積體電路設計

功課三 Sorter for 5 6-bit unsigned intergers

B05901105 陳矩翰

- 設計思路

  利用 gravity sort，亦稱 bead sort，完成 longest path (hopefully)較短的 sorter

- 模組

  主要需要做的事情如下

  a. 將 6-bit 輸入轉換成 63-bit、可以用於 gravity sort 的解碼器
  b. Gravity sort 本體
  c. 將 gravity sort 的輸出轉回 6-bit 的無號二進制表示

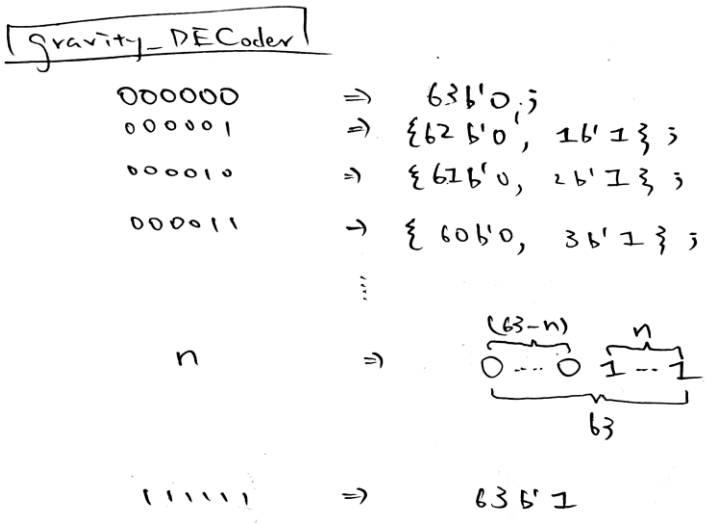
  為此，主要的模組如下

  - gravity_DECoder

    有些類似 6-bit-to-64-bit 解碼器，但是假設某個 bit 為 1 的話，比其 less significant 的 bits 都需要為 1，也就是解碼出來的會是 more significant 的半側為 0、less significant 的半側為 1 的 vector。

    由於 6-bit unsigned 表示範圍為 0~63，解碼出來的結果僅需要 63-bit、而不需要 64-bit。

    每個 bit 都是由該條件的真值表計算出來，在此不再贅述每條訊號線的 gate-level 實現；我所使用的工具網站是這個。假設需要計算在[63:0]其中位址為 n 的 wire 之 SOP 式子，那麼只需要把真值表的 0~n 地方填 0，(n+1)~63 地方填 1，那麼計算出來的 SOP 便是所希望的輸出。



  - gravity_SORT

    一個 non-comparison-based 的 sorter。

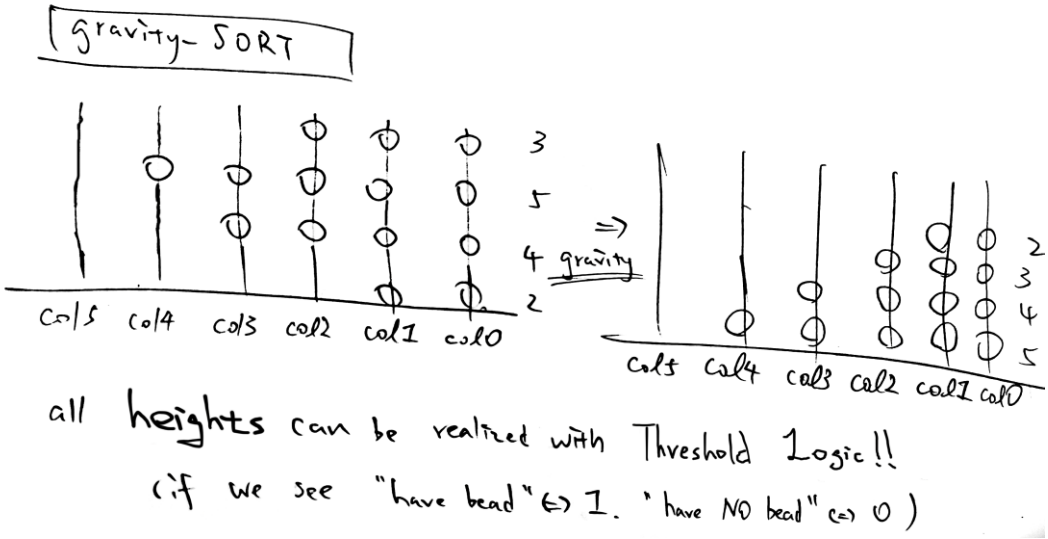
    其原理如下圖：若我們有數個未知數量的珠子集合，那麼如果我們將依序擺在一排柱子的不同高度上，例如第 1 組珠子排在離地 1 m 處，第 2 組珠子排在離地 2 m 處，第 n 組珠子排在離地 n m 處，並且珠子間不能有空著柱子沒排、且需要從第一根柱子開始排，那麼在排好之後，隨著重力作用、珠子掉落，我們在不同高度上自然會得到不同數量的珠子，並且這個數量會是排序完成的。

