適用於低軌道衛星通訊的 LDPC Codes 平行處理解碼器實作

Implementations of LDPC Parallel Decoder for LEO Satellite Communication

指導教授：李昌明　　專題生：張昕茹、王定嵐

摘要

現今的許多通訊標準都使用低密度奇偶檢查碼 (Low-Density Parity-Check codes, LDPC codes) 作為通道編碼方案。LDPC codes 不僅具有卓越的錯誤更正能力，結構也非常適合進行平行化處理。本專題研究了一種基於太空資訊系統諮詢委員會 (Consultative Committee for Space Data Systems, CCSDS) 標準的 LDPC codes。透過研究不同的編碼方式，找到最適合此標準的編碼器方案。使用正規化最小和演算法 (Normalized Min-Sum Algorithm, NMSA)，這種改良過的解碼算法，能在降低計算複雜度的同時保證解碼的精確度。另外，有採用 Efficient chunk 的儲存結構，用更少的記憶體儲存解碼運算中節點間傳遞的外部訊息 (Extrinsic information)。設計了 Simplifier 以改進解碼架構層次，提高部分平行化解碼器的平行度 (Parallelism)，使其能進行高速解碼運算。然而，在將解碼架構作平行化處理的時候會遇到記憶體寫入衝突的問題，以致降低吞吐量 (Throughput)，因此透過預先排程 (Scheduling) 的方式，將解碼過程中會造成衝突的數據錯開計算，避免同時寫入。最終完成 CCSDS 標準的編碼器及部分平行化解碼器，其中解碼器在時脈頻率為 55.56 MHz 的狀態下，吞吐量為 64.8 (Mb/s)。

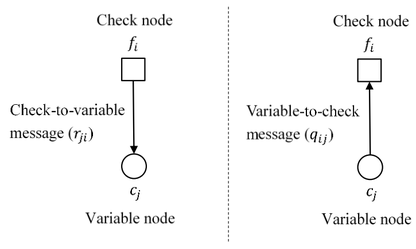
LDPC codes 編碼系統

本專題依據 CCSDS 標準 [1]，利用檢查矩陣架構及循環矩陣的特性，構建完整的檢查矩陣 **H**。接著，將生成矩陣 **G**和匯入訊息相乘，獲得合法碼字 **c**。最後，檢查 $\mathbf{c} \times \mathbf{H}^T = \mathbf{0}$ 是否成立，若成立就表示碼字是無錯誤的。

我們採通道訊號設置為 {0, 1} 的設定，傳送的隨機訊息為等機率，並使用可加性高斯白雜訊 (Additive White Gaussian Noise, AWGN) 的通道模型。分析添加雜訊碼字的 SNR-BER 模擬曲線。

LDPC codes 解碼算法

在 LDPC 碼的解碼過程中，訊息傳遞演算法 (MPA) 透過遞迴方式進行解碼。此演算法描述了訊息的傳送和更新的過程，表示接收到的碼字中每個位元為 1 或 0 的機率。



圖一、兩種訊息更新方式

對於硬體實現而言，MPA 的複雜度較高，因此可以採用簡化過後的正規化最小和演算法 (NMSA) [2]，透過添加正規化因子 α 降低複雜度。解碼的步驟如下：

Step 1：計算事後機率 (A Posteriori Probability)。

Step 2：更新 Check-to-variable message。

$$L(r_{ji}) = \prod_{i' \in V_j \setminus i} \alpha_{i'j} \times \alpha \times \min_{i' \in V_j \setminus i} \beta_{i'j}$$

其中　$\alpha_{ij} = sign[L(q_{ij})],\ \beta_{ij} = |L(q_{ij})|$

Step 3：更新 Variable-to-check message。

$$L(q_{ij}) = L(v_i) + \sum_{j' \in C_i \setminus j} L(r_{j'i})$$

Step 4：完成一次迭代後，判斷碼字是 0 還是 1。

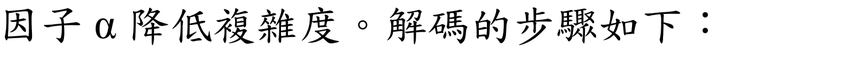
$$L(Q_i) = L(v_i) + \sum_{j \in C_i} L(r_{ji})$$

