A GATE-LEVEL STRATEGY TO DESIGN CARRY SELECT ADDERS

M. Alioto[1], *G. Palumbo*[2], *M. Poli*[2]

[1] DII – Dipartimento di Ingegneria dell'Informazione, Università di Siena,
v. Roma n. 56, I-53100 - Siena (Italy)
Phone ++39.0577.234632; Fax ++39.0577.233602
malioto@dii.unisi.it

[2] DIEES - Dipartimento di Ingegneria Elettrica Elettronica e dei Sistemi, Università di Catania,
viale Andrea Doria 6, I-95125 CATANIA - ITALY
Phone ++39.095.7382313; Fax ++39.095.330793
gpalumbo@diees.unict.it, mpoli@diees.unict.it

ABSTRACT

This paper addresses the gate-level design of Carry Select Adders aiming at minimizing its delay through a proper selection of the Full Adder groups sizes. It starts from a rigorous timing analysis of the Carry Select Adder, from which a preliminary procedure is formulated to build an incomplete nearly-optimum adder. Then, the required number of bits is reached by adding remaining bits into proper blocks minimizing the delay increase. The design strategy proposed also accounts for the dependence of multiplexer (MUX) delay on its fan-out, in contrast to the usual and unrealistic assumption of a constant MUX delay. The strategy proposed is applied in several design cases, whose results shows that the delay achieved is usually minimum, and only in a few cases delay it is lower than 2% of the optimum.

1. INTRODUCTION

Addition is the most frequently operation performed in both general-purpose or application-specific digital systems [1], and generally, it greatly affect the overall speed of these systems. For this reason, many adder architectures have been proposed until now, each of which offering specific tradeoffs in terms of area-delay-power consumption. When a compromise between speed and area is required, Carry Select Adders (CSAs) are a good design choice [2], since they can be very fast [3]-[4] with a reasonable area, especially when reduced-area schemes are used [5].

The architecture of an N-bit CSA, reported in Fig. 1, is based on the consideration that carry propagation in a carry chain of N Full Adders can be speeded up by evaluating the carry output of successive digits without waiting for the carry input arrival of the previous ones. This is achieved by dividing the carry chain into Q different blocks consisting of two M_i Full Adder chains (with i=1…Q), one assuming that the block carry input is 1 and the other assuming the carry input to be 0. In each block, the correct carry (and sum) output is selected through multiplexers according to the value of the carry input, i.e. the carry output of the previous block. Obviously, the first block does not require such a multiplexer, since its carry input coincides with that of the adder, which is immediately available at the beginning of the computation [6].

The speed performance of a CSA strongly depends on the block sizing [7] (i.e. on the number of bits M_i computed by each block), provided that the bit length N of operands is achieved

$$\sum_{i=1}^{Q} M_i = N \tag{1}$$

In this paper, a design strategy to minimize the delay of a Carry Select Adder is proposed. It allows for sizing the number of bits computed by each block, according to criteria derived from a preliminary timing analysis of the CSA. The strategy accounts for the dependence of the MUX delay on its fan-out, which greatly varies depending on the block considered. Optimization of group sizes consists of two design steps, the first of which leading to a nearly-optimum adder with a number of bits lower than the required value N. In the second step, the nearly-optimum adder is completed by keeping the delay increase minimum.

2. TIMING ANALYSIS OF THE CSA

Let us consider the N-bit CSA in Fig. 1 with Full Adders partitioned into Q groups with sizes (i.e., number of computed bits) $M_1 \ldots M_Q$. The generic i-th block with i>1 consists of two chains of M_i Full Adders whose carry input is equal to 0 and 1, respectively. Its carry output (as well as sum outputs) is obtained by selecting the correct result through the multiplexer MUX_i, whose selection signal is generated by the carry output of the previous block, and input signals are the carry outputs of the two Full Adders' chains. Therefore, once all adder input signals are available at time t=0, the worst-case time $t_{in,i}$ (reported in Fig. 1 in gray line) required to generate the two MUX input signals is that needed by the carry propagation through all the M_i Full Adders

