# A1CSA: A Low-Cost and Energy-Efficient Fast Adder Architecture Targeting Cell-Based VLSI Design

Jucemar Monteiro, Pedro P. Campos, José Luís Güntzel
System Design Automation Lab (LAPS) / INE
Federal University of Santa Catarina
Florianópolis, Brazil
{jucemar, pedropaulovc, guntzel}@inf.ufsc.br

Luciano Agostini
Group of Architectures and Integrated Circuits (GACI)/DInfo
Federal University of Pelotas
Pelotas, Brazil
agostini@inf.ufpel.edu.br

***Abstract*— Contemporary battery-operated portable devices demand power efficient fast adders. Although a myriad of fast adder architectures are found in the literature, most of them are based on transistor-level optimizations that cannot be synthesized in cell-based VLSI design flow. This paper proposes two versions of add-one carry-select adders (A1CSA), suited for standard-cells design. Synthesis results for 45nm technology showed that the proposed A1CSA architectures requires between 14% and 26% less area than all other investigated fast adders. Power estimates showed that, for 64 to 256 bit adders, one of the proposed A1CSA versions consumes, on average, less power than other considered fast adders. Power-delay results reveal that, for the same bit widths, both A1CSA versions are either more energy-efficient or as efficient as other fast adders.**

## I. INTRODUCTION

Addition is the most commonly used arithmetic operation within contemporary electronic systems. It is used not only in general purpose CPUs, but also in acceleration blocks, as those present in signal, image and video processing multimedia devices. Besides used as a proper operation, it serves as the basis to many other arithmetic operations.

The choice of which adder architecture to use is of utmost importance, since the performance of adders may determine the whole system performance [1]. Area and power consumption are also relevant figures of merit to be considered, especially when the design targets VLSI realization. Recently, energy-efficiency has also become an important metric due to the growth of battery-powered portable device marked.

Most of works addressing adder architectures are focused on critical delay reduction by optimizing the carry propagation chain [1-3]. Such optimization can be accomplished at the logic-level, at transistor-level or at both. Transistor-level optimizations may use pass transistors, dynamic logic, etc.

Most of VLSI designs made in industry adopt conventional physical design flow which relies on standard-cells for layout generation. Although typical standard-cells libraries contain up to hundreds of cells, they are all designed in static CMOS style. The inclusion of user-created cells with other styles, as pass transistor or dynamic logic, is generally not allowed due to limitations of the design flow itself. Therefore, when designing high performance addition-based arithmetic circuits using cell-based VLSI design, designers must rely on logic-level optimized fast adder architectures.

As main contribution, this paper proposes two versions of Add-One Carry-Select Adder (A1CSA) which are appropriate for conventional VLSI design flow. Synthesis results show that the proposed A1CSA architectures provide significant area savings in comparison to other fast adder architectures. They also showed that the proposed architectures are particularly attractive for long bit width adders, once their energy efficiency is equivalent or superior to that of other fast adders. The A1CSA also presented a delay very close to that obtained with the best fast adders

The paper is organized as follows. Section 2 presents the most commonly used fast adder architectures. It also reviews related works concerning the A1CSA. Section 3 presents our version of A1CSA. It also introduces a new A1CSA. Synthesis results are presented and discussed in section 4. Section 5 draws the most relevant conclusions.

## II. ADD-ONE CARRY SELECT ADDERS (A1CSA)

The carry-ripple adder (CRA) is known as the most area efficient, and at the same time, the slowest adder architecture [2]. Its has both time and area complexities $O(n)$ [3]. In the carry lookahead adder (CLA), the addition is divided into $m$ modules of $k$ bits (usually, $k$=4). The carries within each module are computed in parallel by using the "generate" and "propagate" signals in order to anticipate the module carry out. When CLA modules are connected in a hierarchical manner (CLAH), it presents time complexity $O(\log_k n)$ [3] and area complexity $O(n \log n)$ [4]. In the carry-select adder (CSA), addition is divided into $m$ modules of $k$ bits each. Each module has two $k$-bit wide adders (generally, CRAs) which perform two additions at the same time: one considering a '0' as carry-in and another considering a '1' as carry-in. The CSA is an

intermediate solution between CRA and CLAH, with time and area complexities $O(\text{sqrt}(n))$ and $O(2n)$, respectively.

