Computer Arithmetic

MIDTERM EXAM: MAY 31, 2004

T.J.H. Kluter

May 31, 2004

Contents

Chapter 1

Assignment 1.16

1.1 Part a

Assume a_7 , b_7 and s_7 are the sign bits of a two complements summator, addendum and result respectively. Than an overflow v in the adder occures in case the result is negative whilst the summator and addendum are both positive, or in case the result is positive whilst the summator and addendum are both negative. So v is defined as formulated in Equation 1.1.

$$\begin{aligned} v &= s_7 \cdot \overline{a_7} \cdot \overline{b_7} + \overline{s_7} \cdot a_7 \cdot b_7 \\ &= s_7 \cdot \overline{(a_7 + b_7)} + \overline{s_7} \cdot g_7 \\ &= s_7 \cdot \overline{t_7} + \overline{s_7} \cdot g_7 \end{aligned} \tag{1.1}$$

For the two's complement addition the summation is defined as in Equation 1.2, and the carry out of a bit position is defined as in Equation 1.3.

$$s_7 = p_7 \cdot \overline{c_7} + \overline{p_7} \cdot c_7 \Leftrightarrow \overline{s_7} = \overline{p_7} \cdot \overline{c_7} + p_7 \cdot c_7 \tag{1.2}$$

$$c_8 = g_7 + t_7 \cdot c_7 \Leftrightarrow \overline{c_8} = \overline{t_7} + \overline{g_7} \cdot \overline{c_7} \tag{1.3}$$

Substitution of Equation 1.2 into Equation 1.1, and the knowledge that $\overline{t_7} \cdot p_7 = 0$, and $g_7 \cdot p_7 = 0$, delivers:

$$\begin{aligned} v \quad = \quad & \overline{t_7} \cdot p_7 \cdot \overline{c_7} + \overline{t_7} \cdot \overline{p_7} \cdot c_7 + \\ & g_7 \cdot \overline{p_7} \cdot \overline{c_7} + g_7 \cdot p_7 \cdot c_7 \\ = \quad & \overline{t_7} \cdot c_7 + g_7 \cdot \overline{c_7} \end{aligned} \tag{1.4}$$

By introducing the terms $\overline{g_7} \cdot \overline{c_7} \cdot c_7$ and $t_7 \cdot c_7 \cdot \overline{c_7}$, which both equal to 0, into Equation 1.4, and the usage of Equation 1.3, we see that:

$$\begin{aligned} v &= c_7 \cdot (\overline{t_7} + \overline{g_7} \cdot \overline{c_7}) + \overline{c_7} \cdot (g_7 + t_7 \cdot c_7) \\ &= c_7 \cdot \overline{c_8} + \overline{c_7} \cdot c_8 \\ &= c_7 \oplus c_8 \end{aligned} \tag{1.5}$$

QED.

1.2 Part b

For one's complement numbers the sommation S of two one's complement numbers A and B, is defined as $S = A + B$ when $c_{out} = 0$ and $S = A + B + 1$ when $c_{out} = 1$. We clearly see a non-linearity in the definition of the one's complement summation function. For simplicity let's assume that $c_{in} = 0$. It can easily being showed that the following discussion also holds in case $c_{in} = 1$. Taking the definition of the one's complement summation, we can build a one's complement adder by using a two's complement adder in a "feedback" configuration as shown in Figure 1.1.

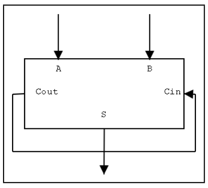

Figure 1.1: One's complement adder build from a Two's complement feedback adder

It can be shown that the one's complement adder of Figure 1.1 stabalizes very quickly, and thus does not form an oscillator. For simplicity let's assume that our A, B and S terms are three-bits one's complement numbers. It can be shown that the following discussion holds for any n-bits one's complement numbers. Under the assumption of the three-bits one's complement numbers we can build a one's complement adder, from the one shown in Figure 1.1, as the two shown in Figure 1.2. Note that the adder shown on the left produces exactly the same result as the one shown on the right!