$$t_{in,i} = M_i \tau_{CARRY} \tag{2}$$

where τ_{CARRY} is the carry delay of each Full Adder (equal for all Full Adders, since they have the same fan-out). Regarding the selection signal of the MUX_i, it is generated by the MUX_{i-1} of the preceding block, having a delay $\tau_{MUX,i-1}$. Therefore, such signal is generated at time tsel,i equal to the sum of $\tau_{MUX,i-1}$ and the arrival time of the latest between the input and selection signal of MUX_{i-1}

$$t_{sel,i} = \max(t_{in,i-1}, t_{sel,i-1}) + \tau_{MUX,i-1} \tag{3}$$

where the MUX delay is assumed to be equal to the sum of its intrinsic delay τ_{int} (associated with its transistors' parasitic capacitances) and a term proportional to its fan-out FO_{i-1} (due to the input capacitance of the driven gates)

$$\tau_{MUX,i-1} = \tau_{\text{int}} + \tau_{FO} FO_{i-1} \tag{4}$$

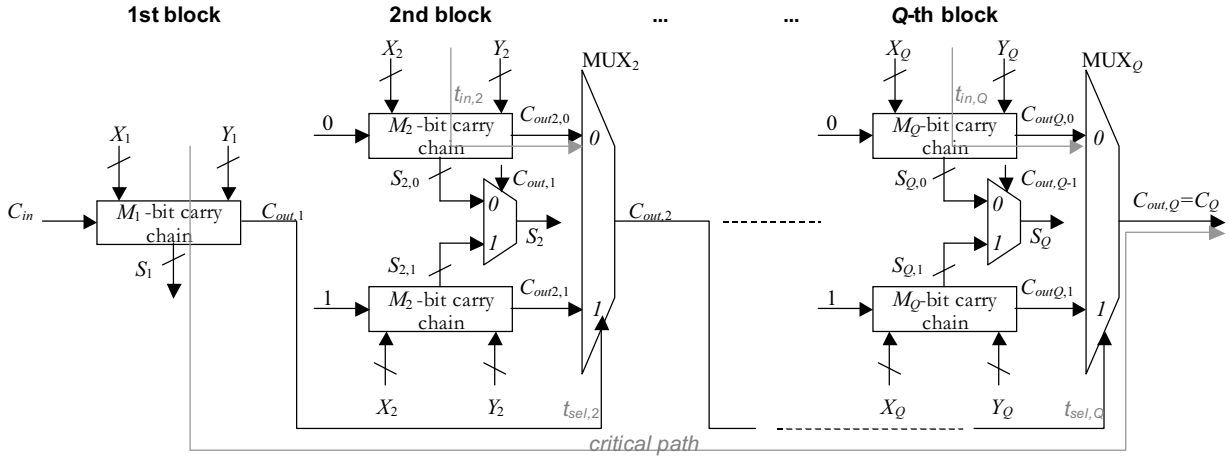

Figure 1. Carry Select Adder architecture.

where τ_{FO} is the MUX delay increase per unity fan-out increase. It is worth noting that the generic MUX_{i-1} has a fan-out of (M_i+1), since it drives the selection input of the M_i multiplexers generating the sum bits of the successive block and its carry output multiplexer. Therefore, by using eq. (4), relationship (3) becomes

$$t_{sel,i} = \max(t_{in,i-1}, t_{sel,i-1}) + \tau_{\text{int}} + \tau_{FO}(M_i + 1) \quad (5)$$

for i=2…Q. In the following, these relationships are used to derive fundamental criteria to optimally design CSAs in two cases, one with a uniform block sizes, being useful to introduce basic ideas and some notation, and the other with a variable sizing.

