Spurious Transitions in Adder Circuits : Analytical Modelling and Simulations

S. J. Abou-Samra[†]*, A. Guyot*[†]*, B. Laurent*[‡]
[†] *TIMA laboratory* [‡] *CSI-INPG laboratory*
46 av. Félix Viallet, F-38031 Grenoble - France
Tel.: (33) 4 76 57 48 12 Fax: (33) 4 76 47 38 14
E_Mail : Abou-Samra@imag.fr

Abstract
Adder architectures are presented here by an unified formalism, and analysed from the delay, complexity and power consumption points of view. An analytical model for the power consumption is derived, assuming that it is proportional to the transition density [DHNT95]. The model is subsequently validated by simulation using a signal transition probabilities propagation tool [Cra89]. Finally, glitches are taken into account when transitions at the input of a cell are separated by one or more cell delays. A redundant to total power ratio is also derived.

Keywords
Adder, BDD, glitch threshold, low power, spurious transition, switching activity

1 INTRODUCTION

Addition is the most frequently used arithmetic primitive, involved not only in simple addition but also in more complex operations like multiplication and division. The present study covers the linear ripple carry adder and different architectures of carry select and carry lookahead adders.

Designing low-power high-speed circuits requires a combination of techniques at four levels : technology, circuitry, architectures and algorithms [BCS92]. This work concentrates on the architecture level and considers a CMOS static technology.

This work is supported by the European Community ESPRIT project 20023 HIPERLOGIC

The paper is organised as follows: the Δ operator introduced by Brent and Kung [BrKu82] is first recalled. Then it is used to describe several well known adder architectures by an unified formalism. An analytical power consumption model is derived first for the ripple carry adder, and extended to other architectures. The notion of Glitch Threshold is then introduced and validated by HSPICE simulations, providing a glitch filtering model usable by the power evaluation software [Cra89]. Finally, concluding remarks and a brief presentation of the future work is given.

2 THE Δ OPERATOR

Let us consider an adder that computes S = A + B, where $A = \sum_{i=0}^{n-1} a_i * 2^i$ and $B = \sum_{i=0}^{n-1} b_i * 2^i$. At every position i, the next carry c_{i+1} is either generated, i.e. $c_{i+1} = 1$, propagated, i.e. $c_{i+1} = c_i$ or killed, i.e. $c_{i+1} = 0$ according to the values of the digits a_i and b_i. So three signals can be defined, one for each case: $g_i = a_i \wedge b_i$, $p_i = a_i \oplus b_i$, and $k_i = \overline{a_i \vee b_i}$.

Then let us note $\mathbf{P}_{i,j}$ the group propagate and $\mathbf{G}_{i,j}$ the group generate, with $n\text{-}1 \geq i \geq j \geq 0$. $\mathbf{P}_{i,j}$ means that the carry propagates from position j up to position i, that is that c_{i+1} is equal to c_j. $\mathbf{P}_{i,j} = \prod_{n=i}^{j} p_n$. $\mathbf{G}_{i,j}$ means that a carry is generated somewhere between j and i and propagated from this location up to position i and yields $c_{i+1} = 1$. $\mathbf{G}_{i,j} = g_i \vee \sum_{n=i}^{j} \left(\mathbf{P}_{i,n+1} \wedge g_n\right)$.

Clearly, one has $\mathbf{P}_{i,i} = p_i = a_i \oplus b_i$, $\mathbf{G}_{i,i} = g_i = a_i \wedge b_i$, $\mathbf{P}_{i,j} \wedge \mathbf{G}_{i,j} = 0$ and $c_{i+1} = \mathbf{G}_{i,0}$. For any k such that $n\text{-}1 \geq i \geq k \geq j \geq 0$, the pair of bits ($\mathbf{P}_{i,j}$, $\mathbf{G}_{i,j}$) can be computed from ($\mathbf{P}_{i,k}$, $\mathbf{G}_{i,k}$) and ($\mathbf{P}_{k\text{-}1,j}$, $\mathbf{G}_{k\text{-}1,j}$) in the following way:

$$(\mathbf{P}_{i,j}, \mathbf{G}_{i,j}) = (\mathbf{P}_{i,k} \wedge \mathbf{P}_{k-1,j}, \mathbf{G}_{i,k} \vee \mathbf{P}_{i,k} \wedge \mathbf{G}_{k-1,j}) \qquad (1)$$

