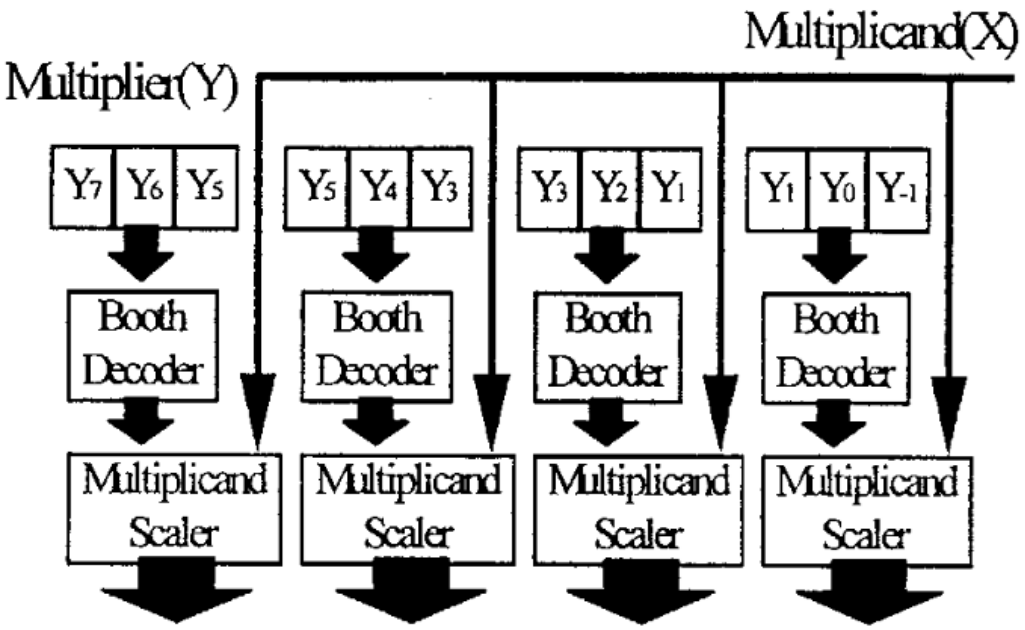
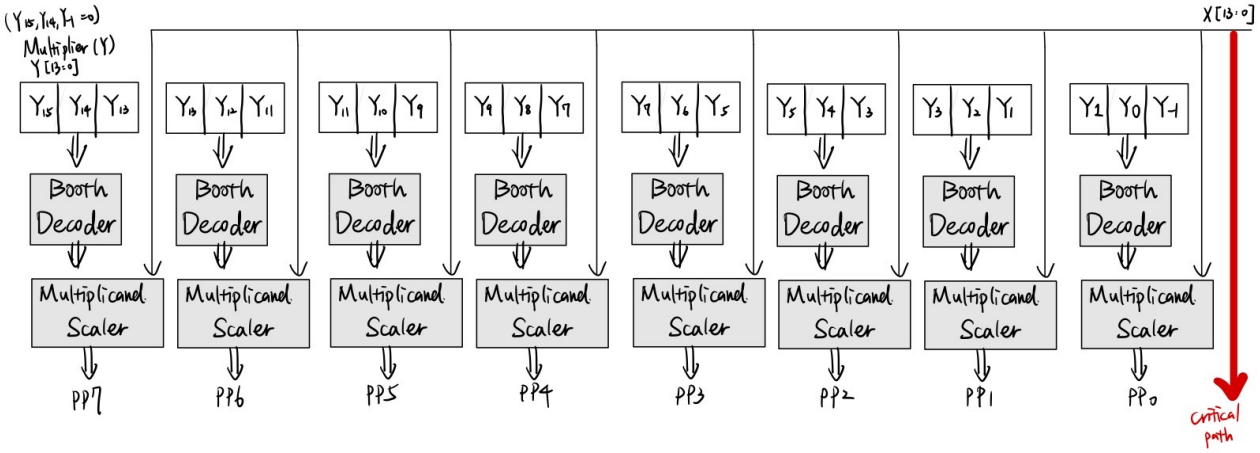
Design a 14-bit unsigned multiplier P=X×Y by using modified Radix-4 Booth multipliers. **The goal is to have minimum critical path delay time with the least gate count.**

(1) Show your block diagram as shown in Fig.1 below (example for the case of 8-bit; Booth decoder and Multiplicand scale corresponding to Booth Encoder and Booth selector of Fig.10.80). For each block, you shall show its logic design diagram. Indicate the critical path. Explain your design concepts (30).





為了解決先前一次看 Y 兩個 bit 來對 X 做乘法產生 3X 的問題。(當 Y=2'b11，此時產生的 3X 需要使用 CPA 來計算 X+2X，會較慢)。這邊使用 radix-4 booth multiplier。透過一次看三個 bit，且每三個 bit 中間會重疊一個 bit 的技巧，來巧妙的避開了 3X 的運算，改以 4X-X 的概念繞過需要使用 CPA 的 3X。將掃描的 3 個 bit 以課本的表格來命名，分別是: $Y_{2i+1}, Y_{2i}, Y_{2i-1}$ (i=0~7)。其中邊界 Y_{-1}, Y_{14}, Y_{15} 皆補 0。而這三個 bit 會先經過 booth encoder 產生 SINGLE, DOUBLE, NEG 的訊號，再經由 booth selector 選擇輸出的 Partial Product 究竟會是-2X, -X, 0 X, 2X。在這題中的 critical path 因為是為同時的平行運算，因此 critical path 為 Y 經過 Booth Decode 再經過 Multiplicand Scalar 的 delay(上圖紅色標註)。

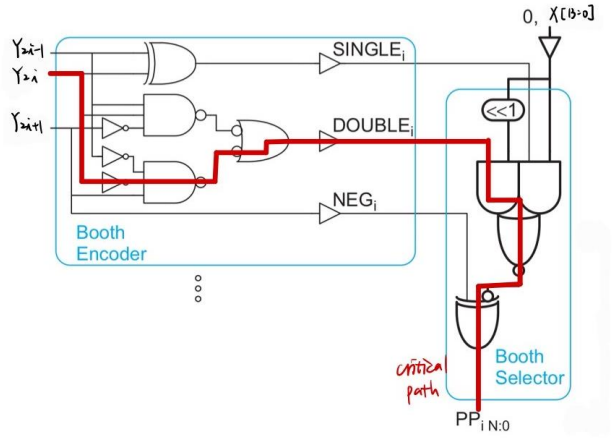
而這邊快速介紹 booth decoder(encoder)及 multiplicand scalar(booth selector)的設計邏輯。先來看看 truth table:

Y_{2i+1}	Y_{2i}	Y_{2i-1}	PP_i	$Single_i$	$Double_i$	Neg_i
0	0	0	0	0	0	0
0	0	1	X	1	0	0
0	1	0	X	1	0	0
0	1	1	2X	0	1	0
1	0	0	-2X	0	1	1
1	0	1	-X	1	0	1
1	1	0	-X	1	0	1
1	1	1	0	0	0	1

由 truth table 或是簡單的數學推導可以發現: $PP_i = -2 \cdot Y_{2i+1} + Y_{2i} + Y_{2i-1}$
而在這之中，由簡單的布林邏輯推導可以得到:

$$\begin{cases} Single_i = Y_{2i} \oplus Y_{2i-1} \\ Double_i = \overline{Y}_{2i+1} \cdot Y_{2i} \cdot Y_{2i-1} + Y_{2i+1} \cdot \overline{Y}_{2i} \cdot \overline{Y}_{2i-1} \\ Neg_i = Y_{2i+1} \end{cases}$$

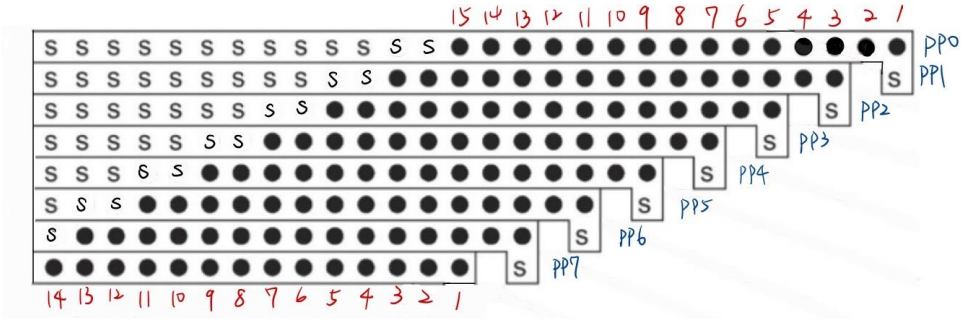
因此可以使用簡單的組合邏輯電路來實現 Booth encoder 及 booth selector:



可以先使用 $Y_{2i+1}, Y_{2i}, Y_{2i-1}$ 的組合算出幾倍及正負的 flag，而後在 booth selector 使用左移 1bit 來達成 2 倍，使用固定一個 pin 為 1 的 XOR2 gate 來實現反向器，使得輸出會事 1's complement，若要使用 2's complement 可以在最後使用 CSA 將各個 partial product 加起來的時候加在下一個 row。而在這之中 booth encoder 及 booth selector 的 critical path 為經過 double 而後再經過 booth selector 而後輸出 partial product 這條路。藉由上圖的這些架構便可以成功設計 14-bit unsigned 的乘法器。透過切分成 8 個 group 平行運算，最終只有 8 個 partial product 需要相加，相比最基礎的乘法器的 14 個 partial product 的相加會有效率一些。

(2) Shown the Radix Booth-4 encoded partial products with simplified sign extension like that shown in Fig.10.82 of B5_Array and Recoded Multuplier. Draw another diagram with the dots or S that replace p_{ij} where i is the ith row and j is the bit location . (20)

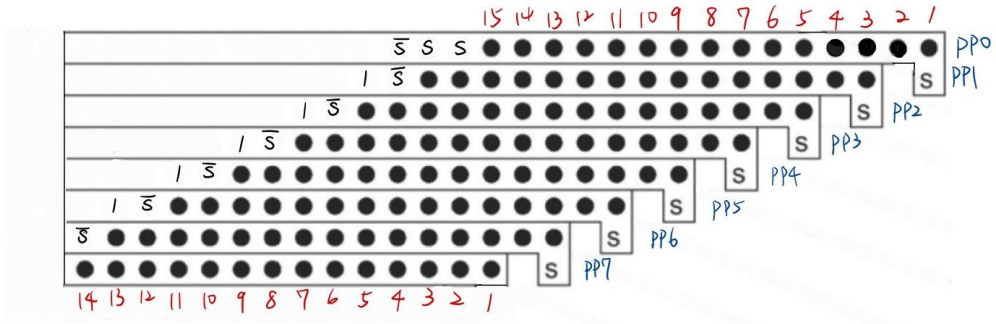
因為在使用 booth encoding 時為了解決 3X 變成 4X-X 來解而產生的負號所需要以 2's complement sign extension 的方法來表示負數。在這些 partial product 中因為最後要相加對其而導致 sign bit 會有相當大的 fan out cost，因此課本介紹了一個蠻厲害的方法來畫簡 sign bit 過多的方法，我認為蠻厲害的。



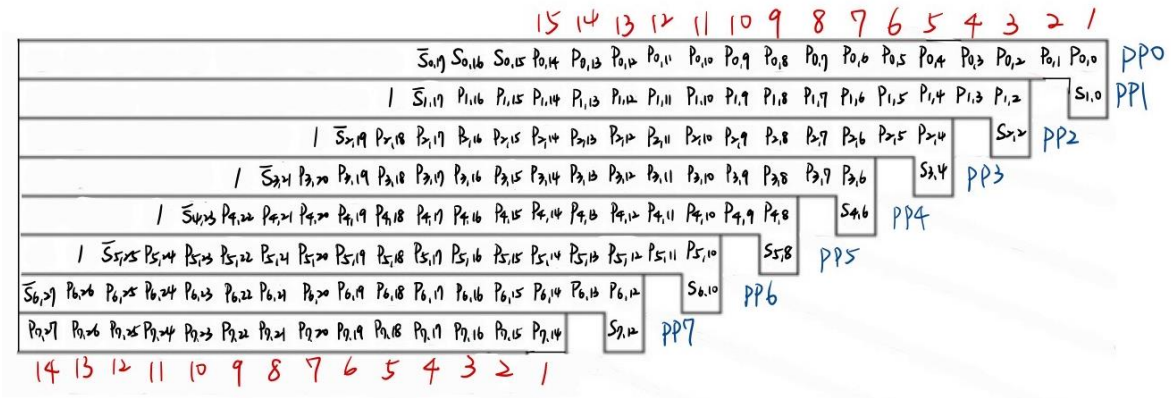
首先，因為前面所介紹的硬體在實作 Negative 時為了偷時間，將 1's complement + 1 = 2's complement 的加法與下一層的 partial product 一起加。因此便可以省下一級 CPA 的時間。此外，由於有可能會有 Double 的可能性，因此每個 partial product 將會是以 15bit 來表示。納因為 Partial Product 7(PP7)因為位處邊界條件(00?)，不可能會是兩倍的可能性，因此這邊可以單純以 14 個 bit 來表示 partial product。而究竟是如何將那麼多的 S 減少的呢?可以參考課本這一段話:"The large number of s bits in each partial product can be replaced by an equal number of constant 1s plus the inverse of s added to the least significant position." 概念化簡如下:

$$\begin{cases} Neg_i = 1, S = 1, \bar{S} = 0, sign\_ext = 1 \\ Neg_i = 0, S = 0, \bar{S} = 1, sign\_ext = 0 \end{cases}$$

