JANUS, an On-line Multiplier/divider for manipulating large numbers.

Alain GUYOT[1], Yvan HERREROS[1] and Jean-Michel MULLER[1,2].

[1]Laboratoire TIM3-IMAG, INPG, 46 Avenue Félix Viallet, 38031 Grenoble Cedex, FRANCE.

[2]CNRS, Laboratoire LIP-IMAG, Ecole Normale Supérieure de Lyon, 46 Allée d'Italie, 69364 Lyon Cedex 07.

Abstract.

This paper deals with the detailed VLSI implementation of a fast bit-serial operator designed to perform very high precision (600 decimal digits) additions, multiplications and divisions, and some of the applications of the circuit.

Introduction.

Our aim is to present a VLSI implementation of an on-line multiplier/divider unit able to manipulate quite large numbers (up to 600 decimal digits). On line arithmetic, presented by Ercegovac and Trivedi in 1977 [6] is a digit serial arithmetic where the digits circulate from the most significant to the least significant. Digit serial arithmetic allows digit-level pipelining (which enable fast computation) and circulation of data most significant digit first is necessary in order to perform some computations like division or maximum of two numbers.

Since in classical number systems the carries propagate from the least significant position to the most significant one, on-line arithmetic needs the use of carry-free redundant number systems, like Avizienis's signed-digits systems [1] or carry-save notation. Frequently, the radix chosen is different from 2 since a carry-free addition algorithm, due to Avizienis [1], may be used in radix $r \neq 2$. For instance, the paste-up system , presented by Irwin and Owens in [15] uses radix 4. In radix 2, carry-free addition is possible, but with two inconvenients : the algorithm seems more complicated, and the delay is larger (see [2] for proof). We shall show here that the first inconvenience vanishes if we choose a good binary representation of the digits in radix-2 signed digit notation, and we shall chose radix 2 for our circuit.

The multiplier uses a three-input parallel adder, a three-input serial adder and a one-digit multiplier which are described first. The divider uses the multiplier to compute the least significant digits of the partial remainder, plus four slices to compute its four most significant digits in a nonredundant form. For clarity reasons, the multiplier is drawn with three digits only, the generalization to any number of digits is straightforward. Most significant digits are at the left. Multiplication and Division are performed following Ercegovac and Trivedi's algorithm [6].

A. Number representation.

Notation. To avoid the carry propagation delay in addition, a signed binary digit (SBD) notation has been adopted. Each SBD x , which is $\overline{1}$, 0 or 1 is represented by a couple (a,b) of negative and positive bits referred to as (-/+) such that x = a-b. So $\overline{1}$ is (1,0), +1 is (0,1) and 0 is either (0,0) or (1,1). To change the sign of a SBD, one can either permutate its two bits or logically complement them. This remark is worth a lot of transistors since most of the elementary combinatorial operations on SBD, like digit multiply or add, are increasing logic functions and the simple restoring gates provided by the technology are decreasing. This remark about change of sign is also useful to change an adder into a subtractor without changing any gate.

B. The operators

B.1 The PPM operator.

This operator computes d and e from its three-input a, b and c, that gives either (a plus b minus c) or (c minus a minus b), depending on the sign affected to the inputs ant the outputs. It is used to build a carry-free parallel (2 gives 1) addition, a carry-save parallel addition (CSA) (3 gives 2) as well as a serial addition.

