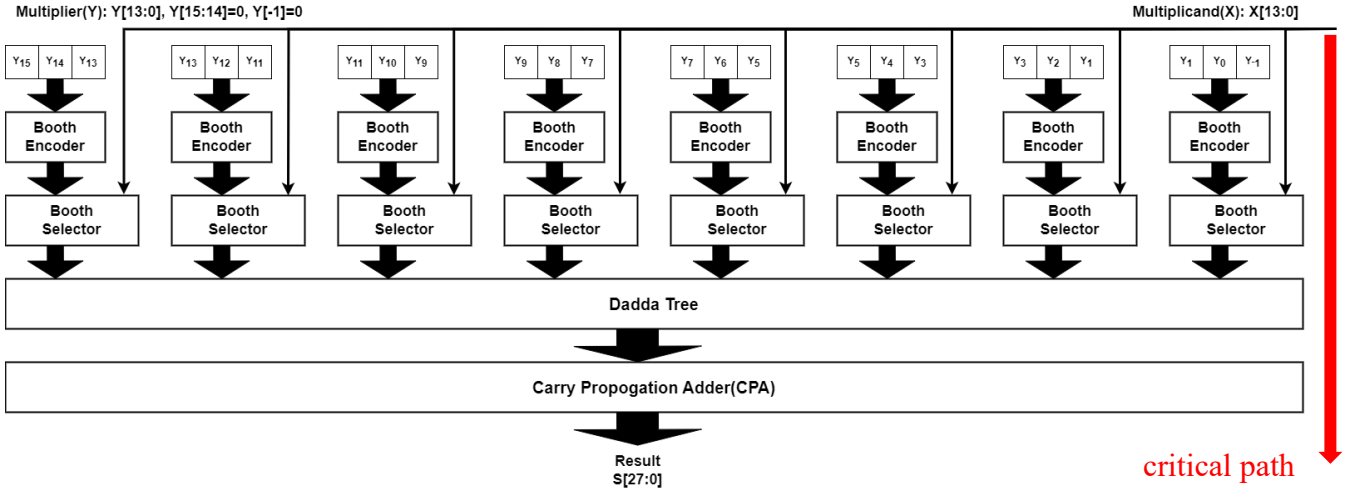
2023 Spring NYCU-EE Digital Integrated Circuits – Homework7

IEE 陳冠瑋 310510221 / May. 15, 2023

Design a 14-bit unsigned multiplier P=X×Y by using 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).

- Logic design diagram & critical path



- **Design concept**
  - **Multiplier**

    Radix-4 Booth multiplier 的操作原理是從第一個 bit (Y_0)開始前後各看一個 bit，一次掃描 3 個 bit Y_{2i+1},Y_{2i}, Y_{2i-1} (i = 0~7)，每三個 bits 中間會 overlap 一個 bit (Y_{2i-1})。其中，這三個 bit 經過 Booth Encoder 會產生一個 3bits 的 value (SINGLEi, DOUBLEi, NEGi) 再送入 Booth Selector，經由三個 bits 的組合，Booth Selector 可以產生-2X, -X, 0, X, 2X 這五種可能的 partial product。而因為是 unsigned multiplier，最後一個 case (Y_{13}, Y_{14}, Y_{15})的 Y_{14}, Y_{15} 要補 0。
  - **Booth Encoder & Booth Selector**

    對應的 truth table 如下:

Y_{2i+1}	Y_{2i}	Y_{2i-1}	Multiplicand Multiples (PPi)	Comments	SINGLEi	DOUBLEi	NEGi
0	0	0	0	No string of 1's	0	0	0
0	0	1	+X	End of previous string	1	0	0
0	1	0	+X	Isolated 1	1	0	0
0	1	1	+2X	End of previous string	0	1	0
1	0	0	-2X	Beginning of string	0	1	1
1	0	1	-X	Beginning and ending of string	1	0	1
1	1	0	-X	Beginning of string	1	0	1
1	1	1	-0(=0)	Center of string	0	0	1

由 truth table 可以得到:

