Aera-time optimal VLSI circuits for convolution

G. Baudet, Franco P. Preparata, J. Vuillemin

IRIA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105 78150 Le Chesnay
France
Tél. 954 90 20

Rapports de Recherche

N° 30

AREA-TIME OPTIMAL VLSI CIRCUITS FOR CONVOLUTION

Gérard M. BAUDET
Franco P. PREPARATA
Jean E. VUILLEMIN

Août 1980

AREA-TIME OPTIMAL VLSI CIRCUITS

FOR CONVOLUTION

Gérard M. BAUDET
INRIA
B.P. 105
78150 LE CHESNAY
FRANCE

Franco P. PREPARATA
Coordinated Science
Laboratory
University of Illinois
Urbana, IL 61801
U. S. A.

Jean E. VUILLEMIN
Université de Paris-Sud
Laboratoire d'Informatique
Bâtiment 490
91405 ORSAY
FRANCE

ABSTRACT

A family of VLSI circuits is presented to perform open convolution, i.e, polynomial multiplication. The circuits are all based on a recursive construction and are therefore particulary well adapted to automated design. All the circuits presented are optimal with respect to the (area)x(time)^2 trade-off, and, depending on the degree of parallelism or pipeline, they range from a compact but slow convolver to a large but very fast convolver.

RESUME

Une famille de circuits VLSI est présentée pour effectuer la convolution plate sur le produit de polynomes. Tous les circuits présentés sont construits à partir de définitions récursives qui se prêtent très facilement à la conception automatisée. Ces circuits sont tous optimaux par rapport au compromis (surface)x(temps)^2, et, selon le degré de pipeline, on trouve un convolveur compact mais lent, et un convolveur de grande surface, mais très rapide.

* This work was partially supported by National Science Foundation Grant MCS-78-13642, by the Joint Services Electronics Program Contrat N00014-79-C-0424, by INRIA, Institut National de Recherche en Informatique et Automatique, 78150 Le Chesnay, France, and by ERA 452 "al Khowarizmi" of Centre National de la Recherche Scientifique.

1 - INTRODUCTION

Very-Large-Scale integration (VLSI) is revolutionizing the methologies of digital system design. The traditional criteria of component count -whether applied to processors or to simpler devices- are no longer adequate to establish a scale of comparison among various solutions to a given problem. Indeed number-of-elements criteria are substantially based on the fact that processing elements and their interconnections are realized by different media. This difference disappears in VLSI, which "integrates" both processing elements and their interconnection in a two-dimensional geometry, the surface of the silicon chip. A meaningful figure-of-merit is represented by the area occupied by the total system, thus capturing the complexity of both computation and data communication. As a result, the VLSI solution to a given computational problem involves the conception of an interconnection architecture, its layout, and the design of an algorithm for that architecture. For any given problem it is of great interest to explore the trade-offs between the production cost (area) and the incremental cost (time) of a dedicated circuit developed to solve that problem.

1.1 - Theoretical considerations

To provide a meaningful gauge for the evaluation of a given design, a computational model of VLSI has been developed recently, through the initial efforts of Mead-Conway [5] and Thompson [8] ; later refinements have been proposed by Brent-Kung [1] and Vuillemin [9]. We briefly recall the model, with the latest amendments.

A circuit is a graph whose nodes are I/O ports or gates, connected by wires. A wire has minimal width λ and at most ν wires overlap at any point ; a gate has minimal area λ^2 and computes boolean function of two inputs ; an I/O port has some minimal area and each input bit is available just once.
As regards computation time, the combined gate-computation and result-transmission (on a wire between two nodes) takes some time τ dependent upon the technology. One relevant global time parameter is the output period P of the circuit, defined

as the maximum time between two successive data passages at any output port when the circuit is used in a pipelined fashion at the highest data rate. Another measure is the time T which elapsed between the beginning and the end of one computation by the chip for one instance (rather than repeated instances) of a given problem. On the basis of arguments on the information transfer inside the VLSI chip of area A realizing the circuit, natural measures of complexity in the given module are the area-time products AP^2 and AP.