Is noted Δ the operator such that:

$$(\mathbf{P}_{i,j}, \mathbf{G}_{i,j}) = (\mathbf{P}_{i,k}, \mathbf{G}_{i,k})\ \Delta\ (\mathbf{P}_{k-1,j}, \mathbf{G}_{k-1,j}). \qquad (2)$$

In the subsequent figures the icon is used for the 4 bit input, 2 bit output Δ-cell.

It is easy to prove that:

∴ Δ is associative, non commutative and idempotent.

∴ Any ($\mathbf{P}_{i,j}$, $\mathbf{G}_{i,j}$) requires (i-j-1) Δ-cells to be computed from the adders inputs. Intermediate results from the Δ-cells may be reused, thus reducing the total number of Δ-cells, but increasing the fan-out of some of them [Zim96].

3. COST AND DELAY MODELS : WORST CASE

The cost of an n-bit adder consists of a linear cost to compute the g_i and p_i from a_i and b_i and the s_i from p_i and $\mathbf{G}_{i-1,0}$ plus a cost varying according to the implementation chosen to get the $\mathbf{G}_{i-1,0}$. This cost is given roughly by the number of Δ-cells, that may range from (n-1) up to ($n \log_2 n$) for regular adders or even go up to 1/2 $(n-1)^2$ for special purpose adders [Zim96].

Note that the $\mathbf{P}_{i-1,0}$ are never used, so the Δ-cells at the bottom of the following figures produce only the $\mathbf{G}_{i-1,0}$ output and since the $\mathbf{P}_{i,j}$ are only useful for the right input of those cells, all the n-1 Δ cells at the bottom of the figures are simplified. This saving is accounted for in the fixed cost .

The adder delay is the sum of the delays of the Δ-cells along the critical path plus a fixed delay to get the g_i and p_i and finally the s_i. In the following, the delay of a Δ-cell is used as the delay unit.

3.1. Some Adder Architectures

Let us examine now some well-known architectures [GBB94], their delay (number of Δ-cells along the critical path), and their cost (the total number of Δ-cells).

3.1.1 Ripple Carry Adder

The ripple carry adder *(figure 1)* delay and its cost are in O(n-1). It is inefficient and easily constructed by mere abutment of Δ-cells.

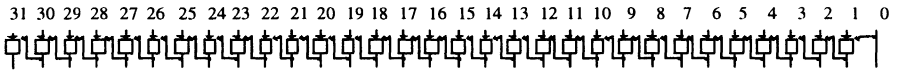

Figure 1 : A 32-bit carry ripple adder

3.1.2 Two Level Carry Select Adder (2-CSA)

The two level carry-select-adder *(figure 2)*, also named conditional-sum-adder or carry increment adder is based on the previous one truncated into blocks of varying sizes. Its cost is in O(2n) and delay $O\lceil\sqrt{2n}\rceil$, more precisely with k Δ-cells along the critical path, an adder can accommodate up to $1+\sum_{i=1}^{k} i = \frac{1}{2}k(k+1)$ bits.

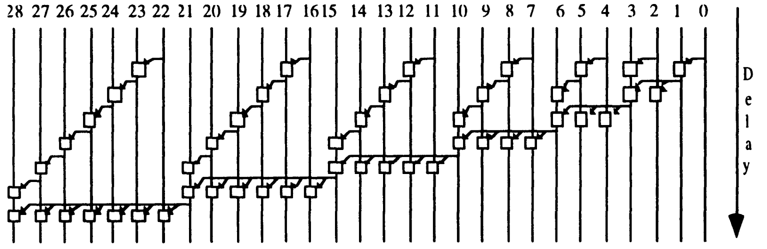

Figure 2 : A Two level carry select adder

3.1.3 Brent and Kung Adder

The Brent and Kung adder [BrKu82] is based on binary Δ-cell trees. The cost is O(2n), the delay $O(2\lceil \log_2(n)\rceil -2)$. One binary tree outputs the $\mathbf{G}_{i,0}$ for all i in the form 2^j-1, then another tree gives the remaining $\mathbf{G}_{i,0}$.

3.1.4 Sklansky Adder

The Sklansky adder [Skla60] has proved to be the fastest architecture. Its cost is $O\lceil \frac{n \log_2(n)}{2} \rceil$, and its delay $O\lceil \log_2(n)\rceil$. The main drawback is that the fan-out grows exponentially from the inputs to the outputs along the critical path and consequently the transistors must be sized.

3.1.5 Kogge & Stone and Han & Carlson Adders

The most significant bit of a Brent and Kung adder as well as in a Sklansky adder is obtained by a perfectly balanced binary tree in time $\log_2(n)$. If the tree for the most significant position is just copied for all other positions, the Kogge and Stone adder [KoSt73] is obtained. The fan-out is reduced to just two, at the expense of a larger number of Δ-cells, that becomes $O(n(\log_2(n) - 1) + 1)$ cells. As for the Sklansky adder, the delay is $O\lceil \log_2 n\rceil$.

In order to reduce the number of cells of the Kogge and Stone adder, Han and Carlson [HaCa87] have proposed to compute only the odd positions, and then to add a layer to compute the even positions from the odd ones. The delay is slightly increased to $O\lceil \log_2(n)\rceil +1$, while the complexity becomes $O\frac{n}{2}(\lceil \log_2(n)\rceil +1)$.

3.2. Comparison

Table 1 [TVG95]

Adder	*# of Δ-cells*	*Delay (Δ-cell)*	*Max. fan-out*	*Useful Activity*
Ripple	n -1	n -1	2	n /2
2-CSA	$\lceil 2n -\sqrt{2n}\rceil$	$\lceil\sqrt{2n}\rceil$	$\lceil\sqrt{2n}\rceil$	$\lceil 2n -\sqrt{2n}\rceil/2$
3-CSA	$5/2\ n -3\log_2(n/2)$	$\lceil\sqrt[3]{6n}\rceil$	$\lceil\sqrt[3]{6n}\rceil$	N.A.
B&K	$\lceil 2n - \log_2(n)\rceil$	$\lceil 2\log_2(n) -2\rceil$	$\lceil 2\log_2(n) -2\rceil$	N.A.
Sklansky	$\lceil n/2 \log_2(n)\rceil$	$\lceil\log_2(n)\rceil$	n /2	$\approx\lceil n/4\log_2(n)\rceil$
K&S	$\lceil n(\log_2(n)-1)+1\rceil$	$\lceil\log_2(n)\rceil$	2	$\approx\lceil n/2\log_2(n)\rceil$
H&C	$\lceil n/2\log_2(n)+1\rceil$	$\lceil\log_2(n)\rceil +1$	2	$\approx\lceil n/4\log_2(n)\rceil$

4. ACTIVITY MODEL FOR THE RCA

In this part of the paper, a model for the activity of a Ripple Carry Adder (RCA) is derived without taking into account the attenuation of the spurious transitions. In the ripple carry adder, when all the inputs are applied at once, the activity is mainly due to the propagation of the carry through a chain of $p_i = 1$. Let us call T(n,k) the number of different chains of k consecutive "1"s in a binary word of length n : It is obvious that : $T(n,0) = 0$ (no "zero bit" chain), $T(n,n) = 1$ (i.e. 111...11) and $T(n,n-1) = 2$ (i.e. 2 possibilities : 011...111 or 11...1110).

Let us now compute the general term T(n,k) for $0 < k < n$. Since the word extremities as well as the bit value 0 act as chain separators, we distinguish two cases. When the chain touches one of the two extremities of the n-bit word, there are $2^{n-(k+1)}$ different values for the n - (k + 1) bits outside the chain.

$$\underbrace{11...10}_{k+1}\ \underbrace{011...01}_{n-(k+1)}$$

There are n - (k+2) possibilities for the chain to be in the middle of the word and for each position there are $2^{n-(k+2)}$ values of the n - (k + 2) remaining bits.

$$\underbrace{01...10}_{k+2}\ \underbrace{011...01}_{n-(k+2)}$$

Thus $T(n,k) = 2^{n-k}\left(1+\frac{n-k-1}{4}\right)$ for $0 < k < n$, $T(n,0) = 0$ and $T(n,n) = 1$

4.1. Activity of the RCA

In the case of the RCA, none of the outputs is obtained from a balanced binary tree, and thus, the activity window of any output is equal to its logical depth. This is not true for other architectures where the outputs are obtained by balanced binary trees like the Kogge and Stone.

The activity caused by a carry propagation over k positions is proportional to $k^2/2$ [MoPa96]. Thus the average activity is

$$A(n) = \frac{1}{2^n}\sum_{k=0}^{n} \frac{k^2}{2} \cdot T(n,k) \ . \tag{3}$$

Let us recall some useful identities [Kre93] :

$$\sum_{i=0}^{n} i \cdot 2^{-i} = 2 - \frac{n}{2^n},\ \sum_{i=0}^{n} i^2 \cdot 2^{-i} = 6 - \frac{n^2}{2^n},\ \sum_{i=0}^{n} i^3 \cdot 2^{-i} = 26 - \frac{n^3}{2^n} \tag{4}$$

since they allow to simplify the expression of

$$A = \frac{3n-4}{4} - \frac{3n^2}{2^{n+3}} \xrightarrow[n\to\infty]{} \frac{3n-4}{4} \ . \tag{5}$$

With these identities, one can also easily verify that :

$$\frac{1}{2^n}\sum_{k=0}^{n} k.T(n,k) = \frac{n}{2} - \frac{3n}{2^{n+2}} \xrightarrow[n\to\infty]{} \frac{n}{2} \qquad (6)$$

which is the known average delay of the ripple carry adder.
In the following, the higher order terms are neglected, i.e. it is assumed that : $A = \frac{3n-4}{4}$. Table 2 shows the relative error for 8, 16, 32 and 64 bits.

Table 2

# of bits	*Activity (%)*	*Delay (%)*	*η (%)*
8	9.3750	2.34	8.2759
16	0.15	0.02	0.04
32	8.94e-06	5.59e-07	4.26e-06
64	8.33e-15	2.60e-16	2.58e-06

Due to the equiprobability of the output vectors, the average number of useful transitions in a RCA is equal to half the number of cells. The total activity A is split in two parts : $A = A_{useful} + A_{redundant}$. Thus, the ratio η of redundant over total activity is:

$$\eta = \frac{A_{redundant}}{A} = \frac{A - A_{useful}}{A} = \frac{n-4}{3n-4} \qquad (7)$$

For large values of n, $\eta \approx 1/3$. This result is consistent with the BDD simulations using a unit delay and with [LMJ95].
By adopting this approach, it is also possible to determine the acticity at a given time t_i *(Figure 3)*.

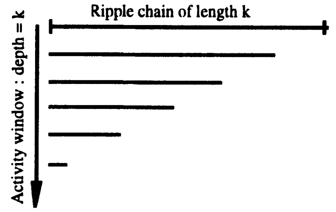

Figure 3: Activity at time t_i.

The activity at t_1 is given by : $A(t_1) = \frac{1}{2^n}.\sum_{k=1}^{n} k.T(n,k)$. The ripple chains of length k that exist at t_2 are those of length k+1 at t_1, thus : $A(t_2) = \frac{1}{2^n}.\sum_{k=1}^{n-1} k.T(n,k+1)$. The sum goes to n-1 because in a word of length n, one cannot have a ripple carry chain of length greater than n.
More generally, the activity at time t_i is given by :

$$A(t_i) = \frac{1}{2^n}.\sum_{k=1}^{n-i} k.T(n,k+i) = 2^{-i}\left(\frac{n-i}{2}+2\right) \qquad (8)$$

5. EXTENSION OF THE MODEL TO OTHER ARCHITECTURES

The previous model is extended to the adders that can be obtained by an association of ripple carry chains - like the carry select adder or the Kogge and Stone adder for example.