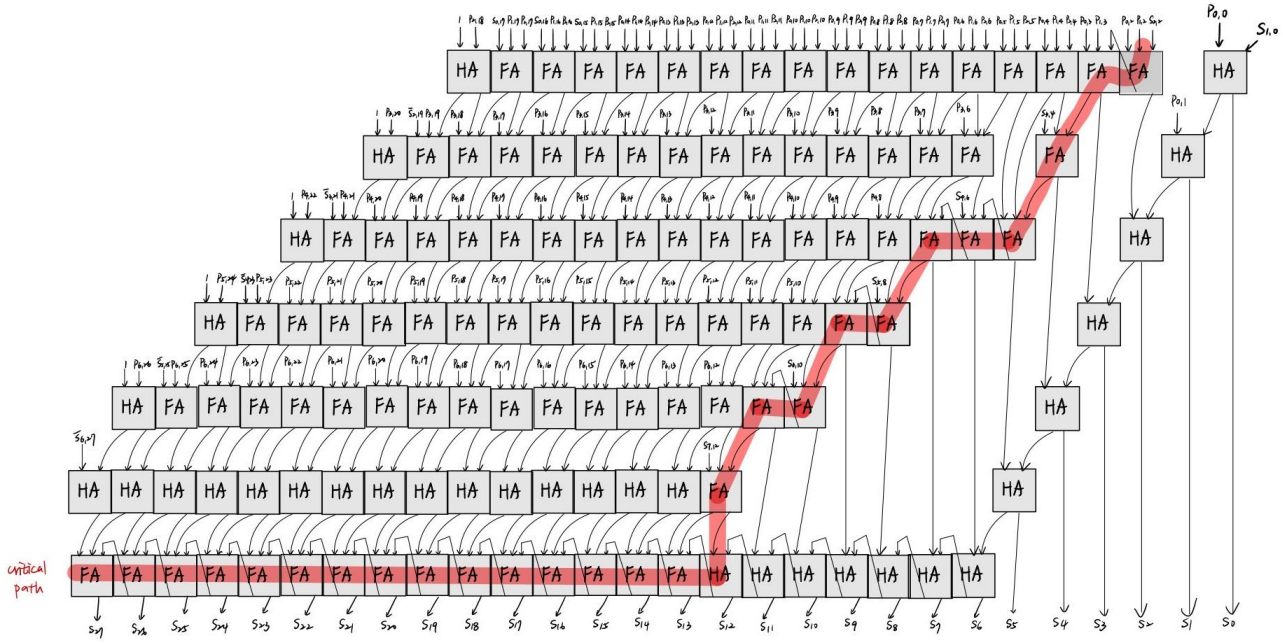
因此把每層整理一下化簡便可以得到:



接下來使用 pij 來表示前面的點點，而 sign extension 則仍然維持使用 S 來表示:



(3) Design the partial products with array multiplier as that shown in Fig.5.2. Draw the block diagram in terms of FA and HA and p_{ij} as inputs. Show the critical path and indicate the delay time in terms of number of HA and FA. For area, show the number of FA and HA used. Explain your design concepts. (25)



將前一題所算出來的所有 partial product 以 full adder 或是 half adder 將每個需要相加的 bit 加起來。在計算中，若是遇到需要做 carry propagate 的運算，便會盡可能的往下一層傳，使其他部分的加法器可以盡可能的平行運算。如此一來才可以使得整體乘法器在最後加法的部分比較快。

