Università di Pisa

Dipartimento di Informatica

Technical Report : TR-99-19

Multipliers based on Wallace trees and pseudo-products

Ciriani Valentina

November 09, 1999

ADDR: Corso Italia 40,56125 Pisa,Italy. TEL: +39-50-887111. FAX: +39-50-887226

Multipliers based on Wallace trees and pseudo-products

Valentina Ciriani

Abstract

The new concept of pseudo-product as extension of the concept of product has been introduced in a recent paper by Luccio and Pagli [LP99]. An arbitrary Boolean function can be expressed as a sum of pseudo-products (SPP), and this expression is, in general, shorter then the SP form. We present new algebraic SPP-expressions for the output functions of parallel integer multipliers based on Wallace trees and column compressions techniques.

1 Introduction: multiplication circuits

The hardware multiply unit is a standard component of computer systems. In fact high performance multipliers are essential in new arithmetic processors.

For an introduction see for example [H79]. A good chapter on arithmetic circuits can also be found in [CLR90].

Binary multiplication compute a $(m+n)$-bit product $p = p_{m+n-1} \dots p_0$ of two m-bit and n-bit numbers $a = a_{m-1} \dots a_0$ and $b = b_{n-1} \dots b_0$. The values **a, b, p** of a, b, p are respectively:

$$\mathbf{a} = \sum_{i=0}^{m-1} a_i 2^i \qquad \mathbf{b} = \sum_{i=0}^{n-1} b_i 2^i$$

$$\mathbf{p} = \mathbf{a} \cdot \mathbf{b} = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (a_i b_j) 2^{i+j} = \sum_{i=0}^{n-1} m_i.$$

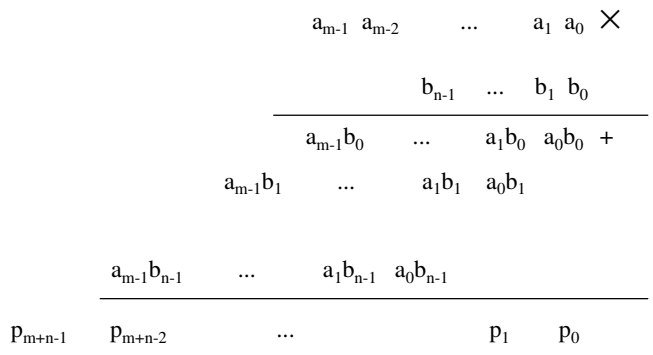

Figure 1: Multiplication is the sum of n partial products.

The terms $m_i = (a_{m-1}b_i \ldots \ a_0b_i) \cdot 2^i$ are called *partial products* and are generated in parallel by $n \times m$ AND gates. The problem of designing a parallel multiplier is then reduced to the multioperand addition problem. Figure 1 shows how the multiplication of a and b is equal to the sum of n partial products m_i .

The method to solve the above problem is via a Wallace tree circuit that reduces the addition of m n-bit numbers to the addition of two numbers.

This circuit is attributed to Wallace [W64], but Ofman [O63] had the same idea independently .

A tree input Wallace tree is the 3-carry-save adder which reduce a three numbers input to two numbers such that the input sum is equal to the output sum (see section 3).

For general m, n the first level of a Wallace tree uses $\lfloor m/3 \rfloor$ 3-carry-save adders in parallel to convert the sum of n numbers to the sum of $\lceil 2m/3 \rceil$ numbers. It does this recursively until there are two numbers left. The last step is a Carry Propagate Adder that sums the two numbers in the final product. Figure 7 shows (on the right) three Wallace trees with 5, 7, 9 input numbers.

Wallace trees are important in building array multipliers in a modular way. For the numbers a, b already defined, the high slice (a_h, b_h) and the

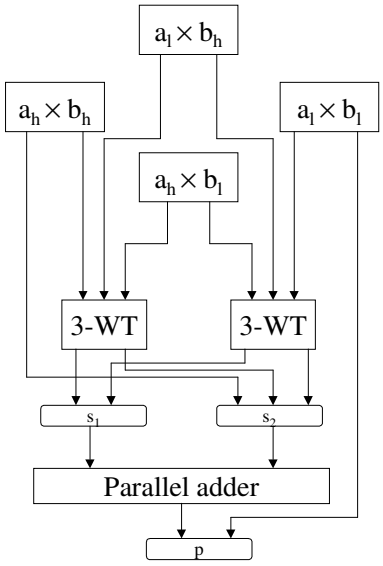

Figure 2: 8×8 modular multiplier build with four 4×4 multipliers, two 3-carry save adders and a parallel adder.

low slice (a_l, b_l) respectively are:

$$\begin{aligned} a_h &= a_{m-1} \ldots a_{m/2} \qquad a_l = a_{m/2-1} \ldots a_0 \\ b_h &= b_{n-1} \ldots b_{n/2} \qquad b_l = b_{n/2-1} \ldots b_0 \end{aligned}$$

We then have:

$$\begin{aligned} p &= a \cdot b = (a_h a_l) \cdot (b_h b_l) = a_h \cdot b_h + a_h \cdot b_l + a_l \cdot b_h + a_l \cdot b_l \\ &= p_{hh} + p_{hl} + p_{lh} + p_{ll} \end{aligned}$$

Figure 2 shows a 8 by 8 array multiplier that uses four 4×4 multiply modules, two Wallace trees and a Carry Propagate Adder; s_1 and s_2 are the input numbers of the parallel adder. The use of 4×4 multiply modules to build bigger multipliers requires some Wallace trees (table 1 where w_i is a i-input Wallace tree). The trees on the right side of figure 7 at page 12 are Wallace trees of size 5,7 and 9.

An other method for parallel multiplication, called column-compression (CC) multiplication, was proposed by Wallace in [W64] and by Dadda in [D65] and [D76]. CC multipliers are also described in [BSS95] and [WJM95].

Multiplier size	**Num. of 4×4 mult. mod.**	**Wallace trees required**
8×8	4	w_3
16×16	16	w_3,w_5,w_7
32×32	64	w_3,w_5,w_7,w_9,w_{11},w_{13},w_{15}

Table 1: Wallace trees required for various modular multipliers.

We describe below the multiplication process of a $n \times n$ product. The process has few stages. The first step reduces the summand matrix S_0 to a new matrix S_1 with less rows.

Note that in S_0: $\forall i \in [1, \ldots, n]$ column i (from the right) has i bits, and $\forall i \in [n+1, \ldots, 2n-1]$ column i has $2n - i$ bits. Figure 3 shows the entire process for a 8×8 multiplication.

The first and the second columns on the right of matrix S_0 are copied in S_1. The tree bits in column 3 are summed with a 3-carry save adder: the least significant output is placed in the third column of $S_1$and the carry is recorded in the fourth column. The process is repeated for the others column with n-carry save adders (for a description see section 3). Dadda in describing this procedure notes that n-carry save adders are difficult to realize. In fact he modified his method to use only 3-carry save adders. In this paper we show a shorter expression for some n-carry save adders and we propose to use them in the CC multiplication.