In the computation of η, the useful activity A_{useful} is assumed to be half the number of Δ cells since "0" and "1" are equiprobable (this is consistant with the BDD simulations).

5.1. Two Level Carry Select Adder

The 2-CSA adder is a RCA truncated into blocks. For n bits, the length of these blocks varies from 1 to $\sqrt{2n}$ - 1. Thus a n bits 2-CSA can be viewed as $\sqrt{2n}$ RCAs of length varying from 1 to $\sqrt{2n}$ - 1 (first level) plus a row of cells that form the second level *(figure 2).*

5.1.1 First level

The total activity of the first level of the 2-CSA is given by the sum of the activities of the ripple carry chains, as they are independent from each other.

$$A_{1^{st}\text{ level}} = \frac{1}{2^n}\cdot\sum_i^{\sqrt{2n}}\left[i.A(i)\right] = \frac{1}{2^n}\cdot\sum_i^{\sqrt{2n}}\left[i.\frac{3i-4}{4}\right] = \frac{n.\sqrt{2n}}{2} - \frac{3.\sqrt{2n}}{8} - \frac{3.n}{4} \quad (9)$$

5.1.2 Second Level

The second level of the 2-CSA is approximated here by a ripple carry chain of length $\sqrt{2n}$, in which, a cell at position i is duplicated i times. This approach neglects the acticity generated at the second level by the ripple of the outputs at the first level. The activity of such a ripple carry chain can be deduced from the activity of the RCA by assuming that the capacitance of the k^{th} cell is k instead of 1.

$$A_{2^{nd}\text{ level}} = \frac{1}{2^n}\cdot\sum_{k=1}^{\sqrt{2n}}\left[k.\frac{k^2}{2}.T(n,k)\right] = \frac{6.\sqrt{2n}-57}{8} \quad (10)$$

5.1.3 Total Activity of the 2-CSA

The total activity of the 2-CSA is the sum of the activities of the first and second levels :

$$A_{Total} = A_{1^{st}\text{ level}} + A_{2^{nd}\text{ level}} = \frac{3.\sqrt{2n}+4.n.\sqrt{2n}-6.n-57}{8} \quad (11)$$

Assuming that the useful activity is given by half the number of cells, the redundant to total activity ratio can be computed:

$$\eta_{2CSA} = \frac{A - A_{useful}}{A} = \frac{\left(7.\sqrt{2n} + 4.n.\sqrt{2n} - 14.n - 57\right)}{3.\sqrt{2n} + 4.n.\sqrt{2n} - 6n - 57} \qquad (12)$$

5.2. Kogge and Stone adder

Each bit of the Kogge and Stone adder is obtained by a balanced binary tree, thus the output of any cell can change only once during a clock cycle - no redundant transitions.

The carry propagation is the result of a logical AND, thus its activity decreases very rapidly with the depth (like 2^{-i}), but the transition probability of the carry generation is almost constant (1/2). These considerations allow us to approximate the activity of the Kogge and Stone adder by half the number of its cells :

$$A_{K\&S} = \frac{n}{4}\log_2 n \text{ , and } \eta_{K\&S} = 0 \qquad (13)$$

6. GLITCH THRESHOLD

HSPICE simulations has been carried out on simple circuit examples *(Figure 4)* in order to quantify the spurious transitions absorption or propagation. The notion of **Glitch Threshold** is introduced here in order to quantify when a glitch becomes a spurious transition

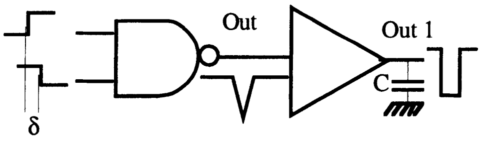

Figure 4: Spurious transition generation and propagation

When the transitions at the inputs of a gate are separated by a delay δ, a glitch is generated at "Out" *(figure 4)*. The amplitude of this glitch is proportional to δ *(figure 5)*. This glitch can be either absorbed, or propagated depending on its width and on the delay of the following gate. As it can be seen in figure 5, there is a threshold for the glitch propagation from "Out" to "Out1".

HSPICE measurements have been carried out (ATMEL-ES2 ECPD07 technology) and ploted. The plot shows that there is a threshold delay under which the glitch is absorbed, and above which the glitch grows into a spurious transition *(figure 5)*. We call this threshold delay G_{th}.

The variation of G_{th} with respect to τ (the buffers' delay) is linear *(figure 6)*, and the slope is approximately 1.89. This means that a glitch of width $\delta = G_{th}$ will become a spurious transition only if $\delta \geq 1.89\ \tau$. The glitch threshold depends on the loading capacitance of OUT1. The threshold phenomenon is attenuated when the capacitance is large.

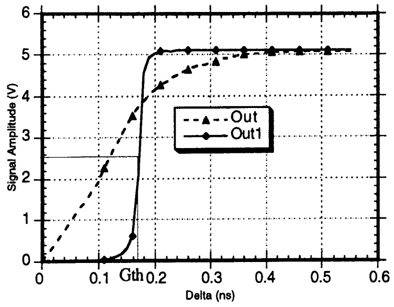

Figure 5 : Glitch Threshold

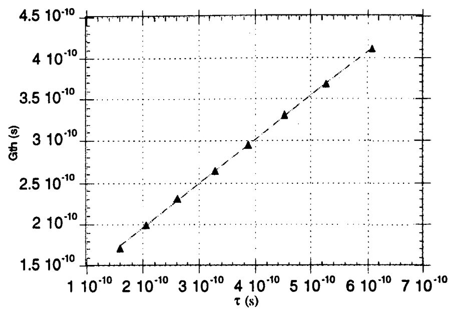

Figure 6 : Variation of G_{th} with τ.

This characteristically behaviour of spurious transitions has been implemented in a BDD simulation tool [Cra89], and in the following section, the above analytical model is compared to the simulations for different adders architectures.

7. BDD SIMULATION AND RESULTS

The tool used for implementation and experiment is ASYL+ [Cra89]. It provides a complete environment for macrogeneration and low-level synthesis. The size of the operands is sufficient to build a netlist for any kind of adder architecture. The mapping and the estimates are then performed with the user library, delay and dissipation model.

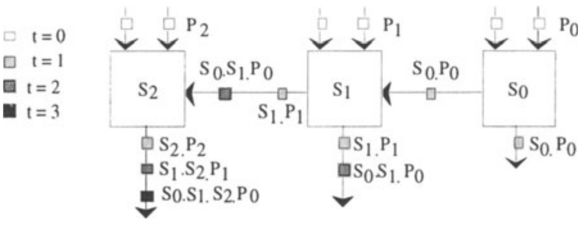

Figure 7 : Switching probabilities example

The power is dependent on the circuit structure as well as the circuit inputs: it is said to be input pattern-dependent. To solve this problem, one can simulate the circuit for a large number of inputs and then average the switching activity. On the other hand, probabilities where introduced [Bur88] to perform the averaging before running the analysis [Najm95] by estimating the number of transitions per clock cycle. Using the Boolean network functionality and connectivity, these input probabilities are propagated through the network. To apply statistical properties, reconvergent fanouts and feedback have to be taken into account. A convenient way to do this is to use binary decision diagrams [Najm91]. As the adder architectures do not contain any reconvergent fanouts, the probability computation is performed without any approximation.

A glitch is created at the output of a gate because of the difference in arrival times at its inputs. Then, the glitch can be propagated to the fanout gates according to their sensitivity. The probability of a switch due to a glitch cannot be estimated the same way as a useful switch for the simple reason that the probability of a node to undergo a transition does not depend only on the Boolean network functionality but also on its structure (path lengths for instance).