Delay 估計:
在 critical path 上共會需要經過 26 個 FA 的 delay。

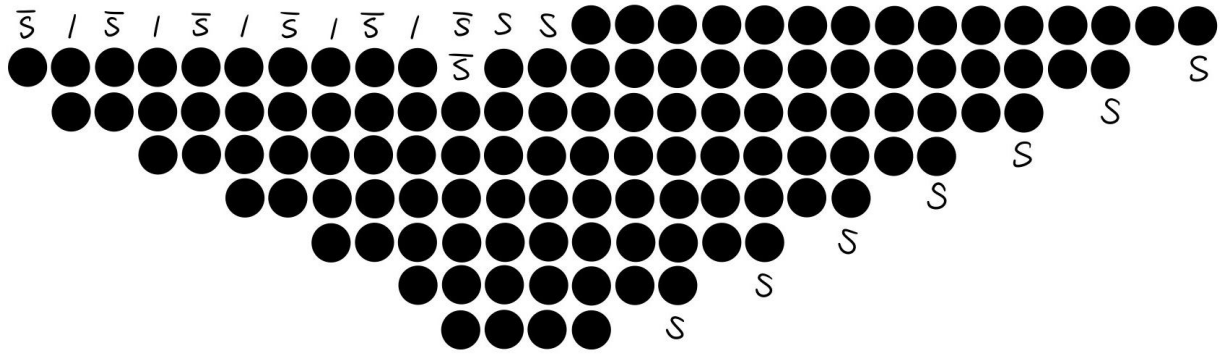
面積估計:

	HA	FA
Numbers	19	110

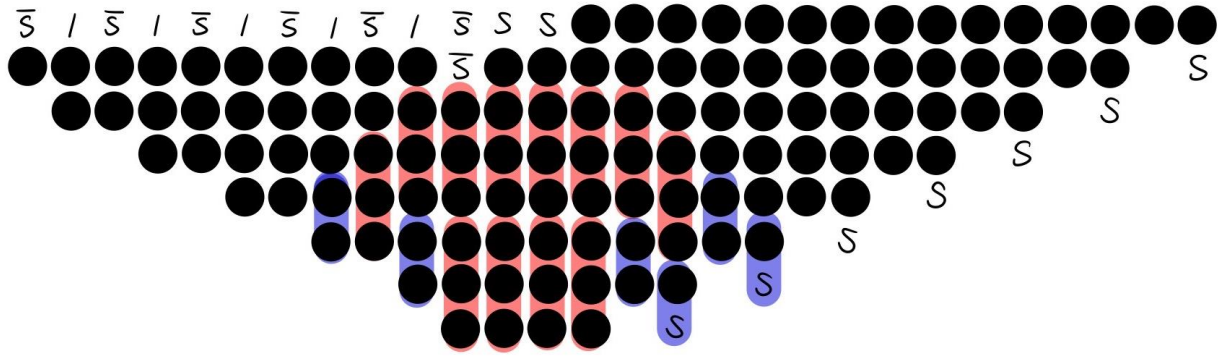
(4) Design the partial products with Dadda method shown in Fig.5.19. To fairly compare with array multiplier, Carry-ripple adder is used in the last stage of CPA. Show the critical path and indicate the delay time in terms of number of HA and FA. For area, show the number of FA and HA used. Explain your design concepts (25)

Dadda tree rule: $d_{i+1} = floor\left(\frac{3}{2} \cdot d_i\right), d_1 = 2$, d = [2, 3, 4, 6, 9, …]

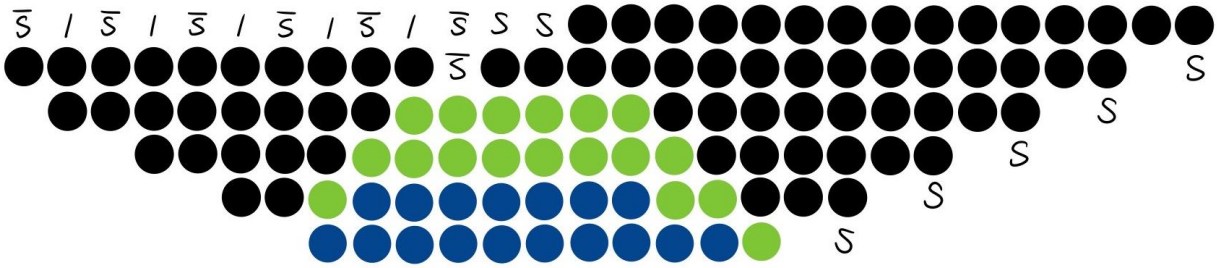
首先，先將前面做出來的 dot diagram 放在一個重力加速度方向為上的空間(因為在哪層加該 bit 不重要，將其往上方靠攏)。形成下圖:



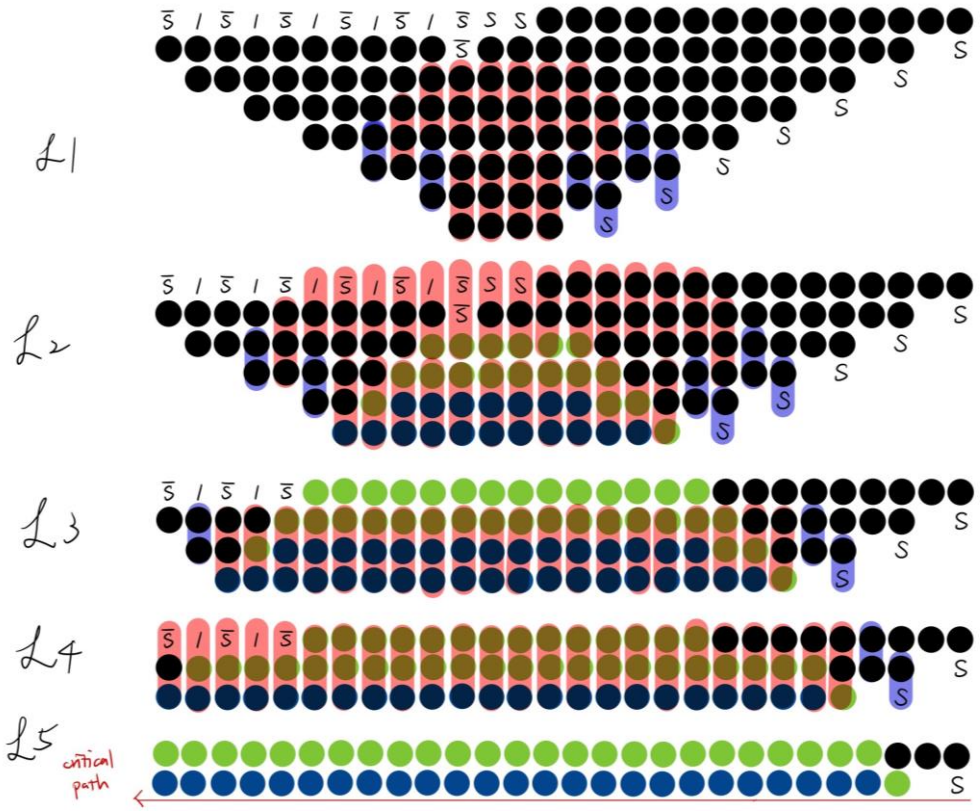
而後從 LSB 往 MSB 看，若是在考慮前一級 column 加法會有進位的可能的情況下，並且該 row 超過 6 個 bit 時使用 HA 或是 FA 將其加起來。(以紅色螢光代表 FA，以藍色螢光代表 HA)。形成下圖:



接著，將每個 column 若是有使用到 HA/FA，將圈起來的黑點替代為綠色的點點表示為 Sum，並且在下一級的 column 產生一個藍色的點點表是為 Carry。



而後重複上面的步驟，只不過這次變成當超過 4 個 bit 就要使用 FA/HA 來運算，而後再以綠色點點及藍色點點代表 Sum 及 Carry。直到每個 column 只剩兩個 bit 就得使用 CPA 來將所有 bit 加起來，輸出最終 Partial Product 的結果了。下一頁為整個流程，分別是考慮 6, 4, 3, 2bit 的情況。



	HA	FA
L1	6	12
L2	6	28
L3	3	20
L4	2	24
L5	2	26
Total	19	110

Critical Path:
在這邊 L5 中，除了兩個 column 只需要 HA 外，因為每一級需要接收前一級的 carry，因此剩餘的 column 都需要使用 FA 來做相加的運算。因此最終 critical path 為經過 26 個 FA 及 2 個 HA。

與 Array Multiplier 的比較: Dadda tree 可以有效的減少所需要使用到的 Adder，但是在這個 case 中，array multiplier 的面積與 critical path delay 皆為較短。但我想這是在助教給定的這個例子中的一個蠻特別的狀況。在大部分的情況下，Dadda tree 可以大幅的減少中間所需要使用到的 FA/HA，使面積有著很好的優化。總而言之，在這次實作 14-bit 的乘法器中，在 array multiplier 與 dadda multiplier 的兩種選擇中，array multiplier 有著較小的面積及延遲，是為較好的設計。但是在其他 M 乘 N 的情況下，則不一定，若是在較大位數的乘法器實作，dadda multiplier 便可以有著較好面積優化空間，此外，若是最後一級的 CPA 可以使用更快速的 CPA 架構，便可以更快速的算出最終的結果，在面積及延遲上可以勝過前面所提到的 array multiplier 的架構。