The outline of this paper is at follows: Section 2 gives a description of a new algebraic synthesis of Boolean formulas introduced by [LP99]. In section 3 the 3-carry save adders are extended for n input and we give the SPP forms of 5 and 7 carry save adders. Section 4 suggests the use of new compact expressions of n-carry save adders for modular multiplication and CC multiplication.

2 Pseudo-products

In the Boolean space the characteristic function of a cube can be expressed as the product (AND) of the variables having constant value in the cube, either in direct or complemented form. Any Boolean function can be expressed in

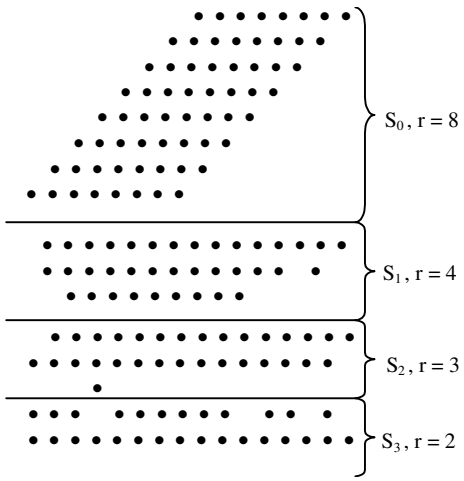

Figure 3: A multiplication process (8×8 product): the matrix S_0 is reduced to the matrix S_3 in 3 steps (r is the number of rows in the matrices).

SP form that is as a sum (OR) of products. Pseudo-cubes, pseudo-products and SPP forms introduced in [LP97] and [LP99] are generalizations of the above.

All the definitions and properties of this section are taken from [LP99].

Definition 1 *A vector* $\mathbf{u}$ *of* 2^m *elements,* $m \geq 0$*, is* ***normal*** *if*

1. $m = 0$*, or*
2. $m > 0$*, and* $\mathbf{u} = \mathbf{v}\widehat{v}$ *with* $\mathbf{v}$ *(hence* $\widehat{v}$*) normal. (*$\widehat{v}$ *denotes* $\mathbf{v}$ *or its complement* $\overline{v}$*).*

Definition 2 *A vector* $\mathbf{u}$ *is* ***constant*** *if* $\mathbf{u} = \mathbf{0}$ *or* $\mathbf{u} = \mathbf{1}$ *(*$\mathbf{0}$ *and* $\mathbf{1}$ *denote vectors of all* 0*'s or all* 1*'s respectively); otherwise* $\mathbf{u}$ *is* ***divided****.* $\mathbf{u}$ *is* ***balanced*** *if it is constant or half of its elements are* 0 *and half are* 1*. Two balanced vectors* $\mathbf{u}$**,** $\mathbf{v}$ *are* ***concordant*** *if* $\mathbf{u} = \mathbf{v}$ *or* $\mathbf{u} = \overline{v}$*; otherwise* $\mathbf{u}$*,* $\mathbf{v}$ *are* ***discordant****.*

In the following definitions a matrix will always have 2^m rows $\mathbf{r}_0, \ldots \mathbf{r}_{2^m-1}$ and n columns $\mathbf{c}_0, \ldots \mathbf{c}_{n-1}$ $n \geq 1$ and $0 \leq m \leq n$.

	$\mathbf{c}_0$	$\mathbf{c}_1$	$\mathbf{c}_2$	$\mathbf{c}_3$	$\mathbf{c}_4$	$\mathbf{c}_5$
$\mathbf{r}_0$	0	1	0	1	0	1
$\mathbf{r}_1$	0	1	0	1	1	0
$\mathbf{r}_2$	0	1	1	0	0	1
$\mathbf{r}_3$	0	1	1	0	1	0
$\mathbf{r}_4$	1	1	0	0	0	0
$\mathbf{r}_5$	1	1	0	0	1	1
$\mathbf{r}_6$	1	1	1	1	0	0
$\mathbf{r}_7$	1	1	1	1	1	1

Figure 4: A canonical matrix with 2^3 rows and $n = 6$ columns. With canonical columns c_0, c_2 and c_4.

Definition 3 *A matrix M is **normal** if $\mathbf{r}_i \neq \mathbf{r}_j$ for $i \neq j$, and all the columns are normal. A normal matrix is **canonical** if its rows, interpreted as binary numbers, are arranged in increasing order.*

Figure 4 shows a canonical matrix, the canonical columns are $\mathbf{c}_0$, $\mathbf{c}_2$ and $\mathbf{c}_4$.

Definition 4 *A normal vector $\mathbf{u} = v_0 \dots v_{2^{m-k}-1}$ is **k-normal**, $0 \leq k \leq m$. A k-normal vector is **k-canonical** if $v_i = 0$ for i even, and $v_i = 1$ for i odd.*

A restriction of a matrix M, composed of the rows whose elements in column c_i are equal to 1 (respectively equal to 0) is denoted by M_i (respectively $M_{\bar{i}}$).

Definition 5 *A matrix M is **balanced** if all its rows are pairwise different, all its columns are balanced and each M_i (hence $M_{\bar{i}}$) is balanced, $0 \leq i \leq n-1$.*

Proposition 6 *A canonical matrix M contains m columns $c_{i_0}, \dots c_{i_{m-1}}$ of increasing indices, such that c_{i_j} is $(m-j-1)$-canonical for $0 \leq j \leq m-1$.*

In a canonical matrix M the columns $c_{i_0}, \dots c_{i_{m-1}}$ of proposition 6 are called *canonical columns* of M. The other columns are the *non canonical* ones. The following propositions show that balanced and normal matrix are the same up to permutation of rows.

Proposition 7 *Any normal (in particular, canonical) matrix M is balanced.*

Proposition 8 *Any balanced matrix M can be transformed into the corresponding canonical one by rearranging the rows, interpreted as binary numbers, in increasing order.*

A set of m points of the Boolean space B^n can be arranged in a $m \times n$ matrix whose rows correspond to the points and the columns correspond to the variables describing B^n. For example the matrix of figure 4 represents eight points in B^6.

Definition 9 *A **pseudo-cube** of degree m is a set of 2^m points whose matrix is balanced, or equivalently it is canonical up to a row permutation.*

The matrix of figure 4 represents a pseudo-cube.

A the characteristic function of a pseudo-cube has a concise algebraic expression denoted by *CEX*.

Definition 10 *Let P be a pseudo-cube of degree m in B^n; let M be the canonical matrix of P; and let $x_{p_0}, \ldots, x_{p_{m-1}}$ and $x_{p_m}, \ldots, x_{p_{n-1}}$ be the canonical and non-canonical variables respectively, with these two sets ordered for increasing values of the indices. The **canonical expression** associated with P, $\mathbf{CEX(P)}$, is given by $f_0 \cdot f_1 \cdot \ldots \cdot f_{n-m-1}$, where each f_i, $0 \leq i \leq n-m-1$, is an EXOR-factor containing the following variables:*

1. *the canonical variables x_{p_j}, $0 \leq j \leq m-1$, such that $M[0, p_{m+1}] \neq M[2^{m-j-1}, p_{m+1}]$; these variables are ordered for increasing indexes;*
2. *the non-canonical variable $x_{p_{m+i}}$, if $M[0, p_{m+i}] = 1$; or $\overline{x}_{p_{m+i}}$, if $M[0, p_{m+i}] = 0$.*

In figure 4 column c_i correspond to variable x_i $(0 \leq i \leq 5)$. The canonical expression for the pseudo-cube is:

$$CEX = x_1 \cdot (x_0 \oplus x_3) \cdot (x_0 \oplus x_4 \oplus x_5)$$

In general, an arbitrary function can be expressed as a disjunction of pseudo-products, giving rise to a sum of pseudo-products (**SPP**) form.

