Optimized Design of Carry-Bypass Adders

Massimo Alioto* and Gaetano Palumbo*

Abstract – In this paper, a simple and systematic procedure to design Carry Bypass Adders (CBA) is proposed. It allows to choose the block sizes of a CBA to minimize the adder delay, and can be used for pencil-and-paper design. Since it derives from rigorous analysis of CBAs, it is general and provides intuitive understanding of the optimum block size.

Compared to optimum results reported in the literature, the optimization procedure proposed leads to a delay which is minimum in actual cases, or very close to optimum (within 7%) even in unrealistic cases.

1 Introduction

The adder is the fundamental block in any arithmetic unit, and is often the speed-limiting circuit in a digital system. Hence, many parallel adder architectures have been proposed to increase speed, with reasonable area and power dissipation features. One of the fastest and efficient architectures in terms of area and power dissipation is the Carry Bypass Adder (CBA) [1].

A N-bit CBA is made up of N full adder gates, which are grouped together into blocks, whose size (i.e., the number of full adders per block) has to be properly chosen to minimize the time needed for a computation [2]-[3]. The CBA architecture can be derived from that of a simple Ripple Carry Adder (obtained cascading N full adders [3]) by stating that, when some contiguous full adders work in propagate (i.e., each of them has a carry output equal to the carry input), they can be bypassed to evaluate the carry output of the last one, since it is equal to the carry input of the first. Hence, in a CBA the full adders are divided into groups, each of them is “bypassed” by a multiplexer if its full adders are all in propagate. The resulting architecture is shown in Fig. 1, where N full adders are grouped into Q blocks. The blocks are connected by 2:1 multiplexers, which can be placed into one or more level structures. The strategies proposed until now to find the optimum distribution of full adders into blocks are based on iterative algorithms [4]-[6], complex pencil-and-paper procedures [2], [7]-[8], or approximate closed-form solutions [1], [9]. However, design strategies proposed do not provide an insight on the optimum block size [4]-[6], are based on unrealistic hypothesis [1], [7], [9], or are tedious to be applied [2], [7].

In this paper a novel optimization strategy of block sizes for one-level Carry Bypass Adders is proposed. It is based on a first analytical sizing of an optimum but incomplete adder, and successively missing bits are properly added to achieve the desired number of bits with minimum delay increase. The strategy proposed is systematic and general, independent of the technology used, and it is suitable for pencil-and-paper design.

The validity of the proposed strategy is checked by comparing the resulting delay to optimum results in [6] for different numbers of bits, full adder and multiplexer delays.

2 Timing analysis of the CBA

Assume the j-th block of the CBA in Fig. 1 is made up by $M_j$ cascaded full adders and a multiplexer, and the number of blocks is Q. The input signal of the j-th multiplexer are $I_j^0$ and $I_j^1$: the former is the output of the full adders chain which forms the j-th block, and it is selected when the carry does not propagate in the block; the second is the carry output of the previous block. When all the full adders of the j-th block work in propagate, its block carry output (or equivalently the carry input of the next block) is equal to $I_j^1$, otherwise it is equal to $I_j^0$.

It is convenient to define $\tau_{\text{MUX}}$ the propagation delay of a multiplexer, $\tau_{\text{CARRY}}$ the full adder carry output delay once its inputs (including the carry input) have been applied, and $\tau_{\text{SUM}}$ the full adder sum output delay. Let $t_j^0$ and $t_j^1$ the worst case delay required to generate signals $I_j^0$ and $I_j^1$, respectively, from the time in which the input signals are provided to the adder. For the sake of clarity, the meaning of the timing parameters introduced is graphically reported in Fig. 1.

By definition, $t_j^0$ is the time required by j-th block to evaluate its carry output signal, $I_j^0$, when its first full adder is not in propagate and the other ($M_j$-1) work in propagate [3]; hence we get

$$t_j^0 = M_j \tau_{\text{CARRY}}$$

* University of Catania, Dipartimento Elettrico Elettronico e Sistemistico, Catania, Italy. E-mail: malioto@dees.unict.it gpalumbo@dees.unict.it Tel: +39-095-7382313, Fax: +39-095-330793.
Moreover, \( t'_j \) is the time when the output of the multiplexer belonging to the \((j-1)\)-th block is evaluated, which depends on the arrival of the slowest input signal among \( t''_j \) and \( I_{j-1} \):

\[
t'_j = \max \left( t''_j, t''_{j-1} \right) + \tau_{\text{MUX}}
\]

To minimize the adder delay, it is necessary [3] to guarantee that \( t'_j \geq t''_j \) for \( j=1\ldots Q \), which means that

\[
t'_j = t''_j + \tau_{\text{MUX}}
\]

which can be iteratively expressed as

\[
t'_j = t''_0 + (j-1)\tau_{\text{MUX}} = M_1\tau_{\text{CARRY}} + (j-1)\tau_{\text{MUX}}
\]

Comparing (1) with (3), it follows that, starting from the first block, the size of the following ones can be progressively increased without violating \( t'_j \geq t''_j \) by guaranteeing

\[
M_j \leq M_1 + (j-1)\alpha
\]

where \( \alpha \) is defined as \( \frac{\tau_{\text{MUX}}}{\tau_{\text{CARRY}}} \). Since the adder is optimum if it is not possible to add bits without increasing the overall delay, in the optimum CBA the block size \( M_j \) must be chosen as high as possible, considering the strict equality in (4a):

\[
M_j = M_1 + j\alpha
\]

Hence, the optimum CBA is made up of a left part with increasing block size, and a right part with decreasing block size. According to [2], the intermediate blocks with the maximum size are named nucleus of the adder.

![Figure 1: Structure and timing analysis of a Carry Bypass Adder.](image)

**3 Optimization of the CBA block size**

As explained in the previous section, an optimum CBA can be designed by adding blocks first with increasing size by \( \alpha \), according to (4), and then blocks with decreasing size by \( \alpha \), according to (5), until the required number of bits, \( N \), is obtained. Hence, a symmetrical block size distribution can be assumed. Unfortunately, \( M_Q \) and \( M_1 \) are unknown, and will be evaluated in the following.

Assuming that the ratio \( \alpha \) is an integer [1], [8], [9], and an equal size for the first and the last block (i.e., \( M_Q = M_1 \)), (4) and (5) lead to a symmetrical distribution of block sizes (i.e., \( M_2 = M_{Q-1} = M_1 + \alpha \) and so on), which vary by a step of \( \alpha \).

Since the number of bits must be equal to \( N \), we have

\[
N = 2(M_1 + M_2 + \ldots + M_{Q/2}) =
\]
where an even number of blocks, \( Q \), was assumed (the same results are obtained with an odd \( Q \)). As observed in the previous section, the MSB of sum output of blocks in the right part are contemporarily generated. In particular, considering the sum output evaluation in the last block, the adder delay is given by

\[
\tau_{PD} = (2M_1 - 1)\tau_{CARRY} + (Q - 1)\tau_{MUX} + \tau_{SUM}
\]  

(7)

where \( M_1 = M_1 \) was used. Evaluating \( M_1 \) from (6), and substituting it in (7), we get

\[
\tau_{ro} = \left(\frac{N - \frac{Q}{2}(Q^2 - 2Q)}{2} - 1\right)\tau_{CARRY} + (Q - 1)\tau_{MUX} + \tau_{SUM}
\]  

(8)

that can be optimized with respect to \( Q \), and gives an optimum value of \( Q \) equal to \( Q_{op} = 2(N/\alpha)^{\frac{1}{2}} \), which substituted into (6) gives the optimum size for the first and last block

\[
M_{1,op} = M_{Q,op} = \frac{\alpha}{2}
\]  

(9)

Hence, using (9) and adding blocks according to (4b) and (5), an optimum CBA is obtained. In general, its number of bits, \( N' \), is lower than the required \( N \), and for this reason the incomplete CBA obtained will be referred as nearly-optimum CBA. Moreover, in actual implementations the ratio \( \alpha \) is lower than unity. Therefore, the introduced criteria have to be properly modified to take into account the non-integer value of \( \alpha \) as discussed in the following.

4 Optimization of the nearly-optimum CBA with non-integer \( \alpha \)

As observed in the previous section, in the optimum CBA the (integer) value of \( M_i \) in (4) must be as high as possible, i.e. equal to the integer part of \( M_i + (j-1)\alpha \). Hence, \( M_1 \) has to be rounded to the closest integer number (which is one in practical cases, since \( \alpha \approx 1 \)), and (4a) becomes:

\[
M_{1,j} = \text{int}[M_1 + (j-1)\alpha]
\]  

(10)

that allows to build the left part of the nearly-optimum CBA, and symmetrically build the right part, adding blocks until the maximum number of bits \( N' \leq N \) is obtained. If \( N' = N \), the optimum CBA built is complete, and the design procedure stops.

If \( N' < N \), we have to properly add other full adders to reach the required number of bits, \( N \), and depending on the number of the full adder which are still to be added, \( N-N' \), and on the size of the last added block, \( M_{Q,2} \), there are three possible cases:

1. If \( N-N' < M_{Q,2} \), no more blocks can be added between the maximum size ones without violating (10), hence the remaining full adders must be properly added with an adder delay increase as low as possible. This problem is tackled in the next section.

2. If \( N-N' = M_{Q,2} \), we can add a block with size \( M_{Q,2} \) between the two maximum size added blocks, resulting to a three blocks nucleus.

3. If \( N-N' > M_{Q,2} \), we can add another block between the two maximum size blocks, its size is determined according to (10), and it will represent the nucleus. As for the case \( N-N' < M_{Q,2} \), eventual remaining full adders to evaluate the remaining bits \( N-N'-M_{Q,2} \) are added following the procedure in the next section.

As an example, consider the case \( N=24 \) and \( \alpha=0.3 \); applying (10), we build the left part of the CBA, whose block sizes \( M_1, \ldots, M_j \) are equal to \( 1, 1, 1, 1, 2, 2 \), respectively, and lead to a total number of bits \( N'=2*(1+1+1+1+2+2+2+2)=20 \), where the factor two accounts for the symmetrical right part. Applying (10), no more blocks can be added: indeed, if we added the 8-th block, for which (10) gives \( M_8=3 \), it would lead to \( N'=26*N \).

The number of remaining bits is equal to \( N-N'=4 \), which is greater than \( N_6 \), thus the scenario is that described by the case 3; therefore we add a one-block 3-bit nucleus as the 8-th block, leading to a nearly-optimum CBA whose \( M_1, \ldots, M_{15} \) are equal to \( 1, 1, 1, 1, 2, 2, 2, 3, 2, 2, 1, 1, 1, 1 \).

Hence, one more full adder must be added to the nearly-optimum CBA to get the required number of bits \( N=24 \). The problem of completing the nearly-optimum CBA is addressed in the following.

5 Completing the nearly-optimum CBA

The remaining bits have to be properly added to the nearly-optimum CBA, in order to minimize the delay increase, \( \Delta\tau_{PD,tot} \), with respect to its delay. To this aim, let us define \( \Delta\tau_{PD,j} \) as the adder delay increase due to insertion of one bit into the \( j \)-th block of the nearly-optimum CBA. To minimize the delay, the remaining bits must be added to blocks with minimum \( \Delta\tau_{PD,j} \).

The search of blocks with minimum \( \Delta\tau_{PD,j} \) is simplified by stating the following observations:

1. \( \Delta\tau_{PD,j} \) is the same for blocks which are symmetrical with respect to the nucleus (i.e., \( \Delta\tau_{PD,Q/2+1} = \Delta\tau_{PD,j} \)), due to the symmetrical sizing

2. for blocks at the left part of the CBA, \( \Delta\tau_{PD,j} \) is due to the increase of \( i \) by \( \tau_{CARRY} \) associated with the

\footnote{For example, for a 0.35-\mu m CMOS process we get \( \alpha \approx 0.32 \).}
insertion of one bit, becoming greater than \( t_i^j \); thus, from (2) \( \Delta \tau_{PD,j} \) is given by the difference between the new value of \( t_i^j \) (i.e., \( M_j +1 \) \( \tau_{CARRY} \)) and \( t_i^j \) (eq. (3)):
\[
\Delta \tau_{PD,j} = \tau_{CARRY} \left[ 1 + \left[ M_i + (j - 1) \alpha \right] - \left[ M_i + (j - 1) \alpha \right] \right]
\]
where (10) was used. From (11), \( \Delta \tau_{PD,j} \) is equal to \( \tau_{CARRY} \) multiplied by one minus the non-integer part of \( [M_i + (j - 1) \alpha] \).

3. only blocks after which the block size changes must be considered, going towards the nucleus; indeed, from (1) adjacent blocks with equal size in the left part of the CBA have equal \( t_i^0 \), but from (3) the closest to the nucleus has the greatest \( t_i^j \); hence, from (11), it has minimum \( \Delta \tau_{PD,j} \).