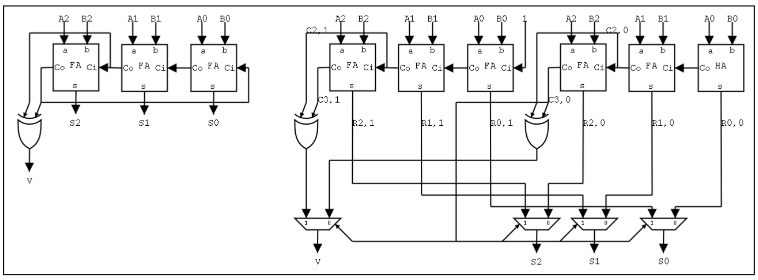

Figure 1.2: Three bits one's complement adder

In Figure 1.2 V denotes the Overflow output. We also know that the summation is lineair and symetric, e.g. $S = A + B \Leftrightarrow S = B + A$. Table 1.1 list all possible one's complement additions, and clearly shows that $v = c_2 \oplus c_3$ is the definition for the overflow in a one's complement addition. Note that the numbers shown underlined are the real outputs of the one's complement addition.

A	B	S	$R_{n,0}$	$C_{2,0}$	$R_{n,1}$	$C_{2,1}$	$C_{3,0}$	V
000_b	000_b	0+0	$\underline{000_b}$	$\underline{0}$	001_b	0	$\underline{0}$	$\underline{0}$
000_b	001_b	0+1	$\underline{001_b}$	$\underline{0}$	010_b	0	$\underline{0}$	$\underline{0}$
000_b	010_b	0+2	$\underline{010_b}$	$\underline{0}$	011_b	0	$\underline{0}$	$\underline{0}$
000_b	011_b	0+3	$\underline{011_b}$	$\underline{0}$	100_b	1	$\underline{0}$	$\underline{0}$
000_b	100_b	0-3	$\underline{100_b}$	$\underline{0}$	101_b	0	$\underline{0}$	$\underline{0}$
000_b	101_b	0-2	$\underline{101_b}$	$\underline{0}$	110_b	0	$\underline{0}$	$\underline{0}$
000_b	110_b	0-1	$\underline{110_b}$	$\underline{0}$	111_b	0	$\underline{0}$	$\underline{0}$
000_b	111_b	0-0	$\underline{111_b}$	$\underline{0}$	000_b	1	$\underline{0}$	$\underline{0}$
001_b	001_b	1+1	$\underline{010_b}$	$\underline{0}$	011_b	0	$\underline{0}$	$\underline{0}$
001_b	010_b	1+2	$\underline{011_b}$	$\underline{0}$	100_b	1	$\underline{0}$	$\underline{0}$
001_b	011_b	1+3	$\underline{100_b}$	$\underline{1}$	101_b	1	$\underline{0}$	$\underline{1}$
001_b	100_b	1-3	$\underline{101_b}$	$\underline{0}$	110_b	0	$\underline{0}$	$\underline{0}$
001_b	101_b	1-2	$\underline{110_b}$	$\underline{0}$	111_b	0	$\underline{0}$	$\underline{0}$
001_b	110_b	1-1	$\underline{111_b}$	$\underline{0}$	000_b	1	$\underline{0}$	$\underline{0}$
001_b	111_b	1-0	000_b	1	$\underline{001_b}$	$\underline{1}$	$\underline{1}$	$\underline{0}$
010_b	010_b	2+2	$\underline{100_b}$	$\underline{1}$	101_b	1	$\underline{0}$	$\underline{1}$
010_b	011_b	2+3	$\underline{101_b}$	$\underline{1}$	110_b	1	$\underline{0}$	$\underline{1}$
010_b	100_b	2-3	$\underline{110_b}$	$\underline{0}$	111_b	0	$\underline{0}$	$\underline{0}$
010_b	101_b	2-2	$\underline{111_b}$	$\underline{0}$	000_b	1	$\underline{0}$	$\underline{0}$
010_b	110_b	2-1	000_b	1	$\underline{001_b}$	$\underline{1}$	$\underline{1}$	$\underline{0}$
010_b	111_b	2-0	001_b	1	$\underline{010_b}$	$\underline{1}$	$\underline{1}$	$\underline{0}$
011_b	011_b	3+3	$\underline{110_b}$	$\underline{1}$	111_b	1	$\underline{0}$	$\underline{1}$
011_b	100_b	3-3	$\underline{111_b}$	$\underline{0}$	000_b	1	$\underline{0}$	$\underline{0}$
011_b	101_b	3-2	000_b	1	$\underline{001_b}$	$\underline{1}$	$\underline{1}$	$\underline{0}$
011_b	110_b	3-1	001_b	1	$\underline{010_b}$	$\underline{1}$	$\underline{1}$	$\underline{0}$
011_b	111_b	3-0	010_b	1	$\underline{011_b}$	$\underline{1}$	$\underline{1}$	$\underline{0}$
100_b	100_b	-3-3	000_b	0	$\underline{001_b}$	$\underline{0}$	$\underline{1}$	$\underline{1}$
100_b	101_b	-3-2	001_b	0	$\underline{010_b}$	$\underline{0}$	$\underline{1}$	$\underline{1}$
100_b	110_b	-3-1	010_b	0	$\underline{011_b}$	$\underline{0}$	$\underline{1}$	$\underline{1}$
100_b	111_b	-3-0	011_b	0	$\underline{100_b}$	$\underline{1}$	$\underline{1}$	$\underline{0}$
101_b	101_b	-2-2	010_b	0	$\underline{011_b}$	$\underline{0}$	$\underline{1}$	$\underline{1}$
101_b	110_b	-2-1	011_b	0	$\underline{100_b}$	$\underline{1}$	$\underline{1}$	$\underline{0}$
101_b	111_b	-2-0	100_b	1	$\underline{101_b}$	$\underline{1}$	$\underline{1}$	$\underline{0}$
110_b	110_b	-1-1	100_b	1	$\underline{101_b}$	$\underline{1}$	$\underline{1}$	$\underline{0}$
110_b	111_b	-1-0	101_b	1	$\underline{110_b}$	$\underline{1}$	$\underline{1}$	$\underline{0}$
111_b	111_b	-0-0	110_b	1	$\underline{111_b}$	$\underline{1}$	$\underline{1}$	$\underline{0}$

