Performance of Digital Adder Architectures in 180nm CMOS Standard-Cell Technology

Luca Pilato, Sergio Saponara, Luca Fanucci
Department of Information Engineering, DII, University of Pisa, Italy
luca.pilato@for.unipi.it sergio.saponara@unipi.it luca.fanucci@unipi.it

***Abstract* – In this paper, we present and compare the design and the performances of ten different implementations for a 16-bit adder in a 180nm CMOS standard-cell technology. Ripple carry adder, increment adder, triangle adder, uniform and progressive carry select adder, uniform and progressive carry bypass adder, conditional adder, ripple carry look ahead adder and hierarchical carry look ahead adder are taken into account. Every architecture is explained, highlighting the pros and cons. Finally, the results of area complexity, worst path timing and average power consumption for each implementation are shown.**

Keywords - CMOS standard-cell technology; adder architectures; adders performance comparison;

I. INTRODUCTION

In a digital ASIC design, a basic element for any kind of application is the adder. Adders are heavily used in data paths, ALU, DSP blocks, FIR and IIR filters, counters, timers, microcontrollers and processors [1]. The performance of the adder as a recurrent block, affects the global performance of the design. Thus, a comparison of the possible implementations for adders is mandatory in order to estimate the design performance. Depends on the target technology, the degree of freedom for the implementation moves on different level. Custom design can optimize the architectures at transistor level while semi-custom design can optimize the arrangement of the building blocks composing the adder. Since the scale of integration for digital ASIC growth up from VLSI to ULSI, a semi-custom approach like standard-cell technology is preferred [2]. The standard-cell design uses different primitive cells to define any architecture implementation. The performances depends on how the cells are connected together, forming the functional block. The analysis of a typical 16-bit adder as a reference block provides us the possible space solution for an application with the selected standard-cell technology.

Hereafter, Section II describes the analyzed adder architectures, showing the concept of some RTL schematics. We take into account some of the most common architectures [3][4][5][6][7] and an optimized design of them. In Section III we present the design and analysis strategies taken for the comparison. In Section IV we summarize the performance results and finally, conclusions are drawn in Section V.

II. ADDER ARCHITECTURES

A. Ripple Carry Adder

The architecture is based on a chain of full-adder cells (FA), Fig. 1. We refer to this adder as RCA. The complexity and the time delay is linear with the number of bits. It is considered the standard and the easier architecture that can be designed, in terms of area complexity, but also the slower one, in terms of propagation delay. The critical path is the carry chain, where each FA must wait the delay of the previous *cout* stage. Equation (1) gives the expected propagation delay of a 16-bit RCA, where t_{FAc} is the worst FA's path delay generating *cout* (*a* to *cout* for the first FA and *cin* to *cout* for others).

$$t_p \approx 16 t_{FAc} \tag{1}$$

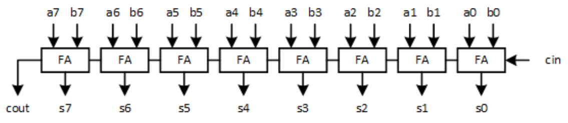

Figure 1. Ripple Carry Adder structure (8-bit)

B. Increment Adder

The increment adder, INCA from now, is the first optimization of the RCA. It splits in half a RCA. The two independent stages compute the least and the most significant part of the addition. An additional line of half adders (HA) solves the *cout* coming from the first stage, incrementing by one the most significant part, Fig. 2. The optimization comes from the reduction of the worst path delay given by the second part of the carry chain, which crosses through HAs instead of FAs. The expected delay for a 16-bit INCA is (2).

$$t_p \approx 8 t_{FAc} + 8 t_{HAc} \tag{2}$$

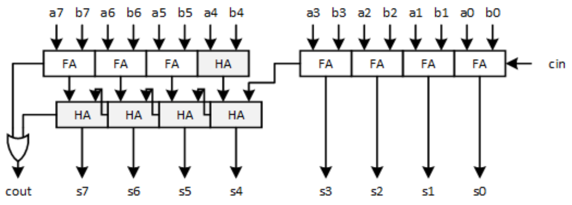

Figure 2. Increment Adder structure (8-bit)

C. Triangle Adder

It is a tree of HA, forming a triangle-like structure, referenced as TRIA. The number of levels of the tree is equal to the word length of the operands and the critical path is linear with the HA sum delay, t_{Has}, over the MSB column, see (3) for the delay of a 16-bit TRIA. Fig. 3 shows the RTL concept.

$$t_p \approx 16 t_{HAs} \tag{3}$$

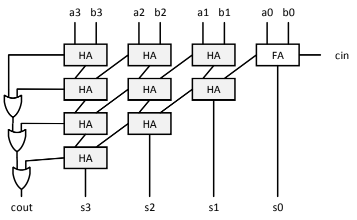

Figure 3. Triangle Adder structure (4-bit)

D. Carry Select Adder, with uniform partitions

The architecture is a further optimization of a RCA. It splits in uniform partitions the operands. Each part is elaborated with two parallel RCAs, excluding the first stage receiving the incoming *cin*. The duplicated RCAs process the sum with the possible incoming *cin* at zero and one. In this way, every sub-unit works concurrently and the result is selected with the previous sub-unit's *cout*. The critical path includes the first RCA and the next mux chain. A concept of carry select adder, from now CSELA-UNIF, is shown in Fig. 4. The partitioning of the considered 16-bit adder in this paper is with 4 groups, each of 4-bit length. The expected delay is (4).

$$t_p \approx 4 t_{FAc} + 3 t_{MUX} \tag{4}$$

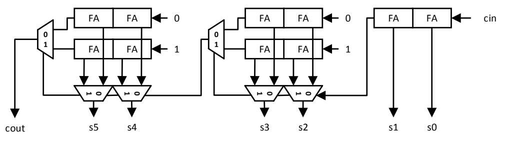

Figure 4. Carry Select Adder with uniform structure (6-bit)

E. Carry Select Adder, with progressive partitions

The structure follows the same concept of a carry select adder, but splits in progressive partitions the size of each sub-unit. The idea is to fill the delay time of the group's mux, using one additional FA in the next partition, improving the critical path. This type of adder is referenced as CSELA-PROG, Fig. 5. The partitioning of the considered 16-bit adder is with 5 groups, respectively of 2-2-3-4-5 bit length. The expected delay is (5).

$$t_p \approx 2 t_{FAc} + 4 t_{MUX} \tag{5}$$

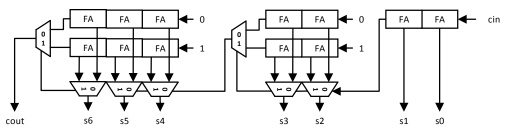

Figure 5. Carry Select Adder with Progressive structure (7-bit)

F. Conditional Adder

The conditional adder, COND from now, is one of the faster in term of speed. It is a structure derived from the carry select, where each FA is a sub-unit. Every result is grouped and selected in a tree of mux, managing the possible 0-*cin* and 1-*cin* sum. Typical size of the mux tree is a 2's power. The s_0 bit comes from the first FA, while its *cout* selects the next result, s_1. The *cout* of this group selects the results of the next 2-bit group, $s_3\, s_2$. Again, the *cout* of this 2-bit group selects the results of the next 4-bit group, and so on… The critical path involves the first FA and the chain of then final mux in the tree, e.g. only 4 mux in a 16-bit structure, (6). The cons is the high complexity in terms of area. A COND concept is shown in Fig. 6, highlighting the 0-paths in blue and the 1-paths in red.

$$t_p \approx t_{FAc} + 4 t_{MUX} \tag{6}$$

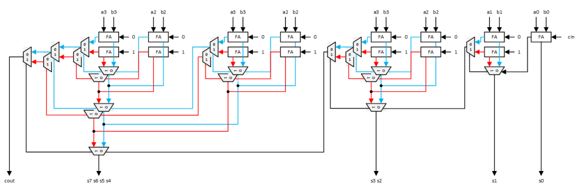

Figure 6. Conditional Adder structure (8-bit)

G. Carry Bypass Adder, with uniform partitions

The carry bypass, also known as carry skip, splits in many parts the operands. For each group it uses a RCA for the partial results, plus some addition logics to evaluate a special bypass function. Starting from (7) the carry generation, G, and propagation, P, functions are defined for each bit position. Then (8) gives the *cout* expression, e.g. in a group of 3-bit length.

$$G_i = a_i b_i \quad P_i = a_i \oplus b_i \tag{7}$$

$$cout = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 cin \tag{8}$$

$$BP = P_2 P_1 P_0 \tag{9}$$

We consider the and-ed propagation functions in the last term of (8) as the bypass condition, (9). When BP=1, the *cin* is propagated as the sub-unit's *cout*, otherwise, *cout* is generated inside the RCA. The BP function is evaluated in parallel for each group. Then, it selects the internally or bypassed *cout* for the next part. This adder, from now CBYPASS-UNIF is shown in Fig. 7. The critical path includes the first RCA, with BP=0, and others with BP=1. Hence, the last group has to wait all the BP mux before solves the last RCA delay. The considered 16-bit adder is with 4 groups of 4-bit length. The expected delay is (10).

$$t_p \approx 8 t_{FAc} + 3 t_{MUX} \tag{10}$$

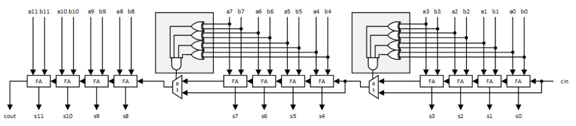

Figure 7. Carry Bypass Adder with Uniform structure (12-bit)

H. *Carry Bypass Adder, with progressive partitions*

Considering the critical path of the bypass structure, the first and last group are always included, as the bypass mux. An optimization is a progressive partitioning, we call it CBYPASS-PROG. Starting and ending with a reduced size group we increase their size until the middle point. Due to the fixed size of our 16-bit adder, the design partitioning will approximate this rule with 1-2-2-3-3-2-2-1 grouping. The expected delay is (11).

$$t_p \approx 2t_{FAc} + 7t_{MUX} \tag{11}$$

I. *Carry Look ahead Adder, with ripple structure*

Using (7) and the generalization of (8) it is possible to computes each *cout* of any bit stage in an addition, without waiting the previous *cout*. Splitting the operands, again, in 4 uniform partitions, and using the pre-evaluated *cout,* each sub-unit is able to get the summing result with a simple *xor*, (12).

$$s_i = cout_{i-1} \oplus P_i \tag{12}$$

The cell for the generation of G_i and P_i signals is a half adder (HA). The block for the $cout_i$ generation is a 4-bit carry look ahead generator (CLG4). The 16-bit carry look ahead adder with ripple structure, from now CLA-RIPPLE, is composed by a series of four CLG4 blocks, Fig. 8. The critical path starts from the first HA and cross through the *cin-cout* chain of the CLG4 blocks, (13).

$$t_p \approx t_{HAs} + 4t_{CLG4} \tag{13}$$

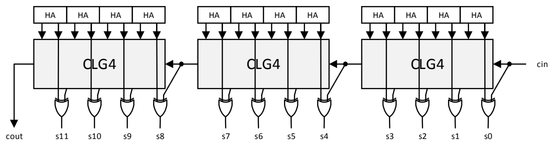

Figure 8. Carry Look ahead Adder with Ripple structure (12-bit)

J. *Carry Look ahead Adder, hieararchical structure*

Starting from a CLG4 block it is possible to group the propagate signal and generate signal to a higher level of carry look ahead generator. Equations (14) and (15) are an example.

$$G_0{*} = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 \tag{14}$$

$$P_0{*} = P_3P_2P_1P_0 \tag{15}$$

The first level of four CLG4 are now transformed in CLG4* for the P* and G* block signals generation. The second level of carry look ahead generator is a standard CLG4 accepting the P* and G* signals. The resulting architecture is a 2-level hierarchical structure, CLA-HIER, Fig. 9. The critical path now cross the first HA, then one stage of the CLG4*, the CLG4 of the 2-level and came back to the last CLG4*, (16).

$$t_p \approx t_{HAs} + 3t_{CLG4} \tag{16}$$

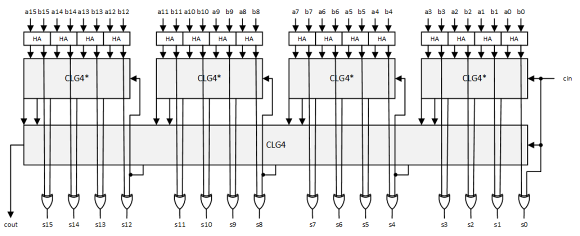

Figure 9. Carry Look ahead Adder, Hierarchical structure (16-bit)

III. Design Analysis Strategies

Since we propose the analysis of the architectures for a generic data path optimization of a standard cell design, the adder under test is placed between two potential pipeline registers. Fig. 10 shows the test bench concept used for the simulations.

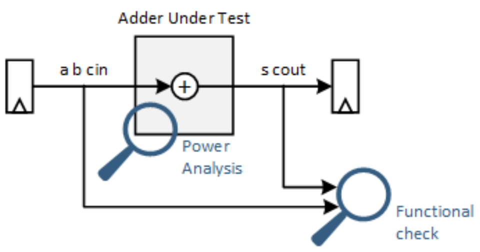

Figure 10. Test bench for design analysis

This test bench architecture guarantees the same condition of fan-in, fan-out and electrical parameter for every considered adder implementation. Starting from a VHDL-RTL description of the blocks the analysis flow is:

RTL simulation, to check the functional correctness.

Gate level synthesis. The synthesis constraints reflect the test bench concept. Input driving strength and output load comes from D-FlipFlop parameters. In addition, we preserve the hierarchical structures, to get the synthesized netlists as close as the RTL concept. The synthesis tool (Cadence Encounter® RTL Compiler) gives us the area and timing performance. The standard cell technology used for the synthesis is the 180nm CMOS @1.8V. The PVT (Process, Voltage and Temperature) corner for synthesis is set to the worst operating condition: worst technology process parameter, 90% of supply voltage, and T=150°C.

Formal verification and gate level simulations, to guarantee the logic equivalence between RTL and Gate level codes.

Power consumption analysis, using random stimuli as input vectors to the gate level netlist. The length of the stimuli guarantees switching nets coverage of 92% in the worst case and up to 100% in best case. A safe clock frequency for the power simulation is selected @10MHz. Power estimation is done saving the VCD (Value Change Dump) file for the whole gate level netlist and elaborating the switching activities with the test bench parameters with Synopsys PrimeTime®.

IV. Performance Comparison

The synthesis results in terms of standard cells area, critical path propagation delay and average power consumption are shown in Tab. I. Column 1 shows the adder references. Column 2 lists the area in terms of absolute values [µm2] and NAND2 gate equivalent [GE] complexity. Column 3 presents the worst path delay [ps]. Column 4 is the average power consumption [µW] with a clock speed @10MHz. The same synthesized architectures were also compared using an older technology, the CMOS 350nm @3.3V. For a sake of space, reported results refer only to the 180nm technology but the analysis prove the CMOS property of scalability, given a scale factor for the 350nm of about x4.5 for the Area, x2 for Timing and x10 for the Power consumption.

TABLE I. Synthesis performance results

CMOS 180nm std-cell @1.8V				
16 bit Adder	Area		Timing	Power
	[µm2]	[GE]	[ps]	[µW] @10MHz
RCA	903	80	8764	5.134
INCA	1140	101	7413	6.369
TRIA	4332	383	9565	21.07
CSELA-UNIF	1809	160	5198	11.64
CSELA-PROG	1947	172	5095	12.74
COND	2893	256	4755	24.93
CBYPASS-UNIF	1301	115	5818	7.417
CBYPASS-PROG	1482	131	4432	9.02
CLA-RIPPLE	1129	100	8189	8.63
CLA-HIER	1439	127	5498	11.66

The results summary is plotted in Fig. 11. The x-axis is the area complexity [µm2]. The y-axis is the worst path delay [ps]. The size of the markers is proportional to the power consumption of Tab. I.

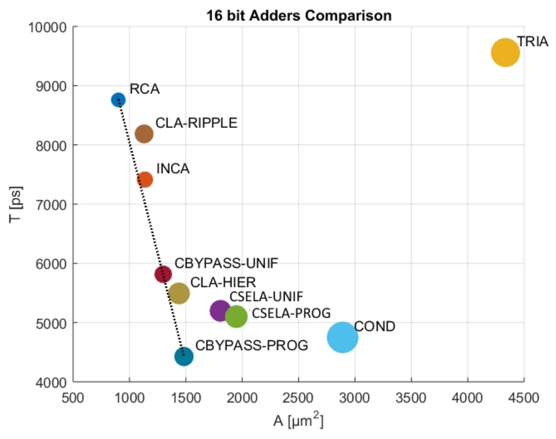

Figure 11. Performance chart

Excluding the COND and TRIA adders, all the absolute power consumptions are approximately around 10µW. This means a computational power up to 100Madd operations per second for a power cost of 100µW. In addition, we can trace the Pareto fronts of the Area-Timing trade off [8]. We can consider the RCA, the INCA, the CBYPASS family and the CLA family adders as the best choice for a design solution. In order to validate this affirmation, Fig. 12 shows the ATP figure of merit: the product of Area Time and Power, for each architecture. Lower is the ATP factor, better is the global trade off for the adder. Sorting the ATP factor, we can see the same results of the Pareto fronts indication.

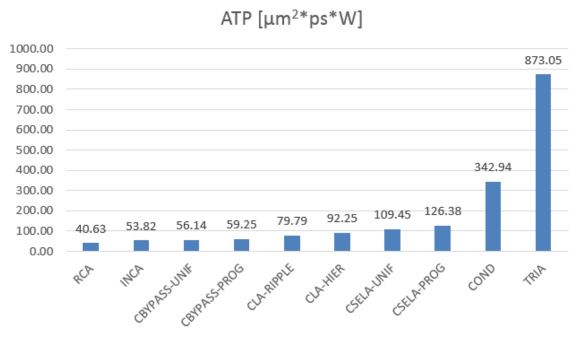

Figure 12. Adders by ATP figure of merit

V. Conclusion

This paper analyzes the real performance in terms of area complexity, critical path timing and average power consumption of ten different 16-bit adder architectures. Ripple carry, increment, triangle, uniform and progressive carry select, uniform and progressive carry bypass, conditional, ripple carry look ahead and hierarchical carry look ahead adders are taken into account. Using the 180nm CMOS standard-cell technology @1.8V, we collect the design performances of the proposed architectures. The best results show a complexity between 80-131GE for the area and 4.4-8.7ns for the maximum propagation delay with average power consumption around 10µW at 10MHz of clock speed. As a good trade-off in terms of area, speed and power the RCA, INCA, CBYPASS and CLA can ensure the sub-optimum design choices.

References

[1] Weste Neil, Harris David, "CMOS VLSI Design: A Circuits and Systems Prospective", 4th edition, Pearson, 2010.

[2] Chinnery David, Keutzer Kurt, "Closing the Gap Between ASIC and Custom", 2002 Springer US.

[3] Saini Jasmine, et al., "Performance, analysis and comparison of digital adders" Computer Engineering and Applications (ICACEA) pp.80-83 2015.

[4] Burgess Neil, "Fast Ripple-Carry Adders in Standard-Cell CMOS VLSI" IEEE Symposium on Computer Arithmetic pp.103-111 2011.

[5] Uma Ramadass, et al. "Area, Delay and Power Comparison of Adder Topologies" VLSICS Journal vol.3, no.1, pp.153-168, Feb. 2012.

[6] Seok-Won Heo, et al., "Study of optimized adder selection" ASIC vol.2 pp.1265-1268 2003.

[7] Kaur Jasbir, et al., "Comparison Between Vatious Types of Adder Topologies" IJCST Journal vol.6, no.1, Jan-March 2015.

[8] Ascia G., et al., "A framework for design space exploration of parameterized VLSI systems" in Design Automation Conference, 2002.