The formula $P_T = 2P_1(1-P_1)$ in no longer valid any more for all the possible transitions. To solve this problem, the probability calculation must be based on real delay models. Since the switching probabilities are supposed to be known at the primary inputs of the circuits, they are propagated to the fanout gates up to the roots of the circuit using the gate delays. Each gate modifies the switching probability according to its ability to propagate the transition from its inputs to its output, what we called sensitivity. The sensitivity calculation rests on the functionality of the gate according to the probabilities at the inputs. Finally, gates have a set of switching probabilities, distant from each other according to the glitch threshold previously introduced, which are added. A simple example of carry ripple adder is illustrated in figure 7.

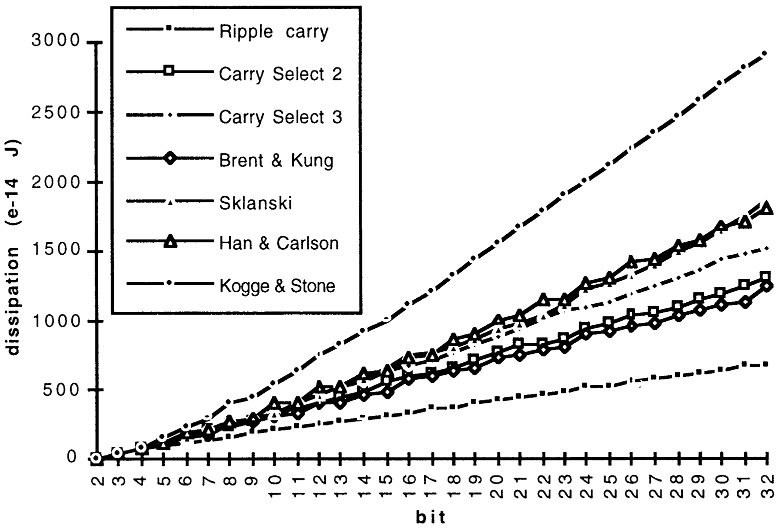

Figure 8 : HSPICE/BDD Power dissipation vs. number of bits

A switch is represented by a square, coloured according to its occurrance time. Its probability is a function of the input switching probabilities P_i and the sensitivities S_i of the cells yet encountered. The authors propose a gate level estimator based on these remarks in [Lau96]. It gives close dissipation estimation to an exhaustive simulation in classical combinational circuits.

The automatic simulation is consistent with the analytical model previously exposed which rests on unit delay and capacitance, and with a power dissipation function linear with respect to the fanout capacitances. However, technology mapped adders have realistic delays as well as a more complex dissipation model at each switching, including for instance the charging of internal capacitances. As a consequence, we built, at transistor level, and simulated a Δ-cell with HSPICE. The submicron technology used is ATMEL ES2 ECPD07. Once the elementary cell is fully characterised, the synthesis tool estimates the total power dissipation of classical adder architectures. These estimate are presented in figure 8.

8. CONCLUSION

In this paper the most frequent adder architectures were compared from the activity, delay and cost points of view. The originality of the approach is that the estimation of the activity was achieved analytically by *implicitly exhaustive enumeration* of all vectors. This is possible thanks to the properties of the Δ operator. Redundant transitions were taken into account, and a redundant to total power ratio was derived. Finally, glitch filtering is taken into account by feeding the BDD tool with technology driven HSPICE simulation results.

In this article it is assumed that all the inputs are ready at the same time, and that all the outputs are desired at the same time. This is not always the case, especially if an adder is associated with other operators that have their own delays. For example in multipliers or dividers, the inputs arrival times are accessible to simulation, thus different adder architectures adapted to these conditions should be examined in order to match the best power-delay-cost trade off.

9. BIBLIOGRAPHY

[BCS92] R. Brodersen, A. Chandrakasan and S. Sheng, "Low-Power Signal Processing Systems", *VLSI Signal Processing* Eds. Yao, Jain, Przytula and Rabaey, IEEE Press, 1992, pp.3-13.

[BrKu82] R. P. Brent, H. T. Kung, "A Regular Layout for Parallel Adders", *IEEE Transactions on Computers*, Vol. C31, pp 261 - 264, March 1982.