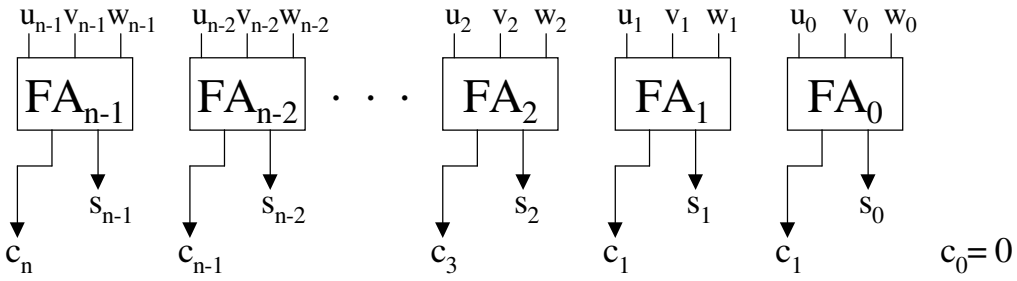

Figure 5: A n-bit carry save adder.

The SPP method is a direct generalization of the one for SP where pseudo-products are substituted for product.

The concept of prime implicant extend to the one of *prime pseudo-product* that is a pseudo product implying f, and not implying any other pseudo-product of f. There is an immediate extension of the classic SP minimization methods to construct SPP form with minimal number of pseudo products or minimal number of literal, composed of prime pseudo products only.

The number of literals in a minimal SPP expression is less equal then in the minimal SP expression for the same function.

On the negative side the SPP circuit is a three level circuit and there is a big effort to produce the SPP minimal forms due to the high number of prime pseudo-products.

3 The Carry save adder

Some arithmetic operations as multiplication require multiple-operand addition.

A 3-carry save adder reduces the problem of adding three numbers to the problem of adding just two numbers; then we only need to perform a two number addition.

The addition of 3 n-bit numbers $u = u_{n-1}u_{n-2}\dots u_0$, $v = v_{n-1}v_{n-2}\dots v_0$ and $w = w_{n-1}w_{n-2}\dots w_0$ is reduced to the addition of n-bit number $s =$

$s_{n-1}s_{n-2}\ldots s_0$ and a $n+1$-bit number $c = c_n c_{n-1} c_{n-2}\ldots c_0$ where:

$$u + v + w = s + c$$

Surprisingly a 3-carry save adder has a limited amount of hardware components and a constant amount of time.

The 3-carry save adder consists of n full adders. For $i \in [0, n-1]$, i-full adder takes input u_i, v_i and w_i; and computes:

$$\begin{aligned} s_i &= parity(u_i, v_i, w_i) \\ c_{i+1} &= majority(u_i, v_i, w_i), \quad c_0 = 0 \end{aligned}$$

such that:

$$\begin{aligned} parity(u, v, w) &= 1 \Leftrightarrow \text{there is an odd number of 1 in } \{u, v, w\} \\ majority(u, v, w) &= 1 \Leftrightarrow \text{there is the majority of 1 in } \{u, v, w\} \end{aligned}$$

Therefore the adder consists of n **independent** full adders: $\{FA_i\}$ as shown in figure 5. A full adder is a combinatorial circuit that takes as input three bits and outputs two bits: s (input sum) and c (carry).

SP and SPP forms for a 3-carry save adder are given by the expressions:

$$\begin{aligned} SPP: \quad & s_i = u_i \oplus v_i \oplus w_i \\ & c_{i+1} = v_i \cdot w_i + u_i \cdot (v_i \oplus w_i) \\ SP: \quad & s_i = u_i \cdot v_i \cdot w_i + u_i \cdot \overline{v_i} \cdot \overline{w_i} + \overline{u_i} \cdot v_i \cdot \overline{w_i} + \overline{u_i} \cdot \overline{v_i} \cdot w_i \\ & c_{i+1} = u_i \cdot v_i + v_i \cdot w_i + u_i \cdot w_i \end{aligned}$$

Comparing SP and SPP expressions for the 3-input circuit, the SPP form is not much more convenient and has one more level. Therefore we decided to use the SP form for 3-carry save adder.

The scheme of the 3-carry save adder generalizes to n inputs as follows.

The n-carry save adder perform parallel addition of n numbers ($n \geq 3$) to yield $\lfloor \log_2 n \rfloor + 1$ numbers output. It then recursively constructs a n-carry save adder on the resulting numbers. Then the set of numbers is reduced until there are only two numbers left.

Figure 6 shows a Wallace tree whit a 7-carry save adder and a 3-carry safe adder. Note that each line represents an entire number, not just a single bit.

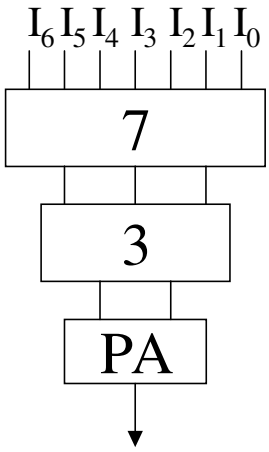

Figure 6: A 7-input Wallace tree. Composed by a 7-carry save adder, a 3-carry save adder and a parallel adder (PA).

The idea is not to add the carries. For a given n (number of operands), the maximum number of carries is $\lfloor \log_2 n \rfloor$. In fact if h_i^j is the i numeral of the j operand ($h_j^i \in \{0, 1\}$) we have:

$$h_1^i + h_2^i + h_3^i + \ldots + h_n^i = f_{\lfloor \log_2 n \rfloor}^{i+\lfloor \log_2 n \rfloor} \ldots f_2^{i+2} f_1^{i+1} f_0^i = h^i$$

Where $f_{\lfloor \log_2 n \rfloor}^{i+\lfloor \log_2 n \rfloor}, \ldots f_{2,}^{i+2} f_1^{i+1}$, f_0^i are the numerals of h^i. In fact carries are $f_{\lfloor \log_2 n \rfloor}^{i+\lfloor \log_2 n \rfloor}, \ldots, f_{2,}^{i+2} f_1^{i+1}$ and f_0^i is the sum.

Then we will have to add $\lfloor \log_2 n \rfloor + 1$ numbers $f_j = f_j^1 f_j^2 \ldots f_j^{n+j}$ with $j \in [0, \ldots, \lfloor \log_2 n \rfloor]$ and $f_j^k = 0 \ \forall k \in [1, \ldots, j]$.

