# A Time Driven Adder Generator Architecture

*M. Aberbour, A. Houelle, H. Mehrez , N. Vaucher, A. Guyot**

*Laboratoire MASI/CAO-VLSI*
*Université Pierre et Marie Curie (PARIS VI)*
*4, Place Jussieu, 75252 Paris Cedex 05 France*
**Integrated Systems design Group, TIMA/INPG,*
*46, Av. Félix Viallet, 38031 Grenoble Cedex, France*
*e-mail: mourad.ABERBOUR@masi.ibp.fr*

**Abstract**

This paper presents the design and implementation of a time driven adder generator architecture. There exists a large variety of adders designed to satisfy different computation requirements, in particular we list the Carry Look Ahead (CLA) adder, the skip adder, the ripple adder, the carry select adder (CSA), etc. These different architectures will offer different delays and it is up to the user to chose among them. The design we present here allows the parametrization of the architecture to fit ones design constraints. From the word length and the wanted delay the generator outputs a suitable architecture.

**Keywords**

Addition VLSI architectures, generators, macro blocks, variable architecture.

## INTRODUCTION

The exists a so large set of different adders generators (Sklansky, 1990), (Bedrij, 1962), (Brent, 1982), (Cavanagh, 1984), (Hwang, 1979), (Muller, 1989), each one implementing a particular architecture, that the choice of an adder may be tough. We present here an alternative which will replace all the others. We impose the time delay criteria to the generator and this one will output the right adder (the architecture of the adder is thus variable).

VLSI: Integrated Systems on Silicon R. Reis & L. Claesen (Eds.)

Moreover, the need for such a generator is justified by the optimization of electrical power consumption and area. In fact, we find addition units in almost all complex circuits. For example, in a general purpose processor, certain adders are allocated for address computation, others are integrated in floating point units and most of the time linked to multipliers. The applications are various and the constraints (computation time, area, power consumption) vary from application to another. However, we find almost the same type of adders (the fastest) in such designs, even though this is not always necessary. For example, in the address computations, we need one clock cycle to carry out the operation, we can thus relax the computation delay requirement and use a slower adder. This will allow a certain gain in power consumption and Silicon area. The generators we designed allow us to fit exactly our performance requirements with the best optimizations possible.

From the word lengths of the two operands and the computation time the generator outputs an adder in four different views (structural, behavioral, physical and placement) it also outputs a certain number of functional patterns. This is illustrated in figure 1.

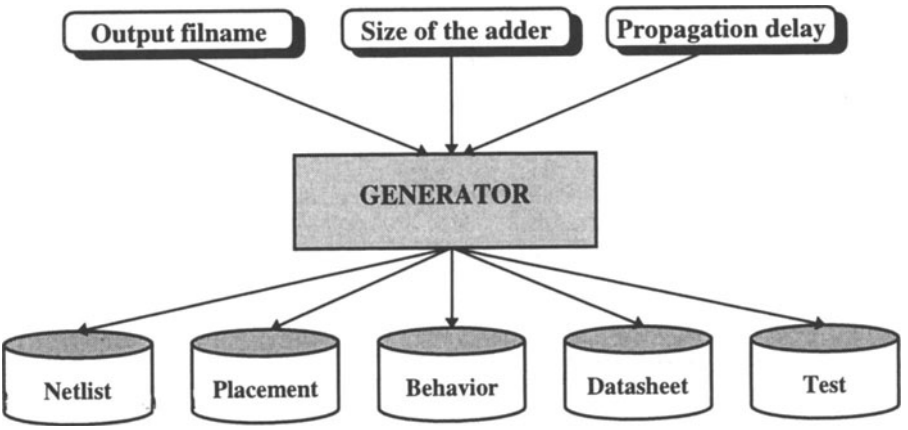


**Figure 1** Generator output

The generator is designed following a methodology developed at the MASI laboratory (Aberbour, 1995), (Houelle, 1994), and directly inherited from the silicon compiler approach (Johansen, 1979).

This paper is composed of three parts. First of all, we review the different types of addition architectures (ripple, CLA, skip adder, ...). Then we select an architecture suitable for the desired delay. Finally we sum up with different VLSI results for a 32 bits adder.

## PRINCIPLE OF THE VARIABLE ARCHITECTURE ADDER

All known adders are constituted of a tree of cells computing the generation (G) and propagation (P) signals in order to determine the values of the intermediate carries. The variable architecture only affects the way this tree is constructed.

The architecture of the tree can in fact be in several forms, more or less parallel. To illustrate this we propose an example of an 8 bits adder in three configurations, as depicted in figures 2(a), 2(b), 2(c).

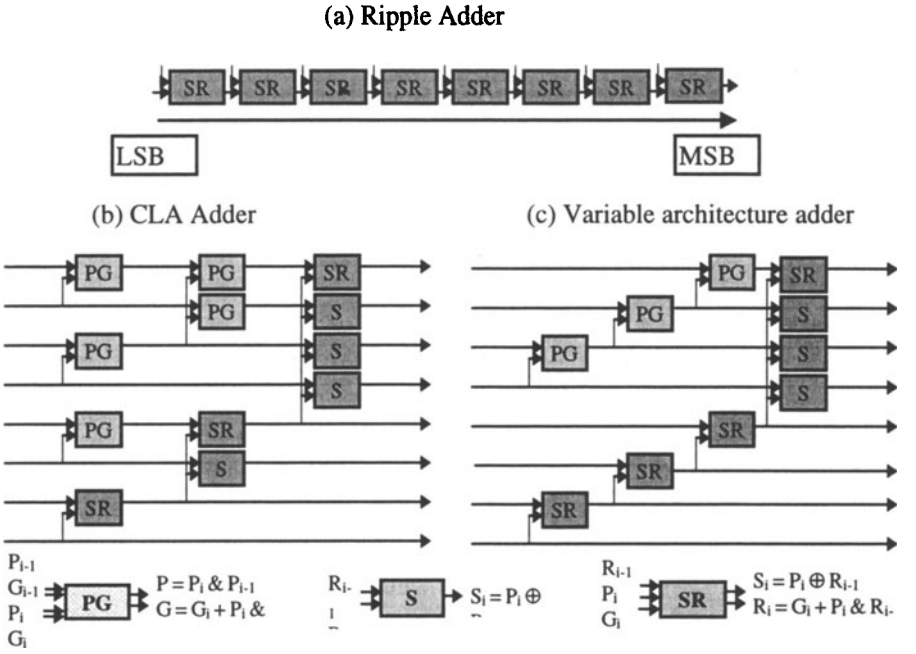


**Figure 2** Alternative adder architectures

The first configuration, figure 2(a), computes the propagation and generation values in a serial fashion and represents a full sequential adder.
This adder contains eight cells and its computation delay in the one of eight combinational stages.

The carry anticipation adder, figure 2(b), uses a binary tree to compute the P and G signals. Its delay evolves logarithmically (Sklansky, 1990), three stages in our case.

The last configuration, figure 2(c), represents an intermediate adder between the full serial adder and the carry anticipation adder. It is built up of ten logic cells and presents a delay of four stages.

The variable architecture adder generator is thus capable of generating a tree containing at the same time a parallel section and a serial section.
For an N bits adder the generator outputs an operator with a number of stages varying from log(N) to N.

## THE MAIN ADDERS ARCHITECTURES

We start the discussion with the ripple adder. It is the slowest ( delay in O(N)), but it occupies the smallest area (in O(N)). It is used where a very small area and power consumption in needed.

Opposite to the ripple adder, is the Carry Look Ahead adder. It has a computation time of the order of $O(\log_2,(N))$, the area is in $O(N*\log_2,(N))$, the fact which makes this adder largest in terms of size. It is readily built in a recursive fashion and this makes it suitable for an implementation as a generator
There exists a large set of architectures with intermediate characteristics. A skip adder architecture offers still better performances.

We find also in the literature the adder with carry selection (Bedrij, 1962). It is broken down into several blocks. Each block carries out two additions in parallel, one anticipating a null carry in and another a 1. The result is then determined depending on the true value of the carry in. The delay is in $O\left(\sqrt{2N}\right)$ and the area is also in $O\left(N - \sqrt{2N}\right)$

## THE CHOSEN ARCHITECTURE

The addition architecture used is introduced by (Slansky, 1990). This operator allows the simple computation of the carry propagation and generation functions $P^i_j$ , $G^i_j$, starting from position i up to position j. The properties of this operator are listed below

- Associativity

$$PG^i_j = ( PG^k_j \Delta PG^l_{k-1} ) \Delta PG^i_{l-1} = PG^k_j \Delta ( PG^l_{k-1} \Delta PG^i_{l-1}) \text{ for } j \geq l \geq k-1 \geq i \quad (1)$$

- Idempotence

$$PG^{i}_{j} = PG^{k}_{j} \Delta PG^{i}_{l} for\ j \geq l \geq k\text{-}1 \geq i \quad (2)$$

- Non Commutativity

$$PG^{i}_{j} \neq PG^{i}_{k\text{-}1} \Delta PG^{k}_{j} for\ j \geq k \geq i \quad (3)$$

The most important property is the way the intermediate propagation and generation functions are computed

$$PG^{i}_{j} = PG^{k}_{j} \Delta, PG^{i}_{k-1} \quad (4)$$

which corresponds to

$$\left(P^{i}_{j}, G^{i}_{j}\right) = \left(P^{k}_{j} \bullet P^{i}_{k-1}, G^{k}_{j} + P^{k}_{j} \bullet G^{i}_{k-1}\right) \text{for } j \geq k \geq i \geq 0 \quad (5)$$

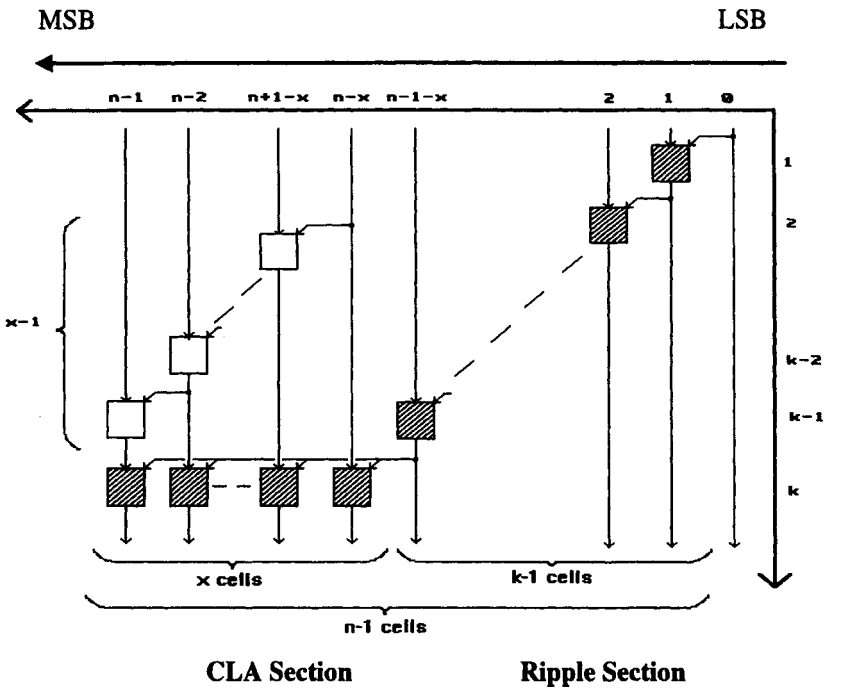


**Figure 3** The configurable adder architecture

The chosen architecture must be modulable depending on the imposed computation time. This means that the length of the critical path must vary from a configuration to another. Moreover, since the generator must be able to generate adders with a propagation delay comprised between a CLA delay time and a ripple delay time, the implemented architecture is hybrid, in between the CLA (the fastest) and RIPPLE (the lowest) architectures. Now suppose that the delay constraint forces us to build an adder in which the critical path is constituted of k $\Delta$ cells; the adder will then be as shown in figure 3.
We notice three distinct parts : Let n be the number width of the adder and k the number of stages (k represents also the number of cells of the Ripple part of the adder).
The first group of x cells in parallel is placed between the position (n-x,k) and (n-1,k); x will be determined later on. The inputs to these cells are

$$PG_i^{n-x}(\forall i, n-1 \geq i \geq n-x) \text{ and } G_{n-x-1}^{0} \tag{6}$$

The first values $PG_i^{n-x}$ are generated by the group of $\Delta$ cells situated at the top of the previous group, precisely from the point $(n+1-x,k-x+1)$ to the point $(n-1,k-1)$.
It is then sufficient to specify the value of x and we get a basis to build the adder. We have seen that cells are placed from the point (1,1) to the point $(n-x-1,k-1)$ included. Since these cells are cascaded, i.e. they are on a diagonal then

$k-1=n-x-1$ which yields $x=n-k$ (7)

Unfortunately, there exists a limiting case which restricts the application domain of the algorithm. Since the cells of the second group start from position (n+1-x,k-x+1), where k-x+1 is the reference number of the stage, and the highest stage number is 1. Then we conclude that

$k-x+1 \geq 1$ with $x=n-k$ we get $k \geq n/2$ or $n \leq 2k$ (8)

However, it can happen that this inequality be violated. In this case we apply the same process as to build a CLA adder. This means that we instanciate n/2 cells from (n/2+1,k) to (n-1,k), then we elaborate two adders of n/2 bits starting from the stage referenced by the number k-1.

## RESULTS

Three different VLSI comparisons have been carried out on a 32 bits adder and this for each configuration, meaning for every value of k varying from 5 to 31. For these tests we used the cells library ECPD07 of the ATMEL-ES2 company.
First of all, we focused our comparisons on the routed circuit area. The automatic placement and routing have been done with the CADENCE tools.

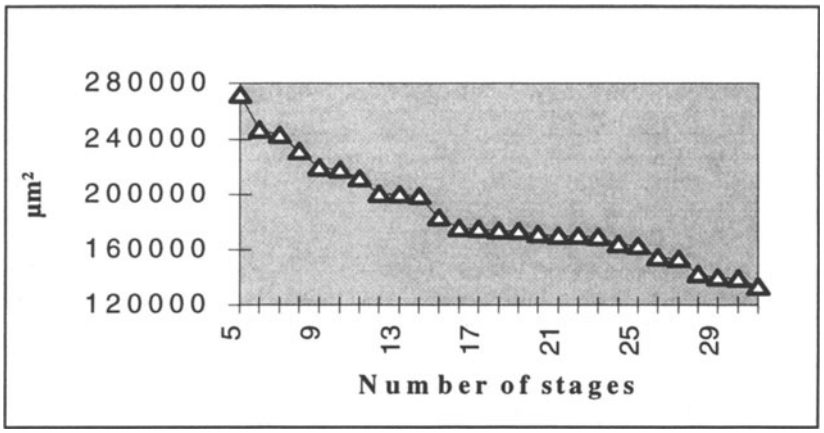


**Figure 4** Area of the routed circuit

As shown in figure 4, and at a first glance, the curve seems not meaningful. However, we can distinguish two intervals. The first, for a number of stages less than 16 (=n/2), the area gain is very interesting, the curve is sharp. Elsewhere, the curve is flat and doesn't constitute an advantageous zone to find the best area-delay compromise for the adder. The curve indicates that the area decays exponentially when the architecture tends to become fully serial.

Now lets focus on the propagation time results for the used technology, illustrated in figure 5.

The curve is quasi-ideal because the delay grows linearly with respect to the number of stages. This proves that the delay grows as expected with respect to the number of stages. The messured delays for a 32 bits adder varies from 9 to 30 nano seconds.

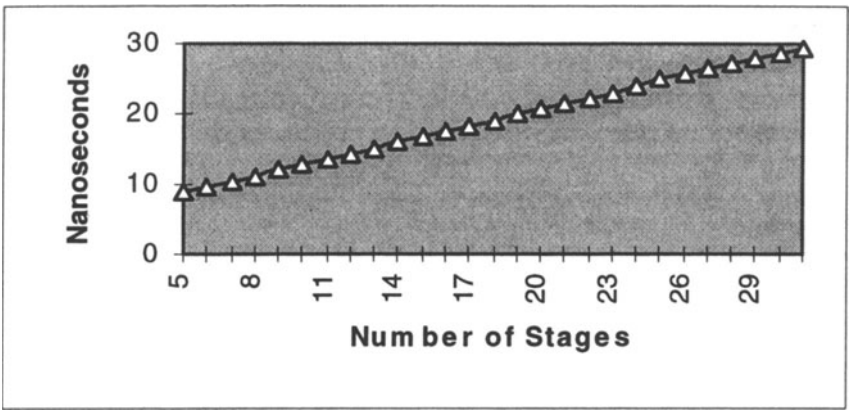


**Figure 5** Propagation delay comparisons

Using the power consumption values of each cell, provided by ES2, we establish the maximal adder's power consumption which corresponds to the case where all input cells toggle at the same time, which is in fact almost impossible. This is illustrated in figure 6.

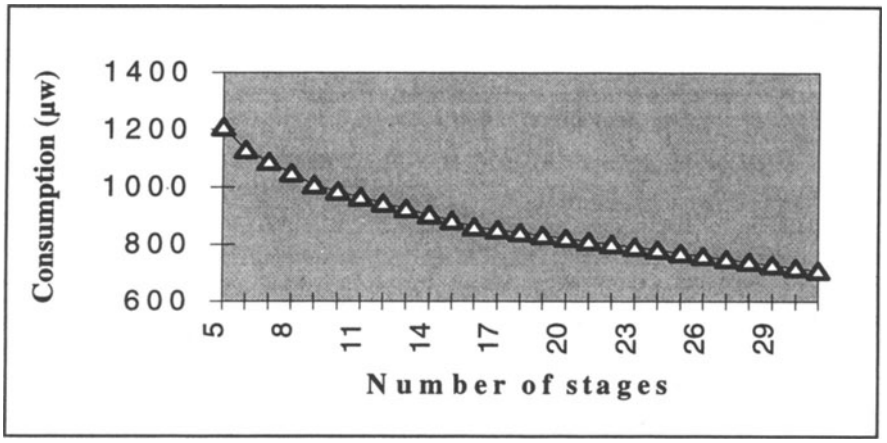


**Figure 6** Electrical power consumption

The obtained curve is smooth and grows exponentially. Once more we can extract two distinct intervals:

A sharp and fast growth part for a number of stages less than 16 (=n/2).