SINGLEi: $\overline{Y_{2i}} \cdot Y_{2i-1} + Y_{2i} \cdot \overline{Y_{2i-1}} = Y_{2i} \oplus Y_{2i-1}$

DOUBLEi: $\overline{Y_{2i+1}} \cdot Y_{2i} \cdot Y_{2i-1} + Y_{2i+1} \cdot \overline{Y_{2i}} \cdot \overline{Y_{2i-1}}$

NEGi = Y_{2i+1}

進入 Booth Selector，將 X 成以對應的倍數

SINGLEi = 1 → 乘 1 倍

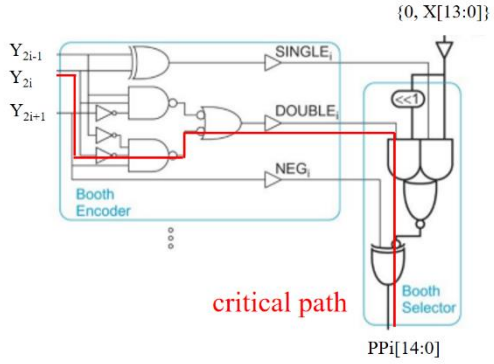
DOUBLEi = 1 → 乘 2 倍

NEGi = 1 → 乘負號

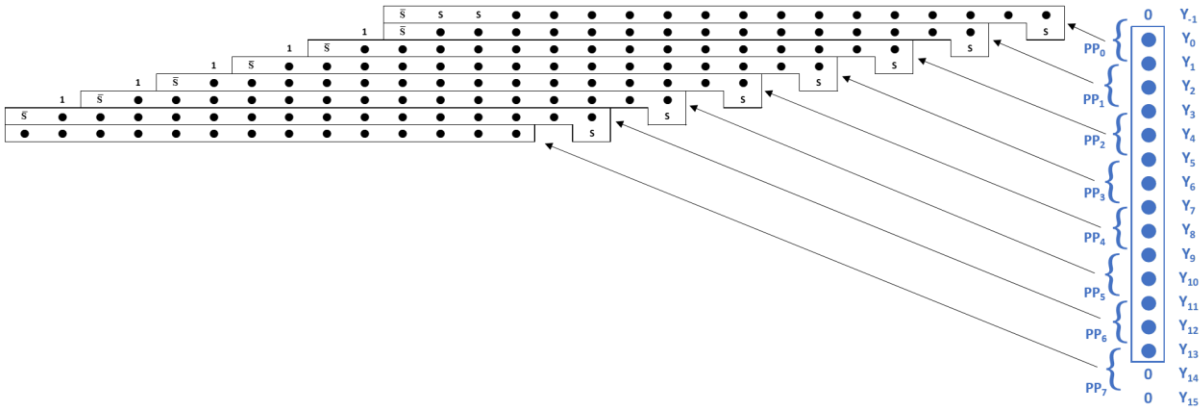
乘完即為那一列的 Partial product。

因為每組 Booth Encoder 和 Booth Selector 是平行處理，因此在計算 critical path 時只需要看一組即可。

下圖為 Both Encoder & Booth Selector 的 logic design diagram 和 critical path:



- **Dadda Tree & CPA**

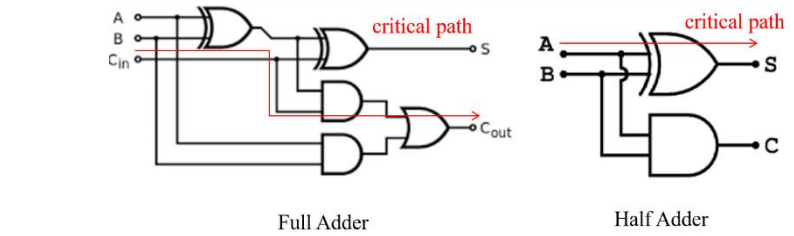


(a) 將 signed extension 化簡可以得到上圖 (推導在第二題)