The carries are not used in the sum of the following numerals then all single numerals are added simultaneously.

The time required by a Wallace tree depends on the depth of the carry save adders.

Typically a Wallace tree constructed whit 3-carry save adders has more levels that the one whit n-carry save adder ($n > 3$).

In order to minimize the number of levels in a n input Wallace tree we used a n-carry save adder as the first level.

Figure 7 shows some Wallace trees with n-carry save adder (on the left) and the correspondent ones with only 3-carry safe adders (on the right).

n-Wallace tree	n-carry save adders	Only 3-carry safe adders
5	5	6
6	5	6
7	5	8
8	7	8
9	7	8
15	9	12

Table 2: Number of levels needed to build a n-Wallace tree with a n-carry save adder (second column) and with only 3- carry save adders (third column).

The number of levels in a n input Wallace tree with a n-carry save adder as the first level and in the one with only 3-carry safe adders are summarized in table 2.

Note that n-carry save adders with $n > 3$ have 3 levels because we used a SPP form and 3-carry safe adders have 2 levels because we used the SP expression. In the number of level we don't consider the parallel adder.

We will show below 5 and 7 input circuits to be used in a 16 input modular multiplier. We then show how such a multiplier can be built.

3.1 5-input circuit

The 5-input circuit is shown in figure 7 on the top left.

The SPP form of the 5-carry save adder is shorter then the SP expression. In fact, given the input size k, SP form has $144k$ literals and SPP expression has $31k$ literals.

The SPP expressions of the output functions of a 5-carry save adder are:

$$
\begin{aligned}
F_5(1) &= I_0 \oplus I_1 \oplus I_2 \oplus I_3 \oplus I_4 \\
F_5(2) &= (I_4 \oplus I_0) \cdot (I_2 \oplus I_1) + (I_4 \oplus I_1) \cdot (I_2 \oplus I_3) + (I_4 \oplus I_3) \cdot (I_2 \oplus I_0) \\
F_5(4) &= I_0 \cdot I_1 \cdot I_2 \cdot (I_3 \oplus I_4) + I_0 \cdot I_3 \cdot I_4 \cdot (I_1 \oplus I_2) + I_1 \cdot I_2 \cdot I_3 \cdot I_4
\end{aligned}
$$

Where I_0, I_1, I_2, I_3, I_4 are the input numerals, $F_5(1)$ is the sum and $F_5(2)$, $F_5(4)$ the two carries.

Figure 7 shows that if we use a 5-carry save adder instead of two 3-carry save adders then we reduce the number of levels of the circuit .

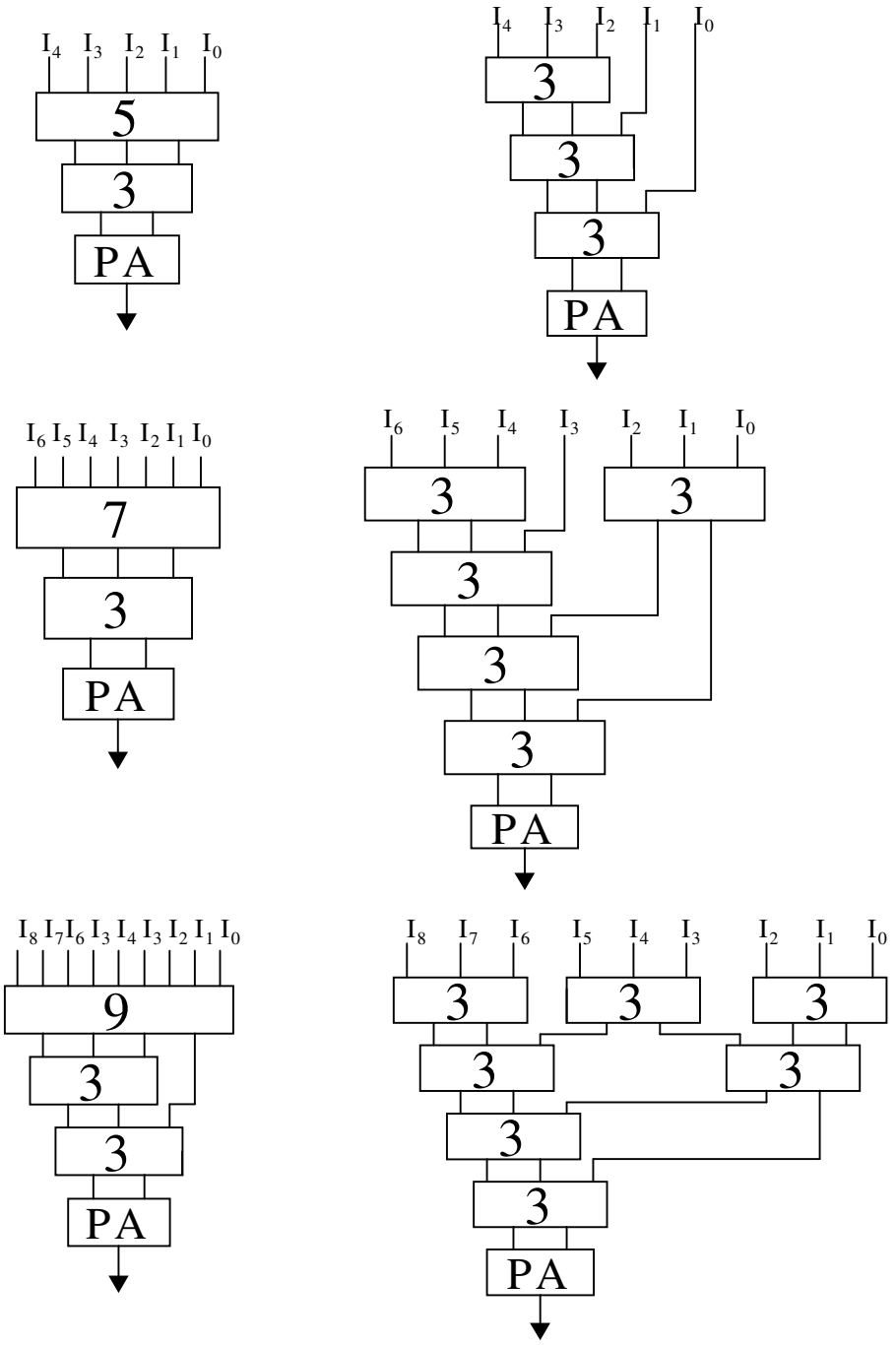

Figure 7: Wallace trees with 5,7,9 input. On the right the one built with a n-carry save adders and on the left the ones built with only 3-carry save adders.

n-Wallace tree	Litterals in SP form	Litterals in SPP form
3	$18k$	$8k$
5	$144k$	$31k$
7	$858k$	$142k$

Table 3: Number of litterals in the SP and SPP expressions (k is the input size).

3.2 7-input circuit

The minimal SP form for the 7-carry save adder has a huge number of implicants. The SPP expression is much more shorter then the corresponding minimal SP (SP form has $858k$ literals and SPP expression has $142k$ literals where k is the input size). But there is a greater effort to produce the SPP minimal form due to the highest number of prime pseudo-products over prime implicants.