Table 1.1: All three bits one's complement additions

Chapter 2

Assignment 2.18

2.1 Part a

From Lings summation and propagate function we know that:

$$s_i = t_i \cdot \overline{h_i} + \overline{t_i} \cdot h_i + g_i \cdot t_{i-1} \cdot h_{i-1} \tag{2.1}$$

and:

$$h_i = g_i + t_{i-1} \cdot h_{i-1} \Leftrightarrow \overline{h_i} = \overline{g_i} \cdot \overline{t_{i-1}} + \overline{g_i} \cdot \overline{h_{i-1}} \tag{2.2}$$

Using these both Equations, and the knowledge that $\overline{t_i} \cdot g_i = 0$, we find:

$$\begin{aligned} s_i &= t_i \cdot \overline{g_i} \cdot \overline{t_{i-1}} + t_i \cdot \overline{g_i} \cdot \overline{h_{i-1}} + \overline{t_i} \cdot g_i + \\ & \quad \overline{t_i} \cdot t_{i-1} \cdot h_{i-1} + g_i \cdot t_{i-1} \cdot h_{i-1} \\ &= t_i \cdot \overline{g_i} \cdot (\overline{t_{i-1}} + \overline{h_{i-1}}) + (\overline{t_i} + g_i) \cdot t_{i-1} \cdot h_{i-1} \\ &= p_i \cdot \overline{(t_{i-1} \cdot h_{i-1})} + \overline{p_i} \cdot t_{i-1} \cdot h_{i-1} \\ &= p_i \oplus (t_{i-1} \cdot h_{i-1}) \end{aligned} \tag{2.3}$$

By definition we know that:

$$h_i = g_i + t_{i-1} \cdot h_{i-1} = c_{i+1} + c_i \tag{2.4}$$

and:

$$c_{i+1} = g_i + p_i \cdot c_i = g_i + t_i \cdot c_i \tag{2.5}$$

Using both equations results in:

$$g_i + t_{i-1} \cdot h_{i-1} = g_i + t_i \cdot c_i + c_i \Leftrightarrow g_i + t_{i-1} \cdot h_{i-1} = g_i + c_i \Rightarrow c_i = t_{i-1} \cdot h_{i-1} \tag{2.6}$$

As $c_i = t_{i-1} \cdot h_{i-1}$ and Lings summation is $s_i = p_i \oplus (t_{i-1} \cdot h_{i-1})$ this results in $s_i = p_i \oplus c_i$ which is the definition of the addition, QED.