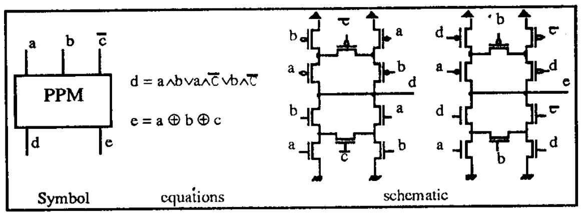

Fig. 1 *The PPM operator*

Parallel two-input adder

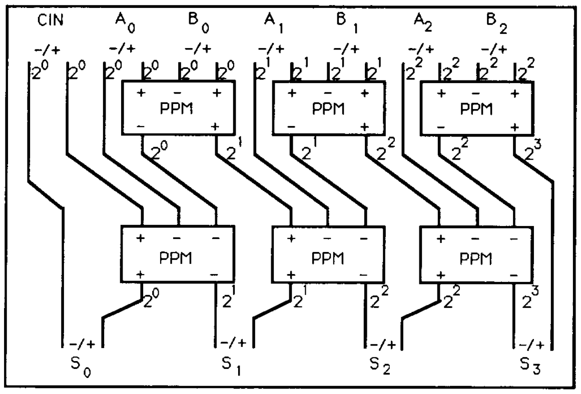

Fig. 2 *A parallel two-input adder*

In this adder that perform S=A+B+Cin, it is easy to verify that the carry does not propagate longer than two positions and that the weight and sign of bits are consistent.

Three-input carry-save adder.

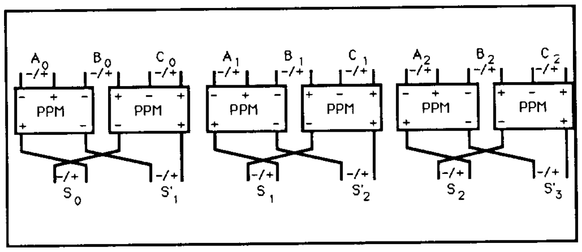

Fig. 3 *A 3-input carry-save adder .*

By combining the carry-free adder with this carry-save adder, a three-input parallel adder is built with a delay of three PPM boxes (i.e., 6 gates).

Most significant bits first serial addition.

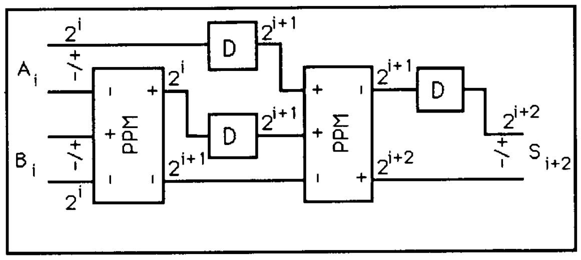

Fig. 4 *An on-line adder .*

A and B are fed in serially, boxes labeled D are one cycle delays (master-slave flip-flops). By combining this circuit with a one SBD CSA and two boxes D, the three-input serial adder used in the multiplier is built.

B.2 SBD multiplier.

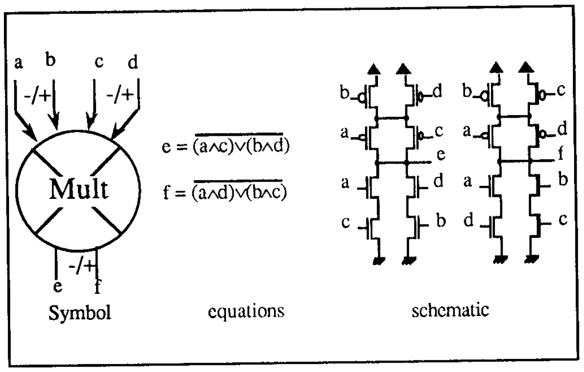

Fig. 5 *Elementary SBD multiplier.*

The choice of radix 2 allows the product of two SBD to be one SBD.

On-line multiplier.

In this picture, double wires carry SBD, while the boxes labeled L are latches to hold digits of A and B. Before computation, there are all cleared (set to zero). During computation a control bit (token) runs backward to load serially the incoming SBD in the A and B latches. The detailed timing of the serial computation of A times B is explained on 3 bits in the table below.

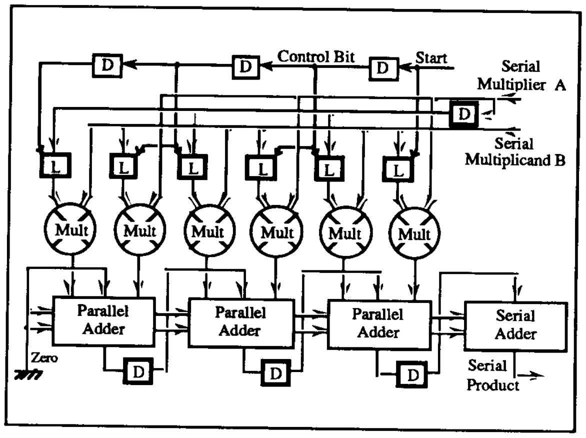

Fig. 6 *On-line multiplier .*

Cycle numb	Incoming bits	Multiplicand latch	Multiplier latch	Inputs of the parallel adder (+ 2 times previous line)
0		0 0 0	0 0 0	0 0 0
1	a_2 b_2	0 0 0	0 0 b_2	0 0 a_2b_2
2	a_1 b_1	0 0 a_2	0 b_1 b_2	0 a_1b_1 $a_1b_2+a_2b_1$
3	a_0 b_0	0 a_1 a_2	b_0 b_1 b_2	a_0b_0 $a_0b_1+a_1b_0$ $a_0b_2+a_2b_0$

Table 1 *Computation of AxB*

Eventually, the product is computed as follows:
$((((a_2b_2)*2 + a_1b_2 +a_2b_1)*2 + a_1b_1+a_0b_2 +a_2b_0)*2$
$+ a_0b_1+a_1b_0)*2 + a_0b_0$

Implementation considerations.

The circuit fig. 6 is assembled from three instances of the identical SBD slice. By adding more slices a circuit for any precision can be obtained, no resizing of the transistors being necessary, since each slice communicates only with the next ones, so the size of the slices can be kept small. Besides the non recoding SBD addition is far less demanding for transistors than the previous solutions known to the authors [10] and few costly master-slave flip-flops are used. Altogether, each slice contains 32 gates and 152 transistors, so the 600 decimal digit (2048 SBDs) multiplier would cost a little over 300k transistors, with a very high regularity.

B.3 SBD Divider.

We implement the algorithm of Ercegovac and Trivedi [6]. As shown on fig.7 the divider consits of two parts. The right part is the same multiplier as described in fig. 6. It multiplies the partial quotient (signed) by the partial dividend, adding one bit to both of them at each clock cycle, and add the signed result to the partial remainder PR (without carry propagation on up to 2048 positions). The right part of the circuit propagates a carry on the 4 most significant positions of the partial remainder in order to simplify comparisons, and from its value predicts the next SBD of the quotient.The division algorithm is listed below. Let us assume that there is a radix point between the two parts of the circuit, i.e., the right part works on the fractional part of the numbers while the left part works with the integer parts. This is purely conventional since the divider does not have to know the real weight of the digits. In the algorithm the divider is assumed to be positive, each of its SBD as well as the result has to be complemented if it turns out to be negative.

Algorithm [6]

```
While D < 8 do D:= D * 2 + next SBD from divisor;
        { loop until the four bits known part D of the divisor is
          big enough to ensure the convergence of the division
          algorithm ,i.e., 7 < divisor< 16  or  8 ≤ D ≤ 15 }
start multiplying ;
loop until dividend is exhausted
begin
    get next SBD from dividend into multiplier ;
    PP :=  PR - 2 ≥ 0 ;                    {0 ≡ false,  1 ≡ true}
    PN := -PR - 2 ≥ 0;
    next SBD of quotient  :=  PN-PP ;
                         { PN and PP are the 2 bits of the SBD}
                    { next SBD of  quotient goes to the multiplier
                              and is output   from the circuit }
    PR := PR + incoming digits from multiplier  +
         next digit from divisor ;          {  | PR |  < 2 * D }
    PR := 2 *
      ( if PP then PR -D else if PN then PR + D else PR ) ;
                                           { | PR | < 2 * D - 4}
end;
```

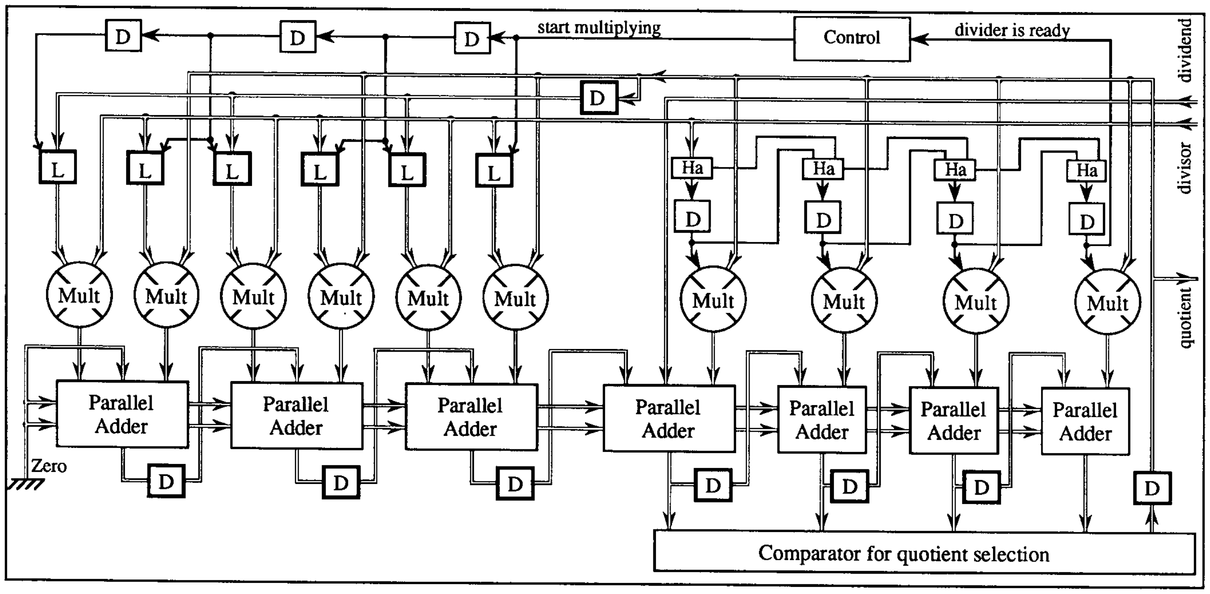

Fig. 7 *On line divider .*

C. Synchronization

From the on-line adder of fig. 4, the multiplier of fig. 6 and the divider of fig. 7 and a few switches, it is easy to assemble a programmable operator which is slightly more complex than the divider and can perform any of the operations. This circuit is designed to be controlled by a Transputer via the links. As in the Transputer, the flow of data is regulated by a token that goes backward to request for data.

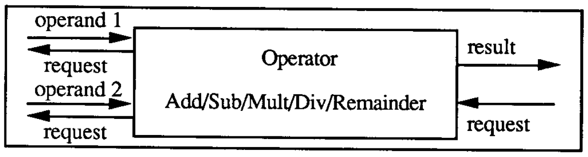

Fig. 8 *Regulation of the data flow .*

The circuit is fitted with two fifo for the input operands to delay one operand when the other is not yet available or when the weights have to be aligned, for addition and subtraction. This feature allows sharing of data.

Example. Let us compute (x+b)x (x is shared).

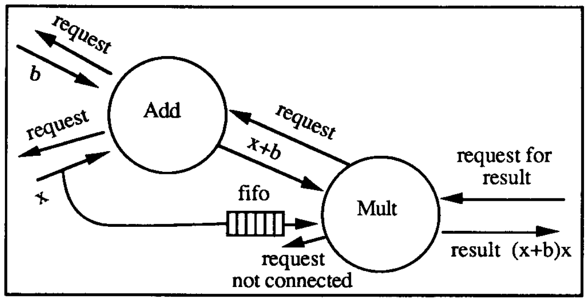

Fig. 9 *Example of data sharing .*

It is easy to see that if the fifo is longer than the delay of the addition, regulation for x in the multiplier is performed through the adder. Duplication of data is allowed provided that it is used to compute the same result.

D. Implementation and performances.

We have already designed (as a test circuit) a 64 digit SBD multiplier. The floorplan and the layout are given in Fig. 10 and 11. For each cycle, less than 10 gates are passed through. With a delay of 3ns per gate, a frequency of 30MHz can be expected that is 30 million digits of result per second. Provided that enough circuits are pipelined and that numbers fit in up to 600 decimal digit, this figure is almost independent of the complexity of the expression.

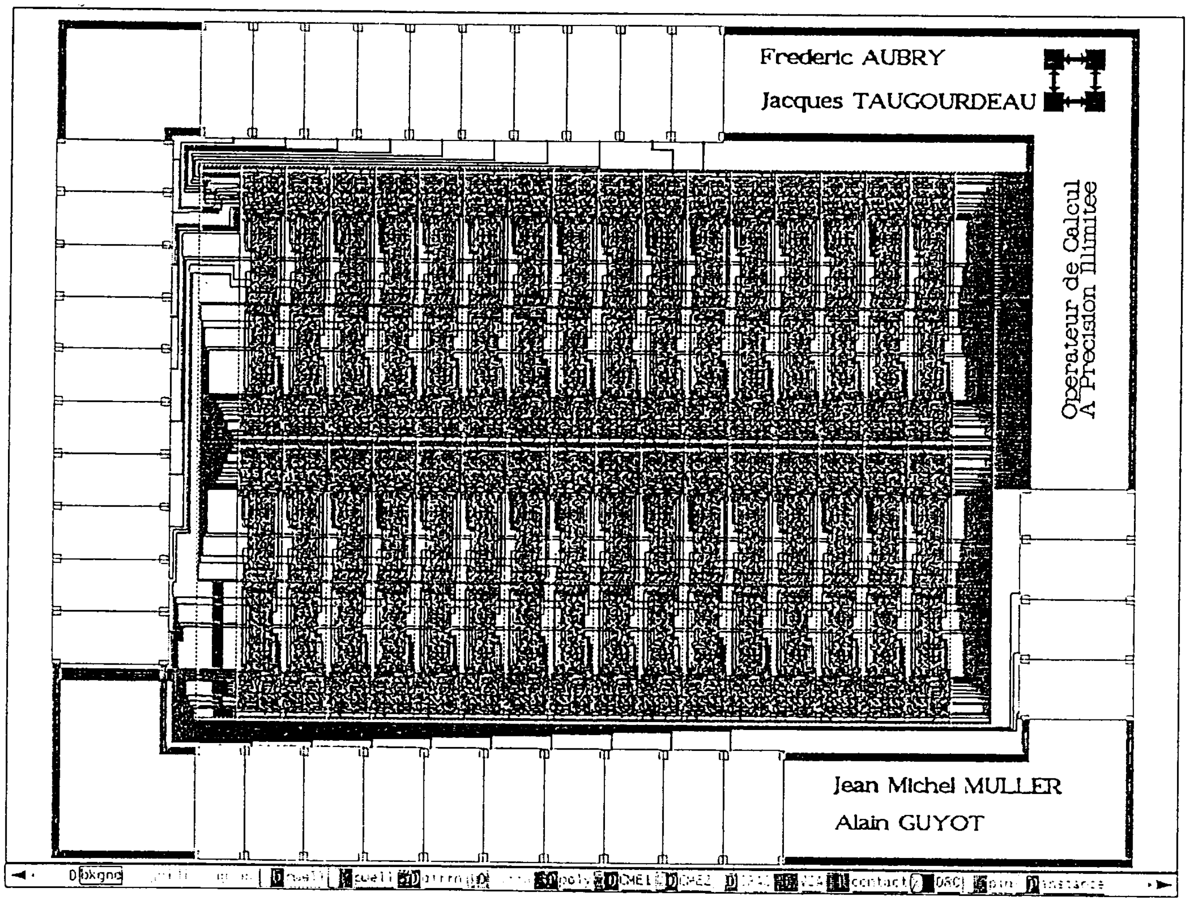

Fig. 11 *Layout* .

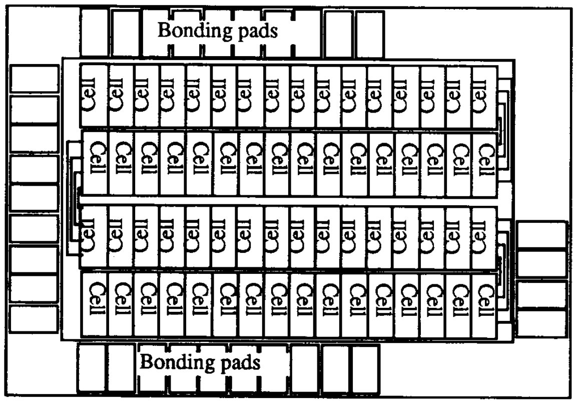

Fig. 10 *Organization of the circuit* .

References

[1] A. Avizienis, *Signed-digit number representations for fast parallel arithmetic* , IRE Transactions on electronic computers, 10, pp. 389-400, 1961.

[2] J. Duprat, Y. Herreros and J.M. Muller, *Some results about on-line computation of functions*, submitted to the 9th symposium on computer arithmetic.

[3] M.D. Ercegovac and T. Lang, A division algorithm with prediction of quotient digits, 7th Symposium on computer arithmetic, Urbana, Illinois, June 1985.

[4] M.D. Ercegovac and T. Lang, On-line scheme for computing rotation factors, 8th Symposium on computer arithmetic, Como, Italy, May 1987, IEEE Publ. No 87CH2419-0.

[5] M.D. Ercegovac, On-line arithmetic : an overview, SPIE Vol. 495, Real time signal processing VII, pp 86-93, 1984.

[6] M.D. Ercegovac and K.S. Trivedi, On line algorithms for division and multiplication, IEEE Trans. on Computers, Vol. C-26 No 7, pp 681-687, July 1977.

[7] M.D. Ercegovac and P.K.G. Tu, A radix-4 on-line division algorithm, 8th Symposium on computer arithmetic, Como, Italy, May 1987, IEEE Publ. No 87CH2419-0.

[8] A.L. Grnarov and M.D. Ercegovac, On the performance of on-line arithmetic, Proc. 1980 Intern. Conference on parallel processing, IEEE Publ. No 80CH1569-3, pp 55-62, Aug.1980.

[9] V.C. Hamacher and J. Williams, A linear-time divider array, Canadian Electr. Engineering Journal, Vol.6, No 4, 1981.

[10] M. d'Hoe, M. Pierre, Ph. Deleuze, A. Vandemeulebroecke, P. Jespers and M. Davio, CASBA: Cryptographic application using signed binary arithmetic. ESSIRC'85 September 1985.

[11] K. Hwang, Computer arithmetic principles, architecture and design, New-York, J. Wiley&Sons Inc., 1979.

[12] M.J. Irwin, An arithmetic unit for Online computation, PhD thesis, tech. report UIUCDCS-R-77-873, Dept. of Computer science, university of Illinois, Champaign-urbana, IL 61801, May 1977.

[13] M.J. Irwin, A pipelined processing unit for on-line division, Proc. 5th symposium on Computer architecture, IEEE Publ. No 78CH1284-9C, pp 24-30, April 1978.

[14] M.J. Irwin and R.M. Owens, On-line algorithms for the design of pipeline architectures, 6th symposium on Computer Architecture, Philadelphia, PA, April 1979.

[15] M.J. Irwin and R.M. Owens, Digit-pipelined arithmetic as illustrated by the paste-up system : a tutorial, IEEE Computer, pp 61-73, April 1987.

[16] H. Lin and H.J. Sips, A novel floating-point online division algorithm, 8th Symposium on computer arithmetic, Como, Italy, May 1987, IEEE Publ. No 87CH2419-0.

[17] J.E. Robertson, A new class of digital division methods, IRE Transactions on electronic computers, Vol. C-7, Sept. 1958.