A significant area reduction may be achieved in the CSA if the adder receiving a carry-in '1' is replaced by a less expensive logic to increment by one the sum result generated by the remaining adder. A few works that investigated the use of the so-called "add-one logic" assumed the use of pass transistor logic [5-7], which prevents their realization in a standard-cells based design flow. Instead of using add-one logic, the approach presented in [4] requires a 50% duty-cycle clock and some extra logic resulting in a sequential adder. The work in [8] proposes a logic-level add-one circuit to realize a fully combinational CSA.

This work improves the ideas of [8] to propose two versions of Add-One CSA (A1CSA) architectures targeting conventional cell-based VLSI design. In one of those versions, named A1CSA, 4-bit adder modules are connected together in a CSA manner. In the other one, named A1CSAH, 4-bit adder modules are connected together in a hierarchical way, similarly to the CLAH. Thus, A1CSA and A1CSAH inherit time and area complexity from CSA and CLAH, respectively.

The add-one logic can be implemented based on the following properties of binary addition [6] [8].

**Property 1:** *In an addition of two n-bit numbers with the least significant bit being "0" for both numbers, if the carry-in bit is changed from one value to another, the LSB of the addition is complemented and the other bits remain unchanged.*

**Property 2:** *If the addition of two n-bit numbers with a carry-in of "0" has m "1s" before the first occurrence of a "0" (starting from the LSB), then the least significant m+1 bits of the addition with a carry-in of "1" will have values complementary to the first m+1 bits of the addition with carry-in "0".*

Fig. 1 shows an example of properties 1 and 2. Such properties are used to derive the logic equations for the add-one block (A1B). The obtained simplified equations are:

$$S_n = \neg S^0_n \quad (1)$$

$$S_{n+1} = S^0_n \oplus S^0_{n+1} \quad (2)$$

$$S_{n+2} = S^0_{n+1} \cdot S^0_n \oplus S^0_{n+2} \quad (3)$$

$$S_{n+3} = S^0_{n+2} \cdot S^0_{n+1} \cdot S^0_n \oplus S^0_{n+3} \quad (4)$$

| Cin | 0 | | Cin | 1 | |
|---|---|---|---|---|---|
| A | 1 1 0 0 | (12) | A | 1 1 0 0 | (12) |
| B | 0 1 0 0 | (4) | B | 0 1 0 0 | (4) |
| Output | 1 0 0 0 **0** | (16) | Output | 1 0 0 0 **1** | (17) |

(a)

| Cin | 0 | | Cin | 1 | |
|---|---|---|---|---|---|
| A | 0 0 0 1 | (1) | A | 0 0 0 1 | (1) |
| B | 0 1 1 0 | (6) | B | 0 1 1 0 | (6) |
| Output | 0 **1 1 1** | (7) | Output | **1 0 0 0** | (8) |

(b)

Figure 1. Example of Property 1 (a) and Property 2 (b)

Fig. 2 shows the structure of an 8-bit width version of the proposed A1CSA. The first 4-bit module uses a 4-bit wide CRA with free carry-in (just as in a conventional CSA). The second (and the next) 4-bit modules are composed by a single 4-bit CRA with carry-in fixed at '0', a four 2-to-1 muxes, the Add-One Block (A1B) and logic to compute the module propagation. Besides addition modules, a dedicated logic is responsible for selecting the correct sum result at the output of the second and higher order modules.

Fig. 3 presents the structure of a 16-bit width version of the proposed A1CSAH. As the A1CSA, the A1CSAH uses a single RCA per module. However, the carry selection is computed in a hierarchical manner, by a dedicated block identified as PGH. Similarly to hierarchical CLAs (CLAH), the PGH receives the "propagate" and the "generate" signals of the module. It is interesting to notice that, since each CRA has carry-in '0', the module "generate" equals to its carry-out signal. Differently from the A1CSA, the Add-One Block of the A1CSAH (A1BH) also incorporates the four 2-to-1 muxes.

## III. Results and Discussion

The proposed adder architectures, A1CSA and A1CSAH, were evaluated and compared against CSA and CLAH (carry lookahead adder) which are fast adder architectures that may be adopted in a cell-based design flow. The CRA was also included in the study to serve as reference.

In order to obtain reliable area, critical delay and power estimates, 8, 16, 32, 64, 128 and 256-bit wide adders were described in VHDL and synthesized for TSMC 45nm standard-cells library using Synopsys Design Compiler Topographical [9]. Synopsys Design Compiler (DC) mapped the CRAs by using the full adder (FA) cell available from the standard-cells library, thus providing the best possible CRA realizations within a fully automated design flow.

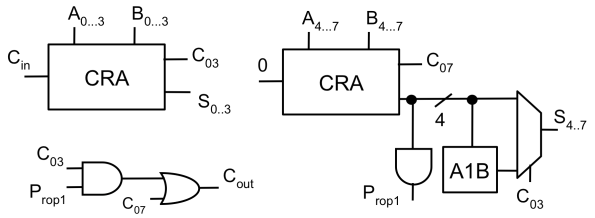


Figure 2. 8-bit A1CSA

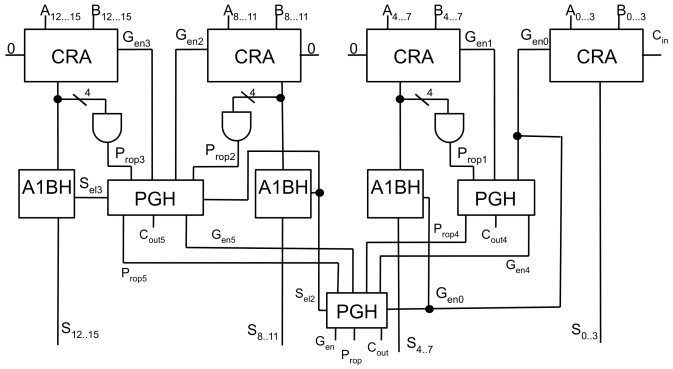


Figure 3. 16-bit A1CSAH

The fast adder architectures were designed by splitting the total bit width into 4-bit wide modules (i.e., m=4). In the case of CLAHs, the 4-bit CLAH module is described in VHDL by instantiating the VHDL descriptions of component sub-circuits: the generate, the propagate, each internal carry and the sums. Such VHDL organization allows one to restrict logic optimization and mapping to the bounds of each sub-circuit, thus preventing Synopsys DC from transforming a CLA into a CRA. For the remaining fast adders (CSA, A1CSA and A1CSAH), a 4-bit module uses one (two, in the case of CSA) 4-bit CRA, which was mapped by Synopsys DC by using FAs and half-adders (HAs) from the standard-cells library. The extra logic required to implement each adder was carefully described and synthesized so to prevent logic optimization from degenerating the structure of each type of adder architecture.

TABLE I shows area estimates provided by Synopsys Design Compiler (DC), in $\mu m^2$. As one would expect, the CRA requires the smallest amount of area among the investigated architectures. Among the remaining architectures, referred to as "fast adders", A1CSAH is the architecture requiring the smallest amount of area, followed by A1CSA. TABLE II compares A1CSA and A1CSAH to CSA and CLAH. The main result of this table is that A1CSA and A1CSAH uses low area for all comparisons since all results are lower than 1. From this table it can be seen that A1CSA requires, on average, 18% less area than CSA, whereas A1CSAH requires, on average, 26% and 23% less area than CSA and CLAH, respectively. Considering only 64 to 256 bit wide adders, A1CSA requires, on average, 19% and 9% less area than CSA and CLAH, respectively, whereas A1CSAH requires, on average, 27% and 18% less area than CSA and CLAH, respectively.