In appendix A you can find the SP and the SPP expressions for the output of a 7-carry save adder.

4 Improving multiplication

For a 16 by 16 multiplication network we need six Wallace trees, two of each size: 3,5,7. With the new SPP expressions for Wallace trees of size 5 and 7 we have an improvement of the multiplication network. In fact the number of levels of the trees is reduced then 16 by 16 array multiplier is smaller.

We use SPP expressions for 5 and 7 input circuits because the number of literals is very low compared to the corresponding minimal SP form as table 3 shows.

The new expressions for 5 and 7 carry save adders can be also used for CC multiplier in order to reduced the number of steps in the multiplication process.

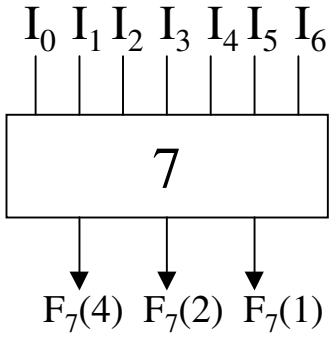

Figure 8: A 7-carry save adder with input I_0, I_1, I_2, I_3, I_4, I_5, I_6 and output functions $F_7(1)$, $F_7(2)$, $F_7(4)$.

A Appendix: SP and SPP expressions for a 7-carry save adder

In this appendix are shown SP and SPP expression for the outputs ($F_7(1)$, $F_7(2)$, $F_7(4)$: see figure 8) of a 7-carry save adder.

A.1 $F_7(1)$ expressions

A.1.1 SP form

$$
\begin{aligned}
F_7(1) = {} & \overline{I_0}\cdot\overline{I_1}\cdot\overline{I_2}\cdot\overline{I_3}\cdot\overline{I_4}\cdot\overline{I_5}\cdot I_6 + \overline{I_0}\cdot\overline{I_1}\cdot\overline{I_2}\cdot\overline{I_3}\cdot\overline{I_4}\cdot I_5\cdot\overline{I_6} \\
& + \overline{I_0}\cdot\overline{I_1}\cdot\overline{I_2}\cdot\overline{I_3}\cdot I_4\cdot\overline{I_5}\cdot\overline{I_6} + \overline{I_0}\cdot\overline{I_1}\cdot\overline{I_2}\cdot\overline{I_3}\cdot I_4\cdot I_5\cdot I_6 \\
& + \overline{I_0}\cdot\overline{I_1}\cdot\overline{I_2}\cdot I_3\cdot\overline{I_4}\cdot\overline{I_5}\cdot\overline{I_6} + \overline{I_0}\cdot\overline{I_1}\cdot\overline{I_2}\cdot I_3\cdot\overline{I_4}\cdot I_5\cdot I_6 \\
& + \overline{I_0}\cdot\overline{I_1}\cdot\overline{I_2}\cdot I_3\cdot I_4\cdot\overline{I_5}\cdot I_6 + \overline{I_0}\cdot\overline{I_1}\cdot\overline{I_2}\cdot I_3\cdot I_4\cdot I_5\cdot\overline{I_6} \\
& + \overline{I_0}\cdot\overline{I_1}\cdot I_2\cdot\overline{I_3}\cdot\overline{I_4}\cdot\overline{I_5}\cdot\overline{I_6} + \overline{I_0}\cdot\overline{I_1}\cdot I_2\cdot\overline{I_3}\cdot\overline{I_4}\cdot I_5\cdot I_6 \\
& + \overline{I_0}\cdot\overline{I_1}\cdot I_2\cdot\overline{I_3}\cdot I_4\cdot\overline{I_5}\cdot I_6 + \overline{I_0}\cdot\overline{I_1}\cdot I_2\cdot\overline{I_3}\cdot I_4\cdot I_5\cdot\overline{I_6} \\
& + \overline{I_0}\cdot\overline{I_1}\cdot I_2\cdot I_3\cdot\overline{I_4}\cdot\overline{I_5}\cdot I_6 + \overline{I_0}\cdot\overline{I_1}\cdot I_2\cdot I_3\cdot\overline{I_4}\cdot I_5\cdot\overline{I_6} \\
& + \overline{I_0}\cdot\overline{I_1}\cdot I_2\cdot I_3\cdot I_4\cdot\overline{I_5}\cdot\overline{I_6} + \overline{I_0}\cdot\overline{I_1}\cdot I_2\cdot I_3\cdot I_4\cdot I_5\cdot I_6 \\
& + \overline{I_0}\cdot I_1\cdot\overline{I_2}\cdot\overline{I_3}\cdot\overline{I_4}\cdot\overline{I_5}\cdot\overline{I_6} + \overline{I_0}\cdot I_1\cdot\overline{I_2}\cdot\overline{I_3}\cdot\overline{I_4}\cdot I_5\cdot I_6 \\
& + \overline{I_0}\cdot I_1\cdot\overline{I_2}\cdot\overline{I_3}\cdot I_4\cdot\overline{I_5}\cdot I_6 + \overline{I_0}\cdot I_1\cdot\overline{I_2}\cdot\overline{I_3}\cdot I_4\cdot I_5\cdot\overline{I_6} \\
& + \overline{I_0}\cdot I_1\cdot\overline{I_2}\cdot I_3\cdot\overline{I_4}\cdot\overline{I_5}\cdot I_6 + \overline{I_0}\cdot I_1\cdot\overline{I_2}\cdot I_3\cdot\overline{I_4}\cdot I_5\cdot\overline{I_6}
\end{aligned}
$$

$$
\begin{aligned}
&+\overline{I_0}\cdot I_1\cdot\overline{I_2}\cdot I_3\cdot I_4\cdot\overline{I_5}\cdot\overline{I_6}+\overline{I_0}\cdot I_1\cdot\overline{I_2}\cdot I_3\cdot I_4\cdot I_5\cdot I_6\\
&+\overline{I_0}\cdot I_1\cdot I_2\cdot\overline{I_3}\cdot\overline{I_4}\cdot\overline{I_5}\cdot I_6+\overline{I_0}\cdot I_1\cdot I_2\cdot\overline{I_3}\cdot\overline{I_4}\cdot I_5\cdot\overline{I_6}\\
&+\overline{I_0}\cdot I_1\cdot I_2\cdot\overline{I_3}\cdot I_4\cdot\overline{I_5}\cdot\overline{I_6}+\overline{I_0}\cdot I_1\cdot I_2\cdot\overline{I_3}\cdot I_4\cdot I_5\cdot I_6\\
&+\overline{I_0}\cdot I_1\cdot I_2\cdot I_3\cdot\overline{I_4}\cdot\overline{I_5}\cdot\overline{I_6}+\overline{I_0}\cdot I_1\cdot I_2\cdot I_3\cdot\overline{I_4}\cdot I_5\cdot I_6\\
&+\overline{I_0}\cdot I_1\cdot I_2\cdot I_3\cdot I_4\cdot\overline{I_5}\cdot I_6+\overline{I_0}\cdot I_1\cdot I_2\cdot I_3\cdot I_4\cdot I_5\cdot\overline{I_6}\\
&+I_0\cdot\overline{I_1}\cdot\overline{I_2}\cdot\overline{I_3}\cdot\overline{I_4}\cdot\overline{I_5}\cdot\overline{I_6}+I_0\cdot\overline{I_1}\cdot\overline{I_2}\cdot\overline{I_3}\cdot\overline{I_4}\cdot I_5\cdot I_6\\
&+I_0\cdot\overline{I_1}\cdot\overline{I_2}\cdot\overline{I_3}\cdot I_4\cdot\overline{I_5}\cdot I_6+I_0\cdot\overline{I_1}\cdot\overline{I_2}\cdot\overline{I_3}\cdot I_4\cdot I_5\cdot\overline{I_6}\\
&+I_0\cdot\overline{I_1}\cdot\overline{I_2}\cdot I_3\cdot\overline{I_4}\cdot\overline{I_5}\cdot I_6+I_0\cdot\overline{I_1}\cdot\overline{I_2}\cdot I_3\cdot\overline{I_4}\cdot I_5\cdot\overline{I_6}\\
&+I_0\cdot\overline{I_1}\cdot\overline{I_2}\cdot I_3\cdot I_4\cdot\overline{I_5}\cdot\overline{I_6}+I_0\cdot\overline{I_1}\cdot\overline{I_2}\cdot I_3\cdot I_4\cdot I_5\cdot I_6\\
&+I_0\cdot\overline{I_1}\cdot I_2\cdot\overline{I_3}\cdot\overline{I_4}\cdot\overline{I_5}\cdot I_6+I_0\cdot\overline{I_1}\cdot I_2\cdot\overline{I_3}\cdot\overline{I_4}\cdot I_5\cdot\overline{I_6}\\
&+I_0\cdot\overline{I_1}\cdot I_2\cdot\overline{I_3}\cdot I_4\cdot\overline{I_5}\cdot\overline{I_6}+I_0\cdot\overline{I_1}\cdot I_2\cdot\overline{I_3}\cdot I_4\cdot I_5\cdot I_6\\
&+I_0\cdot\overline{I_1}\cdot I_2\cdot I_3\cdot\overline{I_4}\cdot\overline{I_5}\cdot\overline{I_6}+I_0\cdot\overline{I_1}\cdot I_2\cdot I_3\cdot\overline{I_4}\cdot I_5\cdot I_6\\
&+I_0\cdot\overline{I_1}\cdot I_2\cdot I_3\cdot I_4\cdot\overline{I_5}\cdot I_6+I_0\cdot\overline{I_1}\cdot I_2\cdot I_3\cdot I_4\cdot I_5\cdot\overline{I_6}\\
&+I_0\cdot I_1\cdot\overline{I_2}\cdot\overline{I_3}\cdot\overline{I_4}\cdot\overline{I_5}\cdot I_6+I_0\cdot I_1\cdot\overline{I_2}\cdot\overline{I_3}\cdot\overline{I_4}\cdot I_5\cdot\overline{I_6}\\
&+I_0\cdot I_1\cdot\overline{I_2}\cdot\overline{I_3}\cdot I_4\cdot\overline{I_5}\cdot\overline{I_6}+I_0\cdot I_1\cdot\overline{I_2}\cdot\overline{I_3}\cdot I_4\cdot I_5\cdot I_6\\
&+I_0\cdot I_1\cdot\overline{I_2}\cdot I_3\cdot\overline{I_4}\cdot\overline{I_5}\cdot\overline{I_6}+I_0\cdot I_1\cdot\overline{I_2}\cdot I_3\cdot\overline{I_4}\cdot I_5\cdot I_6\\
&+I_0\cdot I_1\cdot\overline{I_2}\cdot I_3\cdot I_4\cdot\overline{I_5}\cdot I_6+I_0\cdot I_1\cdot\overline{I_2}\cdot I_3\cdot I_4\cdot I_5\cdot\overline{I_6}\\
&+I_0\cdot I_1\cdot I_2\cdot\overline{I_3}\cdot\overline{I_4}\cdot\overline{I_5}\cdot\overline{I_6}+I_0\cdot I_1\cdot I_2\cdot I_3\cdot I_4\cdot I_5\cdot I_6\\
&+I_0\cdot I_1\cdot I_2\cdot I_3\cdot I_4\cdot I_5\cdot I_6+I_0\cdot I_1\cdot I_2\cdot\overline{I_3}\cdot\overline{I_4}\cdot I_5\cdot I_6\\
&+I_0\cdot I_1\cdot I_2\cdot\overline{I_3}\cdot I_4\cdot\overline{I_5}\cdot I_6+I_0\cdot I_1\cdot I_2\cdot\overline{I_3}\cdot I_4\cdot I_5\cdot\overline{I_6}\\
&+I_0\cdot I_1\cdot I_2\cdot I_3\cdot\overline{I_4}\cdot\overline{I_5}\cdot I_6+I_0\cdot I_1\cdot I_2\cdot I_3\cdot\overline{I_4}\cdot I_5\cdot\overline{I_6}\\
&+I_0\cdot I_1\cdot I_2\cdot I_3\cdot I_4\cdot\overline{I_5}\cdot\overline{I_6}+I_0\cdot I_1\cdot I_2\cdot I_3\cdot I_4\cdot I_5\cdot I_6
\end{aligned}
$$

A.1.2 SPP form

$F_7(1) = I_0 \oplus I_1 \oplus I_2 \oplus I_3 \oplus I_4 \oplus I_5 \oplus I_6$

A.2 $F_7(2)$ expressions

A.2.1 SP form

$$
\begin{aligned}
F_7(2) = &\overline{I_0}\cdot\overline{I_1}\cdot\overline{I_2}\cdot\overline{I_3}\cdot I_5\cdot I_6+\overline{I_0}\cdot\overline{I_1}\cdot\overline{I_2}\cdot\overline{I_4}\cdot I_5\cdot I_6\\
&+\overline{I_0}\cdot\overline{I_1}\cdot\overline{I_3}\cdot\overline{I_4}\cdot I_5\cdot I_6+I_0\cdot\overline{I_1}\cdot I_2\cdot\overline{I_3}\cdot\overline{I_4}\cdot\overline{I_5}\\
&+\overline{I_0}\cdot I_2\cdot\overline{I_3}\cdot I_4\cdot\overline{I_5}\cdot\overline{I_6}+\overline{I_1}\cdot I_2\cdot\overline{I_3}\cdot I_4\cdot\overline{I_5}\cdot\overline{I_6}\\
&+\overline{I_0}\cdot\overline{I_1}\cdot\overline{I_2}\cdot I_4\cdot\overline{I_5}\cdot I_6+\overline{I_0}\cdot\overline{I_1}\cdot\overline{I_3}\cdot I_4\cdot\overline{I_5}\cdot I_6
\end{aligned}
$$