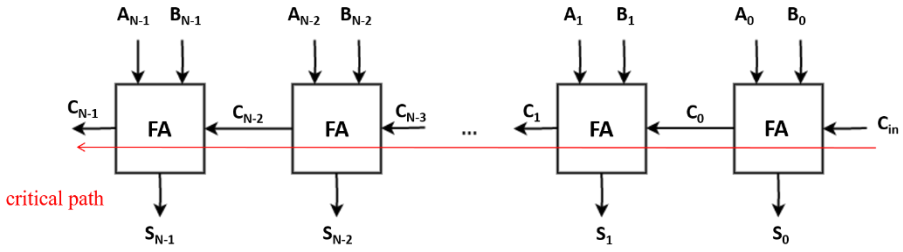
(b) NEGi = 1 時，partial product 為負，PP_0~PP_6 皆有可能為負值，所以需要加上 NEGi。PP_7 因為 Y[15]=0, Y[14]=0，從 truth table 可得知，僅有 0, +X 的情況，所以不需要加上 NEGi。PP_0~PP_6 有 2Y, -2Y 的可能性，所以為 15 bits，而 PP_7 只有可能是 0, +Y，所以為 14 bits。

(c) Dadda Tree 的部分是將上面產生的 8 個 partial product 加起來，Dadda 的方法將每個 level 的 vector 透過 HA 和 FA 組合計算，盡量使用最少的 adder 做完，最後剩下兩個 vector 在經由 CPA(carry ripple adder) 算出答案