If we define the problem size N as the larger of the total number of bits used to specify either the input or the output, a simple argument by Vuillemin [9] shows that the circuit area A, period P and problem size N satisfy the relationships

$$AP^2 \geq N^2$$

and

$$AP \geq N^{3/2}$$

for such fundamental problems as integer multiplication, merging, cyclic shift, cyclic and open convolution and linear transform.

Computing time T and period P are obviously related by $T \geq P$ so the above inequalities imply $AT^2 \geq N^2$ and $AT \geq N^{3/2}$.

Several of the above mentioned problems have been considered elsewhere [1,4,6,7] and circuits have been proposed which are optimal with respect to the AP^2 or AP measure. This paper is devoted to circuits implementing convolution which are optimal with respect to AP^2.

1.2 - Convolution

Convolution of two sequences of numbers is a central problem in signal processing. Digital filtering is indeed the convolution of a fixed sequence -the discretized impulse response of the filter- and the signal sequence.

The present state of VLSI technology makes it quite feasible to integrate a convolver, either as a special purpose chip or as part of a more general signal processing chip. Applications in telecommunications, image processing and other engineering fields have already motivated important and expanding research in this area.

For specificity, we define the convolution A*B of two number sequences $A=(a_o,\ldots,a_{n-1})$, $B=(b_o,\ldots,b_{n-1})$ as $C=(c_o,\ldots,c_{2n-2})$ where $c_i = \sum_{o\le j\le i} a_j b_{i-j}$ for $o\le i<2n-1$. In other words, convolution is a polynomial product : $C(x) = \sum_{0\le i<2n-1} c_i x^i = A(x).B(x)$ where $A(x) = \sum_{0\le i<n} a_i x^i$, $B(x) = \sum_{0\le i<n} b_i x^i$. The input coefficients a_i, b_i for $0\le i<n$ will be assumed to be integers in the range $[0,2^p[$, thus the output coefficients c_i need to be coded on at most $k \triangleq 2p + \lceil \log_2 n \rceil$ bits. Measured as the total number of output bits, our problem size is therefore $N = O(nk)$. For sake of clarity, we will assume throughout that both input and output coefficients are coded over k bits (e.g., input coefficients are padded with sufficiently many zeroes). We observe immediately that $k \ge \log_2 N$.

Other types of convolutions are useful in engineering, but they are all amenable to a treatment very similar to the one which is chosen here for simplicity of exposition.

1.3 - Contribution of this paper

The most widely used circuit for computing convolutions involves a multiplier-accumulator whose function is to compute the successive $a_j b_{i-j}$ and accumulate the corresponding c_j. One also has to provide registers for storing all of the a_i and b_i, plus some data and control paths. If a traditional shift and add multiplier is used, each product can be computed and accumulated in time O(k) with a circuit having O(k) area. The area used for storing A and B is O(kn) and, since there are $O(n^2)$ products to be computed, the resulting chip has area A=O(kn) and computing time (and period) $T=O(n^2k)$. Although such a design occupies a rather small area, its computing time is too slow for many applications.

A more interesting solution has been recently proposed by Kung-Leiserson [4], using a linear array of identical cells which has also found applications in pattern matching (Foster-Kung [2]). Their array consists of 2n linearly connected multiplier-accumulator cells, each of which computes

one of the output coefficients. Input sequences $(a_o,\ldots,a_{n-1})$ and $(b_o,\ldots,b_{n-1})$ advance throught the circuit in opposite directions, so that a_j and b_{i-j} successively meet in cell i. The area of the resulting chip is A=O(nk) while its computation time (and period) is T=O(nk).

Neither of these designs is optimal with respect to the AP or AP^2 measure, and the next section of this paper is devoted to designing AP^2-optimal convolvers.

One key idea in our designs is to define circuits recursively. The principle of recursive convolution is presented in its simplest setting in section 2.1 and the corresponding convolver is analyzed in section 2.2. A refinement of the basic idea in 2.3 yields an AP^2-optimal convolver for a wide class of problem sizes. Finally, in section 2.4, we show that the above circuits can be used in a pipelined fashion to construct convolution circuits achieving the optimal value $AP^2=N^2$ for any area size, ranging form A=N to $A=N^2$

2 - OPTIMAL CIRCUITS FOR CONVOLUTION

In this section, we present successive versions of a convolution circuit. The performances of each circuit are analyzed, and the results of such analysis are used as a guide towards the next improved version of the circuit.

2.1 - The basic recursive convolver

The product $C(x) = A(x) \cdot B(x) = c_o + c_1x + c_2x^2$ of two degree one polynomials $A(x) = a_o + a_1x$ and $B(x) = b_o + b_1x$ can be computed with 4 multiplications and 1 addition :

$$p_1 = a_ob_o,\ p_2 = a_ob_1,\ p_3 = a_1b_o,\ p_4 = a_1b_1$$
$$c_o = p_o,\ c_1 = p_2 + p_3,\ c_2 = p_4. \qquad (2.1)$$

In general, consider two polynomials A, B of degree less than 2n, their product C=A×B, and write $A(x) = A_o(x) + x^n A_1(x)$, $B(x) = B_o(x) + x^n B_1(x)$, $C(x) = C_o(x) + x^n C_1(x) + x^{2n} C_2(x) + x^{3n} C_3(x)$ where all A_i, B_i, C_i have degree less than n. The C_i's can be otbained by first computing recursively the 4 products

$$\begin{aligned} P^1(x) &= A_o(x).B_o(x) = P_o^1(x) + x^n P_1^1(x), \\ P^2(x) &= A_o(x).B_1(x) = P_o^2(x) + x^n P_1^2(x), \\ P^3(x) &= A_1(x).B_o(x) = P_o^3(x) + x^n P_1^3(x), \\ P^4(x) &= A_1(x).B_1(x) = P_o^4(x) + x^n P_1^4(x), \end{aligned} \tag{2.2}$$

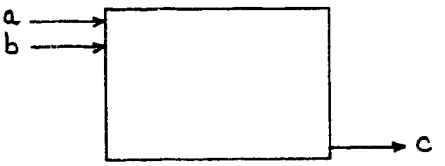
and then performing 4 polynomial additions :

$$\begin{aligned} C_o(x) &= P_o^1(x), \\ C_1(x) &= P_1^1(x) + P_o^2(x) + P_o^3(x), \\ C_2(x) &= P_1^2(x) + P_o^3(x) + P_o^4(x), \\ C_3(x) &= P_1^4(x). \end{aligned} \tag{2.3}$$

Such formulas lead to an obvious recursive algorithm for computing polynomial products. We now show that such an algorithm can be translated into a circuit, so to speak wired into the silicon.

Our aim is to describe a circuit M_n which multiplies two polynomials of degree less than n. Recall that the coefficients a_i, b_i and c_i are all coded with k bits. Circuit M_n has 2n input wires, one for each of the coefficients $a_o,\ldots,a_{n-1}$, $b_o,\ldots,b_{n-1}$ of A and B ; on each of these wires, the k successive bits of a_i or b_i are read in serially. Similarly, M_n has 2n-1 output wires, one for each of the coefficients $c_o,\ldots,c_{2n-1}$ of C, and, on each such output wire, the k bits necessary to encode each c_i appear serially.

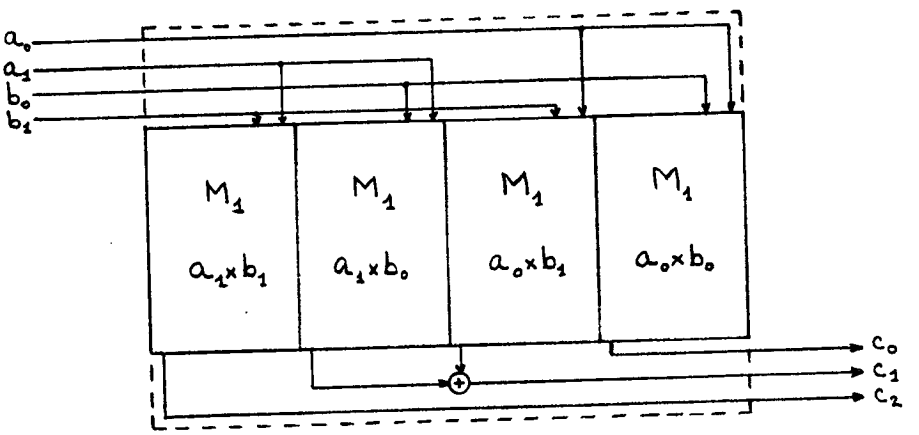
The basis of our recursion is a circuit M_1 (see figure 1) which performs the serial multiplication of two k-bit integers a and b, computing serially the k bits of $c=a\times b \pmod{2^k}$.



Basic serial multiplier

Figure 1

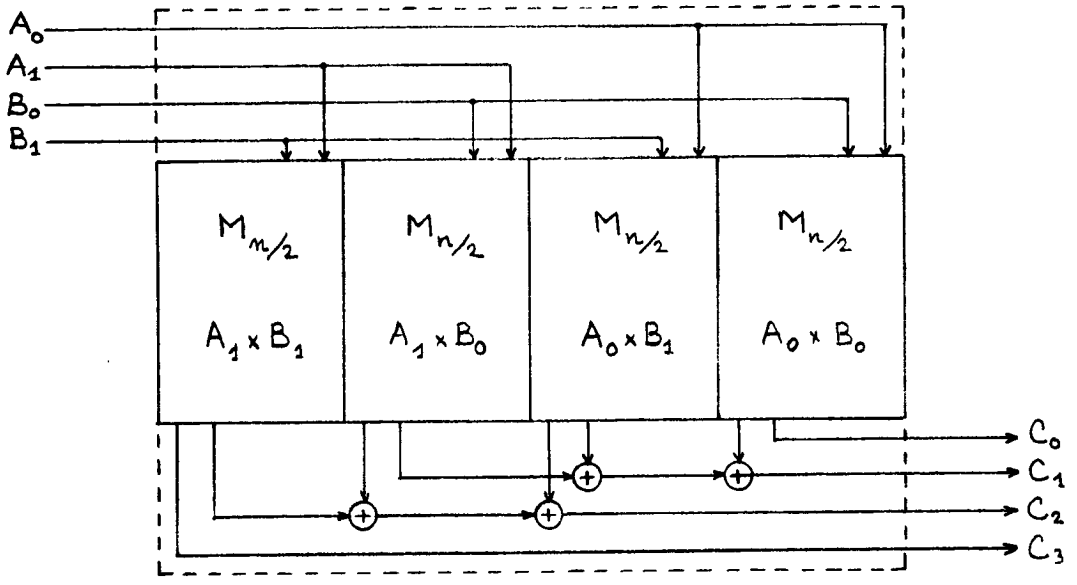
Formulas 2.1 can then be used to define a circuit M_2 combining 4 circuits M_1 and 1 elementary serial adder, as shown in figure 2.



Circuit M_2

Figure 2

In general, formulas 2.2, 2.3 are used to construct a circuit M_n using four circuits $M_{n/2}$ and four adders as in figure 3. There, each wire $A_0, A_1, \ldots, C_3$ represents n/2 parallel wires, each of which carries k bits serially. The k bits of the output wires are added in a serial adder whose surface has the same order of magnitude as that of the intersection of the wires. Note the placement of the modules $M_{n/2}$ within M_n : if we consider standard a module placement where the larger of the two dimensions is the horizontal one, for a standard placement of M_n, each of the $M_{n/2}$ is rotated 90° with respect to the standard.