2.2 Part b

The recursions for the adder generation for a conventional adder and a Ling adder for a group of four is shown in the Table below.
We know that a t_i term has a fanin of 1, and a g_i term has a fanin of 2. We can clearly see from the Table presented that the Ling recursion removes one

Conventional carry:	Conventional sum:	Ling carry:	Ling sum:
$c_1 = g_0 + t_0 \cdot c_0$	$s_0 = p_0 \oplus c_0$	$h_0 = g_0 + c_0$	$s_0 = p_0 \oplus c_0$
$c_2 = g_1 + t_1 \cdot g_0 + t_1 \cdot t_0 \cdot c_0$	$s_1 = p_1 \oplus c_1$	$h_1 = g_1 + g_0 + t_0 \cdot c_0$	$s_0 = p_1 \oplus (h_0 \cdot t_0)$
$c_3 = g_2 + t_2 \cdot g_1 + t_2 \cdot t_1 \cdot g_0 +$ $t_2 \cdot t_1 \cdot t_0 \cdot c_0$	$s_2 = p_2 \oplus c_2$	$h_2 = g_2 + g_1 + t_1 \cdot g_0 +$ $t_1 \cdot t_0 \cdot c_0$	$s_2 = p_2 \oplus (h_1 \cdot t_1)$
$c_4 = g_3 + t_3 \cdot g_2 + t_3 \cdot t_2 \cdot g_1 +$ $t_3 \cdot t_2 \cdot t_1 \cdot g_0 +$ $t_3 \cdot t_2 \cdot t_1 \cdot t_0 \cdot c_0$	$s_3 = p_3 \oplus c_3$	$h_3 = g_3 + g_2 + t_2 \cdot g_1 +$ $t_2 \cdot t_1 \cdot g_0 +$ $t_2 \cdot t_1 \cdot t_0 \cdot c_0$	$s_3 = p_3 \oplus (h_2 \cdot t_2)$
$c_{out} = c_4$		$c_{out} = h_3 \cdot t_3$	

product term of each sum-term of the carry generation, and bubbles it towards the sum calculation. For s_0 we clearly see that there is no difference between the conventional and the Ling adder, as both have no carry generation, and for the sum we have the same structure. For s_1 we see also that there is no difference, as the conventional carry generation uses a fanin scheme of [2,2] , whilst the Ling uses a scheme of [1,1]. In the sum generation the normal scheme is [1,1], whilst the Ling must use here [2,2], making the area and fanin of both scheme's equal. For s_2 we see the first advantage of Ling, the normal carry generation uses here a [2,3,3] scheme, whilst the Ling uses a [2,2,2] scheme. In the summation the normal scheme we still find [1,1,1], whilst the ling gives a [1,3,2] scheme. Overall, we can say that Ling removes here 2 "AND" product terms in the carry generation, and introduces one in the sum generation, which results in the area savings of one "AND" product term, and a fanin saving of 1. Equally for s_3 the Ling recursion removes 3 "AND" product terms in the carry generation, and intruduces one in the sum generation, which results in the area savings of two "AND" product terms, and a fanin saving of 2. Finally for the carry out generation the Ling saves four "AND" product tems, and only intruduces one in the carry out generation. Thus saving 3 "AND" produc terms, and a fanin saving of 3.
We can see that Ling saves a total of 6 "AND" product terms in Area and a total of 6 fanin savings, when we look at a block of four bits.

Chapter 3

Assignment 2.21

Denote $g_i = a_i \cdot b_i$ and $p_i = a_i \oplus b_i$. For the carry we get $c_{i+1} = g_i + p_i \cdot c_{i-1}$ and for the sum $s_i = p_i \oplus c_i$. When we consider the prefix adders than we can formulate:

$$c_1 = g_0 + p_0 \cdot c_0 \quad (3.1)$$

$$c_2 = g_1 + p_1 \cdot c_1 = (g_1 + p_1 \cdot g_0) + (p_1 \cdot p_0) \cdot c_0 = G_1 + P_1 \cdot c_0 \quad (3.2)$$

$$c_3 = g_2 + p_2 \cdot c_2 = (g_2 + p_2 \cdot G_1) + (p_2 \cdot P_1) \cdot c_0 = G_2 + P_2 \cdot c_0 \quad (3.3)$$

$$c_{n+1} = g_n + p_n \cdot c_n = (g_n + p_n \cdot G_{n-1}) + (p_n \cdot P_{n-1}) \cdot c_0 = G_n + P_n \cdot c_0 \quad (3.4)$$

And Thus:

$$s_n = p_n \oplus c_n = p_n \oplus (G_{n-1} + P_{n-1} \cdot c_0) \quad (3.5)$$

In case we have $c_0 = 0$ this reduces to $s_n = p_n \oplus G_{n-1}$ the well known prefix equation, and result for the requested $s = x + y$ addition. In case $c_0 = 1$ this equation states: $s_n = p_n \oplus (G_{n-1} + P_{n-1})$ which can be implemented by only one extra stage, and which delivers the result for the requested $z = x + y + 1$ addition. This means that by only adding n-"OR" gates and n-"XOR" gates we can determine $s = x + y$ and $z = x + y + 1$ with the same prefix adder, which has no carry in functionality.