    以電路實現而言，可以將擺放珠子視為 1、不擺放珠子視為 0，那麼珠子隨重力下落這個行為便可以用 Threshold Logic 來 model，也就是 threshold 為 1 的 threshold logic 可以觀察最低高度的珠子分佈若要觀察掉到次低高度的珠子，可以用 threshold 為 2 的

threshold gate；要觀察掉到第 n 低高度的珠子分佈，用 threshold 為 n 的 threshold logic。已知輸入是 5 個 unsigned interger，也就是重力落下完成後，最多只會有 0~5 的不同高度，因此準備 threshold m out of 5 的 threshold logic 即可。

更具體的來說，input 經過 gravity_DECoder，得到的結果可以視為一組已經排在柱子上的珠子，這也是為什麼我們要求 gravity_DECoder 的輸出會是 less-significant half 為 1、more-significant half 為 0 的形式；input 範圍是 0~63，因此需要準備 63+1 根「柱子」，對於其中 less-significant 的 63 根柱子，每根柱子需要有 5 種 threshold logic；第 64 根柱子對應到 MSB 的輸出，也就是 input 為 64 的情況，然而此情況不會發生，是故固定填 1'b0 即可。柱子上 threshold logic 的求值對應到珠子的落下。

參見 THRES_1_5、THRES_2_5、THRES_3_5、THRES_4_5、THRES_5_5 等 helper module。



- gravity_ENCoder

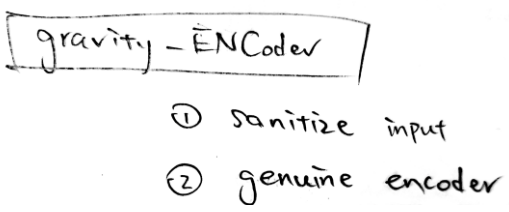
  需要將 gravity sort 的特殊編碼轉換回 6-bit unsigned 表示。

  Gravity sort 的編碼中，必定是 more-significant half 為 0、less-significant half 為 1 的表示方式，因此如果要考慮某根位處[63:0]其中第 n 條 wire 的輸出的話，如果有任何比其 significant 的 wire 為 1，那麼其輸出要是 0。

  亦即，如果該 wire 輸出要為 1，僅有可能是在輸入的值[63:0]中，[63:(n+1)]都是 0、[n:0]都是 1 的情況。

  為此，我們可以利用 helper module “foobar”以及 OR-gate，先把 input 轉換成常見的 one-hot encoding，在此姑且稱作 sanitizing，而後再利用一般的編碼器形式將其轉回 6-bit unsigned 表示式。

  參見 helper model “foobar”以及”Sanitizer”。



以及其他的輔助模組

- foobar

  仔細觀察 gravity_ENCoder 的需求，可以發現針對[63:0]中的第 n 條 wire，如果

[63:(n+1)]這些 wire 中的某第 m 條是 1 的話，那麼該第 n 條 wire 要變成 0。
寫成真值表如下：

Input[m] with m > n (A)	Input[n] (B)	Output[n] (o)
0	0	0
0	1	1
1	0	0
1	1	0

foobar 實現的便是此真值表，腳位如括號中所示。

- Sanitizer
  利用 foobar 輔助，將 gravity sort 的 encoding 轉成一般的 one-hot encoding。
  由於是「任何」比第 n 條 wire 還要 significant 的 wires 為 1 都會導致第 n 條 wire 要為 0，對於每個大於 n 的 m，並且為了加速運算，我對於每個大於 n 的 m 都做了一個對應的 foobar gate，然後再把他們 AND 起來。
- THRES_3_5
  首先寫下(5, 3)=10 的、5 中取 3 共 10 種不同 minterm，然後將其 AND 起來。
  其他數量的 threshold logic 同理，不再贅述。

- 實驗心得
  - 在討論到 Gravity Sort 的時候，常常會提到的是電路難以實現的問題，這次作業算是了解了他的難題：
    - 為了對應 n-bit 的輸入，需要做 2^n 這麼多個「柱子」，也就是光此部份的走線數量就是隨 input 數量呈指數成長，而在本實作中為了加速運算，”foobar”這個 module 的使用數量甚至是呈 Θ(2^(2n))成長的， 可以猜想其成本極高並極其耗電。
    - threshold logic 不管是用 SOP 還是 POS 表示式都偏長，並且需要的數量也是隨 input 個數呈指數成長，單用 gate-level design 的效果應該很差，需要作到 transistor-level 的設計才行。
  - Gravity Sort 的理論複雜度雖然很漂亮，電路跑起來不一定特別快
    - 單單使用 gate-level design 的話，最後編碼的候，longest path 高達 Θ(n)級邏輯閘；假設一般 comparison based sorter 需要做 Θ(log(n))次比較、每次比較是 Θ(log(n))級邏輯閘，容易看出後者很可能更快。
    - 然而，如果使用 transistor-level design，編碼的 longest path 可以降到常數深度：foobar 模組其實可以改成使用 pseudo-NMOS 來實現，如此將可以大幅提昇速度。
  - Verilog 的 variable name 不能用數字當開頭
    - 忘記的話可能會額外花上高達 2 小時在 debug
  - 電機系館很多，但是凡是開給公用的實驗室電腦都很爛，有錢蓋房沒錢買電腦