2.1 Constant block sizing (M_i=M).

In the constant block sizing, all Q blocks compute the same number M of bits, i.e., M_i=M for i=1…Q, where the number of blocks Q is equal to N/M. In this case, Full Adder chains in all blocks generate their output at time $M\cdot\tau_{CARRY}$, thus from (2) and Fig.1 it follows that

$$t_{sel,2} = t_{in,i} = M\tau_{CARRY} \quad (6)$$

By iteratively substituting eq. (6), eq. (5) becomes

$$t_{sel,i} = M\tau_{CARRY} + (i-2)[\tau_{\text{int}} + \tau_{FO}(M+1)] \quad (7)$$

for i=3…Q, which is always greater than $t_{in,i}$ in (6). Therefore, the latest arriving signal of each MUX is always its selection signal. Thus the adder critical path crosses all the (Q-2) multiplexers MUX_2…MUX_{Q-1} and finally the sum and carry output multiplexers of the last block (which are assumed to have a unity fan-out, without loss of generality), as highlighted in Fig. 1 by the gray line. Thus, the worst-case adder delay is

$$\tau_{PD} = M\tau_{CARRY} + (Q-2)[\tau_{\text{int}} + \tau_{FO}(M+1)] + [\tau_{\text{int}} + \tau_{FO}] \quad (8)$$

where the last Full Adder delay to generate its sum output was assumed to be equal to the carry delay without loss of generality (a different value simply adds a constant term to (8), and hence it does not affect the adder optimization).

By substituting Q=N/M and minimizing (8), it can be easily derived that delay is minimized for the following value of M

$$M_{opt} = \sqrt{\frac{N(\tau_{\text{int}} + \tau_{FO})}{\tau_{CARRY} - 2\tau_{FO}}} \quad (9)$$

and the resulting minimum delay is

$$\tau_{PD,opt} = 2\sqrt{N(\tau_{CARRY} - 2\tau_{FO})(\tau_{\text{int}} + \tau_{FO})} + N\tau_{FO} - \tau_{\text{int}} \quad (10)$$

which is essentially proportional to $\sqrt{N}$. Note that, in order to have a speed advantage over the simple Ripple Carry Adder (RCA) (whose delay is equal to $N\cdot\tau_{CARRY}$ [7]), the MUX parameters τ_{int} and τ_{FO} must be much lower than τ_{CARRY}. These conditions are well satisfied in practical cases, as can be verified from timing parameters in Table I of Mirror Full Adders and TG MUXes (symmetrically designed for minimum power consumption) by using a 0.35-μm CMOS process.

TABLE I

τ_{CARRY}	τ_{int}	τ_{FO}
517 ps	97 ps	19 ps

2.2 Variable block sizing ($M_i \geq M_{i-1}$)

In the constant block sizing case, the input signal of each MUX arrive before its selection signal (i.e., $t_{sel,i} > t_{in,i}$ from (7)). Therefore, from relationship (5), the arrival time $t_{in,i}$ of MUX input signals (and hence the number M_i of bits computed in the i-th block, from (2)) can be increased up to $t_{sel,i}$

$$t_{in,i} \leq t_{sel,i} \quad (11)$$

without modifying the critical path. This justifies the well-known higher performance of the variable block size CSA, since it allows for increasing the number of bits computed by each block without significantly increasing the overall delay. Minimum delay is achieved when each block has the maximum number of bits satisfying (11), i.e. with a strict equality. In this case, the first and the second block size is the same

$$M_1 = M_2 = M_{1,2} \quad (12)$$

since they must provide the selection and input signal of to MUX_2 at the same time (i.e., $t_{sel,2}$=$M_1\tau_{CARRY}$=$t_{in,2}$=$M_2\tau_{CARRY}$).

For next blocks (i=3…Q), by substituting eqs. (2)-(5), relationship (11) with strict equality becomes

$$M_i \tau_{CARRY} = M_{i-1} \tau_{CARRY} + \tau_{\text{int}} + \tau_{FO}(M_i + 1) \qquad (13)$$

and thus M_i results

$$M_i = \frac{\alpha + \beta + M_{i-1}}{1 - \beta} \qquad (14)$$

where α and β are the MUX timing parameters normalized to τ_{CARRY}, τ_{in}/τ_{CARRY} and τ_{FO}/τ_{CARRY}, respectively (much lower than unity, in practical cases).

By iteratively applying (14), M_i is easily found to result as

$$M_i = \frac{M_{1,2}}{(1-\beta)^{i-2}} + \sum_{j=1}^{i-2} \frac{\alpha + \beta}{(1-\beta)^j} = \\ = \frac{M_{1,2}}{(1-\beta)^{i-2}} + \left(1 + \frac{\alpha}{\beta}\right)\left[(1-\beta)^{2-i} - 1\right] \qquad i = 3....Q \qquad (15)$$

Now let us minimize the overall delay, by remembering that the critical path must cross all MUXes, hence the latest arriving signal in the CSA is the output of MUXes in the last block. under (11) with strict equality and (2), this signal is generated in a time

$$\frac{\tau_{PD}}{\tau_{CARRY}} = \frac{t_{sel,Q}}{\tau_{CARRY}} + \frac{\tau_{MUX,Q}}{\tau_{CARRY}} = M_Q + \alpha + \beta = \\ = \frac{M_{1,2}}{(1-\beta)^{Q-2}} + \left(1 + \frac{\alpha}{\beta}\right)\left[(1-\beta)^{2-Q} - 1\right] + \alpha + \beta \qquad (16)$$

where the adder delay τ_{PD} was normalized to τ_{CARRY} to simplify notation and (15) was used to evaluate M_Q. For the same reasons as the constant block sizing case, the Full Adder sum delay was assumed to be equal to the carry delay and a unity fan-out was assumed for the last MUX.

Once MUX and Full Adder timing parameters are known, the CSA delay in (16) is a function of $M_{1,2}$ and Q, where the former can be easily expressed as a function of the latter by using constraint (1) after substituting (15)

$$M_{1,2} = \frac{N \cdot \beta^2 - (\alpha + \beta)\left[(1-\beta)^{2-Q} - 1 - \beta(Q-2)\right]}{2 \cdot \beta^2 + \beta \cdot (1-\beta)^{2-Q} - \beta} \qquad (17)$$

After substituting (17) into (16) and performing some tedious calculations, relationship (16) results to be minimum for $M_{1,2}$ equal to

$$M_{1,2opt} = \frac{\alpha + \beta}{\ln(1-\beta) \cdot (2 \cdot \beta - 1)} - \frac{\alpha + \beta}{\beta} \qquad (18)$$

which is plotted in Fig. 2 for some typical values of α and β. As a result, the CSA delay is minimized by setting the size of the first two blocks equal to (18), and the successive blocks as in (15). However, since (15)-(18) provide a non-integer number of Full Adders, this analytical strategy is not adequate in real cases, and some modification must be applied, as will be shown in Section 3.

3. A STRATEGY TO DESIGN CSAS

In this section a practical design procedure that ensures integer values of M_i is presented. Being based on the previous timing analysis, the strategy consists of two basic steps, where a nearly-optimum CSA (whose critical path crosses all MUXes) is first built and then refined by adding bits until the required number N is achieved.

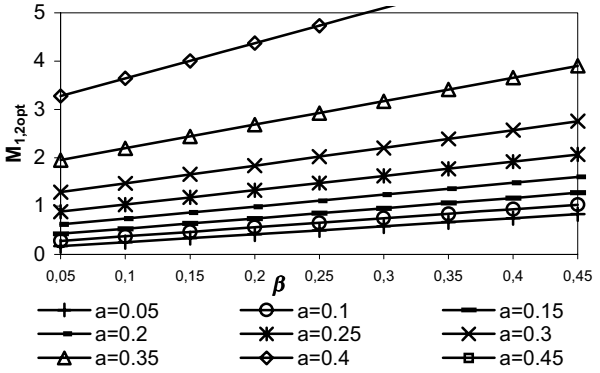

Figure 2. Plot of $M_{1,2opt}$ versus β for some values of α.

3.1 Building a nearly-optimum CSA.

To achieve integer values of $M_{1,2}$, result from (18) must be rounded to the nearest (nonzero) integer, which can be graphically evaluated from the plot in Fig. 2. To ensure that the critical path crosses all MUXes, each (integer) value of M_i for i=3…Q must satisfy (11)

$$M_i \le M_{1,2opt} + \sum_{j=2}^{i-1} \tau_{MUX,j} = M_{1,2opt} + \sum_{j=3}^{i} \left[\alpha + \beta(M_j + 1)\right] \qquad (19)$$

that, solved for the greatest integer value of M_i, leads to

$$M_i = floor\left[\frac{M_{1,2opt} + \beta \sum_{j=3}^{i-1} M_j + (i-2)(\alpha + \beta)}{1 - \beta}\right] \qquad (20)$$

for i=4…Q (for i=3, (20) must be modified by deleting term $\beta \sum_{j=3}^{i-1} M_j$).