其設計如下圖：



Full Adder　　Half Adder

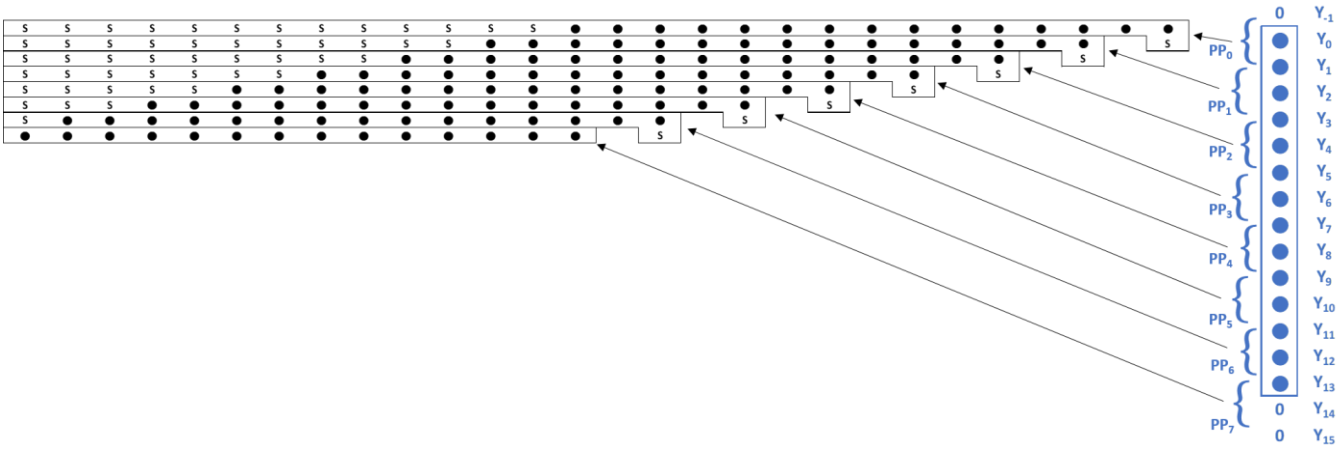


N-bit Ripple Carry Adder

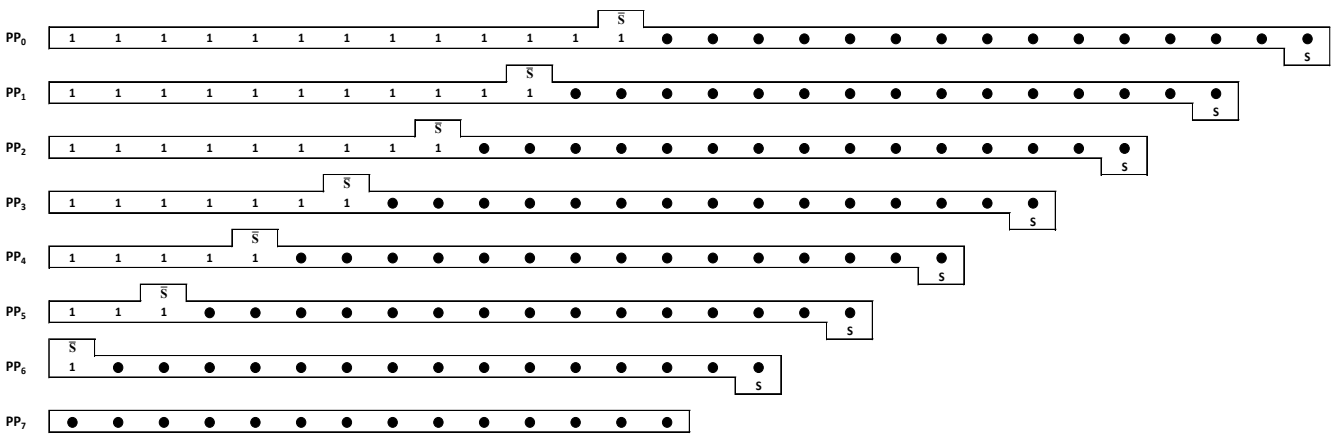
(2) Shown the Radix Booth-4 encoded partial products with simplified sign extension like that shown in Fig.10.82 of B5_Array and Recoded Multiplier (20). Draw another diagram with the dots or S that replace pij where i is the ith row and j is the bit location.