[Bur88] R.Burch, "Pattern-Independent Current Estimation for Reliability Analysis of CMOS Circuits", *In proc. of the Design Automation Conference*, pp. 294-299, 1988.

[Cra89] M.Crastes and al."ASYL : a Logic and Architecture DA System", *Euro ASIC89*, Paris, France, Jan. 1989, pp. 183-209.

[DHNT95] A.C. Deng, X. Huang, S. Napper, J. Tuan, J. Benkoski, "Simulation Algorithms, Power Estimation and Diagnostics in PowerMill", *proc. of Power and Timing Modelling Optimization and Simulation (Patmos)*, Kaiserslautern - Germany, 1995.

[GBB94] A. Guyot, M. Belrhiti, G. Bosco, "Adders Synthesis", *Logic and Architecture Synthesis*, G. Saucier and A. Mignotte, ed. Chapman & Hall, 1994.

[HaCa87] T. Han, D. A. Carlson, "Fast Area-Efficient VLSI Adders", *proc of 8th Symposium on Computer Arithmetic*, pp. 49-56, May 1987

[Kre93] E. KREYSZIG, "Advanced Engineering Mathematics", WILEY, Inc., New York.

[KoSt73] P. M. Kogge and H. S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations" *IEEE Trans. Comput.*, vol. 22, no. 8, pp. 783-791, Aug. 1973.

[Lau96] B. Laurent, "A Low Power Mapping Optimizing Global Power", in Proc. Synthesis and System Integration of Mixed Technologies (SASIMI), pp. 40-46, 25-26 November, 1996, Fukuoka, Japan.

[LMJ95] J. Leijteen, J. van Meerbergen, J. Jess, "Analysis and Reduction of Glitches in Synchronous Networks", *in proc of EDAC*, 1995.

[MoPa96] L. Montalvo and K.K. Parhi, "Estimation of Average Energy Consumption of Ripple-Carry Adder Based on Average Length Carry Chains" *in proc. 11th Design of Integrated Circuits and Systems Conference (DCIS'96)*, Barcelona, Spain, November 1996.

[Najm91] F.Najm, "Transition Density, a Stochastic measure of Activity in Digital Circuits", *In proc. of the Design Automation Conference*, pp. 644-649, june 1991.

[Najm95] F.Najm, "Feedback, Correlation and Delay Concerns in the Power Estimation of VLSI Circuits", *In proc. of the Design Automation Conference*, pp. 612-616, 1995.

[Skla60] J. Sklansky, "Conditional Sum Addition Logic", *IRE Transaction EC-9(2)*, June 1960, pp 226-231.

[TVG95] V.Tchoumatchenko, T. Vassileva and A. Guyot, "Timing Modelling for Adders Optimisation", *In proc. of IPATMOS 95*, Germany, 1995.

[Zim96] R. Zimmermann, "Non-Heuristic Optimization and Synthesis of Parallel-Prefix Adders", *In proc. of IFIP workshop*, France, 1996.

10. BIOGRAPHY

Selim J. Abou-Samra received his Honours degree in Physics from the University of Liverpool, United-Kingdom and his MS degree in Microelectronics from the Université Joseph Fourier in Grenoble, France. He is currently preparing a Ph.D. thesis on low power arithmetic operators under the direction of Prof. Alain Guyot in the TIMA Laboratory. For more information see http://tima-cmp.imag.fr/Homepages/selim

Alain Guyot received his MS, Ph.D. and Habilitation from INPG (Institut National Polytechnique) in Grenoble, France. He his currenltly an associate professor with the same university where he teaches computer architecture, computer arithmetic and VLSI design. He is responsible of the "Integrated System Design" research group in the TIMA Laboratory and responsible of the "Integrated System Design" option of the master in Microelectronics. Prof. Guyot is the author or co-author of over 100 papers in conference proceedings or journals.

Bernard Laurent received engineering degree and MS in computer science and microelectronics from INPG in Grenoble, France. He is currently preparing a Ph.D. in logic synthesis and macrogeneration in the CSI laboratory (INPG) under the direction of Prof. Gabriêle Saucier.