By rounding (18) and then iteratively evaluate (20), we can build a CSA whose blocks have the maximum number of bits which allows for satisfying relationship (11) (i.e. ensuring the critical path to cross all MUXes). Therefore, by iteratively adding such blocks, an increasing overall number of computed bits is achieved. Addition of blocks must be stopped when, adding one more block, the overall number of computed bits becomes greater than the required value N. Let N* the maximum overall number of bits lower than or equal to N that such a CSA can have. If N*=N, the design procedure is concluded. If N*<N, further bits must be added to the nearly-optimum CSA according to criteria discussed in the following subsection.

As an example, consider a CSA with N=32, α=0.33 and β=0.26. From (18), which gives 1.81, M_1 and M_2 are set to 2, while iterative relationship (20) leads to M_3=3, M_4=5, M_5=7 and M_6=11, leading to a 30-bit CSA. Addition of a further

block would lead to a number of bits greater than 32 (since (20) would result to M_7=15), thus the CSA obtained is the nearly-optimum with N^*=30. Its timing analysis is reported in Table II.

TABLE II

i	1	2	3	4	5	6
$t_{in,i}/\tau_{CARRY} = M_i$	2	2	3	5	7	11
$t_{sel,i}/\tau_{CARRY}$	-	2	3.37	5.26	7.93	11.45

3.2 Completing the nearly-optimum CSA.

When the N-N* bits are added to the nearly-optimum CSA, relationship (11) is violated, since block sizes in the nearly-optimum CSA assume the maximum values that satisfy (11). Therefore, such bits must be inserted in proper blocks leading to minimum delay increase compared to the nearly-optimum adder.

When the first bit is added to the i-th block of this CSA, parameter increases by τ_{CARRY} from (2), while its $t_{sel,i}$ increases by τ_{FO} from (5). Since $t_{in,i}$ becomes greater than $t_{sel,i}$, the new critical path starts from Full Adders of the i-th block and crosses all the successive MUXes. Thus, the overall delay increases by the difference of the new value of $t_{in,i}$ (equal to the previous one plus τ_{CARRY}) and the previous value of $t_{sel,i}$. As a result, the best block of the nearly-optimum CSA where the first bit has to be added is that maximizing difference $t_{sel,i}$-$t_{in,i}$ in the nearly-optimum adder. Analogously, eventual successive bits added to complete the adder have to be inserted in blocks that lead to a minimum delay increase, which are identified by simply comparing the delay increase associated with each block.

As an example, let us complete the nearly-optimum CSA introduced in the previous subsection, where two more bits are required. By inspection of Table II, the first block has to be added to the 5-th block (having $t_{sel}^{i} - t_{in}^{i}$=0.93), thus resulting to the CSA shown in Table III. Finally, a simple timing analysis of this adder shows that the last bit should be added to the 6-th block, leading to the final 32-bit adder with block sizes 2, 2, 3, 5, 8, 12. The obtained CSA has the same block sizes as the optimum CSA that was identified through exhaustive research in the design space (i.e., the CSA obtained is optimum).

TABLE III

i	1	2	3	4	5	6
$t_{in,i}/\tau_{CARRY} = M_i$	2	2	3	5	8	11
$t_{sel,i}/\tau_{CARRY}$	-	2	3.37	5.26	7.93	11.71

4. VALIDATION AND CONCLUSIONS

The results obtained with the procedure proposed were compared to optimum results (achieved through exhaustive selection of block sizing that leads to minimum delay). Delay of the former was compared to the latter in more than 100 design cases by considering different values of N (32 and 64), α and β (both ranging from 0.05 to 0.45, to ensure the MUX delay with unity fan-out α+β to be lower than τ_{CARRY}). Results obtained show that in most cases the procedure provides optimum results, and only in a few cases delay achieved is greater than the minimum achievable by less than 2.5% (some results, including all worst cases, are summarized in Table IV). Therefore, for practical purposes, the design strategy proposed leads to optimum results, and can thus be suitably used to optimally set block sizes to minimize the adder delay.