Chapter 4

Assignment 3.24

4.1 Part a

We can determine the range of results by taking $min|z| = -4 - 3 \cdot 3 + 5 \cdot (-4) = -33$ and $max|z| = 3 - 3 \cdot (-4) + 5 \cdot 3 = 30$. Using two's complement representation, the value 30 can be represented in 6 bits, but the value -33 needs 7 bits to be represented. The least number of bits to represent z is therefore 7.

4.2 Part b

We can rewrite the summation as: $z = a - 4b + b + 4c + c$. The bit-matrix for this sommation is (as a , b and c are representable in three bits):

a_2	a_2	a_2	a_2	a_2	a_1	a_0	a
b_2	b_2	b_2	b_2	b_2	b_1	b_0	b
$\overline{b_2}$ 0	$\overline{b_2}$ 0	$\overline{b_2}$ 0	$\overline{b_1}$ 0	$\overline{b_0}$ 1	0 0	0 0	-4b
c_2	c_2	c_2	c_2	c_2	c_1	c_0	c
c_2	c_2	c_2	c_1	c_0	0	0	4c

It can be shown that $\{a_2, a_2, a_2, a_2, a_2, a_1, a_0\} = \{1, 1, 1, 1, 1, 0, 0\} + \{0, 0, 0, 0, \overline{a_2}, a_1, a_0\}$ for sign extenstions. Usign this property results in the first optimizations shown in Table 4.1. Know we can add all constants to form one constant value, as shown in Tables 4.2, 4.3 and 4.4. Finally by rewriting Table 4.4 a little bit we achieve the final bit-matrix shown in Table 4.5.

4.3 Part c

Taking Table 4.5 We can use 5 Full Adders (0..4) to form the reduced Matrix shown in Table 4.6. For the next reduction we can use FA (5) for the row of four, and we can again use two HA's (6..7) to form the reduced Matrix Show in Table 4.7.

The schematic of the realised adder is depicted in Figure 4.1.

0	0	0	0	$\overline{a_2}$	a_1	a_0
1	1	1	1	1	0	0
0	0	0	0	$\overline{b_2}$	b_1	b_0
1	1	1	1	1	0	0
0	0	b_2	$\overline{b_1}$	$\overline{b_0}$	0	0
1	1	1	0	0	0	0
0	0	0	0	1	0	0
0	0	0	0	$\overline{c_2}$	c_1	c_0
1	1	1	1	1	0	0
0	0	$\overline{c_2}$	c_1	c_0	0	0
1	1	1	0	0	0	0

Table 4.1: First table optimization

0	0	0	0	$\overline{a_2}$	a_1	a_0
0	0	0	0	$\overline{b_2}$	b_1	b_0
1	1	1	1	0	0	0
0	0	b_2	$\overline{b_1}$	$\overline{b_0}$	0	0
1	1	1	0	1	0	0
0	0	0	0	$\overline{c_2}$	c_1	c_0
0	0	$\overline{c_2}$	c_1	c_0	0	0
1	1	0	1	1	0	0

Table 4.2: Second table optimization

4.4 Part d

The minimal precision of the CPA in Figure 4.1 is 5 bits, as the lsb is directly provided by the csa-tree, and the sign bit is the inverse of the carry out of the CPA, as result of the constant '1' on position 6 in Table 4.7. The best suited CPA would be a carry-select based CPA.

0	0	0	0	$\overline{a_2}$	a_1	a_0
0	0	0	0	$\overline{b_2}$	b_1	b_0
0	0	b_2	$\overline{b_1}$	$\overline{b_0}$	0	0
1	1	0	0	1	0	0
0	0	0	0	$\overline{c_2}$	c_1	c_0
0	0	$\overline{c_2}$	c_1	c_0	0	0
1	1	0	1	1	0	0

Table 4.3: Third table optimization

				$\overline{a_2}$	a_1	a_0
				$\overline{b_2}$	b_1	b_0
		b_2	$\overline{b_1}$	$\overline{b_0}$		
				$\overline{c_2}$	c_1	c_0
		$\overline{c_2}$	c_1	c_0		
1		1	1			