$$c_i = \begin{cases} 1 & if\ L(Q_i) < 0 \\ 0 & else \end{cases}$$

Step 5：判斷是否符合終止條件 $\mathbf{c} \times \mathbf{H}^T = \mathbf{0}$。

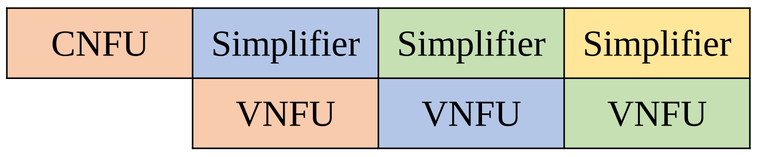
解碼器相關配置

在傳統的解碼器設計中，如圖二，需要等待 Variable-to-check messages 被更新才能計算新的 Check-to-variable messages，因此解碼運算效能較差。



圖二、傳統的解碼排程

本專題採用了 Simplifier 的設計改善解碼器的架構，這使得在計算 Variable-to-check messages 的同時，得以更新 Check-to-variable messages，從而實現高速解碼運算，如圖三所示。

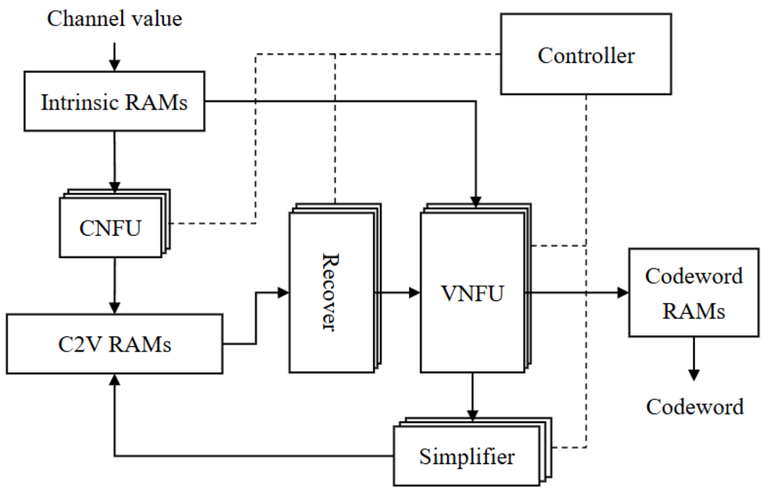


圖三、引入 Simplifier 設計的解碼排程

解碼器硬體架構

在解碼器中加入平行處理設計，使用多個硬體單元同時運算，能獲得更好的吞吐量表現，硬體解碼流程如下：

(1) 將通道輸入的 Channel value 寫入 Intrinsic RAMs。
(2) Check Node Function Unit (CNFU) 計算 Check-to-variable messages，並將結果存入 C2V RAMs。
(3) 開始循環迭代解碼。Recover 將 Check-to-variable messages 送入 Variable Node Function Unit (VNFU) 計算。VNFU 計算 Variable-to-check messages 的同時，Simplifier 也在更新 Check-to-variable messages，並將結果存回 C2V RAMs，供下一次迭代運算使用。
(4) 若是達到預定的迭代次數，則解碼器完成解碼並輸出碼字。



圖四、硬體解碼器架構

成果與結論

使用 Xilinx Artix-7 FPGA，設備型號 xc7a200tsbg484-1 進行實作，硬體資源使用量如表一所示。在迭代解碼次數為 10 次，且 SNR 為 4 dB 以上時能有很好的解碼成功率。本解碼器可以在時脈頻率為 55.56 MHz (P_{clk} = 18 ns) 的狀態下運行，吞吐量為 64.8 (Mb/s) 。

表一、硬體資源使用量

Resource	Utilization	Available	Utilization (%)
LUTs	17278	133800	12.91
FFs	5498	267600	2.05
36k BRAMs	8	365	2.19

[1] Recommendation for Space Data System Standards, TM Synchronization and Channel Coding, Recommend Standard, CCSDS 131.0-B-4, BLUE BOOK, April. 2022.
[2] Jinghu Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier and Xiao-Yu Hu, ”Reduced-complexity decoding of LDPC codes,” IEEE Transactions on Communications, vol. 53, no. 8, pp. 1288-1299, Aug. 2005.