TABLE III shows critical delays (in ns) as estimated by Synopsys DC. As it would be expected, the CRA exhibits the worst delay for all bit widths (except 8). On the other hand, for 8 and 16 bit wide adders the CSA exhibits the shortest delays, whereas for 32 to 256 bit wide adders the CLAH exhibits the shortest delays, followed by the A1CSAH. The delay comparisons shown in TABLE IV reveals that the delay of A1CSA is, on average, 1% shorter and 133% longer than the delays of CSA and CLAH, respectively, whereas the delay of A1CSAH is, on average, 27% shorter and 11% longer than the delays of CSA and CLAH, respectively. For fast adders with bit width ranging from 64 to 256, the delay of A1CSA is, on average, 2% shorter and 258% longer than the delays of CSA and CLAH, respectively, whereas the delay of A1CSAH is, on average, 66% shorter and 6% longer than the delays of CSA and CLAH, respectively. However, it is important to notice that the wider are the adders, the smaller is the difference between the CLAH delay and the A1CSAH delay. Particularly, such difference achieves only 2% for 256 bit wide adders. A similar behavior can be observed with respect to CSA and A1CSA, except that the latter presents slightly shorter delays than the former for adders with 32 to 256 bits. Also worth of mention is the speedup of the A1CSAH over the CSA for 128 and 256 bit width: 69% and 81%, respectively.

TABLE I. CRITICAL DELAY OF ADDERS [ns]

| | Bit-width | | | | | |
|---|---|---|---|---|---|---|
| **Adder** | **8** | **16** | **32** | **64** | **128** | **256** |
| CRA | 0.25 | 0.45 | 0.88 | 1.65 | 3.18 | 6.33 |
| CSA | 0.20 | 0.28 | 0.50 | 0.88 | 1.63 | 3.11 |
| CLAH | 0.23 | 0.29 | 0.35 | 0.41 | 0.48 | 0.57 |
| A1CSA | 0.20 | 0.29 | 0.48 | 0.86 | 1.61 | 3.01 |
| A1CSAH | 0.26 | 0.35 | 0.40 | 0.45 | 0.51 | 0.58 |

TABLE II. CRITICAL DELAY COMPARISONS FOR FAST ADDERS

| **Comparison** | **Bit-width** | | | | | | **Aver.** |
|---|---|---|---|---|---|---|---|
| | **8** | **16** | **32** | **64** | **128** | **256** | |
| A1CSA/CSA | 1.00 | 1.04 | 0.96 | 0.98 | 0.99 | 0.97 | 0.99 |
| A1CSA/CLAH | 0.87 | 1.00 | 1.37 | 2.10 | 3.35 | 5.28 | 2.33 |
| A1CSAH/CSA | 1.30 | 1.25 | 0.80 | 0.51 | 0.31 | 0.19 | 0.73 |
| A1CSAH/CLAH | 1.13 | 1.21 | 1.14 | 1.10 | 1.06 | 1.02 | 1.11 |

TABLE III. ADDERS AREA [$\mu m^2$]

| **Adder** | **Bit-width** | | | | | |
|---|---|---|---|---|---|---|
| | **8** | **16** | **32** | **64** | **128** | **256** |
| CRA | 35.28 | 70.56 | 141.12 | 282.24 | 564.48 | 1128.96 |
| CSA | 56.98 | 135.65 | 293.00 | 607.70 | 1237.09 | 2495.88 |
| CLAH | 66.50 | 135.30 | 272.89 | 548.07 | 1098.44 | 2199.18 |
| A1CSA | 49,39 | 112.90 | 239.90 | 493.92 | 1001.95 | 2018.02 |
| A1CSAH | 43.75 | 101.78 | 215.38 | 442.59 | 896.99 | 1805.81 |

TABLE IV. AREA COMPARISONS FOR FAST ADDERS

| **Comparison** | **Bit-width** | | | | | | **Aver.** |
|---|---|---|---|---|---|---|---|
| | **8** | **16** | **32** | **64** | **128** | **256** | |
| A1CSA/CSA | 0.87 | 0.83 | 0.82 | 0.81 | 0.81 | 0.81 | 0.82 |
| A1CSA/CLAH | 0.74 | 0.83 | 0.88 | 0.90 | 0.91 | 0.92 | 0.86 |
| A1CSAH/CSA | 0.77 | 0.75 | 0.74 | 0.73 | 0.73 | 0.72 | 0.74 |
| A1CSAH/CLAH | 0.66 | 0.75 | 0.79 | 0.81 | 0.82 | 0.82 | 0.77 |

