Baugh-Wooley 2's compliment multiplier; By: Yihao Wang

if we have two 4 bit signed number $a = \{a_3, a_2, a_1, a_0\}$
$b = \{b_3, b_2, b_1, b_0\}$, they can be represent as follow:

$$(a)_{10} = -2^3 \cdot a_3 + 2^2 \cdot a_2 + 2^1 \cdot a_1 + 2^0 \cdot a_0$$

$$(b)_{10} = -2^3 \cdot b_3 + 2^2 \cdot b_2 + 2^1 \cdot b_1 + 2^0 \cdot b_0$$

$$(a \times b)_{10} = \left(-2^3 \cdot a_3 + \sum_{i=0}^{2} \cdot 2^i a_i\right) \cdot \left(-2^3 \cdot b_3 + \sum_{j=0}^{2} \cdot 2^j \cdot b_j\right)$$

$$= 2^6 \cdot a_3 \cdot b_3 + \sum_{i=0}^{2} \sum_{j=0}^{2} 2^{i+j} \cdot a_i \cdot b_j$$

$$- 2^3 a_3 \cdot \sum_{j=0}^{2} \cdot 2^j \cdot b_j - 2^3 \cdot b_3 \sum_{i=0}^{2} \cdot 2^i a_i$$

The range 4bit signed input is $[-8, 7]$
so the range of product is $[-56, 64]$ which needs 8 bits
so the result is 8bit width.

		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
						a_3	a_2	a_1	a_0	
						b_3	b_2	b_1	b_0	
R0:	+	0	0	0	0	0	a_2b_2	a_1b_1	a_0b_0	$\sum_{i=0}^{2} \cdot \sum_{j=0}^{2} a_i \cdot b_j \cdot 2^{i+j}$
R1:	+	0	0	0	0	a_2b_2	a_1b_1	a_0b_1	0	
R2:	+	0	0	0	a_2b_2	a_1b_2	a_0b_2	0	0	
R3:	−	0	0	a_3b_2	a_3b_1	a_3b_0	0	0	0	$-2^3 a_3 \sum_{j=0}^{2} 2^j \cdot b_j$
R4:	−	0	0	b_3a_2	b_3a_1	b_3a_0	0	0	0	$-2^3 b_3 \sum_{i=0}^{2} \cdot 2^i \cdot a_i$
R5:	+	0	a_3b_3	0	0	0	0	0	0	
		P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0	

these two rows must be represented as 2's compliment format:

For example:

The row 3 (R3) can be represented as 2's complement by inversing all bits and add "1" to LSB:

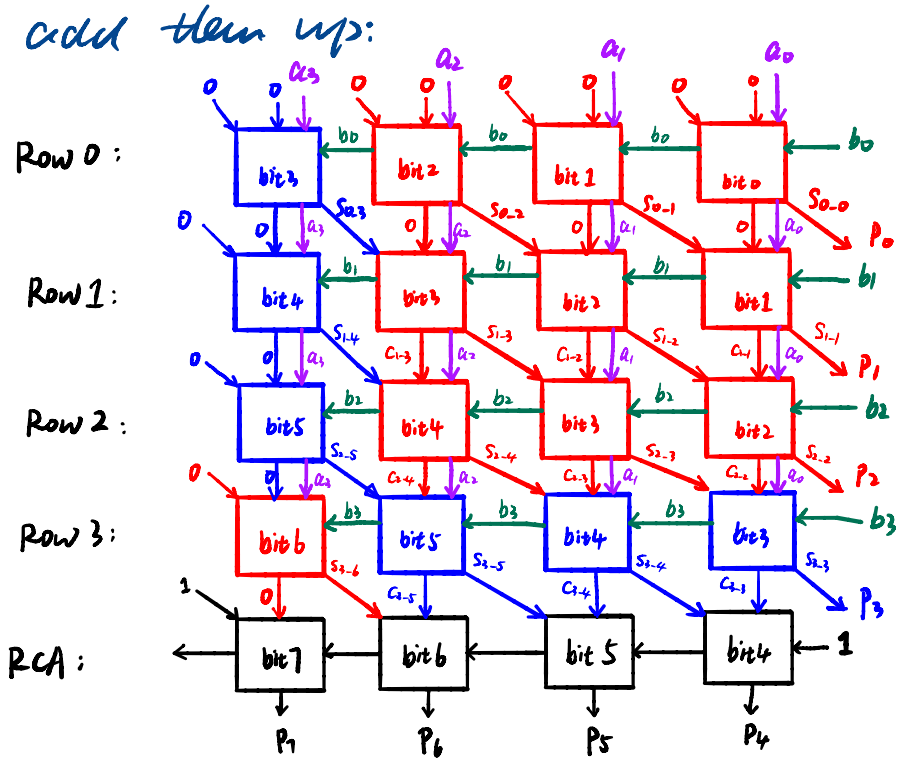
$$R3: \quad + \quad 1 \quad 1 \quad \overline{a_3b_2} \quad \overline{a_3b_1} \quad \overline{a_3b_0} \quad 1 \quad 1 \quad 1+1 \rightarrow LSB$$

$$\Rightarrow R3: \quad + \quad 1 \quad 1 \quad \overline{a_3b_2} \quad \overline{a_3b_1} \quad \overline{a_3b_0}+1 \quad 0 \quad 0 \quad 0$$

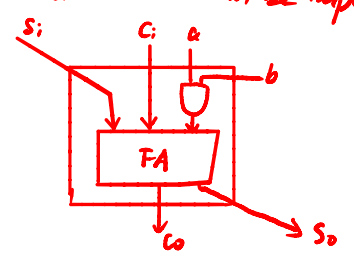
In the same fashion, the intact carry save tree is as follow:

		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
						a_3	a_2	a_1	a_0	
						b_3	b_2	b_1	b_0	
R0:	+	0	0	0	0	0	a_2b_2	a_1b_1	a_0b_0	$\sum_{i=0}^{2}\cdot\sum_{j=0}^{2} a_i\cdot b_j\cdot 2^{i+j}$
R1:	+	0	0	0	0	a_2b_2	a_1b_1	a_0b_1	0	
R2:	+	0	0	0	a_2b_2	a_1b_2	a_0b_2	0	0	
R3:	+	0	0	a_3b_2'	a_3b_1'	a_3b_0'	0	0	0	$-2^3a_3\sum_{j=0}^{2} 2^j\cdot b_j$
R4:	+	1	0	b_3a_2'	$b_3a_1'+1$	b_3a_0'	0	0	0	$-2^3 b_3\sum_{i=0}^{2}\cdot 2^i\cdot a_i$
R5:	+	0	a_3b_3	0	0	0	0	0	0	
		P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0	

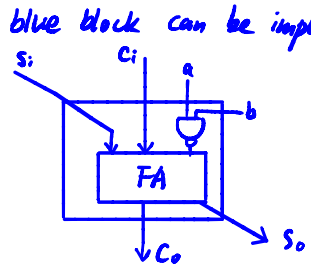
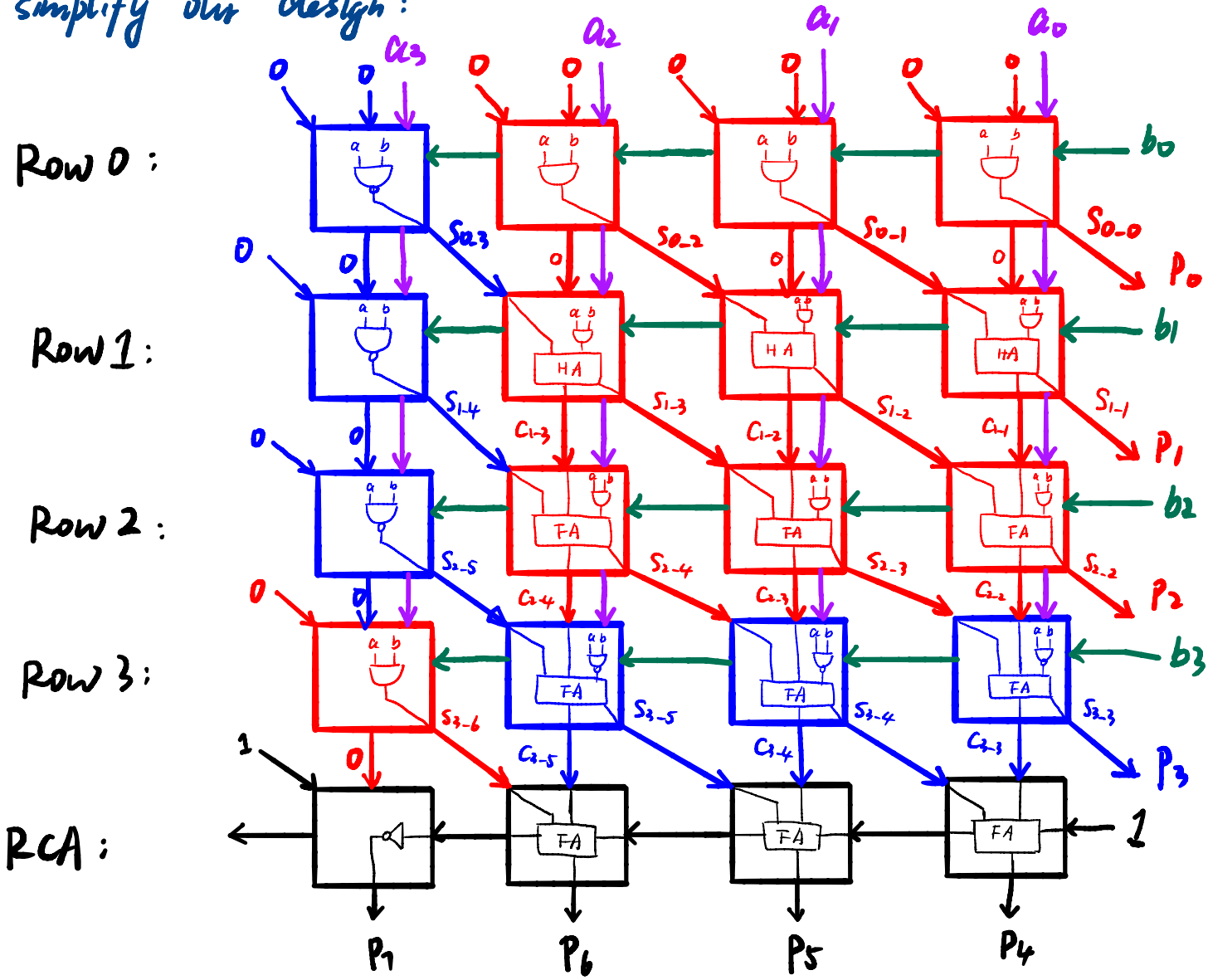
And then we can use general carry-saver adder tree to add them up:



The red block can be implemented as:



The blue block can be implemented as:



Because some inputs are constant '0' or '1', so we can simplify our design: