Two Operand Binary Adders with Threshold Logic

José Fernández Ramos and Alfonso Gago Bohórquez

Abstract—The central topic of this paper is the implementation of binary adders with Threshold Logic using a new methodology that introduces two innovations: the use of the input and output carry of each bit for obtaining all the sum bits and a modification of the classic Carry Lookahead adder technique that allows us to obtain the expressions of the generation and propagation carry in a more appropriate way for Threshold Logic. In this way, it has been possible to systematize the process of design of a binary adder with Threshold Logic relating all its important parameters: number of bits of the operands, depth, size, maximum fan-in, and maximum weight. The results obtained are an improvement on those published to date and are summarized as follows: Depth 2 adder: $s = 2n$, $w_{\text{max}} = 2^s$, $f_{\text{max}} = 2n + 1$. Depth 3 adder: $s = 4n - 2 \left\lfloor \frac{\sqrt{n}}{\log n} \right\rfloor$, $w_{\text{max}} = 2 \left\lfloor \frac{\sqrt{n}}{\log n} \right\rfloor$, $f_{\text{max}} = 2 \left\lfloor \frac{\sqrt{n}}{\log n} \right\rfloor + 1$. Depth $d$ adder (asymptotic behavior): $s = O(n)$, $w_{\text{max}} = O(2^{\sqrt{n}})$, $f_{\text{max}} = O(\sqrt{n})$. If the weights are bounded by $w_{\text{max}}$: $n_{\text{max}} = O(\log^{n-1} w_{\text{max}})$, $d_{\text{min}} = O\left(\frac{\log n}{\log\log w_{\text{max}}}\right)$.

Index Terms—Threshold logic, computer arithmetic, binary adders, logic design, threshold gate, neural networks.

1 INTRODUCTION

A Threshold function is a Boolean function $f(X)$, where $X = (X_1, X_2, \ldots, X_n)$, for which it is possible to find a group of real coefficients $w_i$, called weights, and a real number, such as $T$, called the threshold, that is verified:

\[
f(X) = 1 \iff \sum_{i=1}^{n} w_i X_i \geq T
\]

\[
f(X) = 0 \quad \text{otherwise.}
\]

For any threshold function, there exists an indefinite number of groups of weights and thresholds that verify the previous relationship and it is always possible to find a group of integer values for the weights as well as for the threshold [1]. The vector formed by the group of weights $w_1, w_2, \ldots, w_n$ and the threshold $T$, whose absolute values are the minimum, is called the basic structure of the threshold function and is presented as follows:

\[
[f(X_1, X_2, \ldots, X_n)] = [w_1, w_2, \ldots, w_n; T].
\]

An electronic circuit that implements a threshold function is a threshold gate. A circuit formed by threshold gates is a threshold circuit. A threshold circuit with no feedback is a feedforward threshold circuit. In this paper, only this type of circuits will be considered. The size ($s$) of a threshold circuit is defined to be the total number of threshold gates that it includes. The depth of a gate is the longest path from the inputs to that gate. The circuit can be arranged in layers so that all the gates that have the same depth belong to the same layer. The depth of the threshold circuit ($d$) is the maximum value of the depth of their gates.

The threshold gates forms a complete base that allows any Boolean function to be carried out [1]. It has been well known for a long time [2], [3] that the use of Threshold Logic for carrying out symmetrical functions and, in general, arithmetic functions allows a certain reduction in the size and depth of the circuits. This fact, together with the great importance of the addition operation in digital systems, regularly causes new developments of binary adders based on Threshold Logic.

Although it is well-known that the sum can be carried out with a threshold circuit with depth 2 and a size bounded by an $n$-degree polynomial ( $n$ is the number of bits of the operands), but with exponential weights [4], the optimized designs have been obtained for circuits with depth 3. Among these designs it is necessary to highlight those from Siu et al. [5] and, fundamentally, those of Vassiliadis et al. [6]. In the latter work, two designs for threshold adders with depth 3 are presented. The first of them (PW) has a size of $O\left(\frac{n^2}{\log n}\right)$ and polynomially bounded weights. The second adder (EW) has a size of $O(n) 2^{\left\lfloor \frac{n}{\sqrt{n}} \right\rfloor}$ (optimal complexity $O(n)$), a maximum fan-in of $2\left\lfloor \frac{n}{\sqrt{n}} \right\rfloor + 3$, and a maximum weight of $2\sqrt{n}$. The fact that the weights grow exponentially with $\sqrt{n}$ is the main inconvenience of the EW adder. In the paper mentioned, the problem of whether it is possible to design an adder with size $O(n)$ and polynomial bounded weights is left unsolved.
In this paper, a general methodology for the realization of two operand binary adders is presented. This methodology is not restricted to a concrete value of depth. Size and depth are related in such a way that an increase in depth brings a decrease in size and vice versa.

This paper is organized as follows: In Section 2, a technique that consists of using double carry to obtain the sum bits and its application in the construction of an adder with depth 2 is described. In Section 3, we combine the previous method with the Carry Lookahead method to design an adder with depth 3. Section 4 deals with the suitability of representing the size of a threshold circuit for the total number of gates (parameter s), so we introduce two new parameters that allow us to specify, with much greater precision, the real size of a threshold circuit. In Section 5, a new adder with depth 3 and size optimized is presented. This is based on the coalition of the two adders previously presented. In Section 6, a modification of the Carry Lookahead method that allows the extension from these types of adders to any depth value is presented. The trade-offs between depth and the different parameters that indicate the size of the circuit are indicated. Finally, in Section 7, the contributions of this paper are presented.

2 DOUBLE CARRY THRESHOLD ADDER WITH DEPTH 2 (DCTA2)

The i-bit Si of the sum of a two operands binary adder is given by the expression:

\[ S_i = X_i \oplus Y_i \oplus C_{i-1}, \]  

where \( X_i \) and \( Y_i \) are the i-bits of each operand and \( C_{i-1} \) is the carry output of the previous bits. This operation is a symmetrical function, so to implement it with Threshold Logic, a threshold circuit with depth 2 as minimum is necessary [2]. In several types of parallel adders (Carry Save Adders, Carry Lookahead Adders), the carries of all the bits are calculated simultaneously. Afterward, the sum of each pair of bits \( X_i \) and \( Y_i \) are computed by only using the output carry corresponding to the previous pair \( (C_{i-1}) \), just as indicated in (1). Nevertheless, the fact of having all the carry in advance means that the output carry \( C_i \) can be used besides the \( C_{i-1} \) in the computation of the sum bit \( S_i \).

Although it is not necessary to use this carry in obtaining the sum bits, we will prove that this simplifies this operation to a great extent. The carry \( C_i \) is a function of the variables \( X_i, Y_i \), and \( C_{i-1} \), so if we get the \( S_i \) bit as a function of these three variables and \( C_i \), it is impossible for several input patterns to appear. In Fig. 1, the Karnaugh diagram of \( S_i \) is represented as a function of these four variables. The impossible input patterns have been represented by the symbol \( \Phi \). Since these combinations will never be given, in each one of them we can associate to \( S_i \) the value that we consider more convenient. In this case, the shaded boxes have been taken as \( S_i = 1 \) and those unshaded as \( S_i = 0 \). The reason for making this selection is that we are looking for a threshold function so they can be implemented in only one layer. The resulting function is the following:

\[ f = X_i Y_i C_{i-1} \lor (X_i \lor Y_i) \lor C_{i-1} \overline{C_i}. \]

This is a threshold function with the structure \([f(X_i, Y_i, C_{i-1}, C_i)] = [1, 1, 1, -2; 1]\). Therefore, the fact of using the carry input and carry output of the i-bit allows the i-bit of the sum to be computed in only one layer. To complete the design of an adder using this method it is necessary to previously calculate all the carries in parallel.

The carry of the i-bit is given by the expression:

\[ C_i = 1 \Leftrightarrow C_{i-1} + \sum_{j=0}^{i} 2^j X_j + 2^j Y_j \geq 2^{i+1} \]

\[ C_i = 0 \quad \text{otherwise}, \]

where \( C_{i-1} \) is the carry input to the adder.

This expression corresponds to a threshold function with the structure:

\[ [C_i(C_{i-1}, X_0, Y_0, \ldots, X_{i-1}, Y_i)] = [1, 1, 1, \ldots, 2^{i-1}, 2^i, 2^i, 2^i]. \]

In this way, a two-operands binary adder of length \( n \) can be carried out with a threshold circuit of depth 2 and size \( 2n \).

In the first layer, all the carries are computed by means of \( n \) threshold gates whose structure is given by the expression:

\[ [C_i(C_{i-1}, X_0, Y_0, \ldots, X_{i-1}, Y_i)] = [1, 1, 1, \ldots, 2^{i-1}, 2^i, 2^i, 2^i]; \quad i \in [0, n - 1]. \]

In the second layer, all the sum bits are computed by means of \( n \) threshold gates, all with the same following structure:

\[ [S_i(X_i, Y_i), C_{i-1}, C_i)] = [1, 1, -2; 1]; \quad i \in [0, n - 1]. \]

We call this new structure “Double Carry Threshold Adder with Depth 2” (DCTA2). Its characteristics are:

- **Depth**: \( d = 2 \)
- **Size**: \( s = 2n \)
- **Maximum fan-in**: \( f_{\max} = 2n + 1 \)
- **Maximum weight**: \( w_{\max} = 2^n \)
The size of this circuit thoroughly improves that of the threshold adder with depth 2 proposed by Vassiliadis et al. [6], that is:

\[ s = 5n + 2 \left\lceil \frac{n}{\sqrt{n}} \right\rceil. \]

### 3 Double Carry Threshold Adder with Depth 3 (DCTA3)

In the previous section, a threshold circuit with depth 2 for the sum of two binary numbers of \( n \) bits has been described. Its size parameters and maximum fan-in are optimal because they are linear functions in \( n \). Nevertheless, the main inconvenience of the previous design rests in the fact that the value of the weights increases exponentially with \( n \). Keeping in mind that the real size of a threshold gate directly depends on the value of its weights [7], [8], [9], it is logical to think that although the number of gates is linearly bounded, the real size of the circuit can be exponentially bounded. The most appropriate way to reduce the value of the weights of the gates has already been proven in other papers [4] and consists of increasing the depth of the circuit. The problem is the way that this increase in depth has to be carried out to optimize the value of the weights. Solutions to this problem have already been given in the above-mentioned Siu et al. [5] and Vassiliadis et al. [6], where constructions of threshold adders with depth 3 are described.

The technique that we propose for the realization of an adder with depth 3 consists of the combined use of the Double Carry method together with the Carry Lookahead method. Although not mentioned, this classic method has already been used by Vassiliadis et al. [6] on the PW and EW adders since the parameters “carry force” \((\alpha)\) and “carry preserve” \((\beta)\) that are introduced in this paper respectively coincide with the group-generate \((G_i)\) and group-propagate \((P_i)\) signals used in the Carry Lookahead Adder.

Considering that the operands \( X \) and \( Y \) are divided in groups of \( k \) bits each, the carry generation and propagation signals of each group are defined as follows:

**Carry generation:**

\[ G_i = 1 \iff \sum_{j=0}^{k-1} 2^j X_j + 2^j Y_j \geq 2^k \]

\[ G_i = 0 \text{ otherwise.} \]

**Carry propagation:**

\[ P_i = 1 \iff \sum_{j=0}^{k-1} 2^j X_j + 2^j Y_j = 2^k - 1 \]

\[ P_i = 0 \iff \sum_{j=0}^{k-1} 2^j X_j + 2^j Y_j < 2^k - 1 \]

\[ P_i \text{ is arbitrary } \iff \sum_{j=0}^{k-1} 2^j X_j + 2^j Y_j \geq 2^k. \]

It is very important to note that, later on in this development, the combination \( G_i = 1, P_i = 0 \) is impossible. The output carry of the bits group is given by the expression:

\[ C_i = G_i \lor P_i C_{i-1}, \]

where \( C_{i-1} \) is the carry input to the bits group. This is a threshold function with the structure:

\[ [C_i(C_{i-1}, P_i, G_i)] = [1, 1, 2; 2]. \]

Nevertheless, since the combination \( G_i = 1, P_i = 0 \) cannot be given, the values of \( C_i \) for these inputs can be assigned in the way that we want. If they are assigned as appears in Fig. 2, in which the shaded boxes have been assigned to 1 and the other boxes have been assigned to 0, the resulting function is also threshold, but with a different and much more interesting structure since it is symmetrical (the weights of all the inputs are the same):

\[ C_i = G_i P_i \lor G_i C_{i-1} \lor P_i C_{i-1}, \]

\[ [C_i(C_{i-1}, P_i, G_i)] = [1, 1, 1; 2]. \]

The output carry of the following group, \( C_{i+1} \), can be obtained in exactly the same way:

\[ C_{i+1} = G_{i+1} \lor P_{i+1} C_i = G_{i+1} \lor P_{i+1} G_i \lor P_{i+1} P_i C_{i-1}. \]

Once again, keeping in mind that the combinations \( G_i = 1, P_i = 0, \) and \( G_i = 1, P_i = 0 \) are impossible, and making an appropriate assignment of the values corresponding to these combinations, it is deduced that the carry \( C_{i+1} \) can be obtained from a threshold function with the following structure:

\[ [C_{i+1}(C_{i-1}, P_i, G_i, P_i C_{i-1} G_{i+1})] = [1, 1, 1, 2; 4]. \]

The same process can be extended to the rest of the groups, so all the carries of each group can be obtained from the initial carry \( C_{in} \) and the generation and propagation functions by means of threshold gates with the following structure:
[C_i(C_{in}, P_0, G_0, P_1, G_1, \ldots P_i, G_i)] = [1, 1, 1, 2, 2, \ldots 2^i, 2^i; 2^{i+1}].

It is interesting to note that these structures coincide exactly with the structures of the gates that compute the carries as a function of the input variables.

The adder with depth 3 is based on the previous development. First of all, the operands are divided into \( \lceil \sqrt{n} \rceil \) groups of \( \left\lceil \frac{n}{\sqrt{n}} \right\rceil \) bits each, but only the last one can have a minor number of bits. The bits of the operands will be designated with two subscripts: The first one indicates the group to which it belongs and the second the order number that it occupies in this group:

\[ X_{ij}, Y_{ij}, i \in [0, \lceil \sqrt{n} \rceil - 1], j \in \left[0, \left\lceil \frac{n}{\sqrt{n}} \right\rceil - 1 \right]. \]

In the first layer, the carry-propagation and carry-generation functions are obtained for each bit \( j \) of each group \( i \), by means of threshold gates whose structure is:

\[ [G_{ij}(X_{i0}, Y_{i0}, \ldots X_{ij}, Y_{ij})] = [1, 1, \ldots 2^i, 2^i; 2^{i+1}] \]
\[ [P_{ij}(X_{i0}, Y_{i0}, \ldots X_{ij}, Y_{ij})] = [1, 1, \ldots 2^i, 2^i; 2^{i+1} - 1]. \]

The total number of gates in this layer is \( 2n \) and the maximum weight of the gates is

\[ 2 \left\lceil \frac{n}{\sqrt{n}} \right\rceil. \]

In the second layer, all the carries are computed as a function of \( G_{ij}, P_{ij} \), and the carry input \( C_{in} \). These carries will also be designated with double subscript \( C_{ij} \). All the carries of the bits that belong to one group are computed by gates with an identical structure:

First group (\( i = 0 \)): \([C_{0j}(C_{in}, G_{0j}, P_{0j})] = [1, 1, 1; 2] \)

Second group (\( i = 1 \)): \([C_{1j}(C_{in}, G_{1k}, P_{1k}, G_{ij}, P_{ij})] = [1, 1, 1, 2, 2; 4] \)

\( i \)-group:

\[ [C_{ij}(C_{in}, \ldots G_{i-1,k}, P_{i-1,k}, G_{ij}, P_{ij})] = [1, \ldots 2^{i-1}, 2^{i-1}, 2^i, 2^i; 2^{i+1}], \]

where

\[ k = \left\lfloor \frac{n}{\sqrt{n}} \right\rfloor - 1. \]

The total number of gates in this layer is \( n \) and the maximum weight is \( 2\sqrt{n} \) that belong to the gates that compute the carries of the last group.

In the third layer, the sum bits are obtained using the double-carry method by \( n \) threshold gates, all with the same structure:

\[ [S_{ij}(X_{ij}, Y_{ij}, C_{ah}, C_{ij})] = [1, 1, 1, -2; 1], \]

where \( b = j - 1, a = i \) if \( j > 0 \) and \( b = \lceil \sqrt{n} \rceil - 1, a = i - 1 \) if \( j = 0 \).

We call this new adder “Double Carry Threshold Adder with Depth 3” (DCTA3). The parameters that characterize this adder are the following:

\[ d = 3 \]
\[ s = 4n \]
\[ f_{\text{max}} = 2\lceil \sqrt{n} \rceil - 1 \]
\[ w_{\text{max}} = 2\lceil \sqrt{n} \rceil. \]

Although these results have the same complexity as those of the EW adder of Vassilidis et al. [6], their absolute values are considerably inferior to those of this adder. A comparison among the value of these parameters in the DCTA3, EW, and PW for 32- and 64-bit adders is shown in Table 1.

### Table 1

<table>
<thead>
<tr>
<th></th>
<th>32 bit</th>
<th>64 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( d )</td>
<td>( s )</td>
</tr>
<tr>
<td>PW</td>
<td>3</td>
<td>796</td>
</tr>
<tr>
<td>EW</td>
<td>3</td>
<td>204</td>
</tr>
<tr>
<td>DCTA3</td>
<td>3</td>
<td>128</td>
</tr>
</tbody>
</table>

### 4 Size Evaluation of a Threshold Circuit

In Complexity Theory, it is usually considered that the number of gates of a circuit is a valid reference for indicating the real size that it will occupy on a silicon wafer [10]. Nevertheless, to be valid, this assertion implicitly assumes that all the gates that compose the circuit have approximately the same size. This assumption is safe when the circuit is built with the basic gates AND-OR-NOT with a reduced fan-in, but fails when it is a threshold circuit. This is because the size of a threshold gate is directly related as much as to the number of inputs as to value of its weights. Since a threshold circuit can be formed by gates having very different fan-in and weight values, the fact of representing its size with the number of gates means that all the gates are assigned the same size, which is an evident error.

Based on this reasoning, Beame et al. [11] proposed that the size of a threshold circuit is given as a function of the total number of connections (\( c \)). This number can be obtained as the sum of the number of inputs of all the gates in the circuit. It is clear that this parameter is a better
where the size of a threshold circuit can be defined as follows:

\[ w = \sum_{N_{bt}} \]

where the sum is extended to all the gates of the circuit.

The values of these new parameters for the structures DCTA2, DCTA3, and EW [6] are the following:

**DCTA2:**

\[
\begin{align*}
    c &= n^2 + 6n \\
    w &= 4(2^n - 1) + 6n.
\end{align*}
\]

**DCTA3:**

\[
\begin{align*}
    c &= n \left( \left\lceil \frac{n}{\sqrt{n}} \right\rceil + 2 \left\lfloor \frac{n}{\sqrt{n}} \right\rfloor + 8 \right) \\
    w &= 8 \left\lceil \sqrt{n} \right\rceil \left( 2 \left\lfloor \frac{n}{\sqrt{n}} \right\rfloor - 1 \right) + 4 \left\lceil \frac{n}{\sqrt{n}} \right\rceil \left( 2^n - 1 \right) + 2.
\end{align*}
\]

**EW [6]:**

\[
\begin{align*}
    c &= n \left( \left\lceil \sqrt{n} \right\rceil + 2 \left\lfloor \frac{n}{\sqrt{n}} \right\rfloor + 8 \right) \\
    w &= 8 \left\lceil \sqrt{n} \right\rceil \left( 2 \left\lfloor \frac{n}{\sqrt{n}} \right\rfloor - 1 \right) + 4 \left\lceil \frac{n}{\sqrt{n}} \right\rceil \left( 2^n - 1 \right) + 2.
\end{align*}
\]

**Table 2:**

<table>
<thead>
<tr>
<th>Adder Type</th>
<th>( n_{\text{max}} )</th>
<th>( s )</th>
<th>( c )</th>
<th>( w )</th>
</tr>
</thead>
<tbody>
<tr>
<td>EW [6]</td>
<td>( e^2 )</td>
<td>( 6e^2 + 2e )</td>
<td>( 5e^3 + 17e^2 )</td>
<td>( 10e^{2+1} - 4e^2 - 20e )</td>
</tr>
<tr>
<td>DCTA2</td>
<td>( e^2 )</td>
<td>( 4e^2 )</td>
<td>( 3e^3 + 8e^2 )</td>
<td>( 6e^{2+1} + 2e^2 - 12e )</td>
</tr>
<tr>
<td>DCTA3</td>
<td>( e^2 )</td>
<td>( 4e^2 + 2e )</td>
<td>( 3e^3 + 9e^2 + 6e )</td>
<td>( (6e+1)2^{e+1} + 2e^2 - 6e + 4 )</td>
</tr>
</tbody>
</table>

The approach to the real size of the circuit since the differences in the number of inputs are taken into account, but differences in weight values are not considered. This means that two gates that have the same number of inputs, but very different weight values, would have the same influence on the size of the circuit. However, in most of the integrated designs of threshold gates presented in the literature ([7], [9], [12]), the value of the weights of the threshold gates impacts directly on its size, so it is necessary to introduce a new parameter for measuring the size of the circuit that keeps this in mind. Fernández et al. [9] introduced a new parameter for estimating the size of a threshold gate \((N_{bt})\) that takes into account both the number of inputs and the weight values. This parameter is defined as follows:

\[ N_{bt} = 2 \left( \min \left[ \max \left( \sum_i w_i - T, \sum_j (-w_j) + T \right), \max \left( \sum_i w_i + 1 - T, \sum_j (-w_j) - 1 + T \right) \right] \).

where \( w_i \) are the positive weights, \( w_j \) are the negative weights, and \( T \) is the threshold of the function. According to this, a parameter \((w)\) that offers a better estimation of the size of a threshold circuit can be defined as follows:

\[ w = \sum N_{bt}, \]

where the sum is extended to all the gates of the circuit.

To show that these parameters give a better understanding of circuit size, it is enough to check that, for a 16-bit adder, the values for the DCTA2 are \( s = 32, c = 352, w = 262, 236 \) and, for the DCTA3, they are \( s = 64, c = 320, w = 752 \), from which it is deduced that, although the value of the parameter \( s \) is smaller in DCTA2 than in DCTA3, the real size of the latter will be much smaller, as the parameter \( w \) clearly shows.

It is obvious from the foregoing that the weight values of the gates of a threshold circuit have a decisive influence on the real size of the circuit.

Another fundamental aspect to bear in mind in the evaluation of threshold circuit size is that the maximum value of weight that can perform a real threshold gate is bounded by a number \((w_{\text{max}})\) that depends on the available technology. As a consequence of this, from the designer’s point of view, \( w_{\text{max}} \) is a fundamental parameter in the design of threshold circuits. So, it is useful to establish some relationship between the parameters of the threshold circuit and \( w_{\text{max}} \). Keeping in mind that, in the adders that we are considering, the maximum weight is always a power of 2, we will introduce a new parameter that gives us information about this technological limit:

\[ e = \log_2(w_{\text{max}}). \]

In this way, since the maximum weight in a DCTA2 is \( w_{\text{max}} = 2^n \), it is verified that \( n_{\text{max}} = e \), that is, that the maximum number of bits of a DCTA2 is bounded by the value of parameter \( e \) of the available technology. In the same way, the maximum weight of the DCTA3 and EW adders is \( w_{\text{max}} = 2^{\sqrt{n}} \), that is, \( n_{\text{max}} = 2^e \), so, with the same gate technology, we can design a DCTA3 or EW adder with a number of bits similar to the square of that with which a DCTA2 could be built.

From this point of view, it is interesting to indicate the parameters that specify the size and maximum number of bits of the adders as a function of this parameter. These parameters are shown in Table 2 for several types of threshold adders.
It is possible to improve the DCTA3 if one remembers that the carry input ($C_{in}$) is only used in the second and third layers and is unnecessary in the first layer. If we consider that the output carry of the DCTA2 is obtained in the first layer, we can conclude that a new depth 3 adder can be built with the interconnection of a DCTA2 and a DCTA3 in the following way:

We take the first $\left\lfloor \frac{n}{\sqrt{n}} \right\rfloor$ bits of the operands and the carry input and we make a DCTA2. In the first layer of this adder, all the carries, including the output carry $C_{out}$, are obtained and, in the second layer, the $\left\lfloor \frac{n}{\sqrt{n}} \right\rfloor$ first bits of the sum are computed. With the remaining $n - \left\lfloor \frac{n}{\sqrt{n}} \right\rfloor$ input bits, we make a DCTA3. In the first layer, functions $G_{ij}$ and $P_{ij}$ are computed, the carries are obtained in the second layer using the carry output of the DCTA2 ($C_{out}$) as the carry input, and the remaining sum bits are computed in the third layer. This new adder is called Optimized DCTA3 or DCTA3$^\dagger$.

The parameters of this adder are obtained from the parameters of DCTA2 and DCTA3:

$$w_{max} = 2 \left\lfloor \frac{n}{\sqrt{n}} \right\rfloor$$

$$f_{max} = 2 \left\lfloor \frac{n}{\sqrt{n}} \right\rfloor + 1$$

$$s = 4n - 2 \left\lfloor \frac{n}{\sqrt{n}} \right\rfloor$$

$$c = \left\lfloor \frac{n}{\sqrt{n}} \right\rfloor 6 + \left\lfloor \frac{n}{\sqrt{n}} \right\rfloor$$

$$+ n' \left( \left\lfloor \frac{n}{\sqrt{n}} \right\rfloor + 2 \left\lfloor \frac{n'}{\sqrt{n'}} \right\rfloor + 8 \right)$$

$$w = 4 \left( 2 \left\lfloor \frac{n}{\sqrt{n}} \right\rfloor - 1 \right) + 6 \left\lfloor \frac{n}{\sqrt{n}} \right\rfloor + 2n'$$

$$+ 8 \left\lfloor \sqrt{n'} \right\rfloor 2 \left\lfloor \frac{n'}{\sqrt{n'}} \right\rfloor - 1 + 4 \left\lfloor \frac{n'}{\sqrt{n'}} \right\rfloor \left( 2 \left\lfloor \sqrt{n'} \right\rfloor - 1 \right),$$

where

$$n' = n - \left\lfloor \frac{n}{\sqrt{n}} \right\rfloor.$$
6 Extension to Adders with a Greater Number of Layers

The biggest problem presented when building threshold adders with a high number of bits consists of the difficulty of implementing threshold gates with a high fan-in and high weight values. Indeed, one of the optimum developments for threshold gates, presented by Hidalgo et al. [13], allows the implementation of threshold gates with weight values bounded by $w t_{\text{max}} \leq 40$. This assumes that, for these gates, $e = 5$, which means that the maximum number of bits of a DCTA$^3$ carried out with these gates would be bounded by $n_{\text{max}} = 30$, a value which is insufficient in a great number of applications.

Two solutions can be given to solve this problem: to implement threshold gates that can perform higher weight values or to modify the structure of the threshold adders so that they can have a larger bit number with the same value of $e$. In this section, we investigate the second solution, increasing the depth of the circuit to obtain adders with a greater number of bits.

| TABLE 4 |
| Structures of the Gates of a 16-bit DCTA$^3$ |

| First Layer |
| [ $C_{00}$ ( $C_{00}$, $X_{00}$, $Y_{00}$ ) ] = [ 1, 1, 1, 2 ] |
| [ $C_{01}$ ( $C_{00}$, $X_{00}$, $X_{01}$, $Y_{01}$ ) ] = [ 1, 1, 1, 2, 2; 4 ] |
| [ $C_{02}$ ( $C_{00}$, $X_{00}$, $X_{02}$, $Y_{02}$ ) ] = [ 1, 1, 1, 2, 4, 8 ] |
| [ $C_{03}$ ( $C_{00}$, $X_{00}$, $X_{03}$, $Y_{03}$ ) ] = [ 1, 1, 1, 2, 4, 8, 8; 16 ] |
| Group $i$ (1 ≤ $i$ ≤ 4) |
| [ $G_{0i}$ ( $X_{0i}$, $Y_{0i}$ ) ] = [ 1, 1, 2 ] |
| [ $P_{0i}$ ( $X_{0i}$, $Y_{0i}$ ) ] = [ 1, 1; 1 ] |
| [ $G_{1i}$ ( $X_{0i}$, $Y_{0i}$, $X_{1i}$, $Y_{1i}$ ) ] = [ 1, 1, 2, 2; 4 ] |
| [ $P_{1i}$ ( $X_{0i}$, $X_{1i}$, $Y_{0i}$, $Y_{1i}$ ) ] = [ 1, 1, 2, 2; 3 ] |
| [ $G_{2i}$ ( $X_{0i}$, $Y_{0i}$, $X_{2i}$, $Y_{2i}$ ) ] = [ 1, 1, 2, 2, 4, 8 ] |
| [ $P_{2i}$ ( $X_{0i}$, $X_{2i}$, $X_{1i}$, $Y_{2i}$ ) ] = [ 1, 1, 2, 2, 4, 7 ] |
| Second Layer |
| [ $S_{00}$ ( $X_{00}$, $Y_{00}$, $C_{00}$, $C_{00}$ ) ] = [ 1, 1, 1, -2; 1 ] |
| [ $S_{01}$ ( $X_{01}$, $Y_{01}$, $C_{00}$, $C_{01}$ ) ] = [ 1, 1, 1, -2; 1 ] |
| [ $S_{02}$ ( $X_{02}$, $Y_{02}$, $C_{01}$, $C_{02}$ ) ] = [ 1, 1, 1, -2; 1 ] |
| [ $S_{03}$ ( $X_{03}$, $Y_{03}$, $C_{02}$, $C_{03}$ ) ] = [ 1, 1, 1, -2; 1 ] |
| Group $i$ |
| [ $C_{ij}$ ( $C_{0j}$, $G_{ij}$, $P_{ij}$ ) ] = [ 1, 1, 1, 2 ] $\forall 0 \leq j \leq 2$ |
| Group 2 |
| [ $C_{2j}$ ( $C_{0j}$, $G_{1j}$, $P_{1j}$, $G_{2j}$, $P_{2j}$ ) ] = [ 1, 1, 1, 2, 2; 4 ] $\forall 0 \leq j \leq 2$ |
| Group 3 |
| [ $C_{3j}$ ( $C_{0j}$, $G_{1j}$, $P_{1j}$, $G_{2j}$, $P_{2j}$, $G_{3j}$, $P_{3j}$ ) ] = [ 1, 1, 1, 2, 2, 4, 4; 8 ] $\forall 0 \leq j \leq 2$ |
| Group 4 |
| [ $C_{4j}$ ( $C_{0j}$, $G_{1j}$, $P_{1j}$, $G_{2j}$, $P_{2j}$, $G_{3j}$, $P_{3j}$, $G_{4j}$, $P_{4j}$ ) ] = [ 1, 1, 1, 2, 2, 2, 4, 4, 8; 16 ] $\forall 0 \leq j \leq 2$ |
| $C_{\text{out}} = C_{42}$ |
| Third Layer |
| Group 1 |
| [ $S_{10}$ ( $X_{10}$, $Y_{10}$, $C_{00}$, $C_{10}$ ) ] = [ 1, 1, 1, -2; 1 ] |
| [ $S_{11}$ ( $X_{11}$, $Y_{11}$, $C_{01}$, $C_{11}$ ) ] = [ 1, 1, 1, -2; 1 ] |
| [ $S_{12}$ ( $X_{12}$, $Y_{12}$, $C_{11}$, $C_{12}$ ) ] = [ 1, 1, 1, -2; 1 ] |
| Group $i$ (2 ≤ $i$ ≤ 4) |
| [ $S_{0i}$ ( $X_{0i}$, $Y_{0i}$, $C_{1i}$, $C_{0i}$ ) ] = [ 1, 1, 1, -2; 1 ] |
| [ $S_{1i}$ ( $X_{1i}$, $Y_{1i}$, $C_{0i}$, $C_{1i}$ ) ] = [ 1, 1, 1, -2; 1 ] |
| [ $S_{2i}$ ( $X_{2i}$, $Y_{2i}$, $C_{1i}$, $C_{2i}$ ) ] = [ 1, 1, 1, -2; 1 ] |
To pass from the DCTA2 to the DCTA3, the number of bits of the adder was divided into groups, the group-generation and group-propagation functions of each group were obtained and, from these, the carries of all the bits were computed. The method to increase the depth beyond three layers is similar to the previous one, that is, to carry out new divisions into groups. In this way, for example, to construct a DCTA4, a first division is made into \( \lceil n/3 \rceil \) groups (blocks) of \( \lceil \sqrt{n} \rceil \) bits each. Each one of these blocks is divided again into
bits each. In the first layer, the block-generation and block-propagation functions are obtained by taking the bits of the operands as input variables. Now, we designate the bits of the operands with three subscripts: \( h \) indicates the block number to which belongs, \( i \) indicates the group number inside its block, and \( j \) indicates the bit number inside its group:

\[
\begin{align*}
G_{hij}(...X_{hij}, Y_{hij}) &= \lceil \frac{n}{\sqrt{m}} \rceil \quad \text{[...}, 2^j, 2^j, 2^j+1] \\
P_{hij}(...X_{hij}, Y_{hij}) &= \lceil \frac{n}{\sqrt{m}} \rceil - 1 \quad \text{[...}, 2^j, 2^j, 2^j+1 - 1].
\end{align*}
\]

In the second layer, the group-generation and group-propagation functions are obtained, taking the block-generation and block-propagation functions as input variables. In the classic method of the Carry Lookahead, these functions are obtained from the following expressions:

\[
\begin{align*}
\Gamma_{hij}^2 &= G_{hij}^1 \lor P_{hij}^1 G_{h,i-1,k-1}^1 \lor P_{hij}^1 P_{h,i-1,k-1}^1 G_{h,i-2,k-1}^1 \ldots \\
\Pi_{hij}^2 &= P_{hij}^1 P_{h,i-1,k-1}^1 P_{h,i-2,k-1}^1 \ldots P_{h,i,k-1}^1 P_{h,0,k-1}^1,
\end{align*}
\]

where

\[
k = \left\lfloor \sqrt{\frac{n}{\sqrt{m}}} \right\rfloor.
\]

The carries can be obtained from the carry input to each block according to the expression:

\[
C_{hij} = \Gamma_{hij}^2 \lor \Pi_{hij}^2 C_{h,i-1,j}.
\]

Nevertheless, this classic extension of the Carry Lookahead method is not valid in combination with the double-carry method since, in this case, the combination \( \Pi_{hij}^2 = 0, \Gamma_{hij}^2 = 1 \) is possible, from which it is deduced that:

\[
\left[ C_{hij}(C_{h,i-1,j}, \Gamma_{hij}^2, \Pi_{hij}^2) \right] \neq [1, 1, 1; 2].
\]

A new extension method is necessary for the carry-generation and carry-propagation functions that guarantees
that the combination $P^2(G^1, P^1) = 0, G^2(G^1, P^1) = 1$ is impossible. The following theorem shows the way to obtain these new functions.

**Theorem 1.** Let the following threshold functions be:

$f(X_0, X_1, \ldots, X_n) = [1, w_1 \ldots w_n; T]$

$f_1(X_1, \ldots, X_n) = [w_1 \ldots w_n; T - 1]$

$f_2(X_1, \ldots, X_n) = [w_1 \ldots w_n; T]$

$g(X_0, f_1, f_2) = [1, 1, 1; 2].$

It is verified that the functions $f$ and $g$ are identical.

**Proof.** From the definition of $f_1$ and $f_2$, it is deduced that the combination $f_1 = 0$ and $f_2 = 1$ is impossible for any value of the inputs. So, to prove the theorem, it is sufficient to demonstrate that $f = g$ in the three remaining combinations of $f_1$ and $f_2$.

1. $f_1 = 0, f_2 = 0$

$$\sum_{i=1}^{n} w_i X_i \leq T - 2 \Rightarrow 1 + \sum_{i=1}^{n} w_i X_i \leq T - 1 \Rightarrow$$

$$X_0 + \sum_{i=1}^{n} w_i X_i \leq T - 2 \Rightarrow f_1 = 0$$

$$X_0 + f_1 + f_2 = X_0 \Rightarrow g = 0.$$

2. $f_1 = 1, f_2 = 0$

$$f_1 = 1, f_2 = 0 \Rightarrow \sum_{i=1}^{n} w_i X_i = T - 1$$

If $X_0 = 1 \Rightarrow X_0 + \sum_{i=1}^{n} w_i X_i = T \Rightarrow f = 1;$

$$X_0 + f_1 + f_2 = 2 \Rightarrow g = 1$$

If $X_0 = 0 \Rightarrow X_0 + \sum_{i=1}^{n} w_i X_i = T - 1 \Rightarrow f = 0;$

$$X_0 + f_1 + f_2 = 1 \Rightarrow g = 0.$$

3. $f_1 = 1, f_2 = 1$

$$\sum_{i=1}^{n} w_i X_i \geq T \Rightarrow X_0 + \sum_{i=1}^{n} w_i X_i \geq T \Rightarrow f = 1$$

$$X_0 + f_1 + f_2 = X_0 + 2 \geq 2 \Rightarrow g = 1.$$

From Theorem 1, the group-generation and group-propagation functions of the second layer can be defined as follows:
where $a = 0, 1, 2$ and $b = 0 \ldots h$

\[
b = \left\lfloor \sqrt{\frac{n}{\sqrt{n}}} \right\rfloor - 1
\]

if $a < i$ and $b = j$ if $a = i$.

It is verified that

\[
C_{hij}(C_{h,i-1,j}, P_{hij}^2, G_{hij}^2) = C_{hij}(C_{h,i-1,j}, \ldots, P_{hab}^1, G_{hab}^1)
\]

with

\[
[C_{hij}(C_{h,i-1,j}, P_{hij}^2, G_{hij}^2)] = [1, 1, 1; 2]
\]

and

\[
[C_{hij}(C_{h,i-1,j} \ldots P_{hab}^1, G_{hab}^1)] = [1, 2; 2^2, 2^2]
\]

and the combination $P_{hij}^2 = 0, G_{hij}^2 = 1$ being impossible.

Therefore, in the third layer, all the carries are computed by threshold gates with the following structure:

\[
S_{hij}(X_{hij}, Y_{hij}, C_{hij}^{-1}, C_{hij}) = [1, 1, -2; 1],
\]

where

\[
C_{hij}(C_{i0}, P_{in}, G_{in}) = [1, \ldots, 2^h, 2^h, 2^h+1, 2^h+1],
\]

where $a = 1$ if $(i = 0$ and $b = h)$ and $a = 2$ otherwise

$b = 0 \ldots h$

\[
c = \left\lfloor \sqrt{\frac{n}{\sqrt{n}}} \right\rfloor - 1 \quad \forall b < h
\]

and

\[
d = \left\lfloor \sqrt{\frac{n}{\sqrt{n}}} \right\rfloor - 1 \quad \forall b < h
\]

$c = i$ and $d = j$ if $b = h$.

In the fourth layer, the $n$ sum bits are computed by $n$ threshold gates whose structure is:

\[
S_{hij}(X_{hij}, Y_{hij}, C_{hij}^{-1}, C_{hij}) = [1, 1, -2; 1],
\]
The basic parameters of this new adder are:

The basic parameters of this new adder are:

The basic parameters of this new adder are:

The basic parameters of this new adder are:

As already indicated, the expression of the characteristics of the DCTA4 is simpler as a function of the technological parameter $e$:

Keeping in mind that the input carry is only necessary for the third and fourth layers of the DCTA4 and that the
output carry of the DCTA3+ is obtained in the second layer, a new adder can be built, which we call DCTA4+, by the interconnection of a DCTA3+ and a DCTA4 so that the output carry of the DCTA3+ is connected to the input carry of the DCTA4. The parameters of the DCTA4+ are the following:

\[ \begin{align*}
  n_{\text{max}} &= e^3 + e^2 + e \\
  w_{\text{max}} &= 2^e \\
  f_{\text{max}} &= 2e + 1 \\
  s &= 6e^3 + 2e^2 + 2e \\
  c &= 5e^3 + 13e^2 + 5e^2 + 6e \\
  w &= (5e^3 + 3e + 1)2^{e+2} - 2e^3 - 22e^2 - 6e - 4.
\end{align*} \]

The method to increase the depth 3 threshold adder to the depth 4 threshold adder can be extended to any number of layers \(d\). The characteristics of the DCTA\(d\) and DCTA\(d^+\) adders \((\forall d \geq 3)\) as a function of the parameter \(e\) are the following:

1. DCTA\(d\):

\[ \begin{align*}
  n_{\text{max}} &= e^{d-1} \\
  w_{\text{max}} &= 2^e \\
  f_{\text{max}} &= 2e + 1 \\
  s &= 2e^{d-1}(d - 1) - 2e^{d-2}(d - 3) \\
  c &= (2d - 3)e^d + 2(d + 1)e^{d-1} - 4(d - 3)e^{d-2} \\
  w &= (2d - 3)2^{e+2}e^{d-2} - 12(d - 2)e^{d-2} + 2(7 - 2d)e^{d-1}.
\end{align*} \]

2. DCTA\(d^+\):

\[ \begin{align*}
  n_{\text{max}} &= \sum_{i=1}^{d-1} e^i \\
  w_{\text{max}} &= 2^e \\
  f_{\text{max}} &= 2e + 1 \\
  s &= 2e^{d-1}(d - 1) + 2e \left( \frac{e^{d-2} - 1}{e - 1} \right) \\
  c &= 2(d - 4)e^d + 4(d - 2)e^{d-2} + 6e + 5e^2 \left( \frac{e^{d-1} - 1}{e - 1} \right) \\
  w &= \frac{2e}{e - 1} \left( (2d - 1 - 2^{-1}) + 7e \left( e^{d-2} - 1 \right) \\
  &\quad + 2(2e + 1 - e - 3) \left( \frac{e^{d-2} - 1}{e - 1} - 2 \right) \\
  &\quad + 4(2e - 1) + 6e\right) + 4(2^e - 1) + 6e.
\end{align*} \]

The previous results are better seen in Figs. 4, 5, 6, and 7, where the values of \(c\) and \(w\) are shown as a function of the number of bits for DCTA3+, DCTA4+, and DCTA5+ adders.

From the previous data, we can evaluate the order of complexity of a generic adder DCTA with depth \(d\):

\[ \begin{align*}
  w_{\text{max}} &= O\left( e^{\sqrt{n}} \right) \\
  f_{\text{max}} &= O\left( e^{\sqrt{n}} \right) \\
  s &= O(n) \\
  c &= O\left( e^{\sqrt{n}} \right) \\
  w &= O\left( 2 e^{\sqrt{n}} \right).
\end{align*} \]

From a practical point of view, it is important to know the bounds of the adders that can be implemented as a function of the maximum weight \(w_{\text{max}} = 2^e\) allowed by the available technology. There are two fundamental parameters that show these limitations, one of them, \(n_{\text{max}}\), has already been defined and calculated. The other parameter is the minimum depth \(d_{\text{min}}\) that should have a threshold adder to implement the sum of two operands of \(n\) bits. The values of this parameter for the DCTA and DCTA+ adders are the following:

<table>
<thead>
<tr>
<th>(n) bits</th>
<th>(d_{\text{min}})</th>
<th>(d_{\text{min}})</th>
<th>(d_{\text{min}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n = 32) bits</td>
<td>(n = 32) bits</td>
<td>(n = 32) bits</td>
<td></td>
</tr>
<tr>
<td>(n = 64) bits</td>
<td>(n = 64) bits</td>
<td>(n = 64) bits</td>
<td></td>
</tr>
<tr>
<td>(w_{\text{max}})</td>
<td>(s(d_{\text{min}}))</td>
<td>(c(d_{\text{min}}))</td>
<td>(w(d_{\text{min}}))</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>14</td>
<td>30</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>39</td>
<td>120</td>
</tr>
<tr>
<td>16</td>
<td>20</td>
<td>84</td>
<td>340</td>
</tr>
<tr>
<td>32</td>
<td>30</td>
<td>155</td>
<td>780</td>
</tr>
<tr>
<td>64</td>
<td>42</td>
<td>258</td>
<td>1554</td>
</tr>
</tbody>
</table>
1. DCTA

\[ d_{\text{min}} = 1 + \left\lceil \frac{\log n}{\log e} \right\rceil. \]

2. DCTA+

\[ d_{\text{min}} = \left\lceil \frac{\log(n(e-1)+e)}{\log e} \right\rceil. \]

The orders of complexity of these parameters, as a function of fixed weight \( w_{\text{max}} \), are the following:

\[ n_{\text{max}} = O\left(\log^{d-1} w_{\text{max}}\right) \]
\[ d_{\text{min}} = O\left(\frac{\log n}{\log(\log w_{\text{max}})}\right). \]

Table 6 shows the bounds on the number of input-bits and the size of 32-bit and 64-bit DCTA+ adders for practical values of \( w_{\text{max}} \), evaluated for the \( d_{\text{min}} \) necessary in each case.

The remaining values of complexity are:

\[ f_{\text{max}} = O(\log w_{\text{max}}) \]
\[ s = O(\log^{d-1} w_{\text{max}}) \]
\[ c = O(\log^{d} w_{\text{max}}) \]
\[ w = O(w_{\text{max}}). \]

7 Conclusions

The main contribution of this paper consists of the presentation of a systematic method to develop binary adders with Threshold Logic that introduces a new way to compute the sum bits as a function of the input and output carries and an interesting modification of the classic Carry Lookahead method. Two new parameters have also been introduced to better estimate the real size of a threshold circuit. All the important parameters of the circuit are obtained as a function of the number of bits, the depth and a new technological parameter \( e \) that depends on the characteristics of the available technology of threshold gates.

The results obtained for adders with depth 3 demonstrate an improvement regarding size, maximum weight, and maximum fan-in over all those published to date.

All the adders have a size \( O(n) \) and a maximum weight of \( O\left(2^{\sqrt{d}}\right) \), where \( d \) is the depth of the circuit.

References