NEGi = 1 時，partial product 為負，PP_0~PP_6 皆有可能為負值，所以需要加上 NEGi。PP_7 因為 Y[15]=0, Y[14]=0，從 truth table 可得知，僅有 0, +X 的情況，所以不需要加上 NEGi。PP_0~PP_6 有 2Y, -2Y 的可能性，所以為 15 bits，而 PP_7 只有可能是 0, +Y，所以為 14 bits。

先做 signed extension



利用 S = S' + 1 化簡:



下圖為化簡的結果:

Radix Booth-4 encoded partial products with simplified sign extension

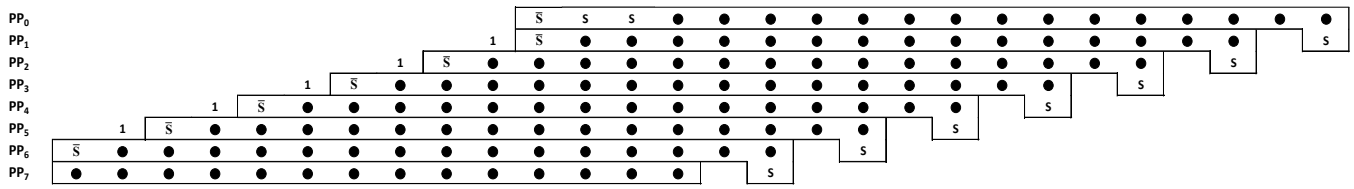
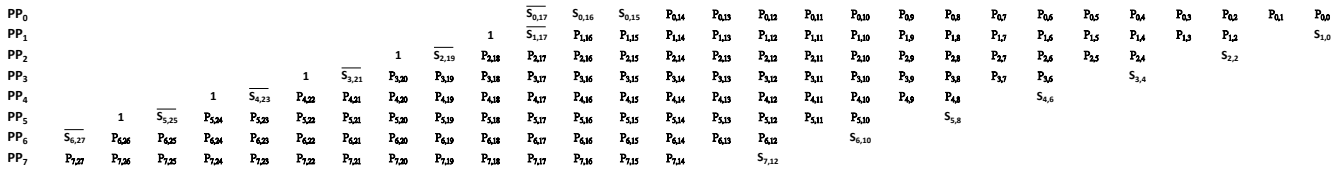
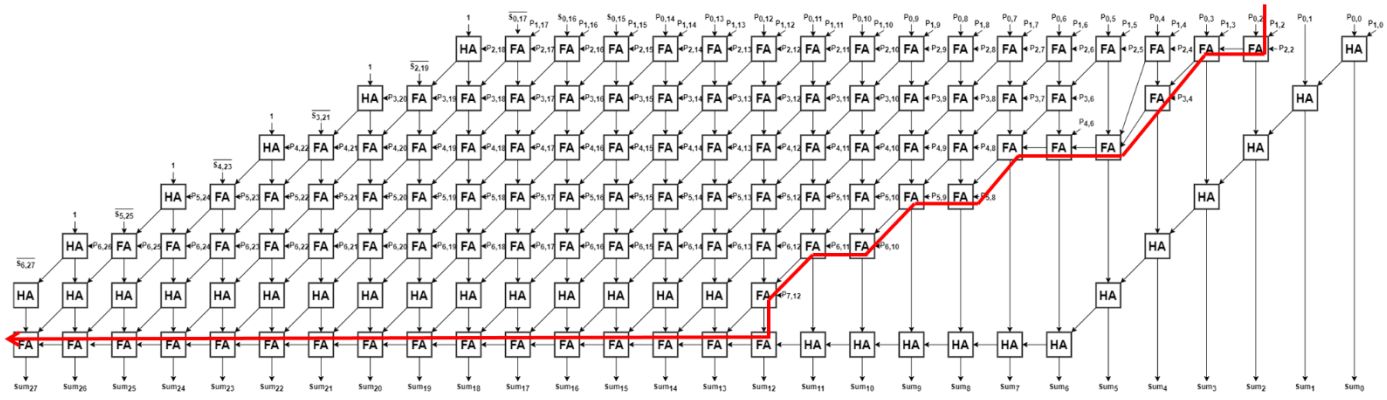


