針對 SEC LUT (查找表) 實現方式優化之分析報告

一、LUT規格分析

* 輸入：7-bit signed (-64 ~ +63)
* 輸出：15-bit unsigned (r)
* 有效資料量：±1 ~ ±43，共86筆
* 總容量：128 entries × 15 bits = 1920 bits ≈ 2kb

二、ASIC Memory Compiler (ROM)分析

(一) 優點

* 面積效率較高，使用高密度bit-cell。
* 低漏電流、佈局規整。
* 讀取速度快且穩定（通常1 cycle）。

(二) 缺點

* 條目數少時，可能面積效益不明顯。
* 資料更新需重新tape-out，缺乏靈活性。

(三) 結論

* 資料未來若大幅擴充 (>256筆)：建議使用Memory Compiler ROM。
* 若維持現狀(86筆)：現有gate-level combinational LUT更省面積。

三、Vivado FPGA BRAM分析

(一) 優點

* 自動推斷，易於使用與修改。
* Routing簡單，固定1 clock cycle延遲。

(二) 缺點

* 最小BRAM單位為18kb，LUT僅約2kb，資源浪費嚴重。

(三) 結論

* 資料若未來擴充達8kb以上：建議使用FPGA的BRAM。
* 目前資料僅2kb：建議使用distributed ROM (LUT資源)，不使用BRAM。

四、RTL建議寫法

使用ROM推斷風格的Verilog RTL：

module SEC\_lLUT28bits(l, r);

input signed [6:0] l;

output reg [14:0] r;

wire [6:0] addr = l + 7'd64; // 轉換地址範圍

reg [14:0] LUT\_ROM [0:127];

initial begin

LUT\_ROM[64 + 1] = 1;

LUT\_ROM[64 - 1] = 17618;

LUT\_ROM[64 + 2] = 2;

LUT\_ROM[64 - 2] = 17617;

// (其他資料以此類推填入)

LUT\_ROM[64 + 43] = 12034;

LUT\_ROM[64 - 43] = 5585;

end

always @(\*) begin

r = LUT\_ROM[addr];

end

endmodule

五、整體建議

| **平台** | **建議做法** | **原因說明** |
| --- | --- | --- |
| ASIC (小規模) | 使用combinational gate-level LUT | gate-level更有效率 |
| ASIC (中~大規模) | 使用Memory Compiler ROM | 面積與power最佳化 |
| FPGA (小規模2kb) | 使用distributed ROM (LUTROM) | 避免BRAM資源浪費 |
| FPGA (中~大規模) | 使用BRAM | Routing效率較高、易擴充 |

52bits:

**重點整理：Combinational LUT 與 Memory-based (ROM/BRAM) 比較**

**📌 1. 你的設計現況**

* **筆數**：136筆
* **資料寬度**：16 bits
* 資料規模介於『使用純組合邏輯』與『Memory-based』方案的交界處。

**📌 2. Combinational Logic 實現問題**

* **面積偏大**：隨著筆數增加，MUX數量快速增加。
* **延遲較高**：多層MUX會導致延遲增加且不穩定。

使用case語句（combinational實現）將會產生很多層MUX，  
這將帶來相當大的延遲與時序上的不確定性。  
在ASIC後端或FPGA synthesis階段，很可能需要額外的優化，  
導致設計與驗證變得較為複雜。

* **功耗較高**：組合邏輯會頻繁翻轉(glitch)，造成較高動態功耗。
* **擴充困難**：未來增加筆數，需要重新合成整個邏輯，設計成本較高。

**📌 3. Memory-based (ROM/BRAM) 優勢**

* **面積更小**：適合較大的資料寬度及筆數。
* **延遲穩定且低**：透過記憶體讀取可保證延遲穩定性。
* **功耗低**：無多餘邏輯翻轉，功耗顯著降低。
* **易擴充**：新增資料只須重新設定記憶體內容即可。

**📌 4. 實務經驗參考**

| **條件** | **一般ASIC設計標準** | **FPGA設計(Vivado)標準** |
| --- | --- | --- |
| 資料筆數推薦 | ≥ 100筆左右即考慮ROM | ≥ 128筆即推薦BRAM |
| 資料寬度推薦 | ≥ 8 bits以上 | ≥ 8 bits以上 |
| 你的情況 | **136筆×16 bits (推薦使用ROM/BRAM)** | **136筆×16 bits (極度推薦使用BRAM)** |

**📌 5. 結論與建議**

 如果未來條目數不會大幅增加且資源與時序不嚴格：  
**可以暫時維持使用純 combinational 實現，快速完成設計與驗證。**

 如果你想有更好的擴充性、更佳的延遲與功耗表現：  
**使用 memory-based 方式（ROM 或 BRAM）會是最佳方案。**