Using these observations, by simple inspection of the CBA the best blocks which remaining bits to be added to are found. As an example, let us complete the nearly-optimum CBA built in the previous section, searching for the blocks with minimum \( \Delta \tau_{PD,j} \). In the left part of the CBA, only blocks 4 and 7 must be considered, since the block size changes after them, whose \( \Delta \tau_{PD,j} \) is equal to 0.1 \( \tau_{CARRY} \) and 0.2 \( \tau_{CARRY} \) from (11); in the right part, only the symmetrical blocks 12 and 9 must be considered, and for observation 1) they also have \( \Delta \tau_{PD,j} \) equal to 0.1 \( \tau_{CARRY} \) and 0.2 \( \tau_{CARRY} \), respectively. Moreover, the nucleus (8-th block) has \( \Delta \tau_{PD,j} = \tau_{CARRY} \) from (11). Hence, the remaining bit has to be added to block 4 or 12, since they have the lowest \( \Delta \tau_{PD,j} \) and the delay increase with respect to the nearly-optimum CBA is \( \Delta \tau_{PD,TOT} = 0.1 \tau_{CARRY} \).

### 6 Validation and conclusions

The results obtained with the procedure proposed were compared to those in [6], which are optimal. More specifically, we compared the delay, to check if the procedure effectively provides optimum results. To this aim, we considered different values of \( N \) (32 and 64) and \( \alpha \) (ranging from 0.1 to 2, with a step of 0.1). Moreover, we considered the examples reported in [6]. For the sake of compactness, in Table I are reported only the cases with \( \alpha < 1 \) and in which the procedure leads to block sizes different from [6].

From inspection of Table I, the procedure proposed leads to a delay greater than that obtained with [6] only for \( \alpha \geq 0.85 \), while for lower values of \( \alpha \) the same delay is achieved, even with a different block size. Therefore, the CBA design strategy proposed leads to optimum speed performance in realistic cases. Even for unrealistic cases, the CBA delay obtained after the optimization strategy proposed differs at most by 7% with respect to [6].

Moreover, it is simple and can be used for pencil-and-paper design. Finally, due to the design criteria analytically derived, it provides the designer with an intuitive understanding of the optimum block size, allowing a good insight of the optimization problem.

### References


<table>
<thead>
<tr>
<th>( N )</th>
<th>( \alpha )</th>
<th>Optimized procedure</th>
<th>Block sizing</th>
<th>( \tau_{PD}/\tau_{CARRY} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>0.2</td>
<td>proposed</td>
<td>1 1 1 1 1 2 2 2 2 2 3 3 2 2 2</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[6]</td>
<td>1 1 1 1 1 2 2 2 2 2 3 3 2 2 2</td>
<td>“</td>
</tr>
<tr>
<td>32</td>
<td>0.7</td>
<td>proposed</td>
<td>1 2 2 3 4 4 4 4 4 3 2 2 1</td>
<td>10.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[6]</td>
<td>1 1 2 2 3 3 4 4 3 3 2 1</td>
<td>“</td>
</tr>
<tr>
<td>32</td>
<td>0.9</td>
<td>proposed</td>
<td>1 1 2 3 4 5 4 5 3 4 2 2 1</td>
<td>11.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[6]</td>
<td>1 2 2 3 4 5 4 5 3 2 1</td>
<td>11.5</td>
</tr>
<tr>
<td>64</td>
<td>0.85</td>
<td>proposed</td>
<td>1 1 2 3 4 5 6 6 7 7 6 6 5 4 3 2 1</td>
<td>15.65</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[6]</td>
<td>1 2 3 4 5 6 7 7 6 6 5 4 3 2 1</td>
<td>“</td>
</tr>
<tr>
<td>64</td>
<td>0.9</td>
<td>proposed</td>
<td>1 2 3 4 5 6 7 7 6 5 4 3 2 1</td>
<td>16.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[6]</td>
<td>1 2 3 4 5 6 7 7 6 5 4 3 2 1</td>
<td>15.9</td>
</tr>
<tr>
<td>128</td>
<td>0.85</td>
<td>proposed</td>
<td>1 2 3 4 5 6 7 8 8 8 9 10 10 9 8 7 6 6 5 4 3 2 1</td>
<td>21.85</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[6]</td>
<td>1 2 3 4 5 6 7 8 8 9 10 10 9 8 7 6 6 5 4 3 2 1</td>
<td>21.65</td>
</tr>
</tbody>
</table>

Table I: results of the procedure proposed vs [6]