$$
\begin{aligned}
&+\overline{I_0}\cdot\overline{I_2}\cdot\overline{I_3}\cdot I_4\cdot\overline{I_5}\cdot I_6+I_1\cdot\overline{I_2}\cdot\overline{I_3}\cdot I_4\cdot\overline{I_5}\cdot\overline{I_6}\\
&+I_0\cdot I_1\cdot I_3\cdot I_4\cdot I_5\cdot I_6+\overline{I_0}\cdot\overline{I_1}\cdot\overline{I_2}\cdot I_4\cdot I_5\cdot\overline{I_6}\\
&+\overline{I_0}\cdot\overline{I_1}\cdot\overline{I_3}\cdot I_4\cdot I_5\cdot\overline{I_6}+\overline{I_0}\cdot\overline{I_2}\cdot\overline{I_3}\cdot I_4\cdot I_5\cdot\overline{I_6}\\
&+\overline{I_1}\cdot\overline{I_2}\cdot\overline{I_3}\cdot I_4\cdot I_5\cdot\overline{I_6}+I_0\cdot I_1\cdot I_2\cdot I_4\cdot I_5\cdot I_6\\
&+\overline{I_0}\cdot I_2\cdot I_3\cdot\overline{I_4}\cdot\overline{I_5}\cdot\overline{I_6}+\overline{I_0}\cdot\overline{I_1}\cdot I_3\cdot\overline{I_4}\cdot\overline{I_5}\cdot I_6\\
&+\overline{I_0}\cdot\overline{I_2}\cdot I_3\cdot\overline{I_4}\cdot\overline{I_5}\cdot I_6+\overline{I_1}\cdot I_2\cdot I_3\cdot\overline{I_4}\cdot\overline{I_5}\cdot\overline{I_6}\\
&+\overline{I_0}\cdot I_1\cdot\overline{I_2}\cdot\overline{I_3}\cdot\overline{I_4}\cdot I_6+I_0\cdot\overline{I_1}\cdot\overline{I_2}\cdot\overline{I_3}\cdot\overline{I_4}\cdot I_5\\
&+\overline{I_0}\cdot\overline{I_1}\cdot I_3\cdot\overline{I_4}\cdot I_5\cdot\overline{I_6}+\overline{I_0}\cdot\overline{I_2}\cdot I_3\cdot\overline{I_4}\cdot I_5\cdot\overline{I_6}\\
&+\overline{I_1}\cdot\overline{I_2}\cdot I_3\cdot\overline{I_4}\cdot I_5\cdot\overline{I_6}+I_0\cdot\overline{I_1}\cdot\overline{I_2}\cdot I_3\cdot\overline{I_4}\cdot\overline{I_5}\\
&+I_1\cdot\overline{I_2}\cdot I_3\cdot\overline{I_4}\cdot\overline{I_5}\cdot\overline{I_6}+\overline{I_0}\cdot\overline{I_1}\cdot I_3\cdot I_4\cdot\overline{I_5}\cdot\overline{I_6}\\
&+\overline{I_0}\cdot\overline{I_2}\cdot I_3\cdot I_4\cdot\overline{I_5}\cdot\overline{I_6}+\overline{I_1}\cdot\overline{I_2}\cdot I_3\cdot I_4\cdot\overline{I_5}\cdot\overline{I_6}\\
&+I_0\cdot I_1\cdot I_2\cdot I_3\cdot I_5\cdot I_6+I_0\cdot I_1\cdot I_2\cdot I_3\cdot I_4\cdot I_6\\
&+I_0\cdot I_2\cdot I_3\cdot I_4\cdot I_5\cdot I_6+\overline{I_0}\cdot I_2\cdot\overline{I_3}\cdot\overline{I_4}\cdot\overline{I_5}\cdot I_6\\
&+I_0\cdot\overline{I_1}\cdot\overline{I_2}\cdot\overline{I_3}\cdot I_4\cdot\overline{I_5}+I_1\cdot I_2\cdot\overline{I_3}\cdot\overline{I_4}\cdot\overline{I_5}\cdot\overline{I_6}\\
&+I_1\cdot\overline{I_2}\cdot\overline{I_3}\cdot\overline{I_4}\cdot I_5\cdot\overline{I_6}+I_1\cdot I_2\cdot I_3\cdot I_4\cdot I_5\cdot I_6\\
&+\overline{I_0}\cdot I_2\cdot\overline{I_3}\cdot\overline{I_4}\cdot I_5\cdot\overline{I_6}+\overline{I_1}\cdot I_2\cdot\overline{I_3}\cdot\overline{I_4}\cdot I_5\cdot\overline{I_6}\\
&+I_0\cdot\overline{I_1}\cdot\overline{I_2}\cdot\overline{I_3}\cdot\overline{I_4}\cdot I_6+I_0\cdot I_1\cdot\overline{I_2}\cdot\overline{I_3}\cdot\overline{I_4}\cdot\overline{I_5}
\end{aligned}
$$

A.2.2 SPP form

$$
\begin{aligned}
F_7(2) = &\ \overline{I_2}\cdot(I_0\oplus I_3)\cdot(I_1\oplus I_4)\cdot\overline{I_5}\\
&+(I_0\oplus I_2)\cdot\overline{I_4}\cdot(I_3\oplus I_5)\cdot\overline{I_6}\\
&+(I_1\oplus I_3)\cdot(I_0\oplus\overline{I_4})\cdot(I_0\oplus I_2\oplus I_5)\cdot(I_2\oplus\overline{I_6})\\
&+\overline{I_0}\cdot\overline{I_3}\cdot(I_1\oplus I_5)\cdot(I_2\oplus I_6)\\
&+(I_1\oplus I_2)\cdot(I_0\oplus I_4)\cdot\overline{I_5}\cdot(I_3\oplus I_6)\\
&+\overline{I_0}\cdot\overline{I_3}\cdot(I_2\oplus I_4)\cdot(I_1\oplus I_6)\\
&+(I_0\oplus\overline{I_2})\cdot(I_1\oplus I_4)\cdot(I_3\oplus\overline{I_5})\cdot(I_0\oplus I_3\oplus I_6)\\
&+(I_1\oplus\overline{I_2})\cdot(I_1\oplus I_3\oplus I_4)\cdot(I_1\oplus I_3\oplus I_5)\cdot(I_3\oplus\overline{I_6})\\
&+\overline{I_1}\cdot(I_0\oplus I_3)\cdot\overline{I_4}\cdot(I_2\oplus I_6)\\
&+(I_0\oplus\overline{I_3})\cdot(I_0\oplus I_1\oplus I_4)\cdot(I_2\oplus I_5)\cdot(I_1\oplus\overline{I_6})
\end{aligned}
$$

