POLITECNICO DI TORINO

NTNU

Norwegian University of Science and Technology

Electronic Engineering Degree

Embedded System Specialization

Master's Thesis

Sub-Threshold Design of Arithmetic Circuits: when Serial might overcome Parallel Architectures

Student:

Simone FINI

Italian Supervisor:

Prof. Luciano Lavagno

Norwegian Supervisors:

Prof. Snorre Aunet

Prof. Trond Ytterdal

Un sincero ringraziamento a tutti i miei relatori:
Prof. Luciano Lavagno, per aver accettato di seguire il mio lavoro da molti kilometri di distanza.
Prof. Snorre Aunet, la prima persona in assoluto a cui mi sono rivolto prima di partire e l'ultima che ho salutato prima di tornare.
Prof. Trond Ytterdal, per la professionalità, la presenza e l'aiuto costante.

A sincere thanks to all my supervisors:
Prof. Luciano Lavagno, for having accepted to follow this thesis from miles away.
Prof. Snorre Aunet, the first person I asked questions to and the last one I greeted before coming back.
Prof. Trond Ytterdal, for his professionalism and the constant presence and help.

Contents

List of Figures

List of Tables

Abstract

Adder circuits are vital for microprocessors; indeed, apart from the addition itself, either subtraction, multiplication or division algorithms may require, at a certain point, the addition of two (partial or not) operands. For this reason, several architectures have been studied and improved over the last decades, in order to speed up the aforementioned operation.

On the other hand, it is well known that having faster circuits means higher complexity, and, therefore, higher power consumption. In addition to this, the downscaling process of transistors has increased the leakage current of these devices, accounting for up to 33% of the total dissipation [10], and due to the little capabilities of batteries with respect to the achievable performance of circuits, the main challenge of engineers and designers is represented by exploiting low power techniques so as to decrease the power consumption of electronic devices as much as possible.

This Master's Thesis work wants to demonstrate that, when working in sub-threshold region, it might be possible to employ simple and repetitive circuits, like ripple carry adders, instead of complex ones, such as Kogge-Stone architectures, having the same propagation time but with a significantly lower energy consumption. In this way, it would be possible to have, at the same time, the performance given by a fast adder and the area and energy dissipation of the simpler and weaker "anchestor".

As an anticipation, and as it will be seen in the final results, the technology employed and the choice of the best available architecture resulted in a great improvement with respect to the study previously conducted [2].

First of all, with the development of a new full adder circuit (the so called "XMAJ3"), it is possible to reduce the energy consumption with respect to ripple carry adders based on both already existing architectures and on the full adder cell contained in the library. This even without the employment of customized gates, but only with standard logic blocks already contained in the library.

Secondly, FDSOI technology makes possible to equalize performance of serial and parallel adders and, at the same time, saving energy, even in super-threshold region, allowing to avoid all the problems that sub-threshold design brings. Particularly, for 32-bit based devices, the average energy saving with respect to the Kogge-Stone adder accounts for 41.48% (with a peak of 56.16%), while for 64-bit adders the mean saving is 50.02%, with a maximum of 56.83%.

1 Introduction

1.1 Motivations

This piece of work wants to be the natural continuation of a study by Prof. Valeriu Beiu, Asbjørn Djupdal and Snorre Aunet, where a serial ripple carry adder and a parallel Kogge-Stone structure were analyzed when operating in sub-threshold region at 100nm and 70nm.
As properly explained in the paper, after several simulations, it was confirmed that wires play a significant role, reducing the speed advantage of the parallel adder from 4.5x to 2.2x-2.4x [2] due to the parasitics that wires introduce. In addition to this, it was noticed that, when running at the same speed, the RCA was more energy efficient than the parallel adder, as it can be seen from Figure 2.

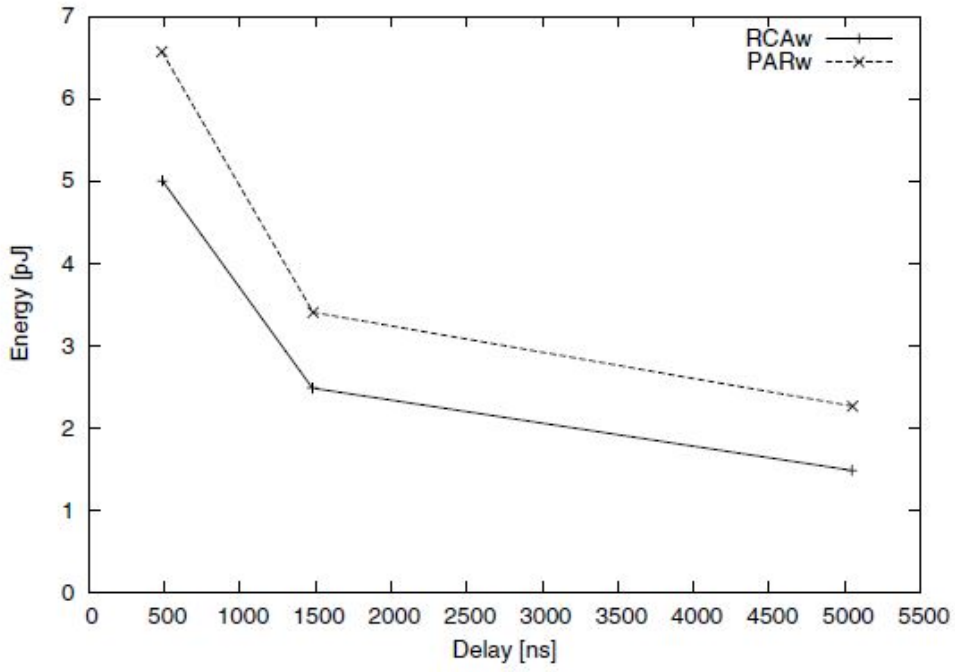


Figure 2: Energy used by 32 bit adders (70nm) [2]

However, this research presents some points that, as suggested in the paper itself, might be improved:

- **Technology Nodes:** back to 2005, the latest available technology node

reached was 90nm, provided by leading semiconductor companies like Intel, AMD, Infineon, IBM [3], while nowadays 7nm technology has been largely spread. In addition to this, FinFET and FDSOI transistors are commonly used, and, therefore, the aforementioned results might not be valid anymore for very fast and little leaky devices.

- **BPTM:** instead of a real technology library, Berkeley Predictive Technology Models were used for all transistors, generated with the parameters suggested in [4]. These did not simulate the behavior of devices under test on account of real physical parameters provided by foundries, but, using statistics about previous technologies, tried to predict what might be the future outcome.

- **Interconnects:** as well as for transistors, all the wires were modeled not taking into account real parameters after place and route, but employing the so called four-segment π model.

  This representation estimates the delay introduced by wires through

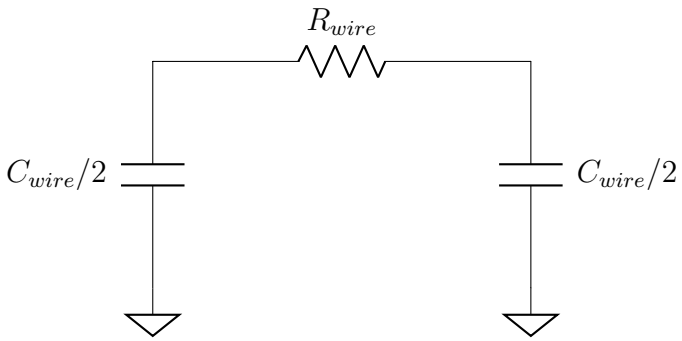


Figure 3: π model

the following formula:

$$\tau = \frac{R_{wire}C_{wire}}{2} = \frac{rcL^2}{2}, \tag{1}$$

where r and c are, respectively, the resistance and capacitance per unit length, while L is the actual length of the interconnection.

However, the main problem of this solution is the choice of the most suitable values for these variables; in fact, as described in [5], several

rules of thumb exist, but since parasitic parameters depend on the kind of metal, the length/distance of wires, their width, their height and so on, the result will always be approximative and will not reflect the real physical effects.

- **Circuit Implementations:** as previously mentioned, two 32-bit structures were investigated, and both the devices were built using the "MIN3" gate (shown in Figure 4) as the basic logic gate.

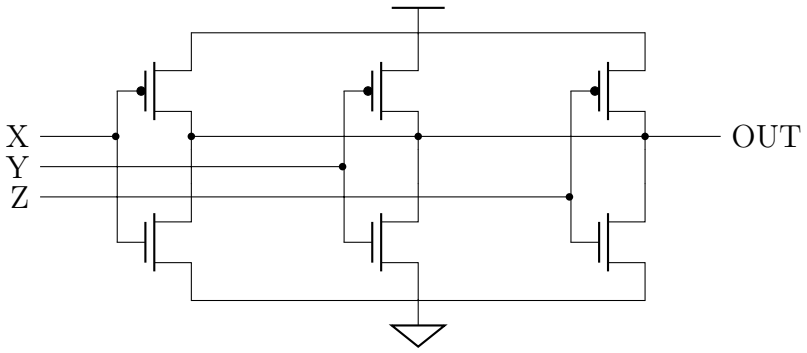


Figure 4: "MIN3" gate

X	Y	Z	OUT
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Table 1: "MIN3" truth table

However, this kind of gate implementation presents some drawbacks, like the fact that it has a high static power consumption due to the possible short-circuit paths that would be present, for instance, in case of

two inputs are high and the third is low; moreover, its functionality and performance heavily depend on the ratio between transistor dimensions.

Having seen all of this, in this thesis work it has been tried to overcome and/or improve the aforementioned aspects, in ways that will be shortly described in the next sub-chapter.

1.2 Thesis Outline

This thesis is organized in the following manner: chapter 2 presents the relevant theoretical background needed in order to analyze and comprehend the subsequent parts, while in chapter 3 a study about 1-bit full adders is detailed, presenting the newly proposed architecture. In addition, in order to understand the goodness of it, simulations of all the previously described structures (and RCA circuits) were carried out, and comparisons in terms of timing, power and energy consumption are shown. chapter 4 describes the way in which RCAs and KS adder are simulated: in fact, instead of applying a unique transition and collecting data retrieved from it, a different technique was employed, since it was tried to have final results that represented the real average values of the different parameters with a higher accuracy.
Finally, Chapter 5 and 6 show, respectively, the place and route of both architectures and the final simulations, while in chapter 7 the derived conclusions are discussed.

2 Theory

The title of this Master's Thesis directly involves two different kinds of arithmetic circuit architectures: serial and parallel...but what is the difference between them? And what about their benefits and drawbacks?

2.1 Serial Architecture (RCA)

When talking about serial arithmetic units, and particularly for this case of study, adders, it means that the entire design starts from a single basic block, which is the so called full adder.

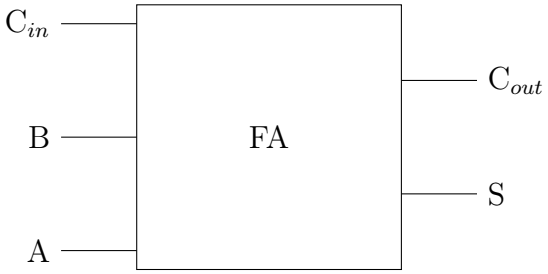


Figure 5: Full adder basic block

A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 2: Full adder truth table

Then this specific circuit is repeated as many times as the required number of bits and each carry-out signal is connected to the subsequent carry-in, as shown in Figure 6. In this way, a serial adder works on each pair of bits (and any carry) at a time; this means that even though the two inputs (which in Figure 6 are "A" and "B") are applied at the same moment, the operation at each stage is completed only when the carry-out bit from the previous stage has been computed.

As a consequence of this kind of structure, the carry propagation (also called rippling through) limits the speed with which two numbers are added, and the output of any such adder arrangement will be correct only if signals are given

enough time to propagate through gates connected between input and output.

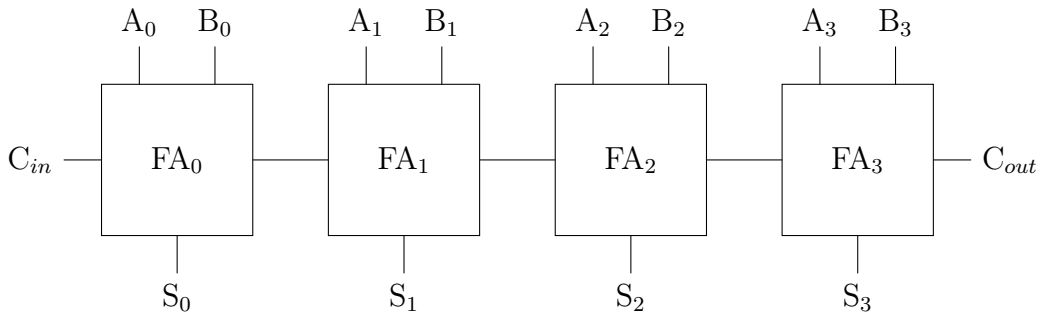


Figure 6: RCA structure

Particularly, the worst case occurs when a carry propagates from the LSB to the MSB, which makes possible to express the critical path delay in the following way:

$$t_{delay} = (N-1)t_{carry} + t_{initial}, \tag{2}$$

where t_{carry} is the delay of a single FA corresponding to the generation the carry-out from when all the three inputs are available, while $t_{initial}$ is the moment when the initial carry-in is applied to the input pin.
One of the possible methods to reduce the carry propagation time is to use faster logic gates, but at a certain point there is a limit below which the gate delay cannot be reduced. For this reason, hardware based on the concept of look-ahead carry have been developed and commonly employed.

2.2 Parallel Architecture (CLA and Kogge-Stone Adder)

2.2.1 Carry Look Ahead Adders

The idea behind carry look-ahead adders is to compute carry signals almost simultaneously, in order to avoid wasting time waiting for intermediate carries to propagate from a full adder stage to the subsequent one.
However, in order to be able to understand how this circuits work, it is essential to introduce two terms: "propagate" and "generate". Specifically, these functions are defined in the following way:

$$p_i = a_i \oplus b_i \longrightarrow propagate \tag{3}$$

$$g_i = a_i \cdot b_i \longrightarrow generate, \tag{4}$$

and it is immediately possible to notice that neither the propagate nor the generate signal depends on any carry bit, but only on the input data.
So, keeping in mind the standard expression for the carry-out signal from a 1-bit full adder:

$$C_{out} = A \cdot B + C_{in} \cdot (A \oplus B), \tag{5}$$

it is straightforward to obtain the new expression for the carry-out:

$$C_{i+1} = g_i + p_i \cdot C_i. \tag{6}$$

Then, upon successive substitutions, it is possible to express every carry-out signal as a function of the initial carry only, as shown below:

$$C_1 = g_0 + p_0 \cdot C_0 \tag{7}$$

$$\begin{aligned} C_2 &= g_1 + p_1 \cdot C_1 \\ &= g_1 + g_0 \cdot p_1 + p_0 \cdot p_1 \cdot C_0 \end{aligned} \tag{8}$$

$$\begin{aligned} C_3 &= g_2 + p_2 \cdot C_2 \\ &= g_2 + g_1 \cdot p_2 + g_0 \cdot p_1 \cdot p_2 + p_0 \cdot p_1 \cdot p_2 \cdot C_0 \end{aligned} \tag{9}$$

$$\begin{aligned} C_4 &= g_3 + p_3 \cdot C_3 \\ &= g_3 + g_2 \cdot p_3 + g_1 \cdot p_2 \cdot p_3 + g_0 \cdot p_1 \cdot p_2 \cdot p_3 + p_0 \cdot p_1 \cdot p_2 \cdot p_3 \cdot C_0 \end{aligned} \tag{10}$$

$$\ldots$$

$$\begin{aligned} C_n &= g_{n-1} + p_{n-1} \cdot C_{n-1} \\ &= g_{n-1} + g_{n-2} \cdot p_{n-1} + g_{n-3} \cdot p_{n-2} \cdot p_{n-1} + \ldots + \\ &\quad + g_0 \cdot p_1 \cdot \ldots \cdot p_{n-1} + p_0 \cdot \ldots \cdot p_{n-1} \cdot C_0. \end{aligned} \tag{11}$$

At the same time, also the sum signal can be expressed as function of the propagate signal, since:

$$S_i = A_i \oplus B_i \oplus C_i = p_i \oplus C_i, \tag{12}$$

which means that the final sum will be computed by xoring the propagate signals with the carries retrieved from the previous equations.

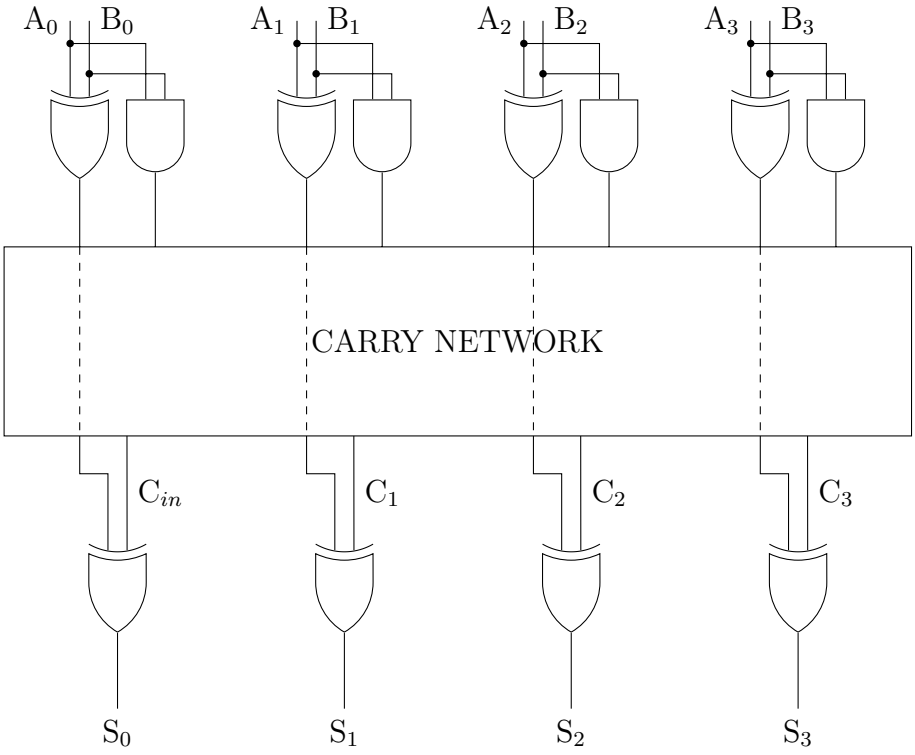


Figure 7: Generic CLA structure

Due to these particular properties, it is then possible to define three blocks, which will compose every kind of CLA structure:

- **PG Network:** a set of "AND" and "XOR" gates that is in charge of generating the p_i and g_i signals from the inputs of the adder;
- **Carry Network:** the block that aims to generate the carry signals from the PG network previously mentioned; talking about this, there are two different possibilities (as can be seen from equation 11): either every carry is generated from the initial carry-in (and in this case the complexity increases with the number of bits) or the dependency of C_n on C_{n-1} is exploited. In the last case, the structure is pretty regular, while the mechanism of rippling the carries decreases the performance, keeping the delay linear with the number of bits [6];

- **Sum Network:** the final structure of the adder, which computes the sum between the propagate signals and the carries.

2.2.2 Tree Adders

The set of solutions that may be adopted in the implementation of a CLA structure is pretty wide, and the main difference mostly consists of the way the carry network is designed.
A specific family is called "tree adders", and is basically based on two structures that implement the following functions:

$$G_{i:j} = G_{i:k} + P_{i:k} \cdot G_{k-1,j} \tag{13}$$

$$P_{i:j} = P_{i:k} \cdot P_{k-1,j}, \tag{14}$$

which must satisfy some specific properties:

- i $\geq$ k $>$ j;
- $G_{x:x} = g_x$;
- $P_{x:x} = p_x$;
- $g_0 = C_{in}$;
- $p_0 = 0$.

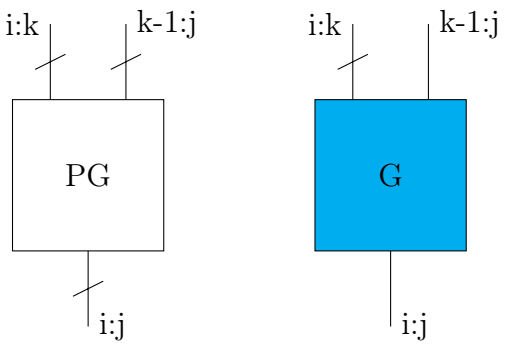


Figure 8: PG and G blocks

Thanks to these structures, the computation delay is turned from linear into logarithmic with n, ensuring faster computation and smaller depth.

In particular, the tree adder circuit taken into account in this work is the so called Kogge-Stone, which is well known for being considered as the fastest adder with the minimal logic depth and small fanout for each stage, increasing performance for typical CMOS process nodes [7], [8].

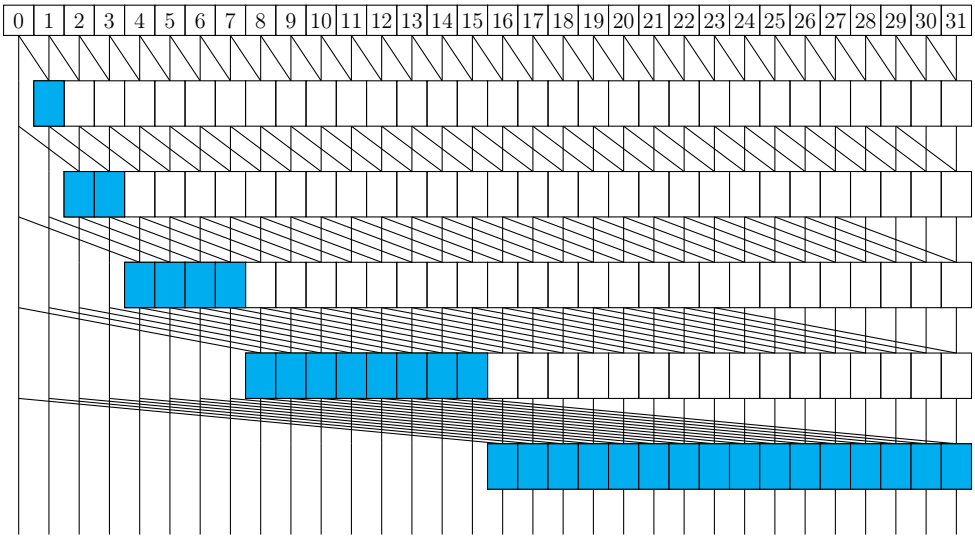


Figure 9: Kogge-Stone PG and Carry networks

On the other hand, as it can be seen from Figure 9, the main drawback of this kind of structure is the wiring congestion, which increases the area needed to built the circuit and its power consumption (together with the increased number of gates with respect to a simpler RCA solution).

2.3 Sub-threshold Design

Having talked about the different kinds of adder circuits that have been considered in this Thesis, now it is time to analyze the initial part of the title of this work: sub-threshold design.

2.3.1 Why Sub-Threshold?

Over the past few years, the extreme growth of portable devices like cellular phones, wireless receivers, communication devices, medical applications and so

on, has significantly expanded the demand for power sensitive design. In addition to this, many times the main requirement for this aforementioned technological equipment is ultra low-power consumption with medium frequency of operation (on the order of magnitude of tens to hundreds of MHz) [13].
For these reasons, and also due to the fact that well-known methods of low-power design may not be sufficient, the design of digital sub-threshold logic has been investigated and gained more interest in recent years.
In addition, the behavior of standard cell libraries, when employed for sub-threshold, has been studied, leading to the fact that already existing libraries do not use much extra energy, and so they provide good solutions for sub-threshold operations [14], allowing many more companies to exploit this kind of technique even though they do not own full custom technologies.

2.3.2 Leakage Current and its Exploitation

First of all, every transistor, due to its construction, has a specific threshold voltage, which, theoretically, is the limit that separates the ideal "non-conducting zone" from the "beginning of conduction". However this is just a rough approximation of the real behavior of these devices: in fact in 1955 Garrett and Brattain [9] mentioned for the first time that, even though the applied voltage has a lower value than the threshold one, transistors still present a very weak drain current, known as leakage current. In any case, back to that period, this kind of current was not a main concern, due to the fact that it was just a small amount with respect to the total magnitude of the saturation current.
As technology scaling continued, voltage supply went on being reduced, starting from 5V down to 1V and below (in order to have a comparison with the present, the standard cell library employed for this piece of work has 0.8V as nominal power supply); however, as the magnitude of saturation current has been decreasing with the supply voltage and the channel length, the leakage current has had an opposite behavior, as shown in Figure 10.

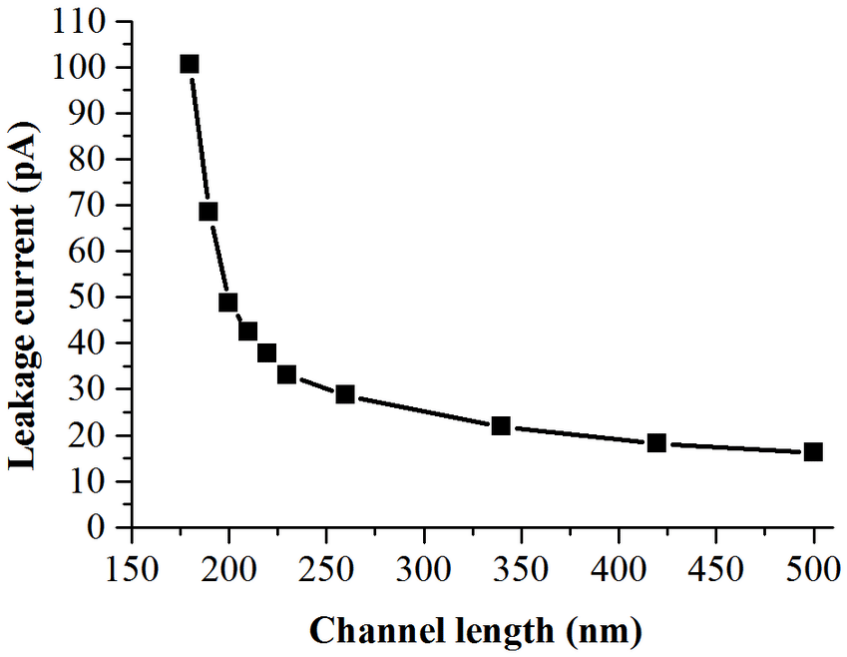


Figure 10: Leakage current with respect to channel length [10]

In addition to this, designers always look for ways to reduce unwanted components of power consumption, since in most of the cases power demanding is pretty high (due to many more components embedded on the same systems like antennas, integrated GPUs and so on), while the life of current batteries is still limited if compared to the possible performance achievable.

Possible solutions might be techniques either applied at architectural level, such as clock and/or power gating, variable frequency domains, precomputation design methods, or processes like multi V_{DD}, Dynamic Voltage and Frequency Scaling, multi and/or variable V_{TH}. However, some of these solutions come at the expense of performance, reliability, chip area, or several of these.

Another possibility, which differs from the previous one since does not require additional hardware and, so, does not affect area and complexity of the chip itself, consists of applying what is called sub-threshold logic: in this case, V_{DD} of the circuit is set at a value lower than or equal to the threshold voltage of that particular process technology. Doing this, transistors will be "switched off", and the leakage current, instead of being only a factor for static power

consumption, will be used for computation.

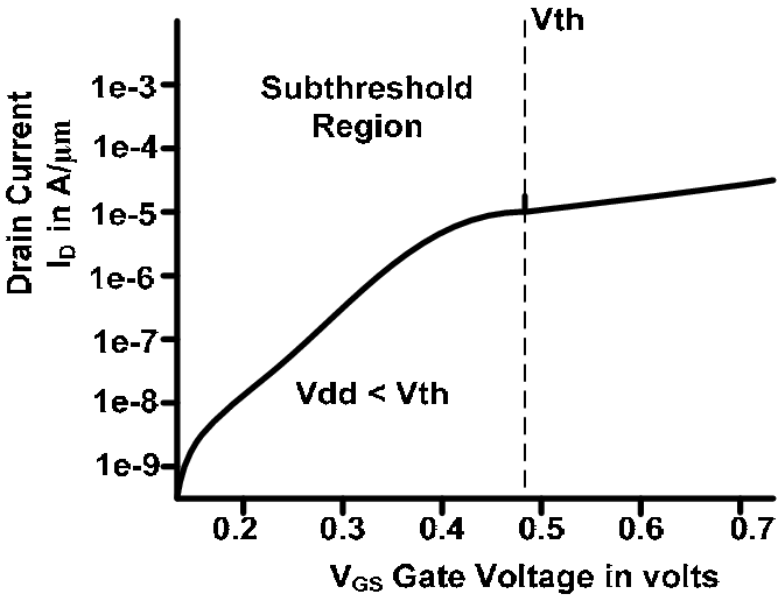


Figure 11: Behavior of drain current with respect to V_{GS} [11]

In fact, as it can be seen from 15, leakage current is proportional to V_{GS} and V_{DS}, still allowing to distinguish a logic '0' from a logic '1'.

$$I_{sub-threshold} = I_0 e^{\frac{V_{GS}-V_{TH}}{\eta V_{TH}}} \left(1 - e^{-\frac{V_{DS}}{V_{TH}}}\right), \tag{15}$$

where I_0 is the drain current.
In this way, dynamic power consumption has the best improvement, due to its direct dependency on the square of V_{DD}.

$$P_{dynamic} = \alpha f C V_{DD}^2 \tag{16}$$

On the other hand, sub-threshold logic presents some drawbacks: first of all, making V_{DD} lower reduces the current significantly, which does not allow circuits to charge load capacitances fast enough. Due to this slow charging of loads, components become slow, as it is possible to see from Figure 12, which represents the simulation of a generic standard cell, taken from the employed standard library, when the supply voltage varies from 0.8V down to 0.2V.

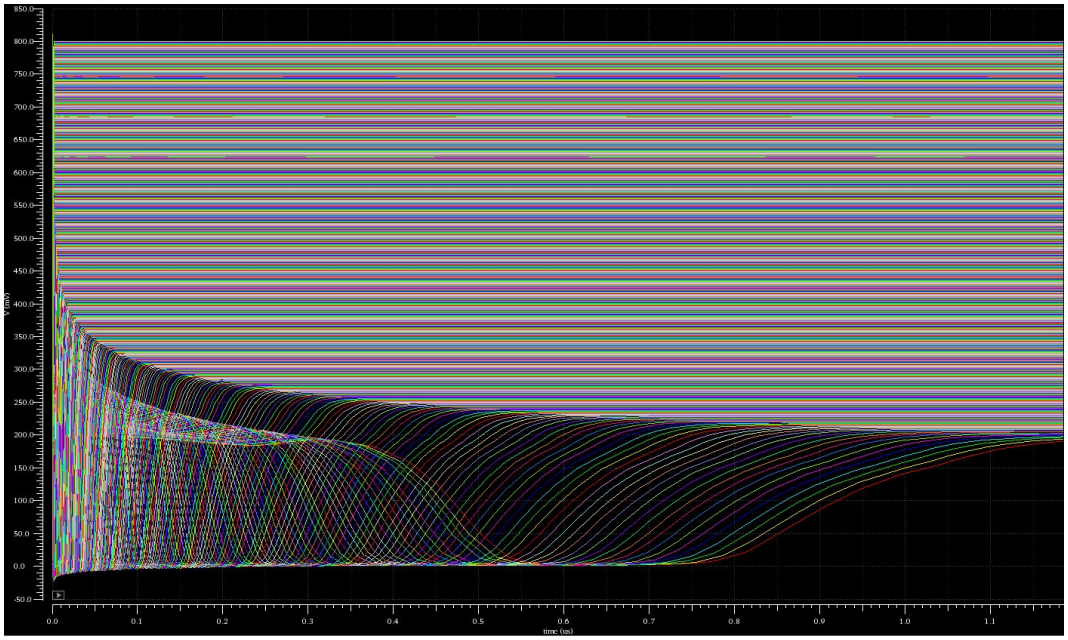


Figure 12: Sub-threshold behavior of a generic standard cell

At the same time, the output response of a transistor in sub-threshold region is subtle, and detecting it requires circuits with great sensitivity. Not only, sub-threshold designs are way more susceptible to process and environmental variations (like changes in temperature) than super-threshold circuits; therefore, they require extra effort to ensure that will operate as expected under all operating conditions [12].

2.4 Interconnects Parasitics

As described in [2], interconnections of circuits play an extremely important role in this case of study, since it is the higher wiring complexity of the Kogge-Stone adder that may make possible for a simple RCA to achieve its performance.

However, how do the aforementioned connection structures affect the performance of circuits?

To start with, a piece of interconnects is a thin-film wire that electrically connects two or more components in an integrated circuit, where the aforementioned structures are stacked on several layers, as shown in Figure 13.

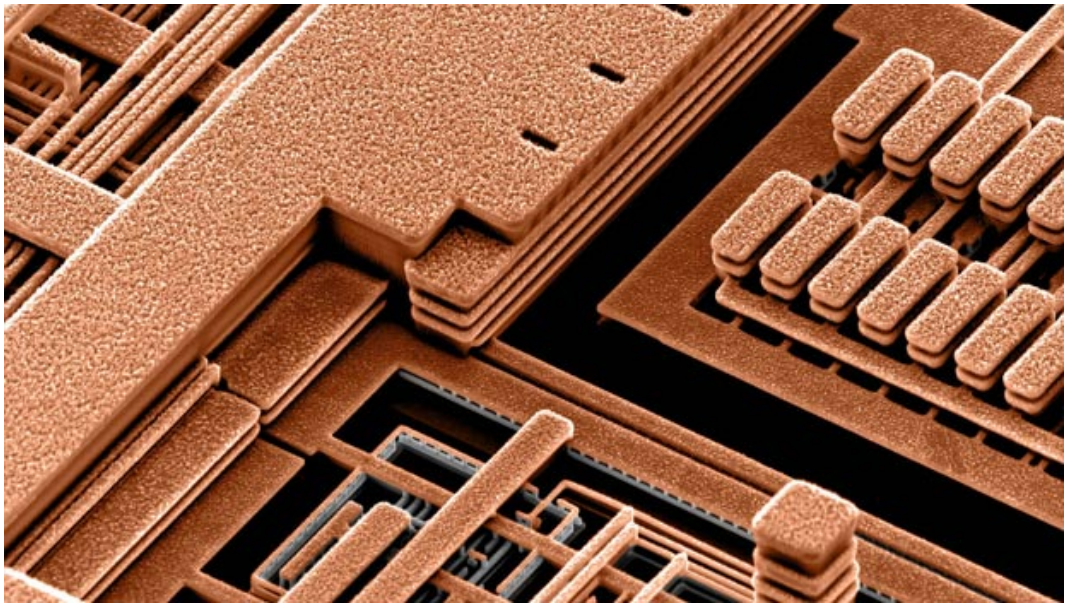


Figure 13: Example of IBM copper interconnects [17]

Due to physical effects and to the aforementioned geometry (nearness, overlap), unwanted parasitic components of resistance, capacitance and inductance are introduced, reducing both performance of ICs and their reliability, by increasing propagation delays, power consumption and presenting extra noise sources.

2.4.1 Resistance

Resistance of a piece of line is given by the following relation:

$$R = \rho \cdot \frac{L}{W \cdot H}, \tag{17}$$

where ρ is the metal resistivity and L, W and H are, respectivly, length, width and height of interconnects. In addition to this, contacts between layers must also be taken into account, since the transition between different stationary interfaces adds even extra resistance [15].

However, wire dimensions are not the only factors that influence the amount of parasitic resistance introduced: in fact, it also depends on the working frequency and temperature.

Talking about the former, for high frequencies it is possible to observe that

the current tends to flow on the surface of the conductor, which means that it works as the actual section of the wire was smaller; therefore, taking into account 17, the product of W and H decreases, and R goes up. This particular behavior is known with the name of "skin effect", and is quantified by the skin depth δ (i.e. the depth from the perimeter of the section at which the current is reduced by a factor of $\frac{1}{e}$ [16]):

$$\delta = \sqrt{\frac{\rho}{\pi \cdot \mu \cdot f}}, \tag{18}$$

where ρ is the aforementioned metal resistivity, μ its magnetic permeability and f the working frequency.

Turning to the latter, when the temperature increases, atoms of the metals get more kinetic energy and start vibrating; this movement hinders the regular electron flow through the crystal lattice; thus, less current is able to transit through the metal wire since the metal resistivity increases according to the following formula:

$$\rho = \rho_0(1 + \alpha(T_m - T_0)), \tag{19}$$

where ρ_0 is the nominal resistivity, α depends on the metal type, T_0 is the reference temperature and, finally, T_m is the metal temperature.

2.4.2 Capacitance

As well as resistance, wire capacitance is function of different factors, such as the distance of the wire from the substrate, the distance from surrounding wires and the shape of the piece of metal itself. So, a first contribution to the total parasitic capacitance is given by the interaction between the wire and the substrate, and it is the main contributor when $W \gg t_{ox}$:

$$C_{sub} = \epsilon_{ox} \cdot \frac{W \cdot L}{t_{ox}}. \tag{20}$$

However, over the past years, in order to lower the resistance (according to 17), advanced processes have a reduced W/H ratio, reaching values lower than 1: these are the so called "fat wires", which introduce a second kind of parasitic

parameter, known with the name of "fringing capacitance". This is due to the capacitive matching between the substrate and the sides of the wires, since their area is not negligible anymore (with respect to the product of W and L).

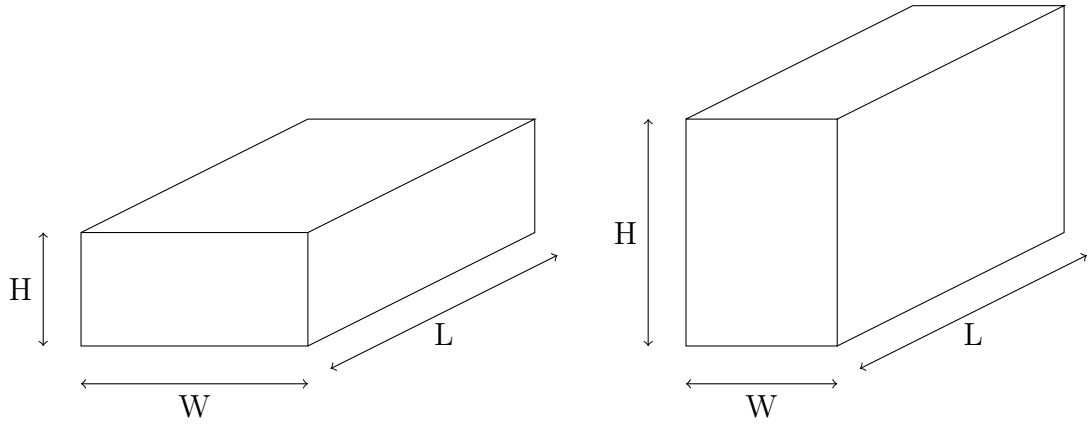


Figure 14: Conventional (left) vs Fat (right) wires

So, capacitances are currently modeled in the following way:

$$C = C_{sub} + C_{fringing} \tag{21}$$

$$= \frac{(W - \frac{H}{2})\epsilon_{ox}}{t_{ox}} + \frac{2\pi\epsilon_{ox}}{\log\frac{t_{ox}}{H}}. \tag{22}$$

Finally, it is essential to notice that these formulas only refer to the capacitive coupling between a specific piece of interconnect and the substrate, but in multilevel interconnects technologies the wires are not completely isolated. So, each wire is also coupled with neighboring wires on the same and on adjacent layers, and, therefore, the more complex the chip design is, the more parasitic capacitance will be introduced.

2.4.3 Inductance

Since, nowadays, working frequencies have largely reached the order of magnitude of GHz, parasitic inductance has started to be seriously considered, since it is responsible for overshoot effects, reflections of signals due to impedance mismatch, inductive coupling between lines, and switching noise due to $L\frac{di}{dt}$ voltage drops [5].

Starting from the geometry of interconnects, it is possible to directly compute

its inductance, thanks to the fact that the capacitance c and the inductance l (per unit length) of a wire are related by the following expression:

$$cl = \epsilon\mu, \tag{23}$$

with ϵ and μ, respectively, the permittivity and permeability of the surrounding dielectric. This is, of course, a simple approximation: in fact, proper modeling requires knowledge of the inductance return path for current. However, in a real circuit, where the interconnection system is very complex, the return path is not easy to resolve and is not even unique.

2.5 Recap

Having quickly spoken about the theory on which this thesis work is based, now it is time to dig into the next chapter, involving the full adder study. Particularly, after a literature exploration and several modifications applied to already existing circuits, a new architecture (based on a "Programmable And/Or (PAO)" standard cell) will be proposed, and with the help of physical simulations with Cadence Spectre, it will be shown the reason why this is the FA chosen to go on with the study.

3 Full Adders

One of the suggestions for future work in [2] concerned about the possibility of retrieving better results with the improvement of the single full adder structure, due to the fact that a quicker and/or less power demanding circuit may reduce the energy consumption, and so the final RCA might have the same performance of a KS with a more narrow gap in terms of voltage supply.

3.1 Initial Full Adder Structures

As seen in 5 and 12, the most conventional full adder can be easily built with three different kinds of logic gates: XOR, OR, AND (Figure 15).

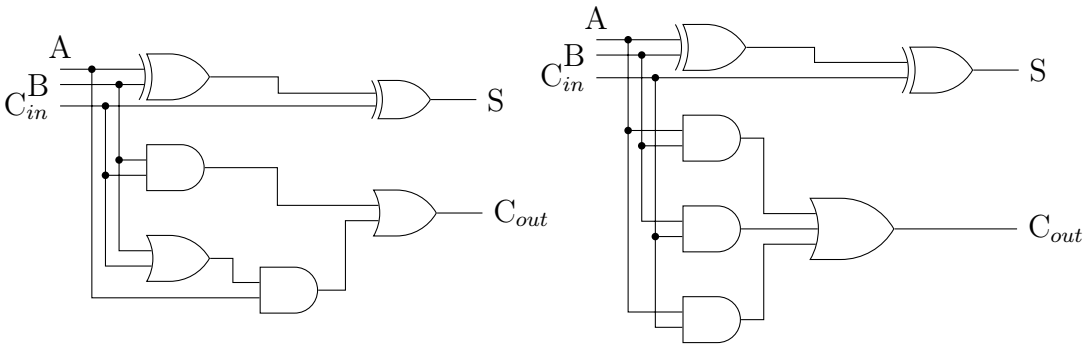


Figure 15: Classic FA Implementations

However, there are many ways of implementation of full adder cells which are more efficient then the conventional structure (either considered as a stand-alone circuit or aiming to improve the final N-bit serial adder); in fact, using simpler, faster or less gates, and/or exploiting the "parallelism" property, the whole architecture improves its performance, still having the same functionality.

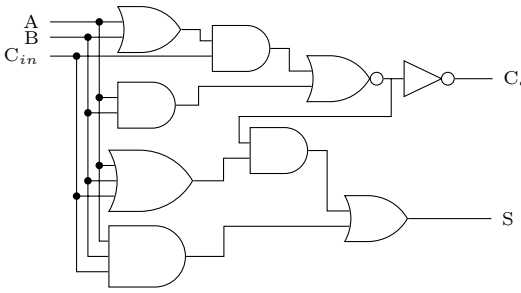
Talking about the aforementioned "parallelism" property, due to the fact that in a RCA structure the operands are presented in parallel, the only signals to be waited for are the intermediate carries. Therefore, the whole FA structure should be designed in order to have as less gates using the carry signal as possible. In this way, when the carry signal is available, the path to the output

pin is shorter.

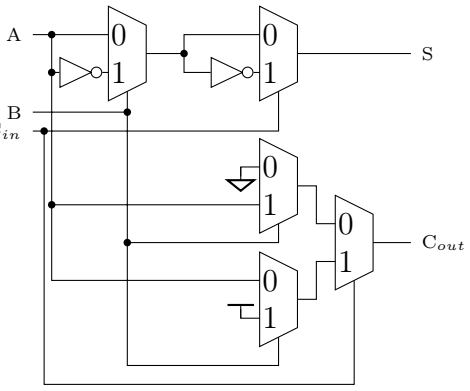
On the other hand, the path from the operands to the first gate using the carry signal does not have to be too long, or there may be the possibility for the carry to wait before having the correct result; however, this is unlikely to happen, and the more bits the RCA structure is based on, the less this effect is a problem for the adder.

3.1.1 Design and Synthesis

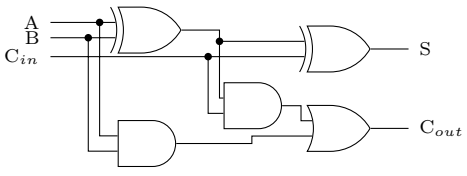
Having considered all of what it has been said, [18], [19] and [20] present several architectures that differ from the conventional full adder cell.



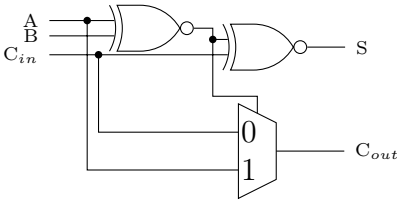
(a) "Logic Sharing" FA Implementation



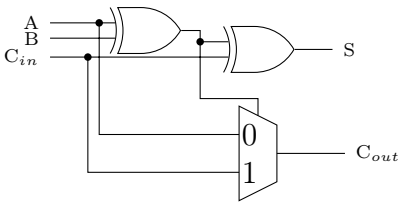
(b) "MUX" FA Implementation



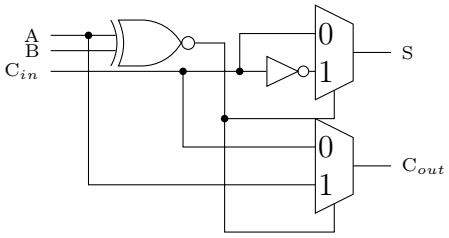
(c) "XAC" FA Implementation



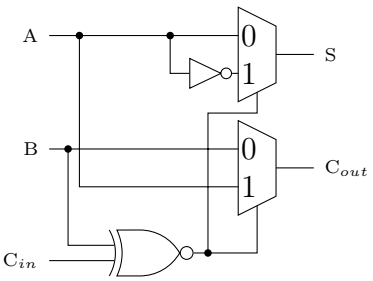
(d) "XNOR" FA Implementation



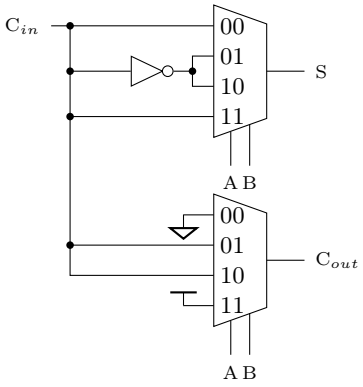
(e) "XOR" FA Implementation



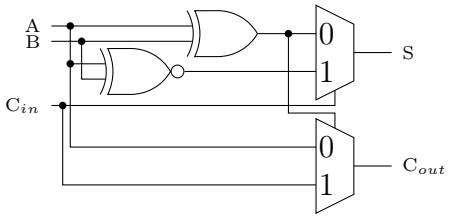
(f) "XNM" FA Implementation



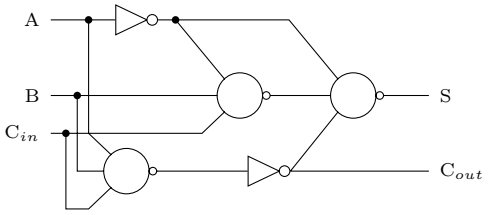
(g) "V1" FA Implementation



(h) "V2" FA Implementation



(i) "V3" FA Implementation



(j) "MAJ3" FA Implementation

Figure 16: Different Full Adder Implementations

It is important to point out the fact that the "logic sharing" structure has been presented in order to reduce the complexity of the full-adder cell in [21], without aiming to use less logic gates than the conventional FA; in addition, since an intermediate signal is shared between C_{out} and S, fault secureness (property for which circuits and networks fail to be totally self-checking generally, but can achieve the totally self-checking goal when the same fault assumptions are precised) is preserved on C_{out} or S if the error also propagates to, respectively, S or C_{out}.

After having structurally described the previously shown full adder circuits in HDL language and synthesized them, their performance had to be evaluated. However, before translating the Verilog netlist into a Spice one, another step had to be carried out: the verification. In fact, the correctness of the circuit behaviors had to be tested, so as to be sure that the synthesis software did not modify the described structures.

Particularly, all the possible combinations of inputs were tested, and as can be clearly seen from Figure 17, where the yellow, red and blue signals are, re-

spectively, inputs, sum and carry-out, all the 1-bit full adders were synthesized correctly.

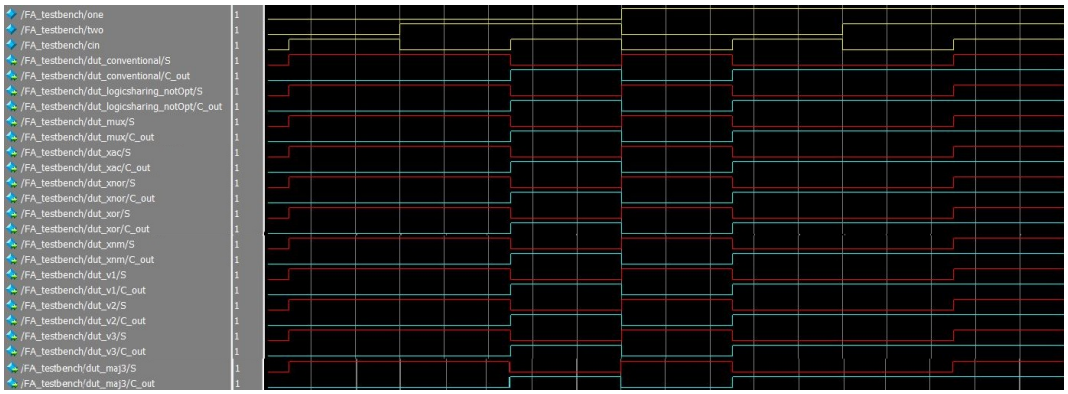
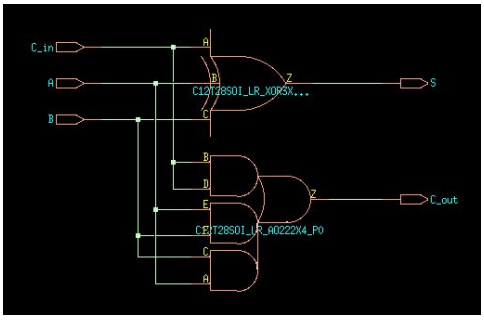
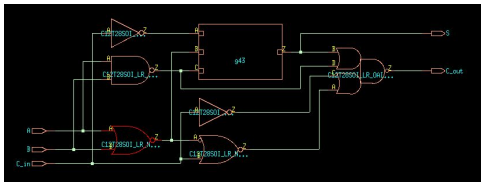


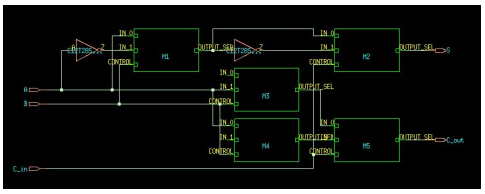
Figure 17: After-Synthesis Verification of the Original FA Structures



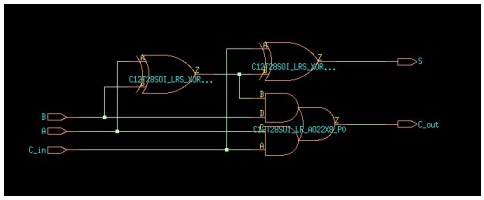
(a) Conventional FA Synthesized



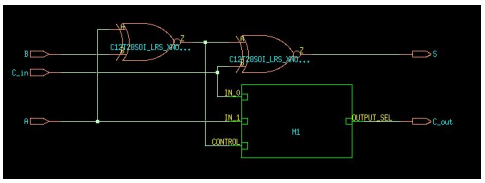
(b) "Logic Sharing" FA Synthesized



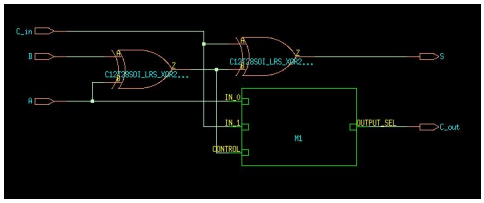
(c) "MUX" FA Synthesized



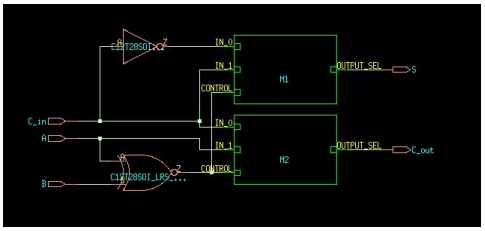
(d) "XAC" FA Synthesized



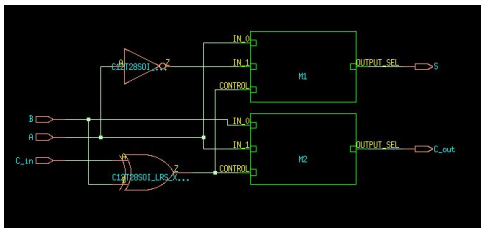
(e) "XNOR" FA Synthesized



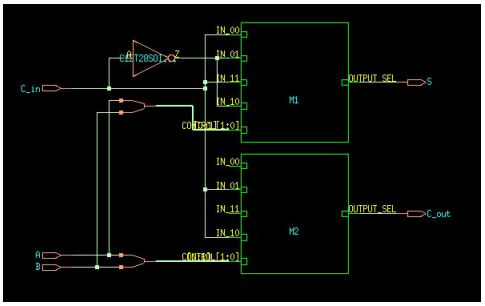
(f) "XOR" FA Synthesized



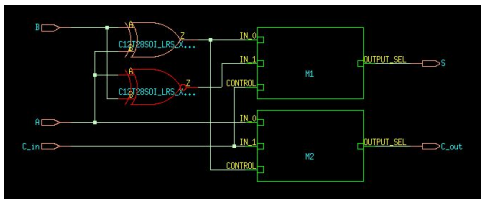
(g) "XNM" FA Synthesized



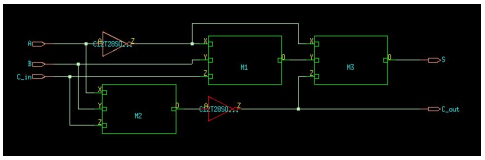
(h) "V1" FA Synthesized



(i) "V2" FA Synthesized



(j) "V3" FA Synthesized



(k) "MAJ3" FA Synthesized

Figure 18: Synthesized FAs Architectures

3.1.2 Spice Simulation

Before showing the results of simulations, it is important to specify that, in order to have the best possible pieces of data, it is not sufficient to physically simulate the truth table shown in 2; in fact, it only takes into account the results that the different combinations of inputs produce, without considering the possible transitions through which those results are obtained.

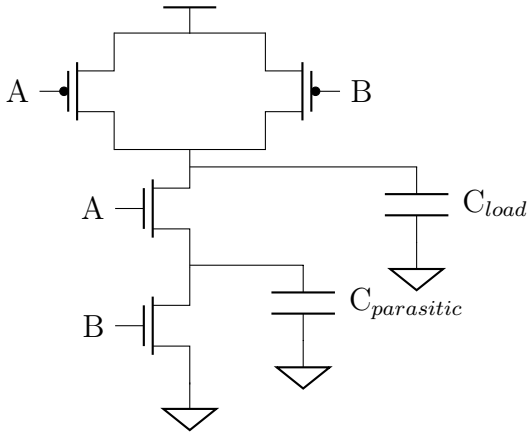


Figure 19: 2 Input NAND

More specifically, it is well known that performance of logic circuits does

depend on the value of parasitic capacitance present in it; however, this value might change according to the different transitions of inputs: an example of this is shown in Figure 19.

To start with, let us consider both "A" and "B" going from logic '1' to '0': in that case the nmos transistors would be switched off, preventing the parasitic capacitance to be charged together with the load. On the other hand, if "A" keeps its high value while "B" switches, the parasitic capacitance slows down the behavior of the entire logic gate.

Therefore, theoretically, all the possible combinations of transitions should be simulated; this is a pretty doable task for a 3-bit input circuit, since the number of transitions is:

$$number\_of\_transitions = 2^{2 \cdot NBIT} = 64. \tag{24}$$

So, after having exported all the Spice netlists, physical simulations were carried out in order to obtain the worst case propagation time, the power and the energy consumption for all the aforementioned full adders at 500MHz (a frequency, computed in advance, high enough that could have let all the devices finish their computation with margin). In particular, all the inputs and the outputs (for all the Spice simulations up to 64 bits) were buffered through conventional CMOS inverters, as shown in Figure 20; this because of the fact that it was wanted to take into account realistic drives and loads and, therefore, make signal slopes as close to reality as possible.

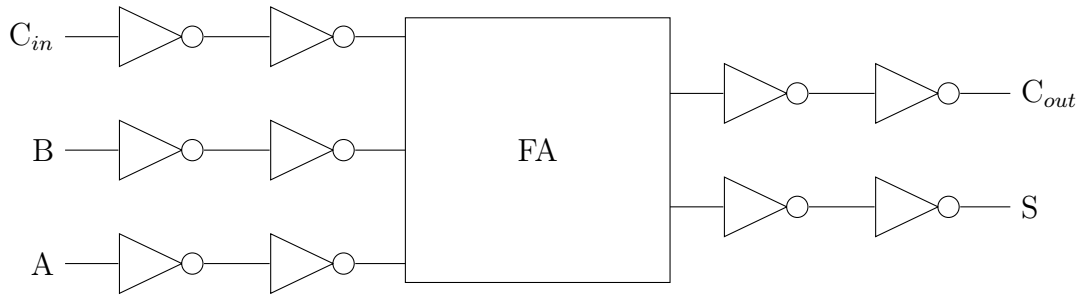


Figure 20: Full adder basic block

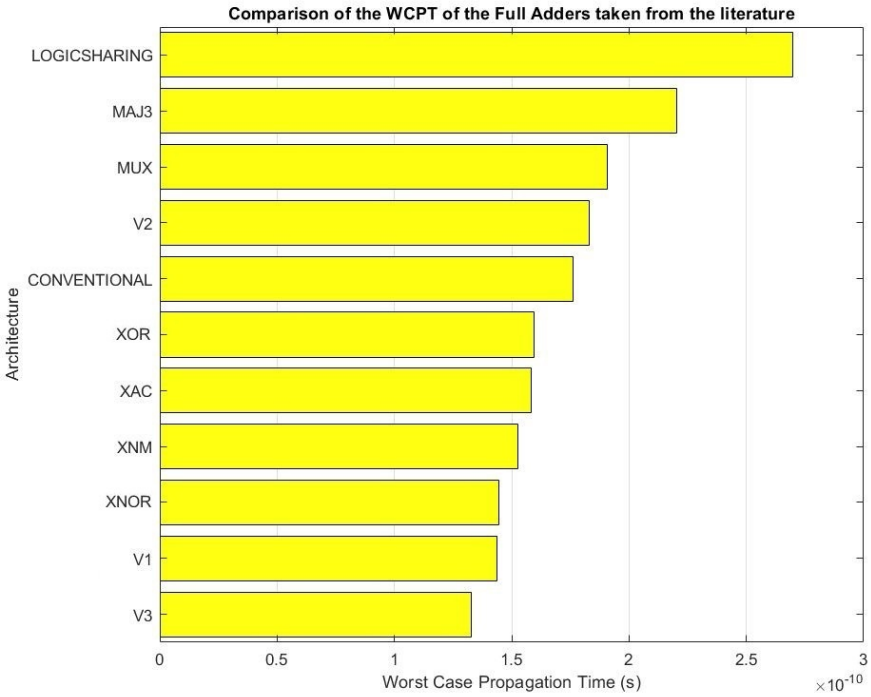


Figure 21: Worst Case Execution Time of the presented FAs

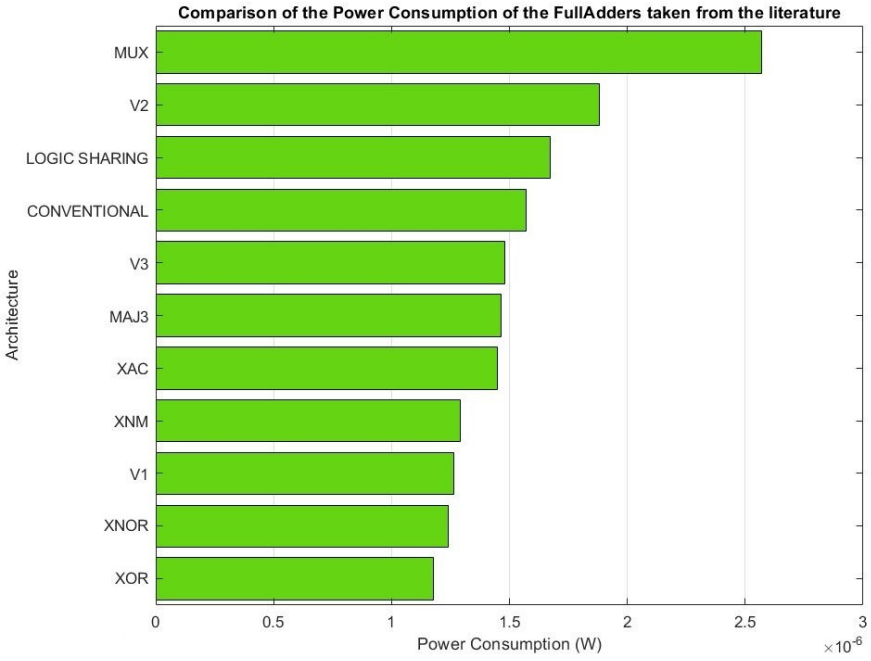


Figure 22: Power Consumption of the presented FAs

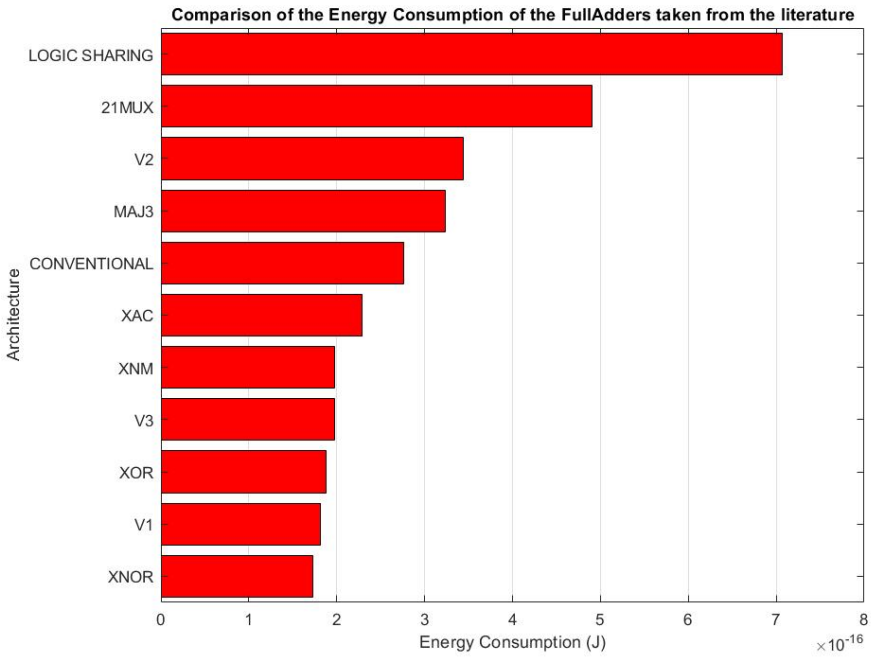


Figure 23: Energy Consumption of the presented FAs

$$P_{AVG} = \frac{1}{T}\int_0^T v_{DD}(t) \cdot i_d(t)dt \tag{25}$$

$$E = \int_0^T v_{DD}(t) \cdot i_d(t)dt \tag{26}$$

As it is possible to notice, due to the fact that the "XNOR" architecture is the second best in terms of power consumption and the third for worst case propagation time (even though the difference between it and the second circuit, the so called "V1", accounts for 1 ps only, and so not that influent for the final energy computation), it ends in being the best solution when the total energy is considered.

However, as it will be explained through the next subchapter, some modifications to the already existing full adders were added, in order to try to optimize the circuits and improve their behavior.

3.2 Optimization of the Existing Full Adders

To start with, it must be pointed out again that the structures that will be presented were not designed from scratch (as it was for the "XMAJ3", which will be covered later), but are just retrieved from modifications (which will be presented in this sub-chapter) applied to the full adders studied from the literature, intending to improve their performance.

Having said this, the first circuit to be taken into account is the "Logic Sharing", due to its initial complexity; in fact, in addition to the high number of gates employed, another problem consists on the fact that two input logic elements (and, particularly, an "AND" and an "OR") have three inputs, which is translated in more area, more input capacitance, and so more power consumption and lower timing performance. In addition to this, the carry-in signal is needed to drive the very early stages of the full adder, and so, as explained in 3.1, this will slow down the future RCAs.

For these reasons, the whole architecture has been rearranged, as shown in Figure 24.

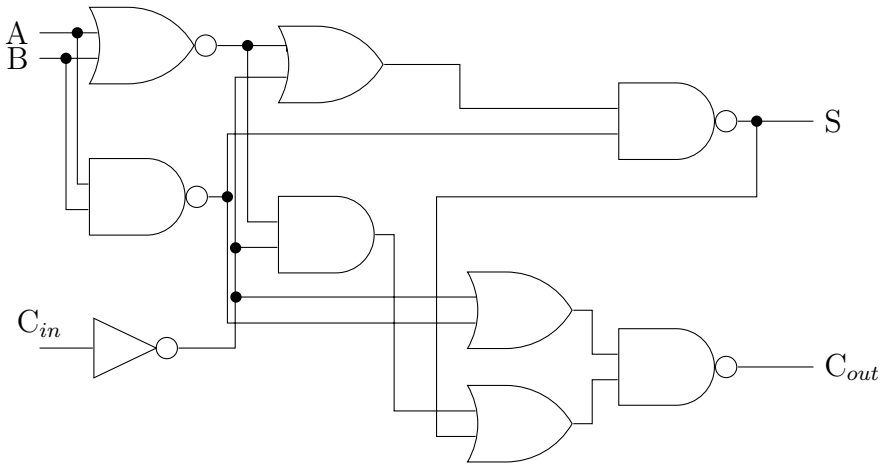


Figure 24: "Logic Sharing" FA After Optimization

Compared to the original architecture, the "Optimized Logic Sharing" FA presents the same number of gates, but each of them has a maximum of two inputs, limiting the effect of additional parasitics on the overall performance.

Furthermore, the correct carry-in signal is not requested sooner than the second stage of gates, allowing the full adder to perform some computation as soon as the operands are present. Finally, even after the application of the aforementioned variations, the logic shared structure is still present, making the circuit have the same benefits as before (described in 3.1.1).

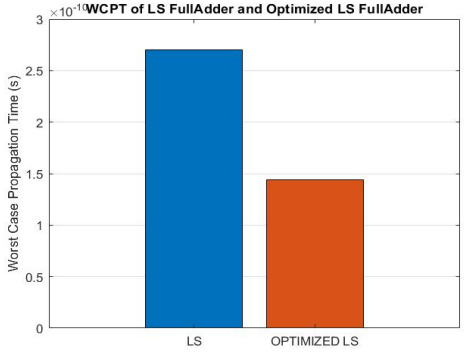


Figure 25: WC Execution Time of the Original and the Modified "Logic Sharing" FA

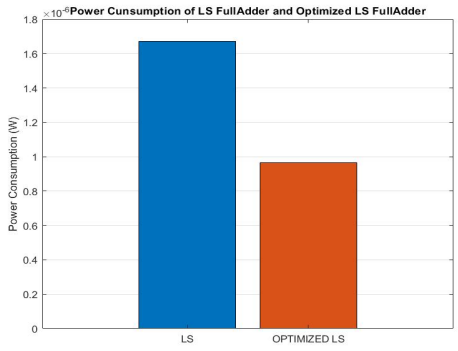


Figure 26: Power Consumption of the Original and the Modified "Logic Sharing" FA

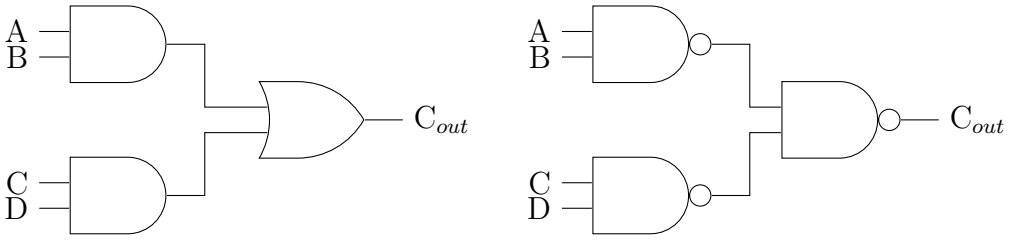
Turning to measured effectiveness, it is possible to see how those simple modifications led to great improvements, both in terms of timing and power consumption: in fact, talking about the former, the optimized full adder presents a decrease of 46.3%, while, taking into account the latter, the percentage drop is 40.6%.

However, this circuit is intended to improve fault secureness in full adder circuits, without being designed for great achievements in terms of performance. For this reason, also other circuits were taken into account in this improvement stage: the "XAC" implementation and the "V3" one.

Talking about the first, the carry-out branch is composed of an "OR" and two "AND" logic gates; however, applying a simple transformation, it is possible to retrieve the following equivalence:

$$\begin{aligned} C_{out} &= (A \cdot B) + (C \cdot D) = \overline{\overline{(A \cdot B) + (C \cdot D)}} \\ &= \overline{\overline{(A \cdot B)} \cdot \overline{(C \cdot D)}} = \overline{(\overline{A} + \overline{B}) + (\overline{A} + \overline{B})}, \end{aligned} \tag{27}$$

consisting in a tree of "NAND" gates, as shown below.



(a) carry-out Branch (b) "Tree of "NAND" Gates

Figure 27: Equivalence between Different Carry-Out Branches

In this way, due to the fact that in general a "NAND" gate is way smaller, faster and less power consuming than either an "AND" or an "OR" port, the worst case propagation time presents a fall of 18.75%, while the power consumption decreases of a quantity of 26.7%.

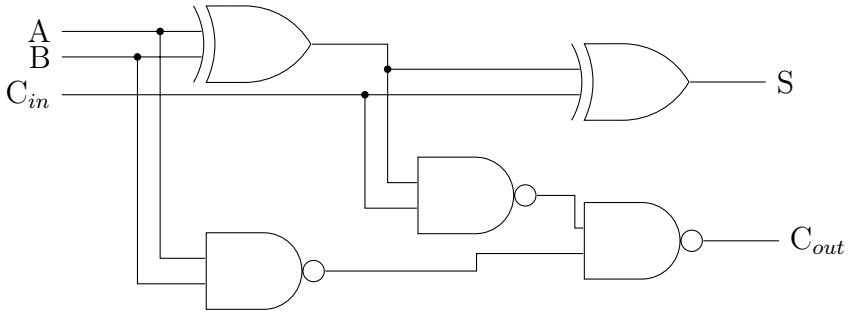


Figure 28: "XAC" FA Implementation After Optimization

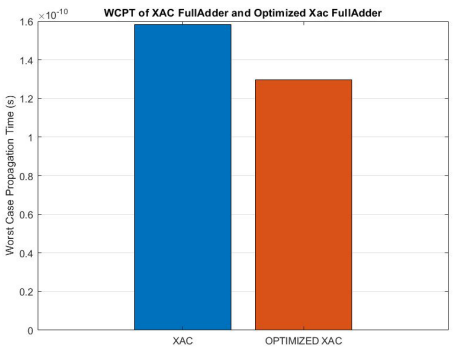


Figure 29: WC Execution Time of the Original and the Modified "XAC" FA

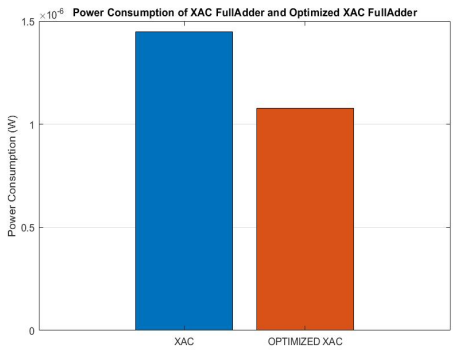
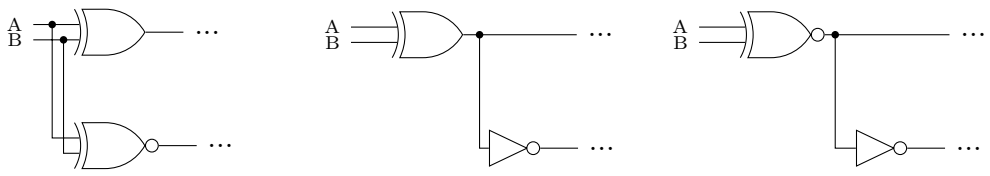


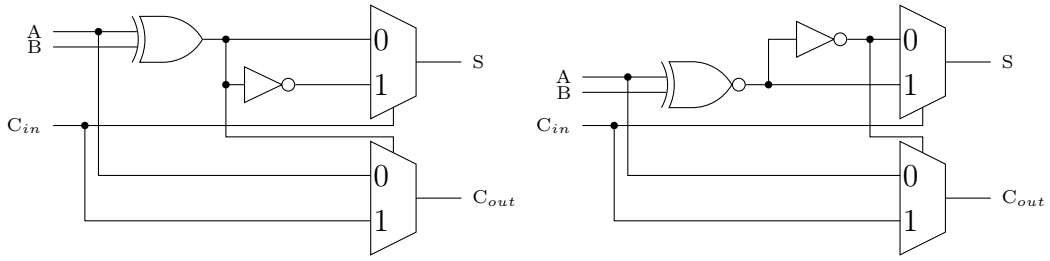
Figure 30: Power Consumption of the Original and the Modified "XAC" FA

Taking into account the so called "V3" implementation, as it can be noticed from Figure 16i, the inputs of the multiplexer in the sum branch of the circuit are the outputs of a "XOR" and a "XNOR" port in parallel; however, thinking about the raw logic, a "XOR" function is implemented in not another way than an inverted "XNOR", which means that the same functionality can be designed with a single "XOR/XNOR" gate plus an inverter on its output, lowering the total number of transistors. On the other hand, this slows down the entire sum computation, since before being able to produce a valid result, the intermediate output of the "XOR/XNOR" gate must be inverted, and only subsequently can feed the following multiplexer.



(a) "XOR/XNOR" (b) "XOR/INV" (c) "XNOR/INV"

Figure 31: Different Solutions for the Sum Branch of the "V3" Full Adder



(a) "V3 XORINV" FA Implementation (b) "V3 XNORINV" FA Implementation

Figure 32: "V3" FA Implementations After Optimization

So, in order to be able to cover all the possible described scenarios, both the aforementioned solutions were implemented and simulated, with a slight rearrangement of the connections, as it is possible to be noticed from Figure 32.

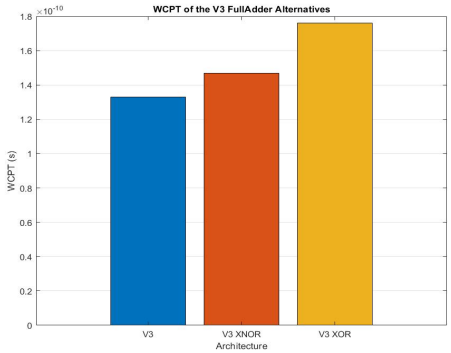


Figure 33: WC Execution Time of the Original and the Modified "V3" FAs

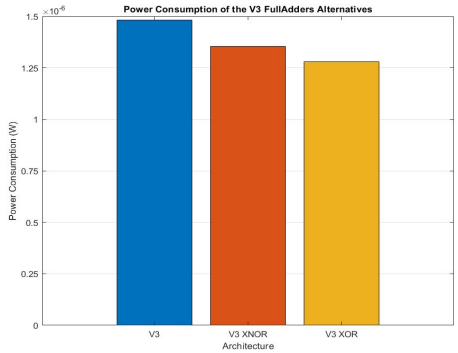


Figure 34: Power Consumption of the Original and the Modified "V3" FAs

As Figures 33 and 34 show, and as expected before the simulation process, the more power is saved (employing, in this case, an inverter instead of a "XOR/XNOR" logic gate), the slower the final full adder is; however, the choice of a "XOR" or a "XNOR" port leads to different results: in fact, due to the internal arrangement of the aforementioned standard cells transistors, the former is smaller, less power consuming but slower than the latter. In this way, the "V3 XOR" solution presents the best power saving, accounting for a 13.7% with respect to the original architecture, but at the same time is the slowest, with a worsening of almost 24.4%. On the other hand, the "V3 XNOR" architecture can be considered as the best trade-off, since it improves the power consumption of 8.8%, but at the same time the timing drop consists of 9.5% only.

However, this modification only is not able to lead to a final improvement in energy consumption, as it is shown in Figure 35. In fact, the power saving is not able to compensate the big difference in terms of worst case propagation time, ending in a heavier energy utilization with respect to the original full adder. For this reason, an additional improvement was employed, which is the basis of the newly proposed full adder circuit, the "XMAJ3" architecture.

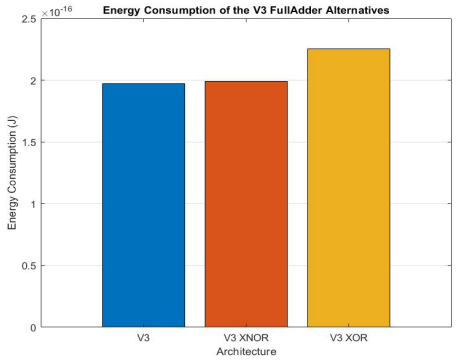
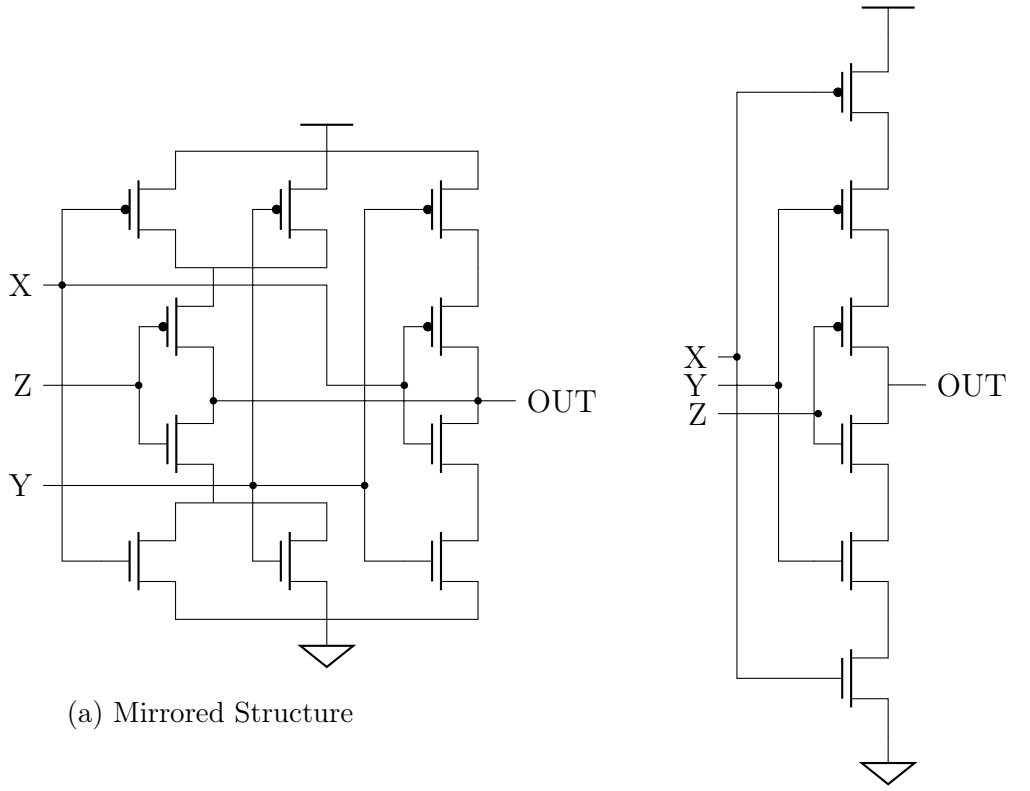


Figure 35: Energy Consumption of the Original and the Modified "V3" FAs

3.3 Majority Function and "Programmable AND/OR"

In the already described full adder circuits, the carry-out branch is composed of several standard cells, disposed as a tree. However, as presented in [2], the carry-out generation may be also possible without the aforementioned tree structure, but using what is called "Majority 3" function.

In fact, as it can be seen from Table 2, the carry-out signal rises any time the number of "1s" is higher than two, which, more generally, can be expressed in the following way: "the carry-out is '1' as soon as the amount of high inputs is the majority within the total number of them". However, in order to reproduce this kind of function, usually some transistor level customized cells must be designed (like the ones shown in Figure 36 plus a CMOS inverter connected to the output), otherwise, employing conventional standard cells, the "Majority 3" function would just be the same tree structure as before.



(a) Mirrored Structure

(b) "Tree of "NAND" Gates

Figure 36: Minority 3 Customized Gates [22]

But this is not the only possibility: in fact, the employed 28nm FDSOI standard cell library provides a special standard cell called "Programmable AND/OR", and even though its layout cannot be shown, it is possible to describe its functionality from an abstract point of view.

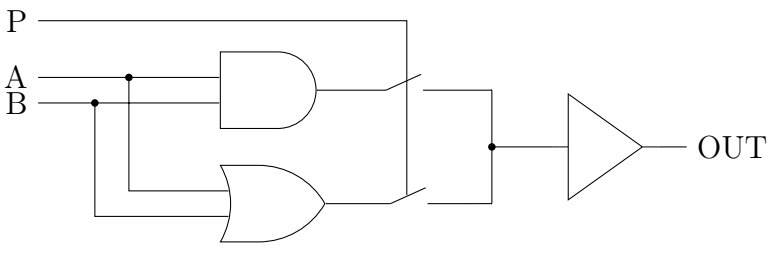


Figure 37: "Programmable AND/OR" Standard Cell Abstract View

Particularly, inputs "A" and "B" are both anded and ored, providing the results almost in parallel (almost since the different propagation times must be always taken into consideration); then, through the "P" signal, which in this

case acts as the "programmable branch", one of the two outputs is selected and sent to the following stage.

So, rearranging Table 2, it is possible to notice that, for C_{in} equal to '0', the output is given by the "AND" function of "A" and "B", while for for C_{in} equal to '1', the result is computed by the "OR".

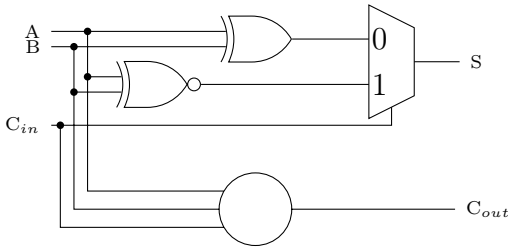
C_{in}	B	A	C_{out}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table 3: Rearranged Full Adder Truth Table

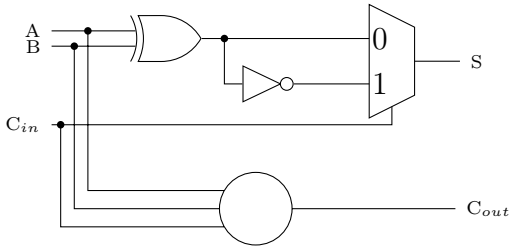
Therefore, this standard cell provides a great solution to implement this kind of function, without the need of designing "ad-hoc gates" and computing the partial results (the output of the abstract "AND" and "OR" gates) in a single stage, which means that there it is not required anymore to further optimize the architecture for as much parallelism as possible.

3.4 Towards "XMAJ3" Full Adder

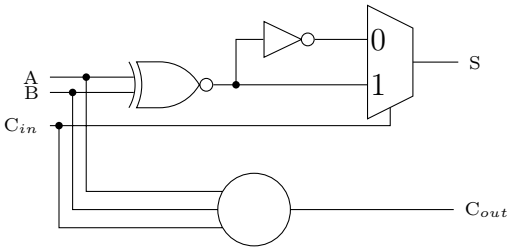
Having presented the "PAO" standard cell, it is possible to restart the description of the optimized full adders: in fact, with the usage of this programmable gate, the previously described "V3", "V3 XOR" and "V3 XNOR" full adder architectures were additionally modified, substituting the whole carry-out branch.



(a) "V3 MAJ3" FA Implementation



(b) "V3 MAJ3 XOR" FA Implementation



(c) "V3 MAJ3 XNOR" FA Implementation

In this way, not only a whole multiplexer is substituted by this programmable standard cell, but also the carry-out computation starts as soon as the carry-in propagates, without waiting for internal intermediate signals to be spread; this means that, in terms of performance, the usage of this kind of carry-out structure speeds up the carry computation by 32.5%, 34% and 36.4% for, respectively, the "V3", the "V3 XORINV" and the "V3 XNORINV" full adders (due to the fact that a multiplexer control signal must pass through the CMOS inverter before being utilized; in the other two cases, the same signal is ready after the "XOR" gate, but since the loads are different, the delays are as well).

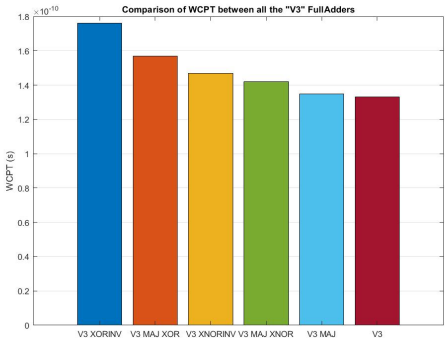


Figure 39: WCPC of the different "V3" FAs

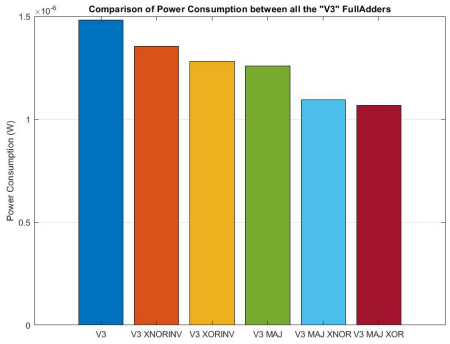


Figure 40: Power Consumption of the different "V3" FAs

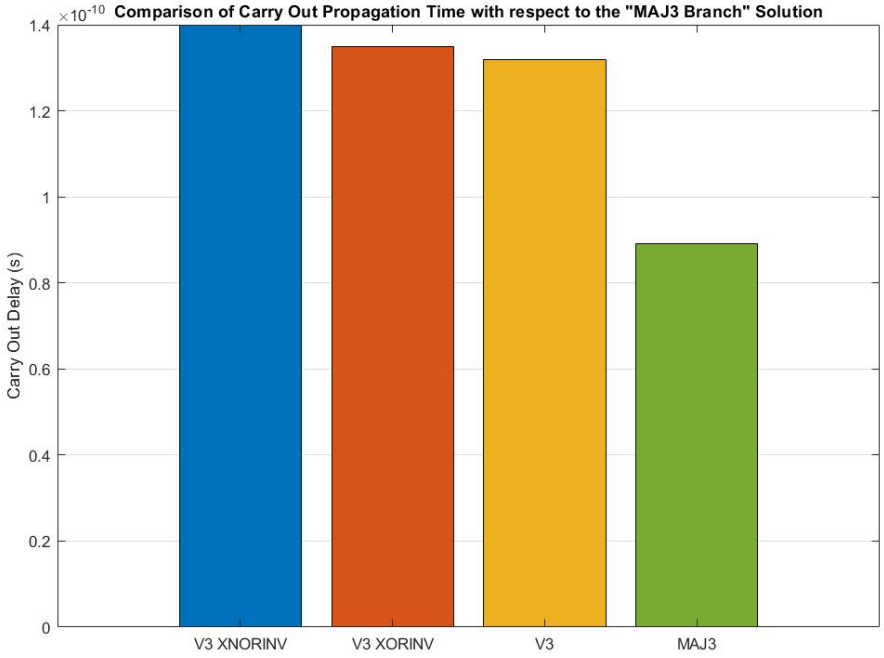


Figure 41: Difference in the carry-out Propagation Time

As it can be possible to be seen from Figures 39 and 40, the utilization of the aforementioned programmable gate leads to a great improvement in terms of power consumption, decreasing it up to 27.7%, but has a lower impact in timing performance: in fact, even though the carry branch has been speed up, and also some parasitic capacitance removed (thanks to the missing intermediate interconnection aiming to drive the carry-out multiplexer), the sum

part has remained almost unchanged, which means that effectiveness of this solution is partial. Particularly, the difference between an architecture and its corresponding "MAJ3" version accounts for 1.48% for the "V3" full adder, 3.4% for the "V3 XNOR" and 10.8% for the "V3 XOR" (these have improvements with the "MAJ3" solution, due to the depth of the control signal paths and the different loads applied to the logic gates).

Therefore, in order to try to exploit the properties of this "PAO" standard cell, also the sum path had to be rearranged, keeping in mind the results previously obtained, which point out that the easiest and most effective solution is to employ a chain of "XNOR" logic gates for the sum path, since they are little more power consuming than the "XOR" cells but so faster that the final energy computation is in favour of the former.

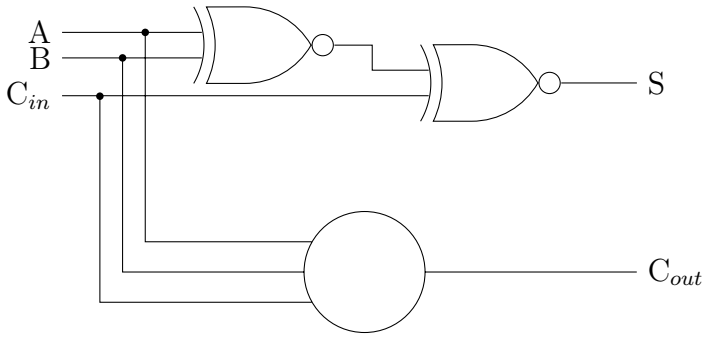


Figure 42: "XMAJ3" FA Implementation

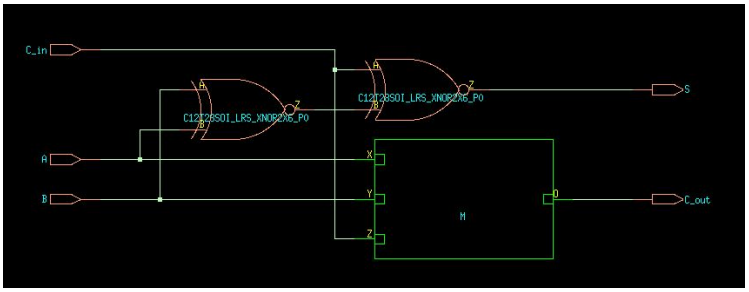


Figure 43: "XMAJ3" FA Synthesized

However, before every simulation and as previously shown, all the modified full adders structures had to be verified after their synthesis, in order to test their correctness.

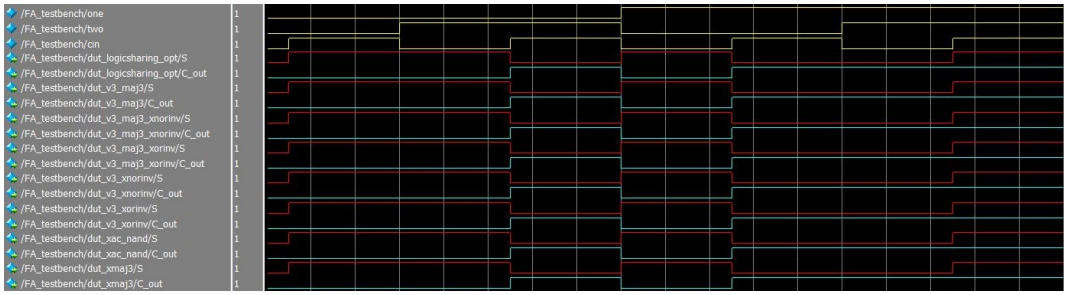


Figure 44: After-Synthesis Verification of the Modified FA Structures

The one shown in Figure 51 is the aforementioned "XMAJ3" full adder, and, as it possible to notice from the following graphs:

- it improves the timing performance of the previously best solution (the "XAC NAND") by 14.6%;
- it is the second best circuit in terms of power consumption, differing from the "LOGIC SHARING" after optimization of a small 8.2%;
- it is the best full adder concerning energy consumption, where it has the biggest improvement: in fact, it performs 15.82% better than the "LOGIC SHARING" device (after optimization).

However, as it will shown in the next chapter, where the RCAs simulation results will be presented and commented, this full adder structure is not considered as being the best candidate to prove the thesis objective and to improve the old results, but that is its "XOR" version.
In fact, for both these 1-bit FA structures, the critical paths include the sum pins, and due to the fact that, as previously mentioned, the "XNOR" gates are faster than the "XOR" ones, the "XNOR XMAJ3" performs better than the "XOR XMAJ3" in terms of timing and energy consumption.

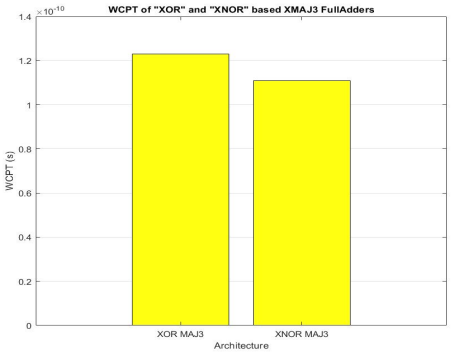


Figure 45: WCPT of "XOR" and "XNOR" based XMAJ3

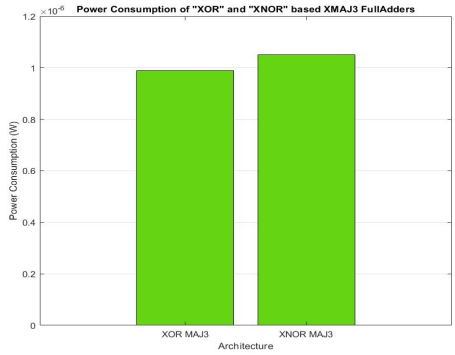


Figure 46: Power Consumption of "XOR" and "XNOR" based XMAJ3

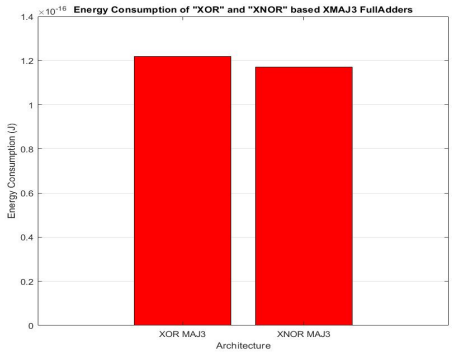


Figure 47: Energy Consumption of "XOR" and "XNOR" based XMAJ3

On the other hand, when RCAs are structured, the critical paths change, including the carry-out signal. In addition to this, these kinds of architectures allow a precomputation of partial results up to the last gate before the final sum signal, and at the same time their carry branch is composed of a single logic port only; this means that, when the carry-in has not propagated yet, both the sum and the carry-out need one last logic gates to be computed in order to present the correct results.

These are, respectively, a "XOR/XNOR" (depending on the achitecture) and a "PAO", but due to the fact that the latter is 61.26% and 65.74% slower than the former ones (values computed without loads), the worst case propagation time (either employing "XOR" or "XNOR" cells) will be given by the path

composed of the various carry-out intermediate and final signals, driven by the same logic gates through the carry branches.

This means that even if "XOR" cells are slower, the power saved with their utilization decreases the amount of energy required by the whole RCA circuit. For this reason, despite the better performance of the "XNOR" based XMAJ3, the "XOR" architecture (which, as it is possible to see from Figure 50 is still the best solution within the original and modified full adders) will be the basic block of the XMAJ3 based RCA structure.

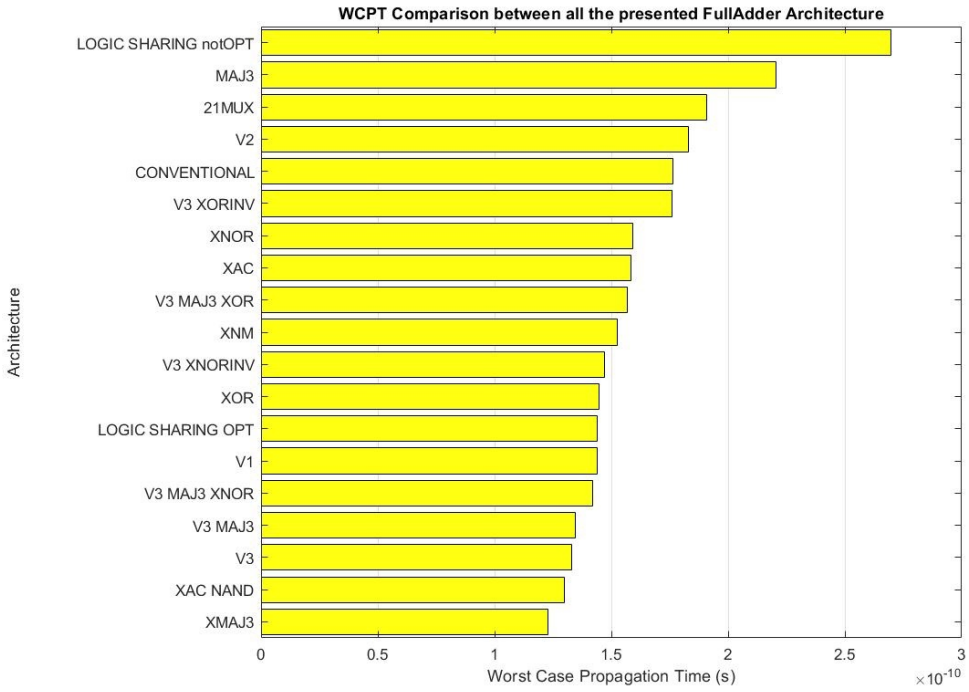


Figure 48: WCPT of the Proposed Architectures with "XOR" based XMAJ3

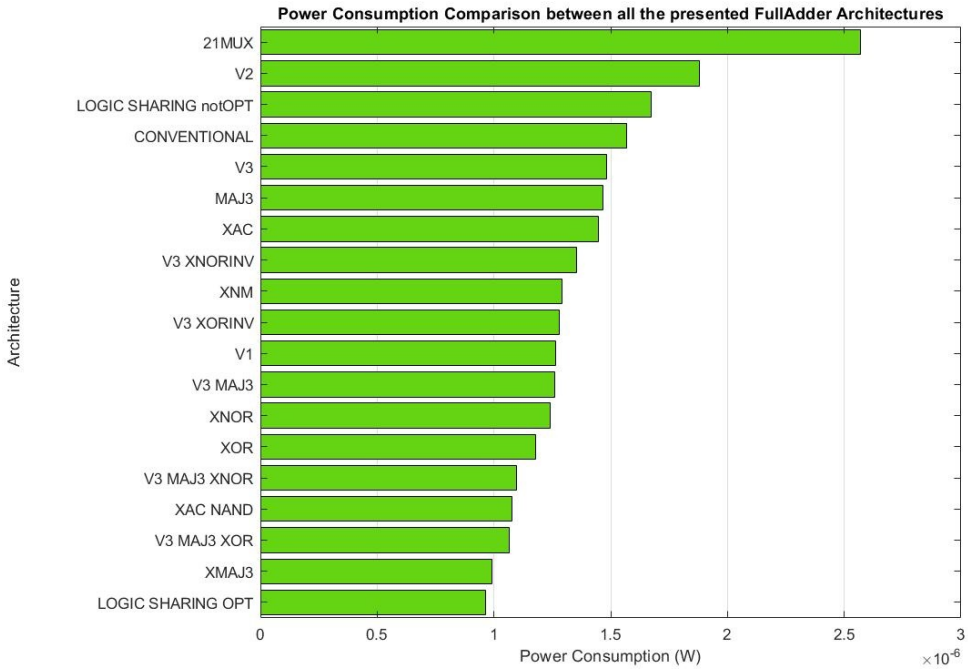


Figure 49: Power Consumption of the Proposed Architectures with "XOR" based XMAJ3

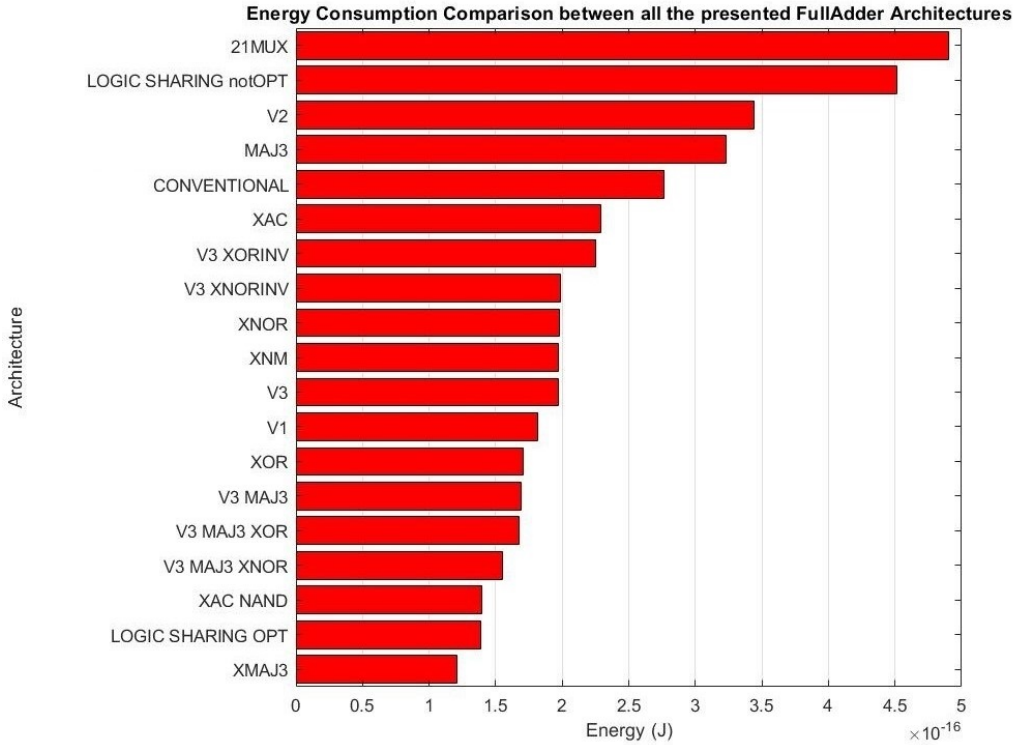


Figure 50: Energy Consumption of the Proposed Architectures with "XOR" based XMAJ3

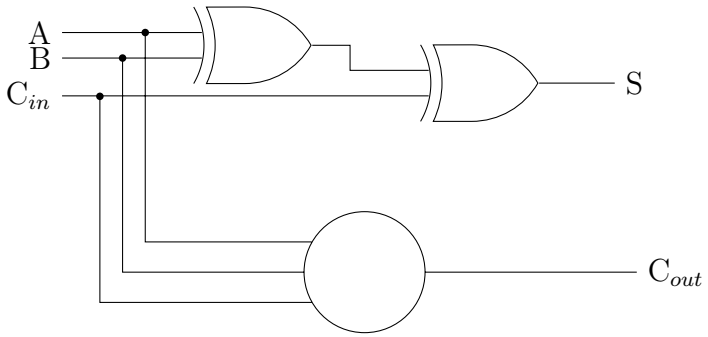


Figure 51: "XOR" based "XMAJ3" FA Implementation

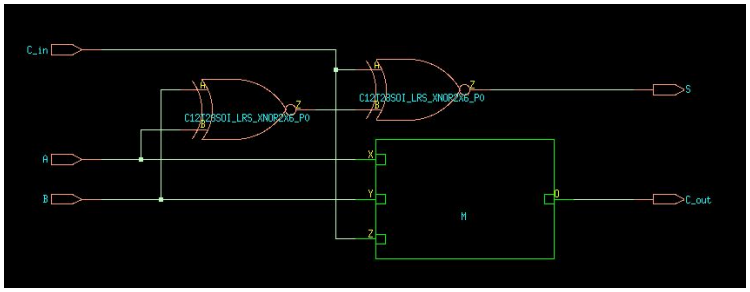


Figure 52: "XOR" based "XMAJ3" FA Synthesized

However, even though this and only this will be taken into account to compare its performance with the Kogge-Stone one, it was chosen to simulate the behavior of several RCAs, so as to understand the goodness of the proposed architectures up to 64 bits.

The results will be presented in the next chapter, after having described how the input stimuli were generated, due to the fact that, going towards 64 bits, the number of all the possible combinations of inputs rises exponentially, and it is impossible to simulate all of them.

4 Input Transition Generation

After having simulated and measured performance of all the full adder circuits presented so far, RCAs had to be characterized. This procedure is a task which grows in complexity as the number of bits increases. In fact, as described in subchapter 3.1.2, in order to have precise measurement (precise according to the stage of circuit development) of device performance, all the possible input transitions should be considered; however, as shown in (24), their number rises exponentially, extending the simulation period until a point where the time requested to finish a process would be on the order of magnitude of several years.

Number of Bits	Transitions
3	64
5	1024
9	262144
17	1.717986918e10
33	7.378697629e19
65	1.361129468e39

Table 4: Number of Possible Transitions with respect to RCA Number of Bits

For this reason a trade-off has to be found: more input stimuli means higher precision, but also longer simulation time; on the other hand, less stimuli would be translated into lower precision, but quicker retrieving of results.
In addition to this, another problem regards the way the inputs are generated: how to chose the best subset of all the possible combinations?
So, in order to do this, three different methods were explored, and these will be presented in the next part of this work.

4.1 "Constrained" Input Generation

The first method which will be presented has been called "constrained", due to the fact that its input generation process is based on a specific property of the transition distribution, correlated to the different kinds of bit changes.
First of all, on account of the number of bits of a device and on how many bits change in a single transition, it is possible to describe a Gaussian distribution that the entire set of them follows. It is important to be noticed that both in the following Figure 53 and in Table 4 unconventional values (3, 5, 9, 17, 33, 65) for the numbers of bits of devices were shown; this is due to the fact that, instead of generalizing, it was wanted to specifically target the description for FAs and RCAs, where, in addition to the normal operators (whose lengths are power of 2), the carry-in signal must be taken into account.

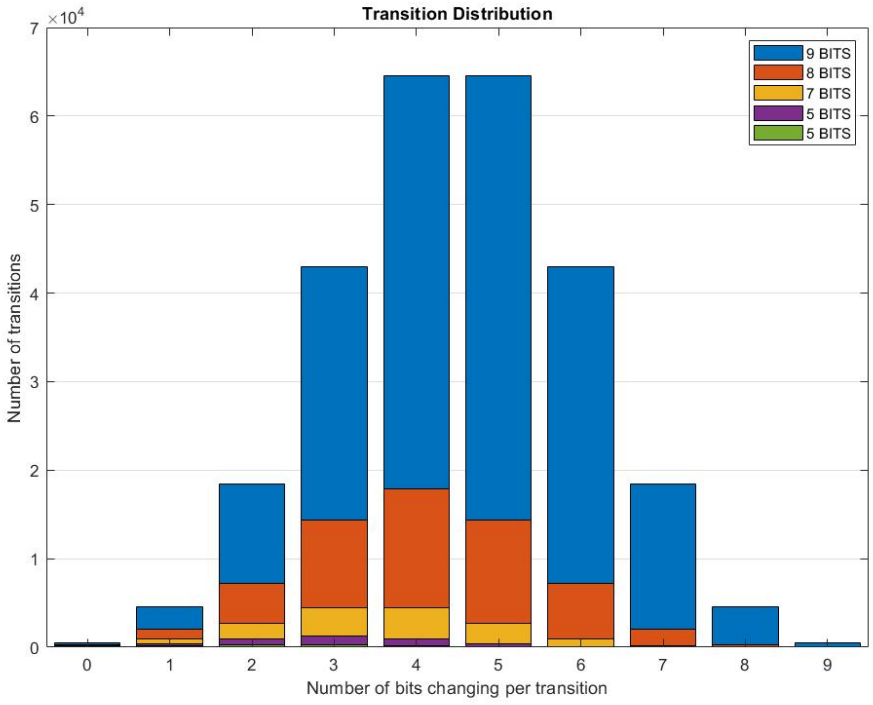


Figure 53: Distribution of Input Transitions

So, to give an example of what this means, five transitions are considered on four bits:

- 0011 → 0011 : in this case, no bit changes, and so it will be said that

the number of bits changed is 0;

- 0011 → 0010 : here only the least significant bit is different, so the number of bits changed is 1;
- 0011 → 1010 : 2 bits changed;
- 0011 → 1110 : 3 bits changed;
- 0011 → 1100 : 4 bits changed.

Therefore, connoting the case of 0 transitions with "0t", 1 transitions with "1t" and so on, it is possible to fill the following table, representing numerically the distributions graphically shown in Figure 53 (ending where it starts).

	0t	1t	2t	3t	4t	5t	...
2 Bits	4	8	4				
3 Bits	8	24	24	8			
4 Bits	16	64	96	64	16		
5 Bits	32	160	320	320	160	32	
⋮	⋮	⋮	⋮	⋮	⋮		

Table 5: Number of Transitions with respect to Number of Bits Changing

In addition to all of this, it is also possible to define two different kinds of transitions:

- **"UP"** transition: a '0' becomes a '1';
- **"DOWN"** transition: a '1' becomes a '0';

and looking at the aforementioned groups, the number of "UPs" and "DOWNs" in a subset of transitions where the number of bits changing is the same (for instance, the previously described "0t", "1t", ...) is constant, and accounts for 50% of the total number of transitions in that group. This means that, having for example twenty-four possible transitions where a single bit changes, twelve of them include an "UP" transition, while the other twelve perform a

"DOWN" one.

Having said all of this, the basic idea of this "constrained" input generation method is to create a set of stimuli where the number of transitions with n bits changing is proportional to the Gaussian distribution, and where the number of "UP" transitions is equal to the number of "DOWN" transitions for each number of bit changes. In this way, even though not all the possibilities are simulated, the proportion between transitions will be kept unaltered, and as it will be seen in the comparisons between the proposed methods, this ends up being the most accurate when power consumption has to be estimated; on the other hand, the same cannot be said regarding worst case propagation time.

4.2 "Random" and "Random Constrained" Input Generation

The second and third input stimuli generation methods are, respectively, the so called "random" and "randomly constrained".

Talking about the former, its implementation is pretty straightforward: transitions are randomly chosen between all the possibilities, without taking into account the fact that (with a smaller probability as the number of bits increases) some transitions might be equal to each other.

On the other hand, the latter is still based on a random generation of inputs but, differently from the previous, it was decided to apply a simple constraint: all the couples of stimuli must be different (either from each other and from the other couples in the same subset). In this way, all the possible transitions where there is no change in bit values (the "0t" transitions) are neglected, leading to a double result:

- due to the fact that there will always be at least a single transition for each stimulus, the previously described distribution is not followed, which makes the power consumption measurement overestimated. However, this may be considered as an useful property from a specific point of view: in fact, if a targeted study was conducted and the average percentage

of overestimation was computed, it would be possible to subtract this quantity to the measured one, retrieving statistically a piece of data closer to the actual one (of course with all the correlated uncertainties);

- since the worst case propagation time corresponds to a transition where bits change their value, not having input stimuli where no value varies increases the probability to generate the transition which produces the worst case performance in the same subset of inputs.

4.2.1 Comparison of the Input Generation Methods

After having described and implemented the three aforementioned possible procedures to generate input stimuli for simulations, a comparison had to be performed, in order to be able to select the one which presented the best trade-off between effectiveness and computational time and complexity.
So, first of all, a generic 2-output standard cell was chosen and its worst case propagation time and power consumption measured, so as to be able to compare the actual values with the ones generated with the input transitions coming from different methods.
Then, 200 sets of stimuli were generated with each method, and each of these sets contained $\frac{1}{4}$ of the total number of possible transitions for that specific cell; for each set of transitions then, both the timing and power parameters were retrieved and, as it is represented by Figure 54 and 55, compared to the actual value. However, in the following figures, only the first 50 measurements are displayed, so as to let capture better the differences between real and measured data.

Figure 54: WCPT Measurements with Different Generation Methods

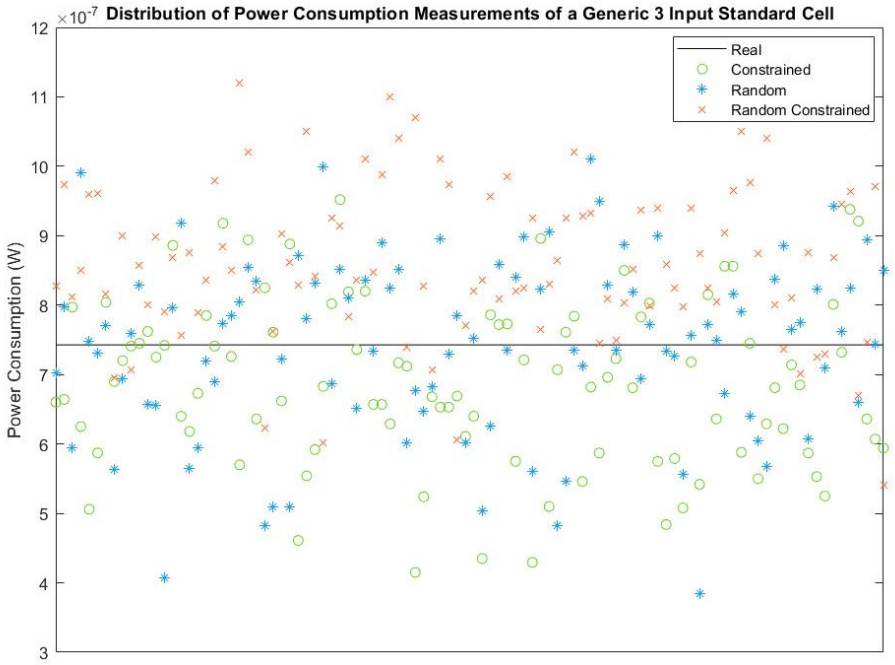


Figure 55: Power Consumption Measurements with Different Generation Methods

Talking about propagation time, it is possible to notice that, employing the so called "Constrained" method, the distribution of measurements is more distant from the actual value with respect to the "Randomly Constrained" procedure; this is directly correlated to the property described at the end of 4.2: due to the fact that input stimuli include also 0-transition combinations, the probability of generating the transition which corresponds to the worst case is lower. On the other hand, it improves the more simple "Random" method, since it does not take into account any restriction about repetitions and number of transitions; therefore, within the same set of stimuli, there might be equals inputs, all inputs with no bit change, all inputs with the same number of bit changes and so on, reducing the precision of the produced results.
Turning to power performance, the same property leads to a better quantification if compared to the other methods, due to the proportionality between stimuli and real transition Gaussian distribution.
Particularly, after having taken into account all the simulations run, the average error percentage was computed, both for timing and power consumption measurements, leading to the following analytic results:

Method	Timing Error	Power Error %
Constrained	4.550	13.708
Random	4.791	14.308
Randomly Constrained	3.905	18.079

Table 6: Error Percentage on Different Measurements

As it can be seen from Table 6, the errors on power consumption are pretty high for all the aforementioned methods, while worst case propagation times can be estimated with a smaller inaccuracy. However, due to the fact that the "Randomly Constrained" method provides the best approximation in terms of timing, with further studies a value close to the actual power consumption might be statistically retrieved and combinations of stimuli are easier to be generated (in terms of computational complexity), this was chosen as principal

input generation methodology.

In any case, there is still a question that rises and has to be answered: due to all these approximations and errors, is it still possible to trust the results obtained from analysis of circuits?

The answer is yes, but also no. This because of the fact that considering values as absolute indicators of performance, the same are, as said before, just approximations of the actual values, with the correlated uncertainties on account of the way the input stimuli are generated. On the other hand, if what is important is the difference of performance between devices, the comparisons can be done as long as the inputs are the same, and this is the case of this piece of work: in fact, in all the next graphs (where the performance of several RCAs will be presented) and also in the conclusive comparison with the Kogge-Stone structure, the main point is not the absolute value, but the gap between different circuits.

5 RCAs and Kogge-Stone Simulations

In this section, the RCAs performance will be compared to the Kogge-Stone one, carrying out Spice simulations for 8, 16, 32 and 64 bits.
Specifically, since many FA structures had been presented, it was chosen not to simulate possible ripple carry adders based on all the aforementioned circuits, but just the most performing ones, which are the so called "V3 MAJ3", "V3 MAJ3 XOR", "V3 MAJ3 XNOR", "XAC NAND", "LOGIC SHARING OPTIMIZED" and the proposed "XMAJ3" (refer to chapter 3.2 and 3.4 for schematics) . In addition to these, two more circuits were added in the comparisons: the "CONVENTIONAL" full adder (shown in 18a) and the standard cell contained in the 28nm FDSOI library, a structure optimized at transistor level for low power consumption. This because both of them are perfect terms of comparisons:

- the former to be able to notice the advantage of the different architectures and, particularly, of the "XMAJ3";
- the latter so as to pay attention to the fact that with the proposed "XMAJ3" it is possible to save energy even though its design was based on a RTL description.

However, before going on, the comparison between the "XOR" and "XNOR" based XMAJ3 architecture is shown, which helps to understand better what described in the end of section 3.4.

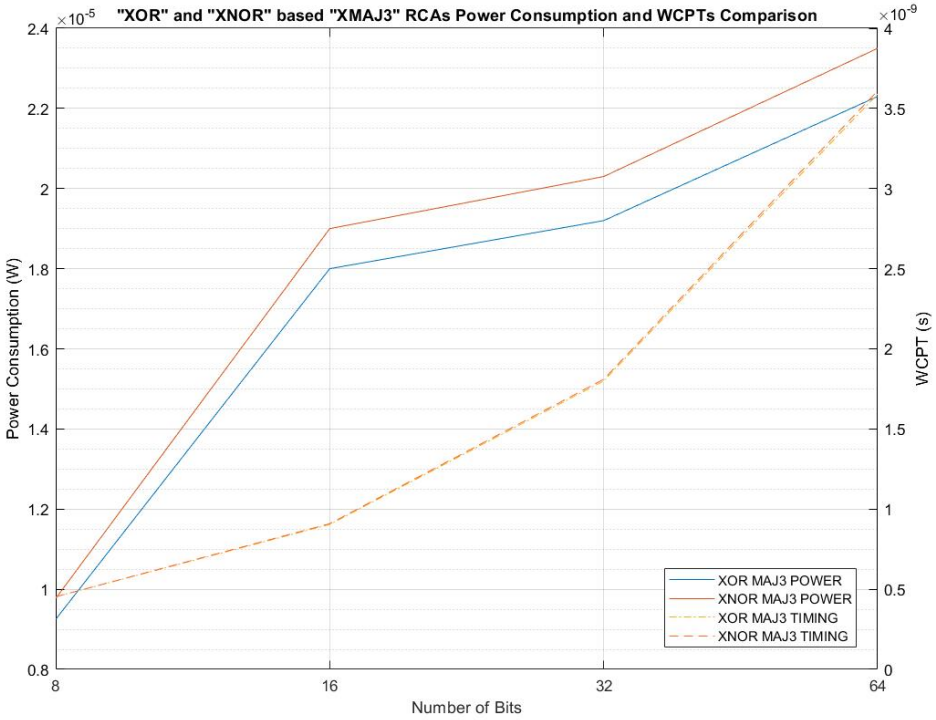


Figure 56: Power Consumption and WCPT Comparison between "XMAJ3" solutions

In fact, the figure above shows what previously explained about the choice of implementing a RCA with the "XOR" based "XMAJ3" full adder: while the worst case propagation time is the same for both architectures (and this can be noticed by the overlap of the two yellow lines), the power consumption is always higher for the "XNOR" based circuit, leading to a higher overall energy demand (Figure 57). It is also important to point out that the non-linearity of the power consumption growth is due to the approximations derived from the set of input transitions generated for simulations.

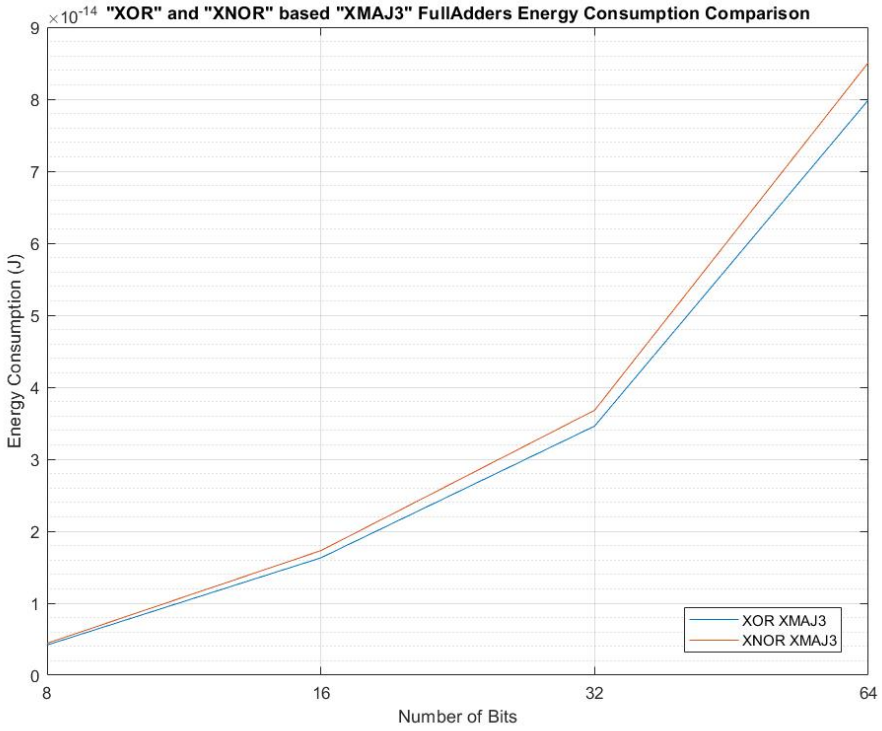
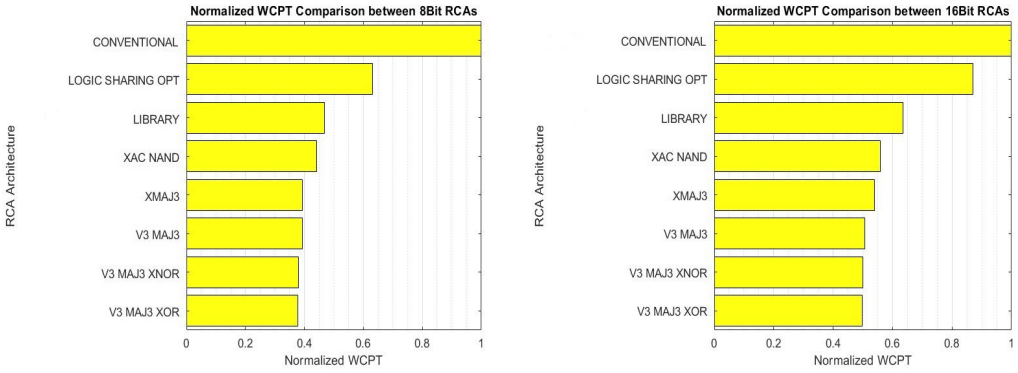
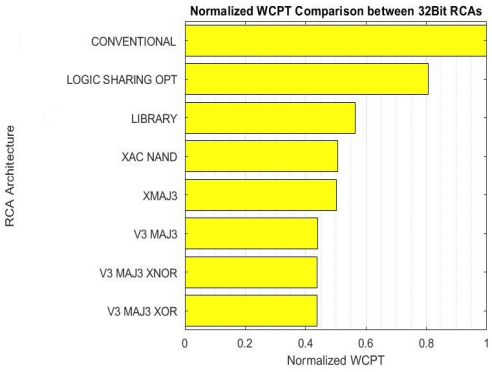


Figure 57: Energy Consumption Comparison between "XMAJ3" solutions

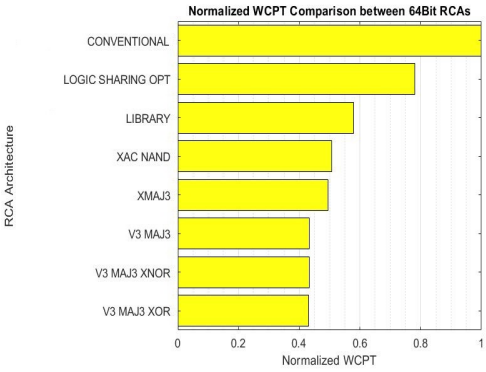
Subsequently, the different RCAs were evaluated and their results normalized with respect to the worst structure, which is the "CONVENTIONAL" one. This because the actual parameters of standard cells contained in the employed library cannot be directly shown due to agreements with the foundry.



(a) Normalized WCPT of 8-bit RCAs (b) Normalized WCPT of 16-bit RCAs

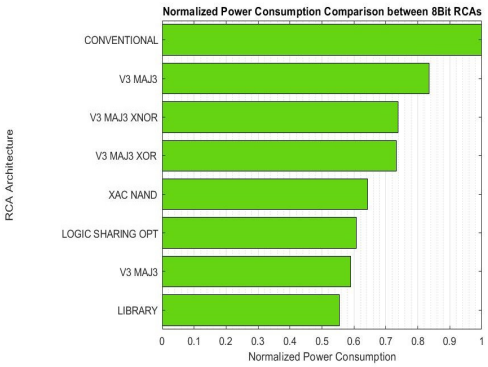


(c) Normalized WCPT of 32-bit RCAs

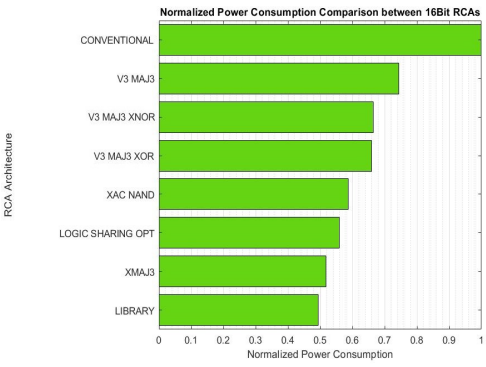


(d) Normalized WCPT of 64-bit RCAs

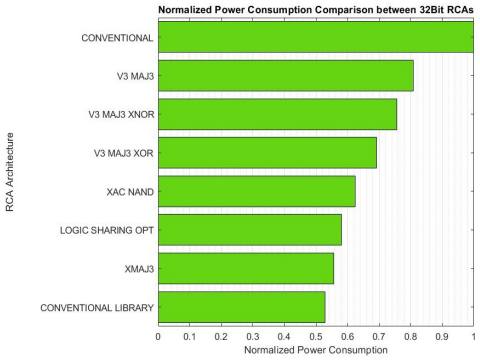
Figure 58: Normalized WCPT of RCAs up to 64 bits



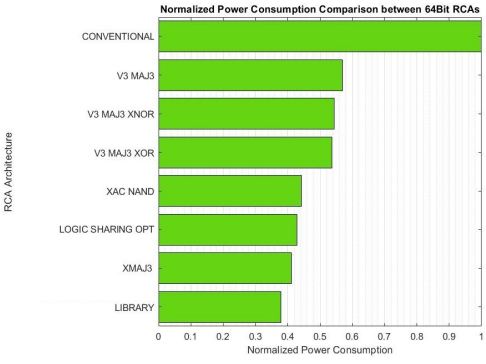
(a) Normalized Power Consumption of 8-bit RCAs



(b) Normalized Power Consumption of 16-bit RCAs

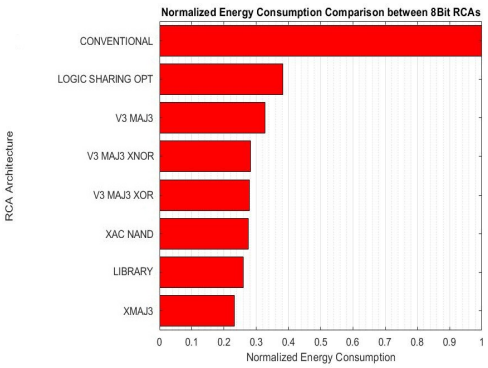


(c) Normalized Power Consumption of 32-bit RCAs

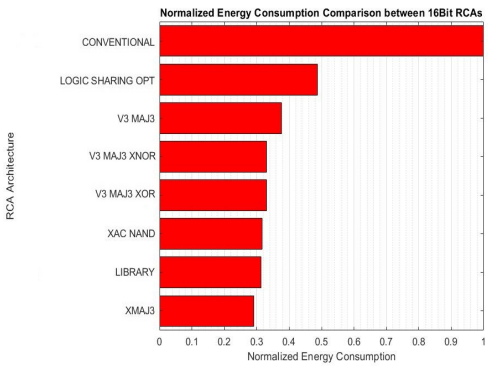


(d) Normalized Power Consumption of 64-bit RCAs

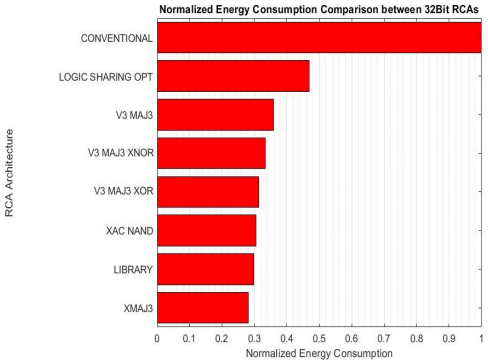
Figure 59: Normalized Power Consumption of RCAs up to 64 bits



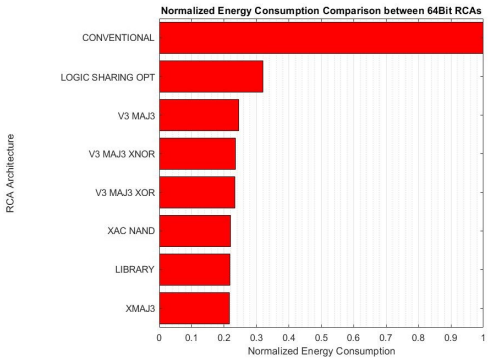
(a) Normalized Energy Consumption of 8-bit RCAs



(b) Normalized Energy Consumption of 16-bit RCAs



(c) Normalized Energy Consumption of 32-bit RCAs



(d) Normalized Energy Consumption of 64-bit RCAs

Figure 60: Normalized Energy Consumption of RCAs up to 64 bits

Figures 58, 59 and 60 help to understand how the performance of RCAs depends on the kind of FA utilized but also on the aforementioned parallelism effect and the change in the critical path. For instance, while on 1-bit FAs the proposed "XMAJ3" was the fastest, here its timing performance drops, so as to demonstrate what explained in 3.4. However, its energy consumption remains the smallest between all the architectures, and, most important, it outperforms the standard full adder cell contained in the library.

In parallel to this, Kogge-Stone adders were developed and synthesized on 8, 16, 32 and 64 bits, following the architecture shown in Figure 9, and subsequently simulated.

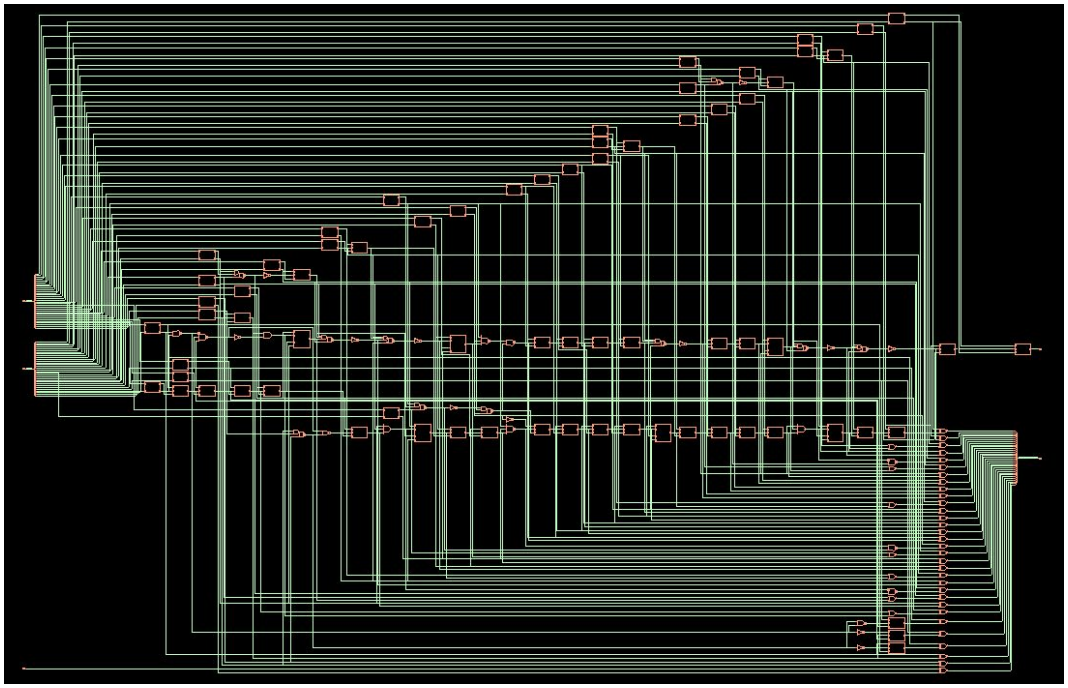


Figure 61: Synthesized 32-bit Kogge-Stone Adder

Particularly, not all the possible input combinations were tested (since, as before, they would have been too many), but 4 couples of values:

- 000...0 + 000...0;
- random couple that does not produce a carry-out;
- random couple that does produce a carry-out;
- 111...1 + 111...1.

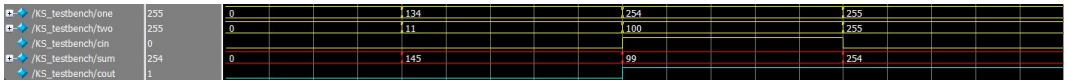


Figure 62: Simulation of 8-bit KS Adder after Synthesis

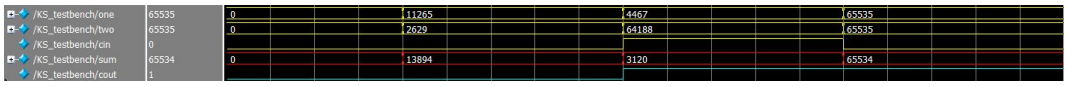


Figure 63: Simulation of 16-bit KS Adder after Synthesis

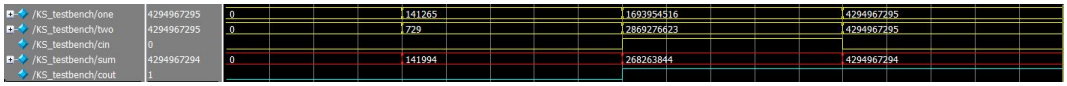


Figure 64: Simulation of 32-bit KS Adder after Synthesis

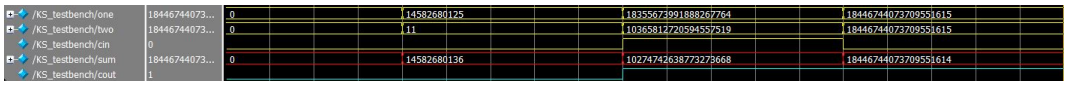


Figure 65: Simulation of 64-bit KS Adder after Synthesis

After having ascertained that the structures were synthesized correctly, they were simulated using Cadence Spectre. In order to have consistent results and be able to compare these adders to the previous RCAs, the same input transitions were employed. However, not all the structures have been added to the graphs, but only the most performing ones in terms of, respectively, energy consumption, worst case propagation time and power consumption: "XMAJ3", "V3 MAJ XOR" and the structure provided with the library.

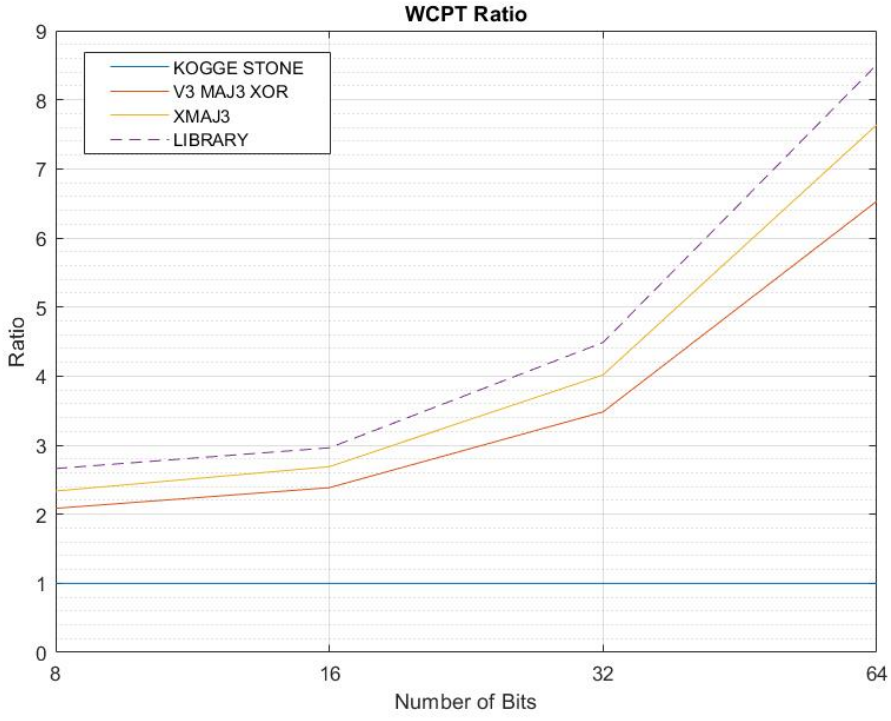


Figure 66: WCPT Ratio Comparison between Kogge-Stone and Ripple Carry Adders

As it can be seen, and as known from theoretical knowledge, the Kogge-Stone architecture, when supplied with nominal voltage, outperforms RCAs when talking about timing performance, and the difference increases as the number of bits grows. However, improvements can be done by designing optimized circuits, as Figure 66 shows: starting from a ratio of 4.43x on 32 bits with the circuit from the standard library (result that also proves the theoretical delay of RCA vs. Kogge-Stone [2]) it is possible to decrease it down to 3.46x. In addition to this, the more bits the devices are based on, the more the performance ratio goes up, but at the same time the more the same ratio can be decreased.

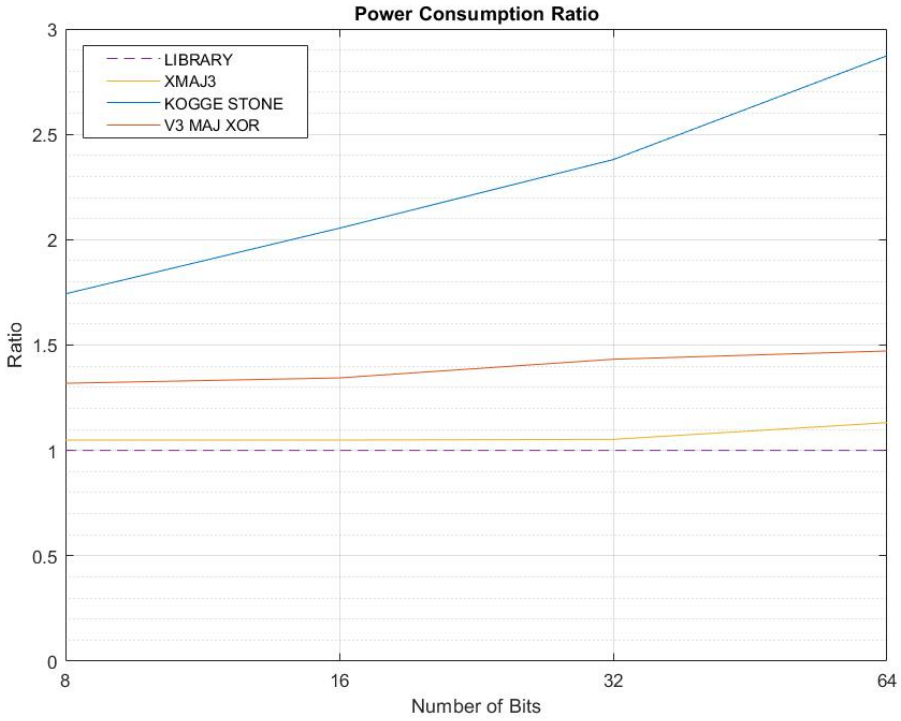


Figure 67: Power Consumption Ratio Comparison between Kogge-Stone and Ripple Carry Adders

On the other hand, an opposite trend can be found in power demanding, where the Kogge-Stone adder is by far the most consuming circuit (up to 2.75x more than the library cell based RCA).

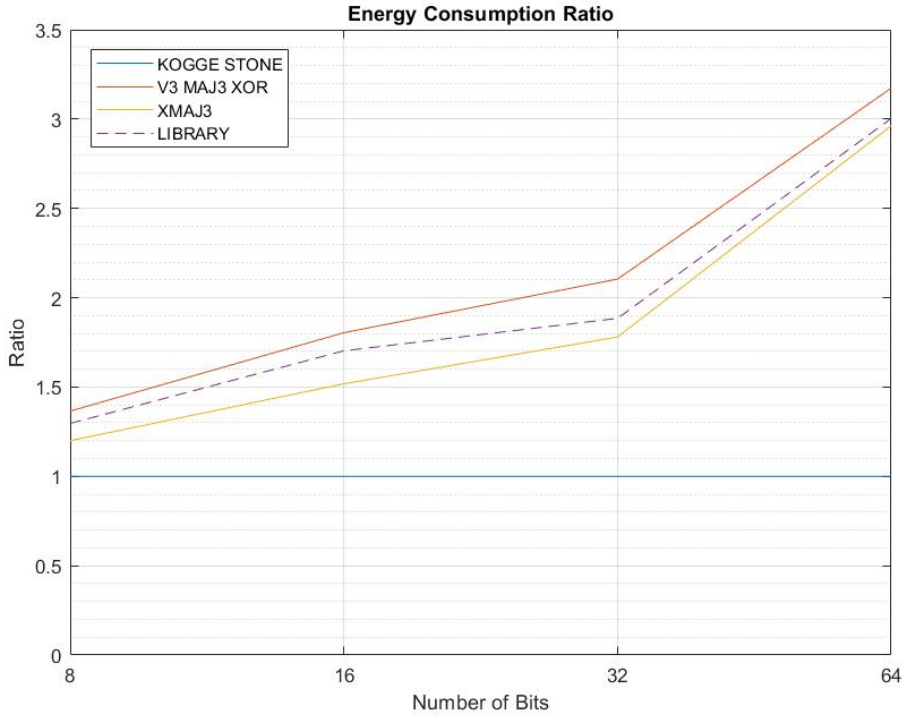


Figure 68: Energy Consumption Ratio Comparison between Kogge-Stone and Ripple Carry Adders

Finally, the energy consumption has been compared, and as it is possible to notice, there is a kind of paradox: the Kogge-Stone circuit is the one presenting the lowest energy consumption.
This is the main effect of the presence of no interconnects in these simulations.

Architecture	8 Bits	16 Bits	32 Bits	64 Bits
V3 XMAJ3 XOR	32	64	128	256
XMAJ3	24	48	96	192
Library	8	16	32	64
Kogge-Stone	51	131	323	771

Table 7: Number of Standard Cells composing the different Adder Structures

In fact, the number of cells composing the Kogge-Stone structure is, in case of the best candidate "XMAJ3", just almost 4x higher, but at the same

time the logic gates are more simple and less power demanding. Particularly, "AND" and "AO" gates (a single logic gate that embeds and "OR" and an "AND" in the same structure) are:

- 34 on 51 for the 8-bit structure;
- 98 on 131 for the 16-bit structure;
- 258 on 323 for the 32-bit structure;
- 642 on 771 for the 64-bit structure,

accounting for 66.7%, 74.8%, 79.9% and 83.3% of the total number of logic ports. Additionally, "ANDs" are 14.6% and 31.9% less energy consuming than "XORs" and "PAOs", while "AOs", respectively, 8.8% and 27.3%.
For these reasons, while the KS structure is up to 7.6x faster than the aforementioned RCA structure, it is just less than 3x more power demanding, leading to an apparent energy saving with respect to the serial architectures.
Therefore, interconnects must be inserted in the designs to be simulated, performing the Place & Route presented in the next chapter.

6 Place & Route

Place and route is a stage in the development of integrated circuits involving the decision of where to place all logic elements and, after this, the design of all the connections needed to make the placed components implement the final device. More specifically, for this piece of work, only the interconnects between the different standard cells were implemented, due to the availability of the "dummy" library, which does not take into account connections inside the logic gates (this because of proprietary licenses).

6.1 32 Bits

The first devices to be placed and routed were the 32-bit "XMAJ3" ripple carry and Kogge-Stone adder. For these structures, and also for the 64-bit based ones, a single optimization step was carried out, concerning the core utilization (so, trying to leave as little unfilled space as possible without creating congestion for routing). This because it was wanted to prove that, even without particular disposition of input/output pins and/or optimized dimensions of power supply and ground lines (which are techniques that would be employed ad-hoc), the initial thesis could still have been reached.
In particular, table 8 shows the height/width ratio of the final circuits and their corresponding core utilization.

Device	H/W Ratio	Core Utilization %
XMAJ3	0.13	70.588235
Kogge-Stone	0.42	70.583097

Table 8: Optimized Core Utilization Percentages with Correlated H/W Ratios for 32-bit Adders

After this step, two rings for power supply and ground were created, both 0.7 μm wide and 0.7 μm distant to each other. Moreover, the pins were disposed in the most general way possible, which consisted of having the 3

input signals on the top of the structure and the 2 output ones at the bottom. This configuration is helped by the fact that, due to the circuit architectures, the routed devices have rectangular aspects.

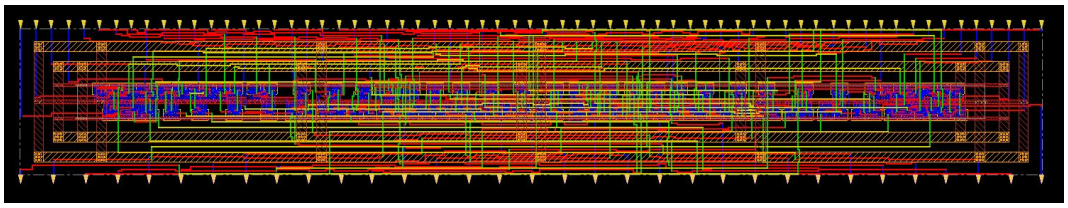


Figure 69: Placed and Routed "XMAJ3" FA based 32-bit RCA

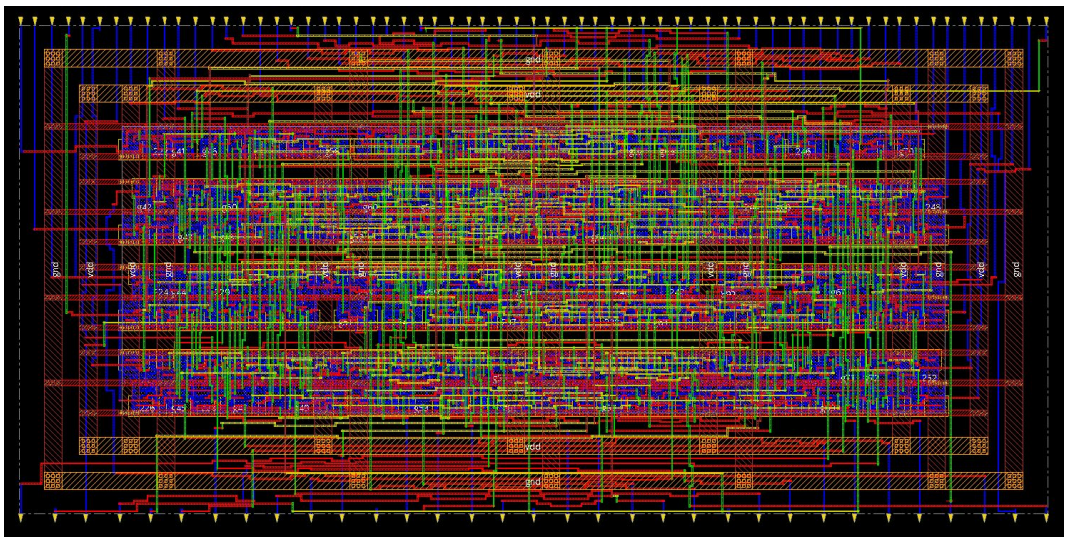


Figure 70: Placed and Routed 32-bit Kogge-Stone Adder

Figures 71 and 70 show the final adders implementations; it is important to point out that the ratio between areas of the proposed RCA architecture and the KS adder is way higher than the expected. This is due to the fact that the library FA based ripple carry adder presents more simple interconnects if compared to the "XMAJ3" FA based one, since the composition of the 1-bit full adders on which the aforementioned 32-bit devices are built is different: while the former is composed of a single standard cell, the latter presents 3 of them. This means that, in addition to the interconnects linking the different blocks of the RCA, additional connections inside the FAs themselves are needed. Particularly,if taking into account the library FA based RCA the previously

mentioned value is 6.43 (which is aligned to the theoretical 7 amount), for the "XMAJ3" based structure it drops to 3.31.

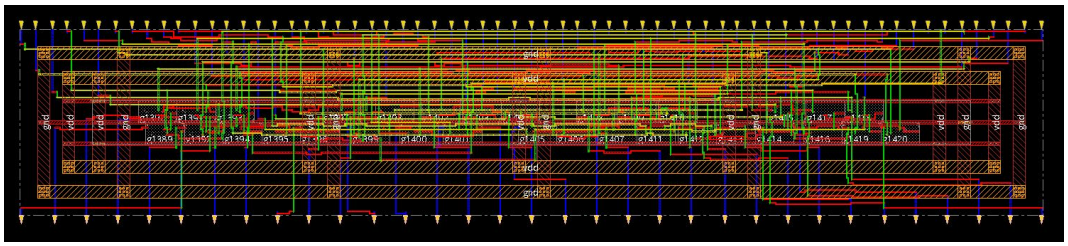


Figure 71: Placed and Routed Library FA based 32-bit RCA

6.2 64 Bits

In the same way as before, and with the same parameters, the 64-bit circuits were placed and routed, leading to comparable results.

Device	H/W Ratio	Core Utilization %
XMAJ3	0.2	70.57588
Kogge-Stone	0.99	70.583856

Table 9: Optimized Core Utilization Percentages with Correlated H/W Ratios for 64-bit Adders

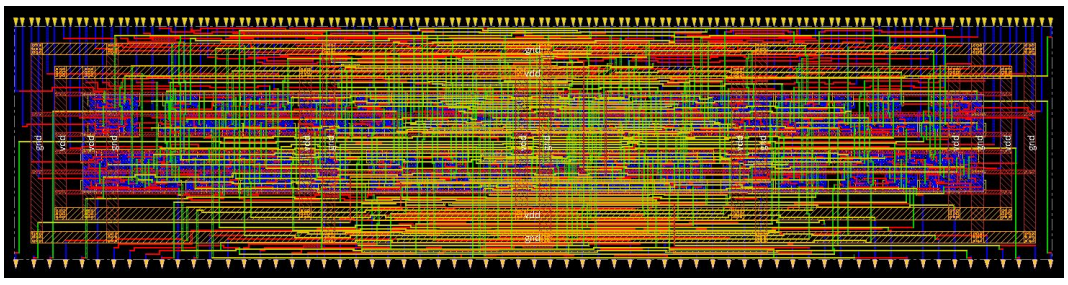


Figure 72: Placed and Routed "XMAJ3" FA based 64-bit RCA

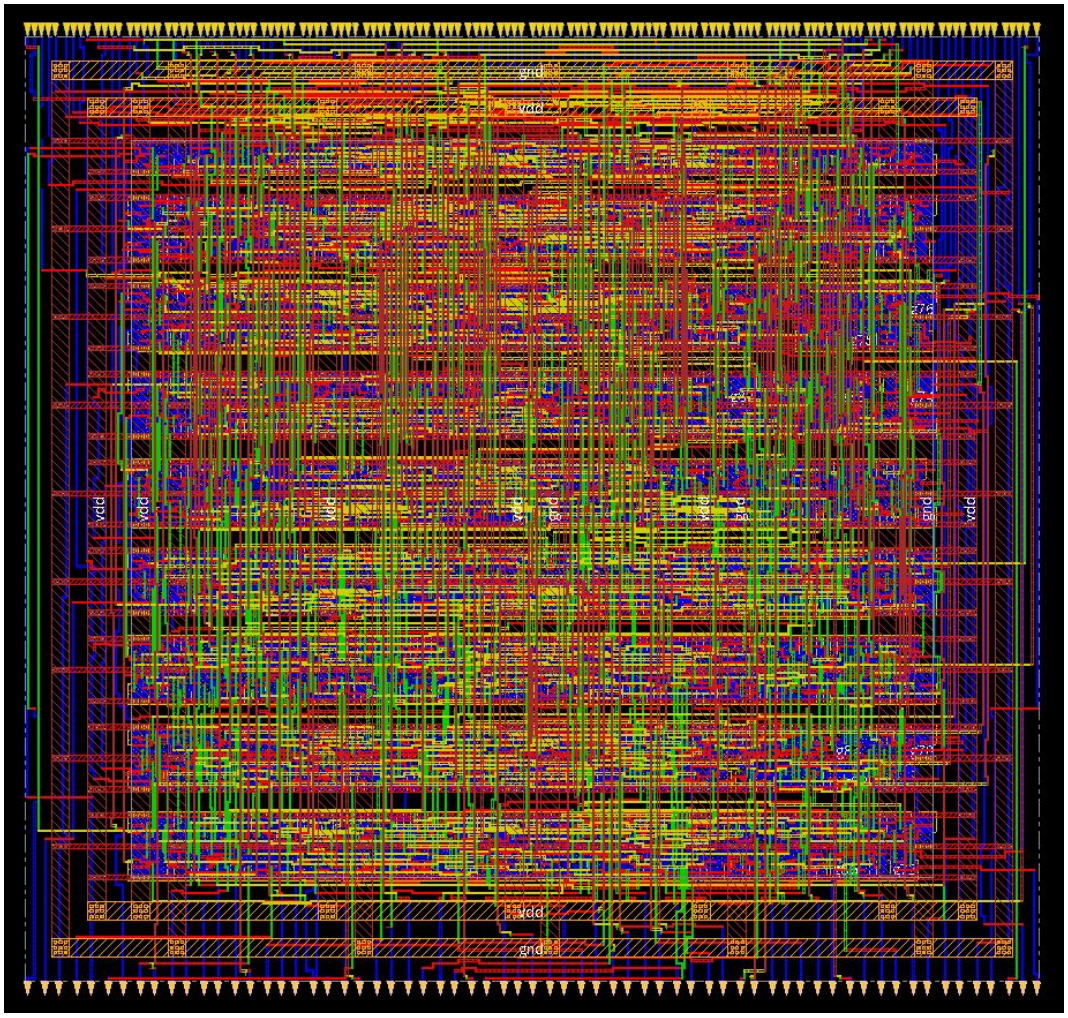


Figure 73: Placed and Routed 64-bit Kogge-Stone Adder

Finally, for the "XMAJ3" based and the Kogge-Stone adders, the .spf (Standard Parasitic Format) files were extracted, meaning that, in a Spice based netlist, interconnects are modeled as described in Figure 74.

```
*|NET TWO[26] 0.002726PF
*|P (TWO[26] I 0.000000PF 19.448 3.100)
*|I (g1394:B0 g1394 B0 I 0.002589PF 5.077 11.825)
*|S (TWO[26]_2 18.972 3.100)
*|S (TWO[26]_3 15.436 4.500)
*|S (TWO[26]_4 8.480 4.500)
C1_8231 TWO[26] 0 0.000017PF
C2_8231 g1394:B0 0 0.000646PF
C3_8231 TWO[26]_2 0 0.000283PF
C4_8231 TWO[26]_3 0 0.000700PF
C5_8231 TWO[26]_4 0 0.001080PF
R1_8231 g1394:B0 TWO[26]_4 66.658257
R2_8231 TWO[26]_4 TWO[26]_3 33.384323
R3_8231 TWO[26]_3 TWO[26]_2 34.403084
R4_8231 TWO[26]_2 TWO[26] 1.954794
```

Figure 74: Description of Interconnects in .spf Files

6.3 Simulations at Nominal Voltage

After having added the structure described in 3.1.2 (the adder with the addition of input/output inverter chains), a first simulation at nominal voltage of 0.8V was carried out for all the aforementioned placed and routed structures, in order to collect quantitative data about how much interconnects influence the parameters (delay, power and energy consumption) that will be measured.

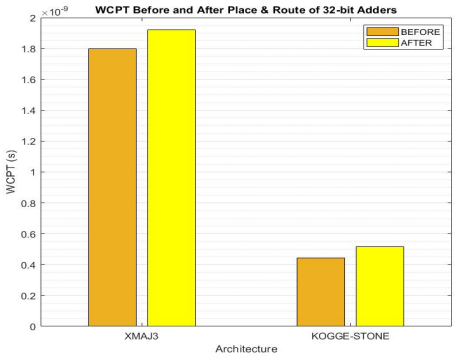


Figure 75: WCPT Before and After P&R of 32-bit Architectures

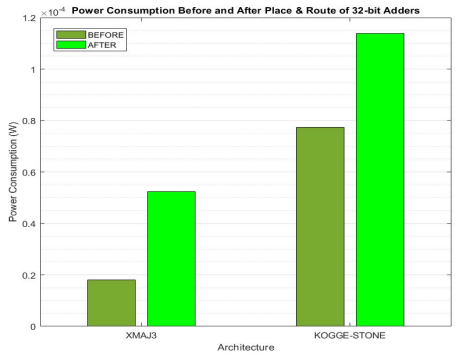


Figure 76: Power Consumption Before and After P&R of 32-bit Architectures

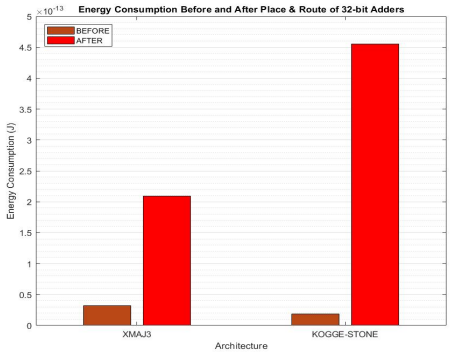


Figure 77: Energy Consumption Before and After P&R of 32-bit Architectures

As it can be seen form Figures 75 and 76, difference in delay is not as marked as in the case of power consumption; in fact, while rise and fall times are exponentially proportional to the inverse of the propagation constant, power consumption is the direct sum of all the possible sources of power consumption.

In addition, after having taken into account connections between logic gates, the Kogge-Stone circuit is the most energy consuming one, proving what anticipated at the end of 5.

Particularly, table 10 describes the difference in terms of percentage before and after the introduction of interconnects and, therefore, parasitics.

Device	WCPT %	Power Consumption %	Energy Consumption %
XMAJ3	6.67	191.11	444.43
Kogge-Stone	17.19	47.29	2400

Table 10: Percentage of Difference in Measured Parameters Before and After P&R for 32-bit Adders

Of course the same effect is found on 64-bit circuits, but in this case the error percentages are even higher than in the case of 32-bit architectures, due

to the fact that more parasitics are added.

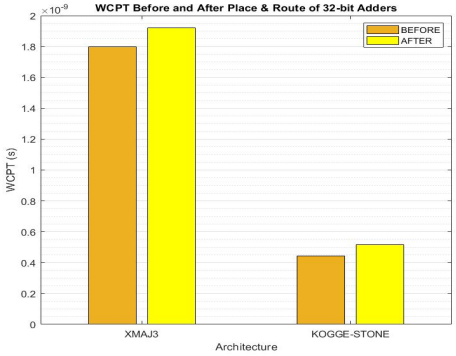


Figure 78: WCPT Before and After P&R of 64-bit Architectures

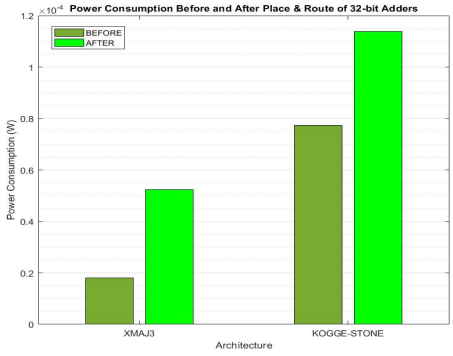


Figure 79: Power Consumption Before and After P&R of 64-bit Architectures

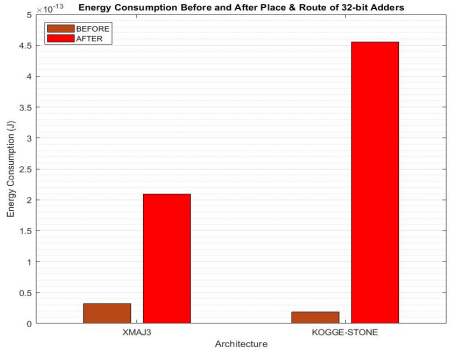


Figure 80: Energy Consumption Before and After P&R of 64-bit Architectures

Device	WCPT %	Power Consumption %	Energy Consumption %
XMAJ3	8.36	387.67	546.29
Kogge-Stone	44.47	99.18	4185.71

Table 11: Percentage of Difference in Measured Parameters Before and After P&R for 64-bit Adders

Therefore, after the first simulations at 0.8V, which helped to prove that interconnects have a vital role in the overall behavior of digital circuits, the

final results would have been retrieved running other processes, decreasing the power supply voltage and reaching the sub-threshold region.

7 From Super to Sub-Threshold

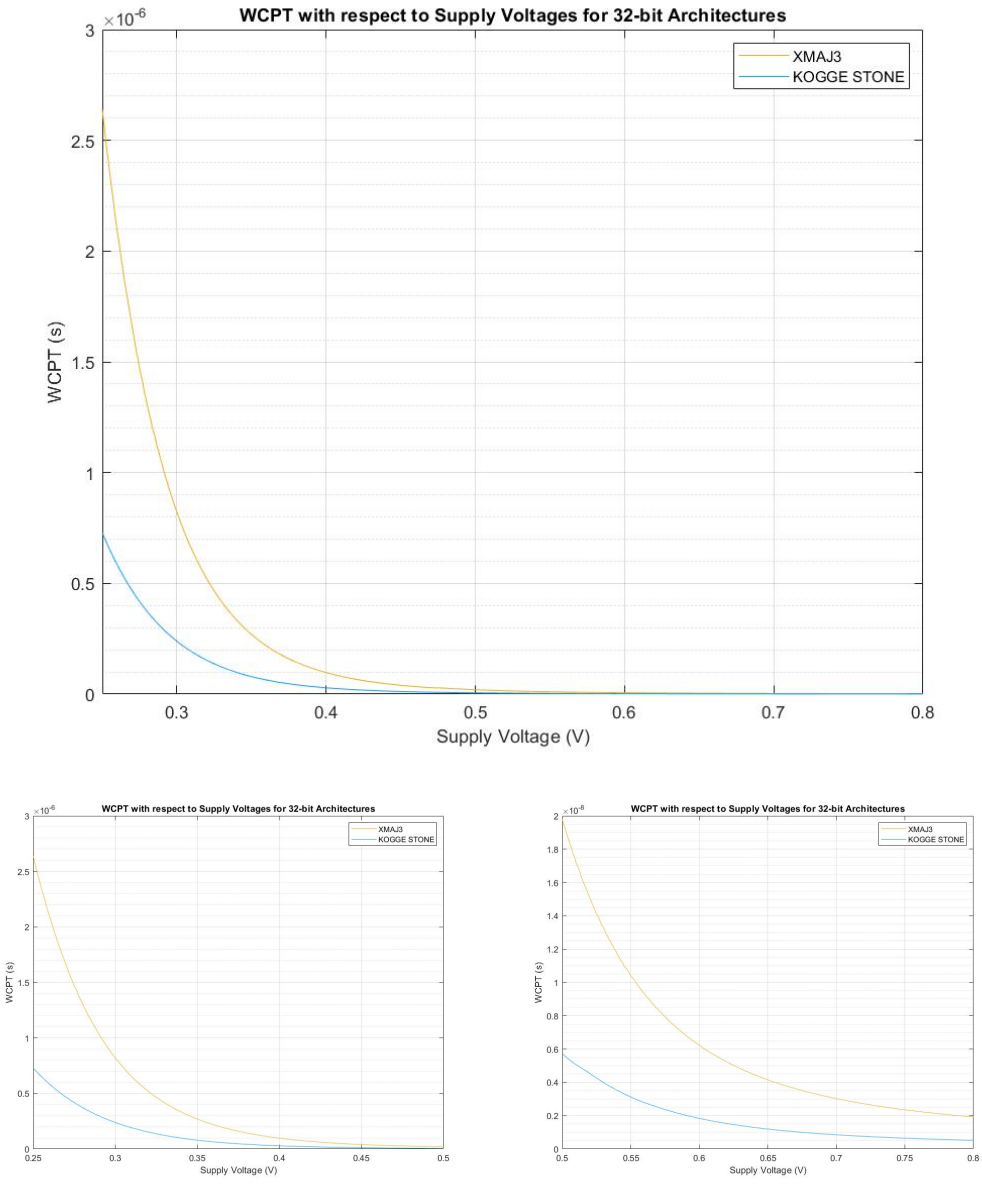
As final step of this work, after having studied and proposed a new 1-bit FA circuit, having taken into account different input generation methods to employ the one which fitted the requirements better, having compared different RCAs with a Kogge-Stone adder and performed the place and route, simulations had to be run in order to measure worst case propagation time, power and energy consumption going to sub-threshold region.
In particular, for both 32 and 64-bit based architectures, 550 simulations were performed, going from 0.8V (nominal voltage for the employed standard cell library) down to 0.25V, limit value after which the circuits would not have been working correctly anymore. Therefore, for each step, the supply voltage was decreased by 0.001V.

7.1 32 Bits

Starting from the aforementioned 32-bit adders, performance were measured at each step, but in a different way if compared to the previous physical simulations. In fact, if until now every device was simulated at a fixed frequency (for instance 500MHz up to 16 bits and 250MHz until 64 bits), applying the same technique for the final simulations would have driven to misleading results. For instance, if the frequency had been set so as to let the circuit finish its computation at the lowest voltage (0.250V), for the highest supplies this would have meant measuring leakage mostly, since their delay would have been orders of magnitudes smaller. On the other hand, choosing the highest possible frequency for high voltages would not have let sub-threshold computations finish.
Therefore, for each voltage step, an ad-hoc frequency was chosen, and in particular the worst case propagation time delay was employed as time limit for both power and energy consumption, as shown in Figure 81.

```
.MEASURE TRAN Qtot INTEGRAL i(vdd_dut) FROM=0ns TO=C_C1
.MEASURE Etot PARAM='VOLTAGE*Qtot'
.MEASURE pwr PARAM='Etot/C_C1'
```

Figure 81: Variable Frequency Measurement in .spf Files



(a) From 0.25V To 0.5V

(b) From 0.5V To 0.8V

Figure 82: WCPT Change with respect to the Decreasing Supply Voltages for 32-bit Adders

In Figure 82 the dependence of delay with respect to supply voltage is

shown, and the more the aforementioned quantity is decreased, the more the speed of devices goes down (with an exponential correlation, as described in 15). However, comparing the different behaviors of the two circuits under test, it is possible to see that, with higher supply voltages with respect to the ones applied to the Kogge-Stone architecture, the ripple carry adder is able to reach its same timing performance.

In addition to this, and most importantly, equalizing the worst case propagation times of the different adders, it is possible to prove the initial thesis of this piece of work: in fact, as Figure 83 describes, the energy consumption of the serial circuit is lower than the parallel device one.

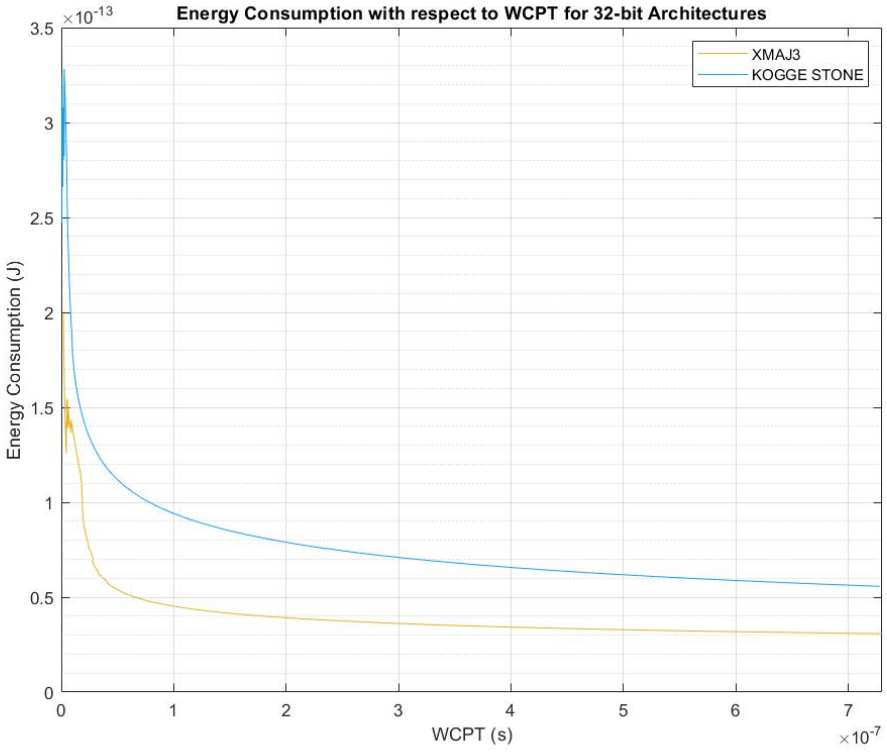


Figure 83: Energy Consumption with respect to WCPT on 32 Bits

In addition to this, Figure 84 describes the difference in terms of supply voltage between the RCA and the KS architectures that is required for the timing performance to be equalized.

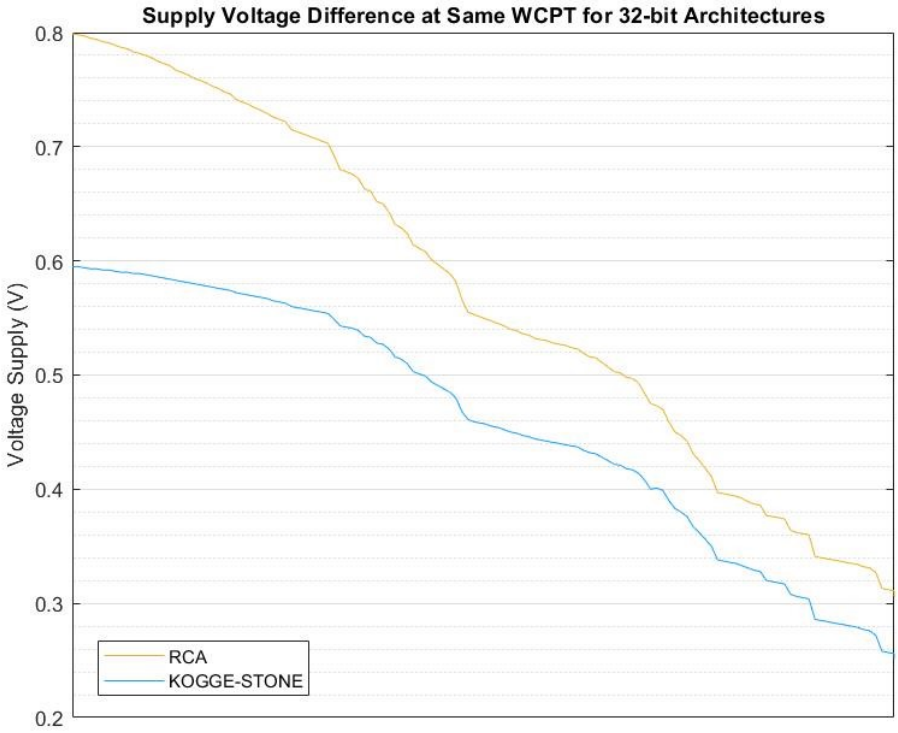


Figure 84: Difference of V_{DD} for Same WCPT on 32 Bits

Particularly, it is possible to notice that, starting from the maximum gap of 0.204V, it starts decreasing exponentially, reaching the final minimum difference of 0.055V, due to the exponential nature of sub-threshold current.
Having said this, Figure 85 illustrates the energy saving percentage of the "XMAJ3" ful adder based RCAs with respect to the Kogge-Stone solution.
First of all, it is important to point out that the drops of the blue line (which describes the energy saving percentage of the devices) are due to approximations in the comparisons of worst case propagation times. In fact, in order to check for equal timing performance, ranges of value had to be considered, and this implied to have, for certain values, smaller differences in energy consumption.
Secondly, while the average power saving is 41.48% with respect to the parallel adder, the trend of the graph can be brought back to a common path (as il will be seen for the 64-bit case of study): there are two peaks, one of which is present even before going to sub-threshold region, and as it is described in Ta-

ble 12, for the 32-bit based devices this accounts for the highest energy saving possible. Before going to sub-threshold, the energy saving decreases, reaching values below the average, increasing again for lower voltages and reaching a second peak, whose value is lower than the first one.

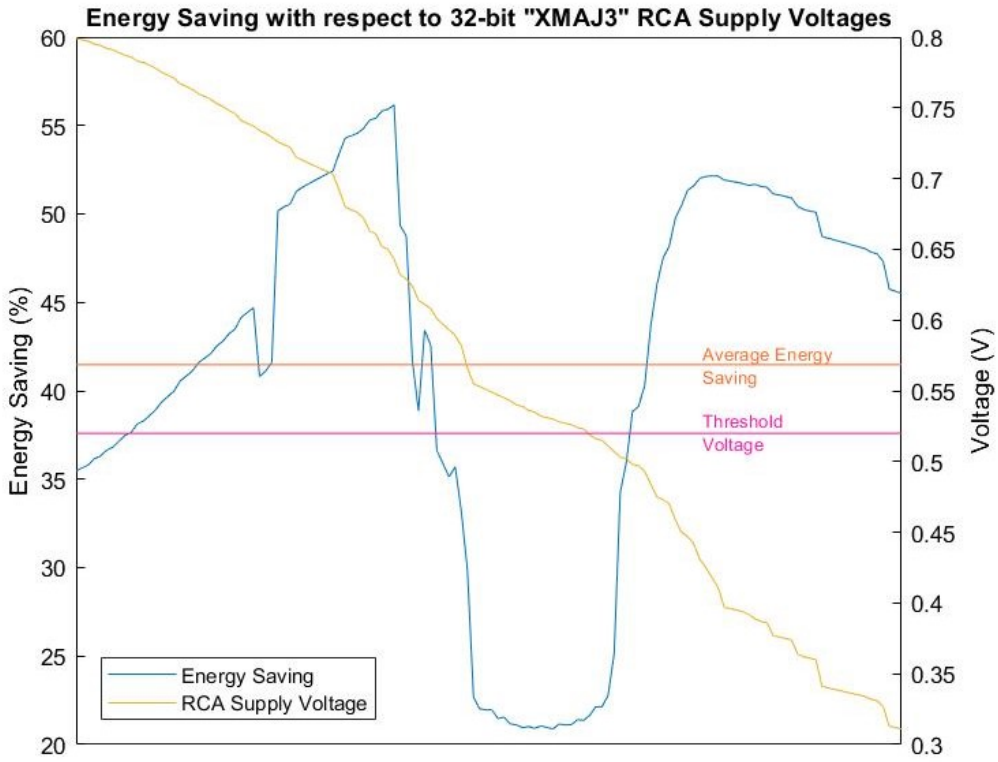


Figure 85: Energy Saving % of "XMAJ3" based RCA with respect to KS on 32 Bits

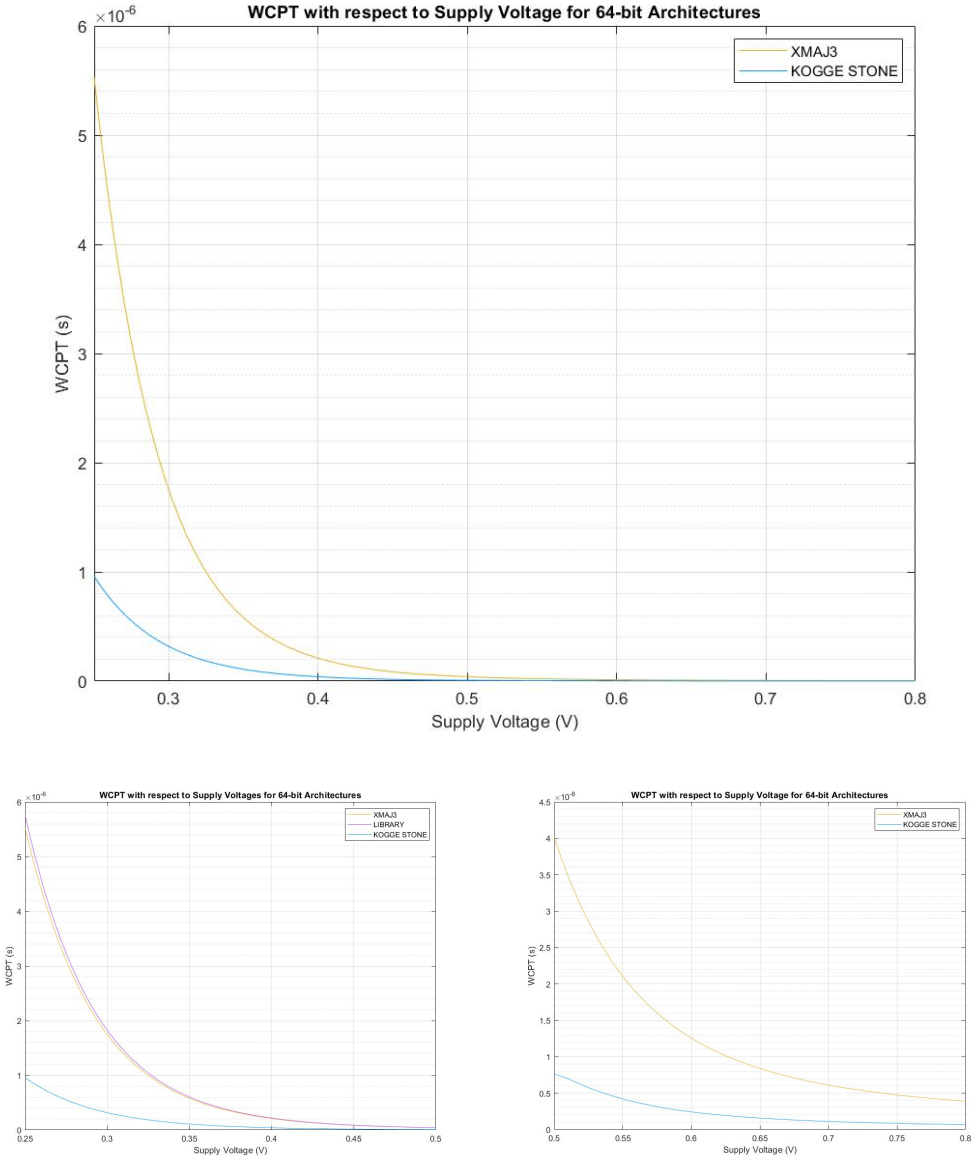
Device	Energy Saving %	Power Supply (V)%
XMAJ3	52.14	0.411
XMAJ3	56.16	0.643

Table 12: Energy Saving Peaks for 32-bit Adders

This is an important result, mostly because knowing this, and taking into account the entire trend, it would be possible to design microelectronic devices, aiming to low-power purposes, that minimize their energy consumption without dealing with all the problems that sub-threshold design implies.

7.2 64 Bits

As for the 32-bit devices, the same steps were carried out for the 64-bit adders, retrieving results that present both similarities and differences.



(a) From 0.25V To 0.5V

(b) From 0.5V To 0.8V

Figure 86: WCPT Change with respect to the Decreasing Supply Voltages for 64-bit Adders

To start with, the same worst case propagation time tendency can be found in Figure 86, where the Kogge-Stone, at the same voltage, always outperforms

the RCA structure. But, as previously seen, downscaling the supply voltage and equalizing the WCPTs, the energy consumption of the former is, again, higher than the latter one, proving the initial thesis on 64 bits as well.

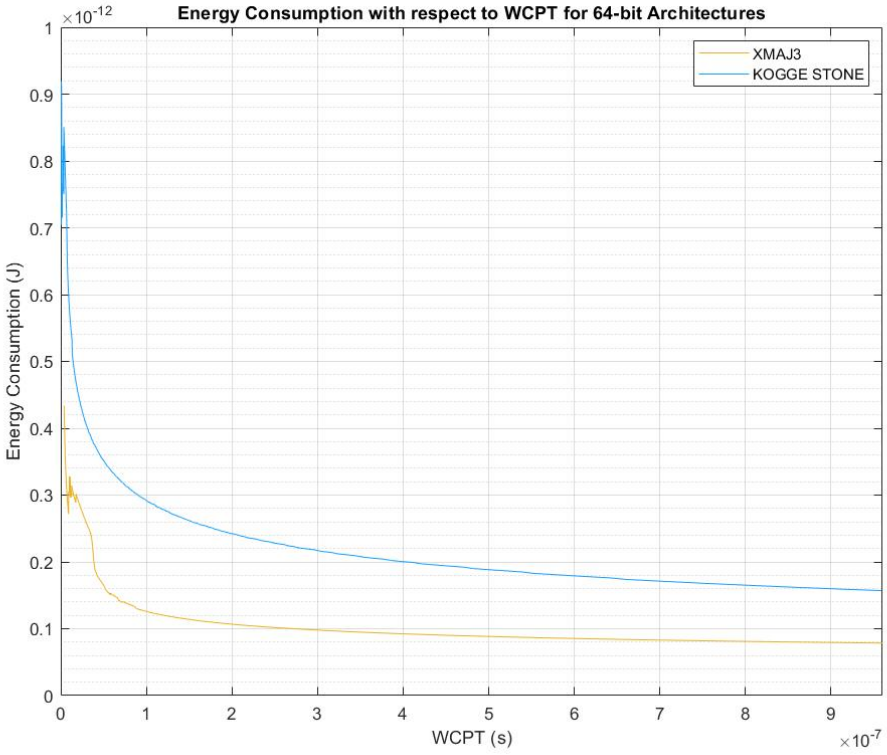


Figure 87: Energy Consumption with respect to WCPT on 32 Bits

However, in this case, the initial difference of supply voltages, which is shown in Figure 88, is higher than the previous, accounting for 0.240V (17.65% more), leading to the result that the more bits the architectures are based on, the higher the aforementioned gap will be in order to equalize their performance.

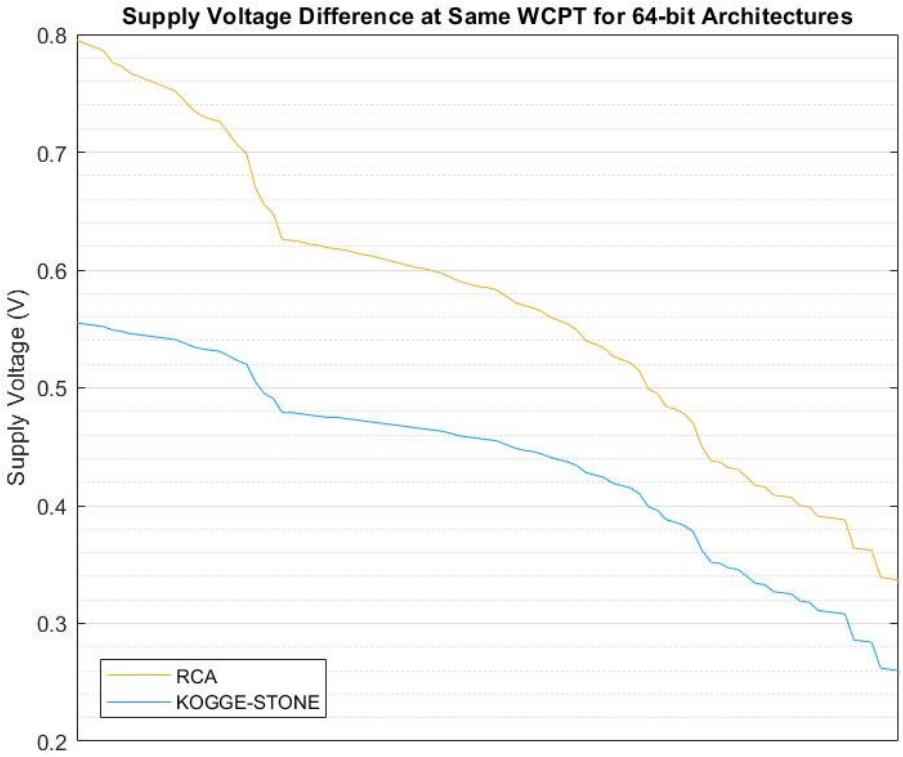


Figure 88: Difference of V_{DD} for Same WCPT on 64 Bits

At the same time, the energy saving is higher as the number of bits will be increased, as it can be observed in Figure 89.

First of all, the average energy saving with respect to the Kogge-Stone adder accounts for 50.02% and, in addition to this, the two peaks (super and sub-threshold) of saving have higher percentage values if compared to the 32-bit case: 56.72% and 56.83%.

Device	Energy Saving %	Power Supply (V)%
XMAJ3	56.83	0.443
XMAJ3	56.72	0.683

Table 13: Energy Saving Peaks for 64-bit Adders

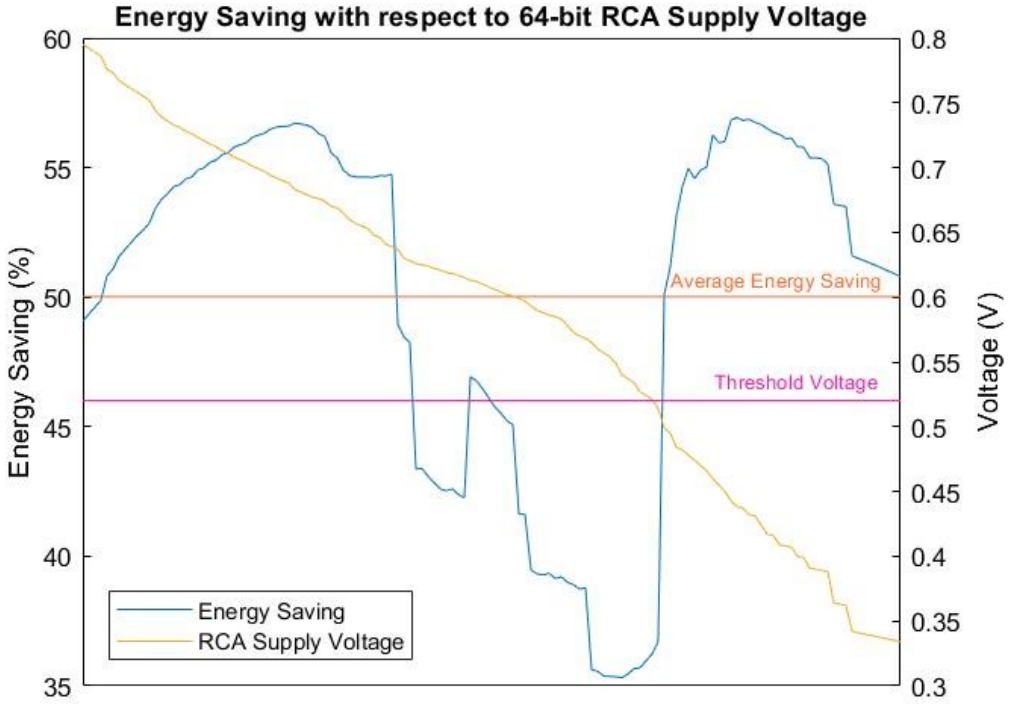


Figure 89: Energy Saving % of "XMAJ3" based RCA with respect to KS on 64 Bits

Finally, it is possible to notice that, when the number of bits is increased, the sub-threshold energy saving peak correlated to the "XMAJ3" based RCA overcomes the super-threshold one, even though of a quantity that might be considered irrelevant (a difference of 0.11%).

8 Conclusions

Having seen the final simulation results and analyzed the piece of data retrieved from them, it is possible to draw the conclusions about the work done. First of all, with the newly presented full adder cell, the overall energy consumption of the already existing ones can be reduced. In addition to this, extending the analysis to RCA structures, the simulations lead to the same result, meaning that it would make sense to explore different transistor level implementations of the proposed architecture in order to maximize its performance. In fact, when interconnects are taken into account, the advantage of the RCAs based on the proposed full adder block is greatly reduced.

Secondly, when the voltage supply is decreased, reaching the sub-threshold region, it has been proved that it is possible to equalize the same timing performance of a Kogge-Stone adder and, at the same time, saving energy even if the voltage applied to the serial architecture is higher than the one applied to the parallel circuit.

Finally, this aforementioned behavior leads to the most important result of this piece of work: in addition to what presented in the study [2], there are two peaks of energy saving: one in super-threshold and the other one in sub-threshold region, meaning that, employing FDSOI technology, energy can be saved even without designing circuit for sub-threshold operations. More particularly, up to 32 bits the super-threshold peak corresponds to the maximum energy saving possible, while from 64 bits it is overcome by the sub-threshold vertex, leading to the hypothesis that the more bits serial adders are based on, the more sub-threshold voltages can save energy.

However, on 64 bits, the difference is so small that it would be better to stick to higher voltages, which allow to reach higher frequencies and computational performance.

8.1 Future Work

Further continuation on this project would include:

- implementing the proposed "XMAJ3" full adder cell employing customized gates instead of using standard logic cells taken from libraries, in order to improve its performance;
- studying the "Constrained" and "Random Constrained" input generation methods, so as to retrieve better data about percentages of error on the measurements;
- carrying out super to sub-threshold simulations on more than 64-bit adders. In fact, it may be proved the fact that the more the number of bits, the more working in sub-threshold will allow to save energy with respect to super-threshold region.

References

[1] N.H.E. Waste, D.M. Harris, *CMOS VLSI Design - A Circuit And System Perspective*. Pearson; 4 edition, 2010.

[2] V. Beiu, A. Gjupdal, S. Aunet, *Ultra Low-Power Neural Inspired Addition: When Serial Might Outperform Parallel Architectures*. IWANN 2005, LNCS 3512, pp.486-493, 2005.

[3] Wikipedia, *90 nanometer*. https://en.wikipedia.org/wiki/90_nanometer

[4] Cao, Y., Sato, T., Orshansky, M., Sylvester, D., Hu, C., *New paradigm of predictive MOSFET and interconnect modeling for early circuit design*. Proc. IEEE Custom Integrated Circuits Conference, pp.201–204, 2000.

[5] Borivoje Nikolic, *The Wire*. Course Notes for Digital Integrated Circuits at Berkeley University, 1999.

[6] M. Horowitz, *Adders*. Course Notes for Computer Systems Laboratory at Stanford University, 2006.

[7] Shilpa C. N, Kunjan D. Shinde, Nithin H. V., *Design, Implementation and Comparative Analysis of Kogge-Stone Adder using CMOS and GDI design: A VLSI Based Approach*. 8th International Conference on Computational Intelligence and Communication Networks, 2016.

[8] R. Kiran, S. Nampally, *ANALYZING THE PERFORMANCE OF CARRY TREE ADDERS BASED ON FPGAS*. International Journal of Electronics Signals and Systems, 2012.

[9] C.G.B. Garrett, W.H. Brattain, *Physical Theory of Semiconductor Surfaces*. Physical Review, vol.99, p.376, 1955.

[10] L. Zhong, H. Song, X. Lai, D. Xu, *Differential Capacitive Readout Circuit Using Oversampling Successive Approximation Technique*. IEEE Transactions on Circuits and Systems I, 2018.

[11] D. Jennifer Judy, V s kanchana Bhaaskaran, *Review and Analysis of the Impacts and Effects on Low Power VLSI Circuits Operating in Sub-threshold Regime*. International Journal of Engineering and Technology, pp.3870-3883, 2013.

[12] *Ultra-low-power CMOS Technologies*. G. Schrom, S. Selberherr, International Semiconductor Conference, 1996.

[13] B. Liu, *Standard Cell Library Design for Sub-Threshold Operation*. Doctoral Thesis, Eindhoven University of Technology, 2014.

[14] A. Wang, B.H. Calhoun, A.P. Chandrakasan, *Sub-Threshold Design For Ultra Low-Power Systems*. Springer, 2006.

[15] R.S. Timsit, *Electrical Contact Resistance: Properties of Stationary Interfaces*. AMP Global Technology, 1998.

[16] M. Graziano, *Microelectronic Systems - Lecture Notes*. Politecnico di Torino, 2013.

[17] IBM100, *Copper Interconnects, The Evolution of Microprocessors*. https://www.ibm.com/ibm/history/ibm100/us/en/icons/copperchip/

[18] K.A.K. Maurya, K.B Sindhuri, Y.R. Lakshmanna, N.U. Kumar, *Design and Implementation of 32-bit Adders Using Various Full Adders*. International Conference on Innovations in Power and Advanced Computing Technologies, 2017.

[19] Z. Tabassum, M. Shahrin, A. Ibnat, T. Amin, *Comparative Analysis and Simulation of Different CMOS Full Adders Using Cadence in 90nm Technology*. 3rd International Conference for Convergence in Technology, 2018.

[20] N. Mastorakis, *High Speed Gate Level Synchronous Full Adder Designs*. WSEAS Transactions on Circuits and Systems, 2009.

[21] R. Duarte, S. Manich, J.Figueras, *Fault-Secure Parity Prediction Arithmetic Operators*. IEEE Design and Test of Computers, 1997.

[22] S. Aunet, Y. Berg *Three Sub-fJ Power-Delay-Product Subthreshold CMOS Gates*. IFIP VLSI SoC, 2005.