Diagram with the dot or S replaced with p_{ij} :



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

- **Block diagram & critical path**



critical delay time = $26 \times \tau_{FA} + 1 \times \tau_{HA}$

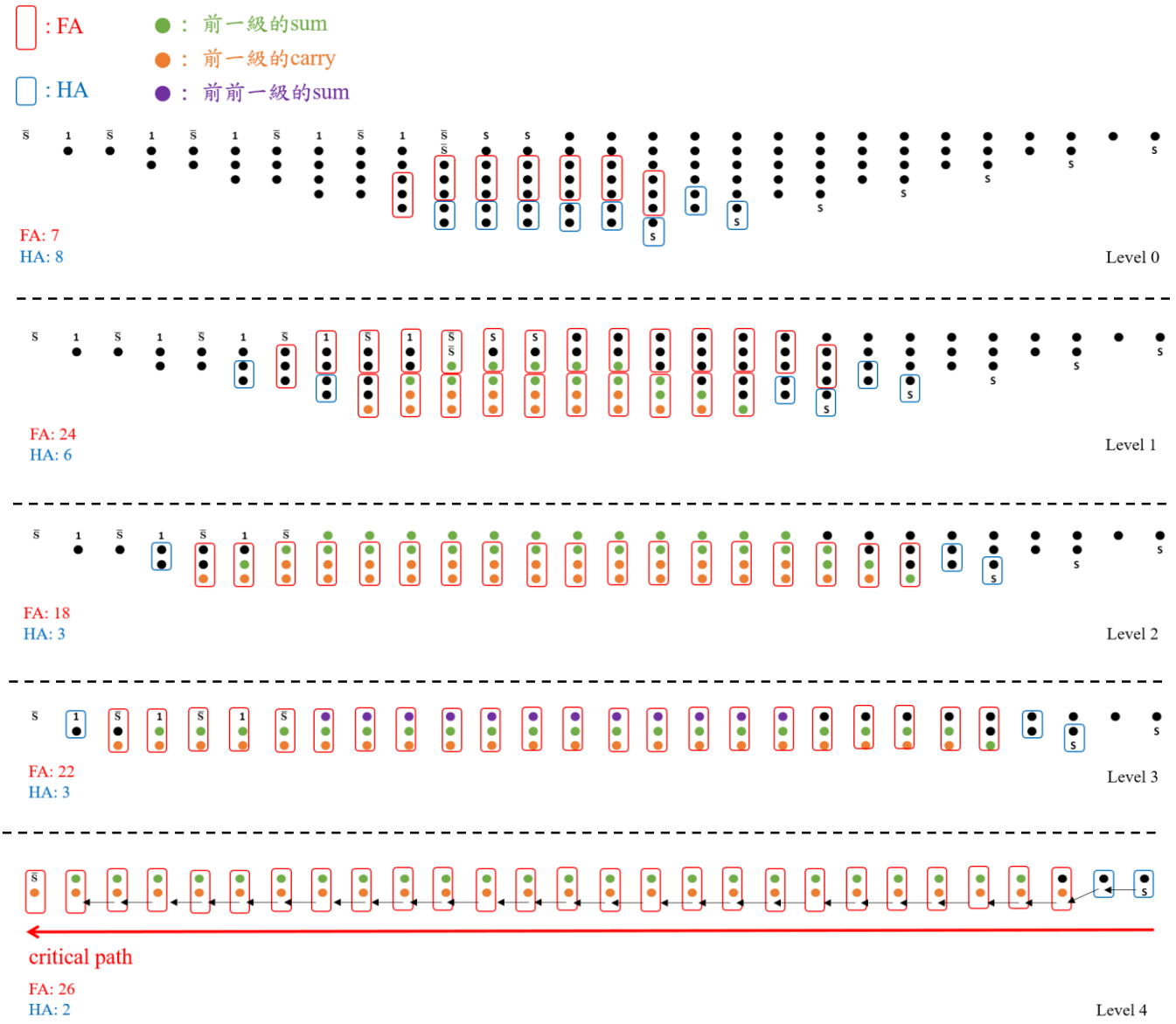
- **Number of FA & HA**

Full Adder	Half Adder
97	32

- **Design concept**

根據第二小題的結果來設計，將前幾項產生的 partial product 和前面已產生的 carry 相加。另外，若是只有兩個輸入或者沒有 carry in 的加法，則改用 Half Adder (HA)來節省面積並提升速度，其餘需要三個輸入的加法一律使用 Full Adder (FA)。如果有需要 propagate carry out 的加法部分則盡量往底部走，這樣可以提早運算完，或是有需要等待 carry in 的部分就盡可能使其平行運算，最後再 propagate 最長的 carry out 來得到各 bit 的 sum。

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



- **Design concept**
  - **Level 0:** 最初的 partial products array 共有 8 層，Dadda 每層的級數差 3/2 倍，因此設計採用 2, 3, 4, 6 級合併。
  - **Level 1:** 八層合併為六層，由右邊往左邊看，超過六層就用 FA 或 HA 合併到六層，並產生 sum 在本行和 carry 下一行。
  - **Level 2:** 六層合併為四層，設計概念如 Level 1。
  - **Level 3:** 四層合併為三層，設計概念如 Level 1。
  - **Level 4:** 三層合併為二層，設計概念如 Level 1，並在最後由左至右使用 carry ripple adder，產生最後的 sum 完成 Dadda method。

- **Number of HA & FA (Area)**

Level	HA	FA
0	8	7
1	6	24
2	3	18
3	3	22
4	2	26
Total	22	97

- **Critical path**

  Critical path = Carry Ripple Adder = Level 5 = 26 FA + 2 HA

- **Compare with array multiplier**

	Array multiplier	Dadda method
Area	33 HA + 96 FA	22 HA + 97 FA
Critical path	26 FA + 1 HA	26 FA + 2 HA

上表中可以觀察到，Dadda 方法的 critical path delay 比 Array multiplier 多了一個 HA，但在面積上卻減少了很多 HA 的數量，從而達到減少面積的需求。因此，Dadda 方法是一個能夠節省面積並且仍能保持一定速度的選擇（假設最後一級使用 carry ripple adder）。

如果兩種方法最後一層都統一使用速度較快的 adder，那麼由於 Dadda 方法到達最後一層的時間只需要 4 層，而 Array multiplier 的版本需要更多層，所以在這種情況下，Dadda 的架構較優。