TABLE V. ADDERS POWER ESTIMATES [$\mu W$]

| **Adder** | **Bit-width** | | | | | |
|---|---|---|---|---|---|---|
| | **8** | **16** | **32** | **64** | **128** | **256** |
| CRA | 31.4 | 63.2 | 127.5 | 254.3 | 510.0 | 1032.1 |
| CSA | 42.2 | 94.1 | 200.0 | 409.9 | 835.0 | 1684.8 |
| CLAH | 40.6 | 80.3 | 160.6 | 321.4 | 646.4 | 1305.5 |
| A1CSA | 39.4 | 85.3 | 178.4 | 364.1 | 742.5 | 1497.3 |
| A1CSAH | 36.2 | 75.9 | 155.9 | 313.1 | 632.3 | 1277.4 |

TABLE VI. POWER COMPARISONS FOR FAST ADDERS

| **Comparison** | **Bit-width** | | | | | | **Aver.** |
|---|---|---|---|---|---|---|---|
| | **8** | **16** | **32** | **64** | **128** | **256** | |
| A1CSA/CSA | 0.93 | 0.91 | 0.89 | 0.89 | 0.89 | 0.89 | 0.90 |
| A1CSA/CLAH | 0.97 | 1.06 | 1.11 | 1.13 | 1.15 | 1.15 | 1.10 |
| A1CSAH/CSA | 0.86 | 0.81 | 0.78 | 0.76 | 0.76 | 0.76 | 0.79 |
| A1CSAH/CLAH | 0.89 | 0.94 | 0.97 | 0.97 | 0.98 | 0.98 | 0.96 |

TABLE V shows the average power consumption of each adder, expressed in $\mu W$, as estimated by Synopsys DC. The CRA is the adder architecture requiring the least power, followed by the A1CSAH and by the CLAH. TABLE VI shows that A1CSA consumes, on average, 10% less power than CSA and 10% more power than CLAH, whereas A1CSAH consumes, on average, 21% and 4% less power than CSA and CLAH, respectively. For adders with 64 to 256 bit width, the power consumption of A1CSA is, on average, 11% lower than that of CSA, and 14% higher than that of CLAH,

whereas the power consumption of A1CSAH is, on average, 24% and 2% lower than that of CSA and CLAH, respectively.

To establish a comparison in terms of energy efficiency, the power-delay product (PDP) of each adder was calculated. The PDP of an adder can be interpreted as the amount of energy it requires, to perform each addition. TABLE VII shows PDP values, in fJ. The CLA presents the lowest PDP values for bit widths ranging from 16 to 256. The A1CSAH presents the second lowest PDP values, considering bit widths from 32 to 256 (for 16 bit wide adders, the A1CSA as the second lowest PDP). The CRA has the highest PDP among all adders from 16 to 256 bits. TABLE VIII compares the PDP of fast adders. It shows that the PDP of A1CSA is, on average, 11% smaller than that of CSA and 162% greater than that of CLAH, whereas the PDP of A1CSAH is, on average, 41% smaller than that of CSA and 6% greater than that of CLAH. For the adders with bit width ranging from 64 to 256, one observe that the PDP of A1CSA is, on average, 13% smaller than that of CSA and 310% greater than that of CLAH, whereas the PDP of A1CSAH is, on average, 74% smaller than that of CSA and 4% greater than that of CLAH. It is important to notice that, the wider is the adder, the smaller is the difference between A1CSAH PDP and CLAH PDP. Indeed, this difference reaches only 4% for 126 bit wide adders. For 256 bit wide adders, the A1CSAH is as energy efficient as the CLAH.

## IV. CONCLUSION

This paper presented two Add-One Carry-Select Adders, A1CSA and A1CSAH, suited for cell-based VLSI design flow. Those architectures were synthesized for TSMC 45nm standard-cells library by using Synopsys Design Compiler Topographical. Ripple-carry adders (CRA), conventional carry-select adders (CSA) and hierarchical carry lookahead adders (CLAH) were also synthesized.