A.3 $F_7(4)$ expressions

A.3.1 SP form

$$
\begin{aligned}
F_7(4) = &\overline{I_0}\cdot\overline{I_1}\cdot I_3\cdot I_4\cdot I_5\cdot I_6 + I_1\cdot I_2\cdot\overline{I_3}\cdot I_4\cdot I_5\cdot I_6\\
&+\overline{I_0}\cdot I_1\cdot I_2\cdot I_3\cdot\overline{I_4}\cdot I_6 + \overline{I_0}\cdot\overline{I_1}\cdot I_2\cdot I_4\cdot I_5\cdot I_6\\
&+I_0\cdot I_1\cdot\overline{I_2}\cdot\overline{I_3}\cdot I_5\cdot I_6 + I_0\cdot I_1\cdot I_2\cdot I_3\cdot\overline{I_4}\cdot\overline{I_5}\\
&+\overline{I_0}\cdot\overline{I_1}\cdot I_2\cdot I_3\cdot I_5\cdot I_6 + I_0\cdot\overline{I_1}\cdot I_2\cdot\overline{I_3}\cdot I_4\cdot I_5\\
&+I_0\cdot I_1\cdot I_2\cdot\overline{I_3}\cdot\overline{I_4}\cdot I_5 + I_1\cdot I_2\cdot I_3\cdot\overline{I_4}\cdot I_5\cdot\overline{I_6}\\
&+\overline{I_0}\cdot I_2\cdot I_3\cdot I_4\cdot\overline{I_5}\cdot I_6 + I_0\cdot I_1\cdot\overline{I_2}\cdot\overline{I_3}\cdot I_4\cdot I_6\\
&+I_0\cdot\overline{I_1}\cdot I_2\cdot I_3\cdot\overline{I_4}\cdot I_6 + \overline{I_0}\cdot I_2\cdot I_3\cdot I_4\cdot I_5\cdot\overline{I_6}\\
&+\overline{I_1}\cdot I_2\cdot I_3\cdot I_4\cdot I_5\cdot\overline{I_6} + \overline{I_0}\cdot I_1\cdot\overline{I_2}\cdot I_4\cdot I_5\cdot I_6\\
&+I_1\cdot I_2\cdot I_3\cdot I_4\cdot\overline{I_5}\cdot\overline{I_6} + I_0\cdot\overline{I_1}\cdot\overline{I_2}\cdot I_4\cdot I_5\cdot I_6\\
&+\overline{I_0}\cdot I_1\cdot\overline{I_2}\cdot I_3\cdot I_5\cdot I_6 + I_0\cdot I_1\cdot\overline{I_2}\cdot I_3\cdot\overline{I_4}\cdot I_5\\
&+I_0\cdot I_1\cdot I_2\cdot\overline{I_3}\cdot\overline{I_4}\cdot I_6 + \overline{I_0}\cdot I_1\cdot\overline{I_2}\cdot I_3\cdot I_4\cdot I_6\\
&+I_0\cdot\overline{I_1}\cdot\overline{I_2}\cdot I_3\cdot I_5\cdot I_6 + I_0\cdot\overline{I_1}\cdot I_2\cdot I_3\cdot\overline{I_4}\cdot I_5\\
&+I_0\cdot I_1\cdot\overline{I_2}\cdot\overline{I_3}\cdot I_4\cdot I_5 + I_0\cdot\overline{I_1}\cdot\overline{I_2}\cdot I_3\cdot I_4\cdot I_5\\
&+I_1\cdot\overline{I_2}\cdot I_3\cdot I_4\cdot I_5\cdot\overline{I_6} + \overline{I_0}\cdot I_1\cdot I_2\cdot\overline{I_3}\cdot I_5\cdot I_6\\
&+I_0\cdot I_1\cdot I_2\cdot\overline{I_3}\cdot I_4\cdot\overline{I_5} + I_0\cdot\overline{I_1}\cdot I_2\cdot I_3\cdot I_4\cdot\overline{I_5}\\
&+\overline{I_0}\cdot I_1\cdot I_2\cdot\overline{I_3}\cdot I_4\cdot I_6 + I_0\cdot\overline{I_1}\cdot I_2\cdot\overline{I_3}\cdot I_5\cdot I_6\\
&+I_0\cdot I_1\cdot\overline{I_2}\cdot I_3\cdot I_4\cdot\overline{I_5} + I_0\cdot I_1\cdot\overline{I_2}\cdot I_3\cdot\overline{I_4}\cdot I_6\\
&+I_0\cdot\overline{I_1}\cdot I_2\cdot\overline{I_3}\cdot I_4\cdot I_6
\end{aligned}
$$

A.3.2 SPP form

$$
\begin{aligned}
F_7(4) = &(I_2\oplus I_3)\cdot(I_1\oplus I_4)\cdot I_5\cdot I_6\\
&+(I_0\oplus I_3)\cdot I_4\cdot I_5\cdot(I_2\oplus I_6)\\
&+(I_1\oplus I_3)\cdot(I_0\oplus I_4)\cdot I_5\cdot I_6\\
&+I_0\cdot I_2\cdot(I_4\oplus I_5)\cdot(I_1\oplus I_6)\\
&+I_2\cdot(I_0\oplus I_3)\cdot(I_1\oplus I_5)\cdot(I_4\oplus I_6)\\
&+I_1\cdot I_3\cdot(I_2\oplus I_4)\cdot(I_0\oplus I_5)\\
&+(I_0\oplus I_1)\cdot I_3\cdot I_4\cdot(I_2\oplus I_5)\\
&+I_2\cdot(I_1\oplus I_3)\cdot I_4\cdot(I_5\oplus I_6)\\
&+I_0\cdot(I_1\oplus I_2)\cdot I_3\cdot(I_5\oplus I_6)\\
&+I_1\cdot(I_0\oplus I_3)\cdot I_4\cdot(I_5\oplus I_6)
\end{aligned}
$$

References

[BSS95] K. C. Bickerstaff, M. J. Schulte and E. E. Swartzlander Jr. Parallel reduced area multipliers. *Journal of VLSI signal processing* 9 (1995) 181-191.

[CLR90] T. H. Cormen, C.E. Leiserson and R.L. Rivest. *Introduction to algorithms*. MIT Press, 1990.

[D65] L. Dadda. Some schemes for parallel multipliers. *Alta Frequenza* 34 (1965) 349-356.

[D76] L. Dadda. On parallel digital multipliers. *Alta Frequenza* 45 (1976) 574-580.

[EGP56] G. Estrin, B. Gilchrist and J.H. Pomerene. A note on high-speed digital multiplication. *IRE Transactions on electronic computers* 5, 3 (1956) 140.

[H79] Kai Hwang. *Computer arithmetic: principles, architecture and design*. John Wiley & Sons, 1979.

[LP97] F. Luccio and L. Pagli. Normal matrices, pseudo-cubes and pseudo-products. *Congressus Numerantium* 127 (1997) 33-35.

[LP99] F. Luccio and L. Pagli. On a new Boolean Function with applications. *IEEE transactions on computers* 48, 3 (1999)

[O63] Yu Ofman. On the algorithmic complexity of discrete functions. *Soviet physics doklady* 7, 7 (1963) 589-591. English translation.

[W64] C. S. Wallace. A suggestion for fast multiplier. *IEEE transcriptions on electronic computers* EC-13, 1 (1964) 14-17.

[WJM95] Z. Wang, G. A. Jullien and W. C. Miller. A new design technique for column compression multipliers. *IEEE Transactions on computers 44, 8 (1995) 962-970.*