程度地減少記憶體的使用，還能實現高效的數據傳輸。（換句話說，從記憶體到處理器板之間傳輸的每一位數據都會直接影響最終結果）。此外，固定順序 SPIHT 研究顯示，通過對現有算法進行細微調整，可以顯著提升定制硬件實現的性能，同時幾乎獲得與原始算法相同的結果。使用固定順序 SPIHT，系統的吞吐量大約提高了一個數量級，同時仍與原始算法的 PSNR 曲線相符。

表 1 Clock cycles per pixel for 512x512 images

Phase	Clock cycles / 4 pixels	Clock rate	Throughput	FPGA area
Wavelet	138,615	75 MHz	100 MPixels/sec	62%
Magnitude	92,788	73 MHz	146 MPixels/sec	34%
SPIHT	46,339	56 MHz	224 MPixels/sec	98%

参考文献

1. D. G. Lowe, "Object recognition from local scale-invariant features," Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 1999, pp. 1150-1157 vol.2, doi: 10.1109/ICCV.1999.790410.

2. Introduction to SIFT(Scale Invariant Feature Transform)
https://medium.com/data-breach/introduction-to-sift-scale-invariant-feature-transform
-65d7f3a72d40

3. T. W. Fry and S. A. Hauck, "SPIHT image compression on FPGAs," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 15, no. 9, pp. 1138-1147, Sept. 2005, doi: 10.1109/TCSVT.2005.852625.

4. A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 6, no. 3, pp. 243-250, June 1996, doi: 10.1109/76.499834.