TABLE VII. ADDERS PDP [fJ]

| Adder | Bit-width | | | | | |
|---|---|---|---|---|---|---|
| | 8 | 16 | 32 | 64 | 128 | 256 |
| CRA | 7.9 | 28.4 | 112.2 | 419.6 | 1621.9 | 6533.2 |
| CSA | 8.4 | 26.4 | 100.0 | 360.7 | 1361.1 | 5239.8 |
| CLAH | 9.3 | 23.3 | 56.2 | 131.8 | 310.3 | 744.2 |
| A1CSA | 7.9 | 24.7 | 85.6 | 313.1 | 1195.5 | 4506.9 |
| A1CSAH | 9.4 | 26.5 | 62.4 | 140.9 | 322.5 | 740.9 |

TABLE VIII. PDP COMPARISONS FOR FAST ADDERS

| Comparison | Bit-width | | | | | | Aver. |
|---|---|---|---|---|---|---|---|
| | 8 | 16 | 32 | 64 | 128 | 256 | |
| A1CSA/CSA | 0.93 | 0.94 | 0.86 | 0.87 | 0.88 | 0.86 | 0.89 |
| A1CSA/CLAH | 0.84 | 1.06 | 1.52 | 2.38 | 3.85 | 6.06 | 2.62 |
| A1CSAH/CSA | 1.12 | 1.01 | 0.62 | 0.39 | 0.24 | 0.14 | 0.59 |
| A1CSAH/CLAH | 1.01 | 1.14 | 1.11 | 1.07 | 1.04 | 1.00 | 1.06 |

Results showed that A1CSAH requires, on average, 26% and 23% less area than CSA and CLAH, respectively, whereas A1CSA requires, on average, 18% and 14% less area than CSA and CLAH, respectively. Such area reduction comes from the use of an add-one logic (replacing one of the two 4-bit CRA modules encountered in conventional CSA).

For adders with bit width ranging from 64 to 256, critical delay estimates showed that A1CSAH is significantly faster than CSA but slightly slower than CLAH, whereas A1CSA is slightly faster than CSA but significantly slower than CLAH. Such behavior is a consequence of the carry logic used by each of those architectures. While A1CSA and CSA compute the carry with a factorized AND-OR logic, A1CSAH and CLAH do that in a hierarchical fashion. Such hierarchy makes its delay complexity $O(\log_k n)$ [3]. On the other hand, A1CSAH consumes less power than CSA and CLAH, whereas A1CSA consumes less power than CSA. Power-delay results showed that A1CSAH is significantly more energy-efficient than CSA and practically as energy-efficient as CLAH, whereas A1CSA is more energy-efficient than CSA.

Cell-based synthesis results showed that A1CSAH is an excellent alternative to CLAH and to CSA, since it requires less area and consumes less power. Additionally, for higher order adders (64 to 256 bits), A1CSAH provides about the same energy efficiency as CLAH, with a very small delay penalty, but with an important reduction in area. A1CSA, by its turn, surpasses CSA in all metrics for the whole bit range considered.

## REFERENCES


[1] J. M. Rabaey, A. Chandrakasan, B. Nikolic, Digital Integrated Circuits: a design perspective, 2nd ed., Upper Saddle River, N. J.: Prentice Hall, 2003, pp.559-586.

[2] M. D. Ercegovac and T. Lang, Digital Arithmetic, San Francisco: Elsevier Science, 2004.

[3] V. Oklobdzija, "High-speed VLSI arithmetic units: adders and multipliers," in Design of High-Performance Microprocessor Circuits, A. Chandrakasan, W. J. Bowhill and F. Fox, Eds. Piscataway, N. J.: IEEE Press, 2001, pp. 181-204.

[4] B. Amelifard, F. Fallah and M. Pedram, "Closing the gap between carry select adder and ripple carry adder: a new class of low-power high-performance adders," *ISQED 2005*, pp. 148-152.

[5] T. Y. Chang and M. J. Hsiao, "Carry-select adder using single ripple-carry adder," Electronics Letters, vol. 34, no. 22, pp. 2101-2103, Oct. 1998.

[6] Y. Kim and L. S. Kim, "64-bit carry-select adder with reduced area," Electronics Letters, vol. 37, no. 10, pp. 614-615, May 2001.

[7] Y. He, C.-H. Chang, and J. Gu, "An area efficient 64-bit square root carry-select adder for low power applications," *IEEE ISCAS 2005*, pp. 4082–4085.

[8] E. Mesquita, et al, "RIC fast adder and its SET-tolerant implementation in FPGAs," *IEEE FPL 2007*, pp. 638-641.

[9] Synopsys's Design Compiler User Guide, Version C-2009.06, June 2009.