It is worth noting that the proposed procedure is simple and can be used for pencil-and-paper design. In addition, being based on a rigorous timing analysis of the adder, the strategy provides the designer with an intuitive understanding of the optimum block size, affording a deeper insight into the optimization process.

5. REFERENCES

[1] C. Nagendra, M. J. Irwin, M. Owens, "Area-Time-Power Tradeoffs in Parallel Adders," *Trans. on CAS- part II*, vol. 43, no. 10, pp. 689-702, Oct. 1996.

[2] O. Bedrij, "Carry Select Adder," *IRE Trans. on Electronic Computers*, vol. EC-11, pp. 340-346, 1962.

[3] Y. Huang, J. B. Kuo, "A High-Speed Conditional Carry Select (CCS) Adder Circuit with a Successively Incremented Carry Number Block (SICNB) Structure for Low-Voltage VLSI Implementation," *IEEE Trans. on CAS – part II*, vol. 47, no. 10, pp. 1074-1079, Oct. 2000.

[4] M. Suzuki, N. Ohkubo, T. Shinbo, T. Yamanaka, A. Shimizu, K. Sakai, Y. Nakagome, "A 1.5-ns 32-b CMOS ALU in Double Pass-Transistor Logic," *IEEE J. Solid-State Circuits*, vol. 28, pp. 1145-1150, Nov. 1993.

[5] A. Tyagi, "A Reduced-Area Scheme for Carry-Select Adder," *Trans. on Computers*, vol. 42, no. 10, pp. 1163-1170, Oct. 1993.

[6] B. Parhami, *Computer Arithmetic, Algorithms and Hardware Designs*, Oxford University Press, 2000.

[7] J. M. Rabaey, A. Chandrakasan, B. Nikolic, *Digital Integrated Circuits: a Design Perspective*, 2nd Ed., Prentice-Hall, 2003.

TABLE IV

N	α β	Design procedure	Block sizing	τ_{PD}/τ_{CARRY}
32	0.05 0.33	proposed	2 2 3 5 8 12	12.76
		optimum	2 2 3 5 8 12	"
32	0.2 0.21	proposed	1 1 2 3 4 5 7 9	9.93
		optimum	1 1 2 3 4 5 7 9	"
32	0.21 0.10	proposed	1 1 1 1 2 3 3 4 5 5 6	6.85
		optimum	1 1 1 1 2 3 3 4 5 5 6	"
32	0.30 0.08	proposed	1 1 1 2 2 3 4 5 6 7	7
		optimum	1 1 1 1 2 3 3 4 5 5 6	6.82
32	0.33 0.30	proposed	3 3 5 8 13	13
		optimum	2 2 3 5 8 12	12.92
32	0.48 0.40	proposed	9 9 14	15.48
		optimum	9 9 14	"
64	0.20 0.25	proposed	1 1 2 4 5 7 10 14 20	20.25
		optimum	1 1 1 2 3 5 7 10 14 20	20.1
64	0.33 0.25	proposed	1 1 2 3 5 7 10 15 20	20.58
		optimum	1 1 2 3 5 7 10 15 20	"
64	0.42 0.08	proposed	1 1 1 2 3 4 4 5 6 7 9 10 11	11.68
		optimum	1 1 1 2 3 4 4 5 6 7 9 10 11	"
64	0.42 0.21	proposed	1 1 2 4 5 8 10 14 19	19
		optimum	3 3 3 6 8 11 13 17	18.96
64	0.42 0.35	proposed	3 3 6 10 16 26	26.51
		optimum	3 3 6 10 16 26	"
64	0.48 0.21	proposed	1 1 2 4 5 8 10 14 19	19.21
		optimum	1 1 2 4 5 8 10 14 19	"