Table 4.4: Final table optimization

				$\overline{a_2}$	a_1	a_0
				$\overline{b_2}$	b_1	b_0
1		1	1	$\overline{c_2}$	c_1	c_0
		b_2	$\overline{b_1}$	$\overline{b_0}$		
		$\overline{c_2}$	c_1	c_0		

Table 4.5: Reordered final table optimization

				s_2	s_1	s_0
			ca_2	ca_1	ca_0	
1		s_4	s_3			
	ca_4	ca_3		$\overline{b_0}$		
				c_0		

Table 4.6: Reduced final table optimization 1

				s_2	s_1	s_0
			s_6		ca_0	
1		ca_6				
	ca_4	s_7		s_5		
	ca_7		ca_5			

Table 4.7: Reduced final table optimization 2

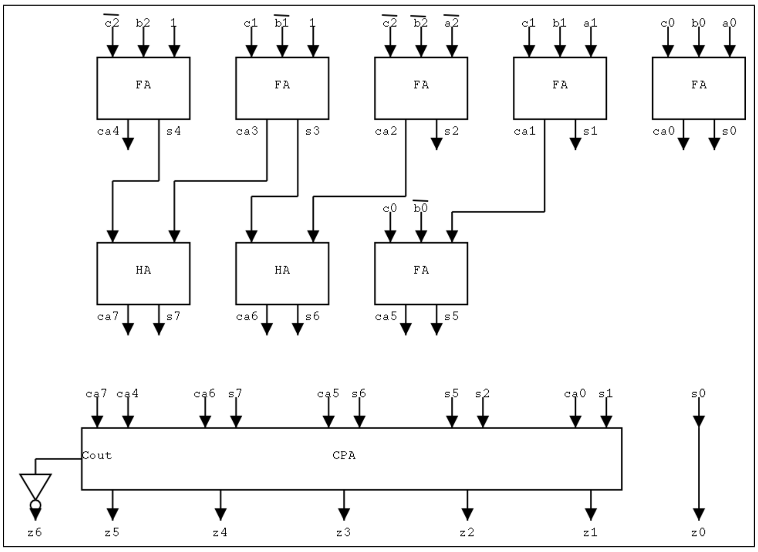

Figure 4.1: Schematic of the realized addition

Chapter 5

Assignment 4.18

5.1 Part a

Asume that the 12x12 two's complement multiplication is defined as $M = A$, where M is the multiplication result, A is the multiplier and B is the multiplicant. The bit-matrix for this multiplication without sign reduction, and the matrices for the 5x5 two's complement multiplication are shown in Figure 5.1. The number of 5x5 multiply modules needed are 9.

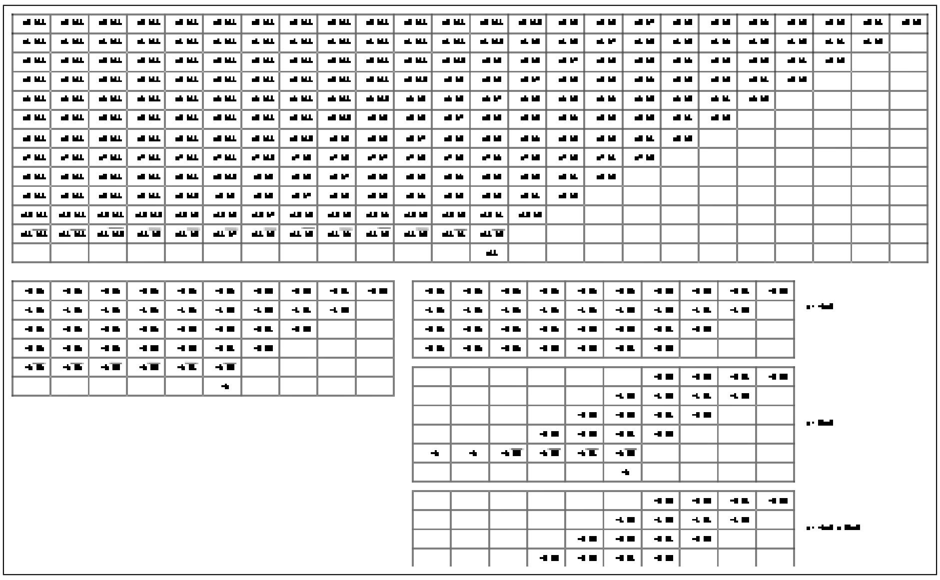

Figure 5.1: Multiply matices