And an almost straight line, representing a not really interesting power consumption-delay compromise.

## CONCLUSION

The main goal achieved in this work is the replacement of all possible adders generators by a generator with a parametrized time driven addition architecture. This is not possible only if we impose the addition computation time to the generator. In fact, the generator provides very good results since we can adjust very precisely the computation time, by using different numbers of intermediate combinatorial stages.

The study of the curves representing the performances of the adders comes up with a conclusion that the compromise is optimal for a number of stages less than n/2, where n is the precision of the adder. In fact, the area and power consumption decrease very rapidly in this interval, whereas the delay grows slowly. This means that if we can tolerate increasing the computation time of the addition by about 10%, this will be equivalent to increasing the number of stages by a few units, then we can achieve a power consumption and area gain of about 15%. Outside this interval ($\geq n/2$), the area and power decrease slowly, and this presents a negligible profit.

## REFERENCES

Aberbour, M., Gounoud, S., Houelle, A., Mehrez, H., Vaucher, N, (1995) A Fully Parametrized IEEE Floating Point Operators Library For Use In Digital Signal Processing. *Proc. ICSPAT 95*, Boston MA USA.

Bedrij, O. J., (1962) Carry Select Adders, *IRE Transactions*, EC-11 No.3, pp340-346.

Brent, R., Kung, H. T., (1982) A regular Layout for Parallel Adders. *IEEE Transactions on Computers*, vol C-31.

Brent, R., (1970) On the Addition of Binary Numbers, *IEEE Transactions on Computers*, vol C-19, pp 758-759.

Cavanagh, J.J.F., (1984) Digital Computer Arithmetic, Design and Implementation, *McGraw-Hill*, computer science series.

Houelle, A., Mehrez, H., Vaucher, N., (1993) Méhodologie de conception de générateurs portables de macro-blocs fonctionnels optimisés en surface et en performance, *MASI report.*

Houelle, A., Mehrez, H., Vaucher, N. (1994) On portable Macro-cell generators using the fully 754-IEEE standard , *proc. ICSPAT 94*, Dallas Texas USA, Oct. 18-21, V2 pp1749-1754.

Hwang, K., (1979) Computer Arithmetic Principles, Architecture and Design, *Wiley*, New York.

Johansen, D., (1979) Bristle blocks: a silicon compiler, *proc* $16^{th}$ *ACM IEEE DAC*, San Diego.

Muller, J-M., (1989) Arithmetique des ordinateurs, operations et fonctions elementaires , *Masson.*

Sklansky, J., (1990) Conditional Sum Addition Logic, *Computer Arithmetic*, vol. 1, pp57-62.

## BIOGRAPHY

Mourad ABERBOUR is a Ph.D. student at the LIP6 laboratory in the CAD-VLSI team of the Pierre and Marie Curie University of Paris. His research interests concern the mapping of Computer Vision algorithms onto VLSI architectures and neural network architectures and their hardware implementations.

Alain HOUELLE is a computer science researcher at the LIP6 Laboratory of PARIS VI. He obtained the PhD. degree from the Pierre and Marie Curie University in 1997. His research work deals with the development of GenOptim: an environment for the design of portable VLSI blocks optimised in both performance and area.

Habib MEHREZ is currently a computer science researcher and teacher at the Paris VI University. He obtained the "Doctorat d'Etat" degree from the Same university in 1991. His research concerned mainly the elaboration of a unified approach of FFT algorithms and their efficient VLSI implementation. Dr. MEHREZ is leading the computer arithmetic and pattern recognition VLSI architectures groups within the CAD&VLSI team of the LIP6 Lab.

Nicolas VAUCHER is a computer science researcher at the LIP6 Laboratory of PARIS VI. He obtained PhD. degree from the Pierre and Marie Curie University in 1997. He is mainly interested in the development of design methodologies for the VLSI generic architectures targeted to signal processing.

Alain Guyot received his MS, Ph.D. and Habilitation from INPG (Institut National Polytechnique) in Grenoble, France. He his currently an associate professor within 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. Prof. Guyot is the author or co-author of over 100 papers in conference proceedings or journals.