A circuit M_n constructed from 4 circuits $M_{n/2}$

Figure 3

2.2 - Performance of the basic circuit

Assume now that we use a circuit M_n (with n a power of 2) to compute the product of two degree n-1 polynomials, each coefficient being coded on k bits, and let A_n, P_n, and T_n denote respectively the area, period and time of M_n. The area $A_n = H_n . W_n$ is the product of the width W_n by the height H_n of M_n. Refering to figure 3, where height and width of M_n are respectively the vertical and the horizontal dimension, for n>1 these quantities are recursively defined by :

$$H_n = W_{n/2} + 8 . \frac{n}{2} . \lambda \ , \ W_n = 4H_{n/2}.$$

With regard to T_n and P_n, we obtain :

$$T_n = T_{n/2} + k.t,$$

where kt is the time for the serial addition,

$$P_n = \max(P_{n/2}, kt),$$

where we assume that the circuit is clocked so that it can be used in a pipelined fashion.

The basis for these recurrences is provided by a classical shift and add multiplier, having parameters $A_1 = W_1 H_1 = ak$, $T_1 = P_1 = t_1 k$, where a and t_1 are suitable constants dependent upon the actual design. Solving the previous recurrences yields :

$$A_n = W_n H_n = (4\lambda n \log n + W_1 n).(2\lambda n \log n + H_1 n)$$

$$P_n = \max (t_1 k, tk)$$

$$T_n = t.k \log n + t_1 k.$$

If we let N=nk represent the total problem size, measured in bits, the measure AP^2 for M_n is expressed by :

$$AP^2 = O(N^2k^2\log^2 N) \qquad \text{for } k \leq \log^2 N$$

$$AP^2 = O(N^2k^3) \qquad \text{for } \log^2 N \leq k.$$

In particular, AP^2 is never asymptotically equal the optimal value $O(N^2)$.

2.3 - An optimal convolver for the measure AP^2

Keeping the recursive convolver of section 2.1 in mind, our next objective is to optimize its AP^2 measure. The basic idea is to adopt a scheme where the number of recursive multiplications grows at a slower rate than the square of the decimation ratio. (In our previous example, the decimation ratio was 2 and the number of recursive multiplications was 4.)

A first instance of such an idea is a well-known method, attributed to Karatsuba [3], whereby the product of two first degree polynomials can be obtained with three, rather than four, elementary multiplications. With the same notation as in section 2.1, we have

$$p_1=a_o.b_o,\ p_2=(a_o+a_1).(b_o+b_1),\ p_3=a_1.b_1\ ;$$

$$c_o=p_1,\ c_1=p_2-(p_1+p_3),\ c_2=p_3.$$

This method can now be used to obtain the product $C(x)=A(x).B(x)$ of two polynomials A(x) and B(x), each of degree less than 2n. Again, letting $A(x)=A_o(x)+x^nA_1(x)$, $B(x)=B_o(x)+x^nB_1(x)$, $C(x)=C_o(x)+x^nC_1(x)+x^{2n}C_2(x)+x^{3n}C_3(x)$ (with degree $(A_i, B_i, C_i)<n$), we have :

$$P^1(x)=A_o(x).B_o(x)=P_o^1(x)+x^nP_1^1(x)$$
$$P^2(x)=(A_o(x)+A_1(x))(B_o(x)+B_1(x))=P_o^2(x)+x^nP_1^2(x)$$
$$P^3(x)=A_1(x).B_1(x)=P_o^3(x)+x^nP_1^3(x)$$

whence :

$$C_o(x)=P_o^1(x)$$
$$C_1(x)=P_1^1(x)+P_o^2(x)-(P_o^1(x)+P_o^3(x))$$
$$C_2(x)=(P_1^2(x)+P_o^3(x))-(P_1^1(x)+P_1^3(x)).$$
$$C_3(x)=P_1^3(x).$$

Thus, this scheme uses three multiplications and eight additions-subtractions of polynomials of degree less than n.

As regards the circuit layout, here again we adopt the scheme illustrated in the previous section, i.e., convolver M_n is assembled from adders, subtracters, wires, and convolvers $M_{n/2}$, the latter rotated 90° with respect to their standard placement. The actual layout of M_n is shown in figure 4.

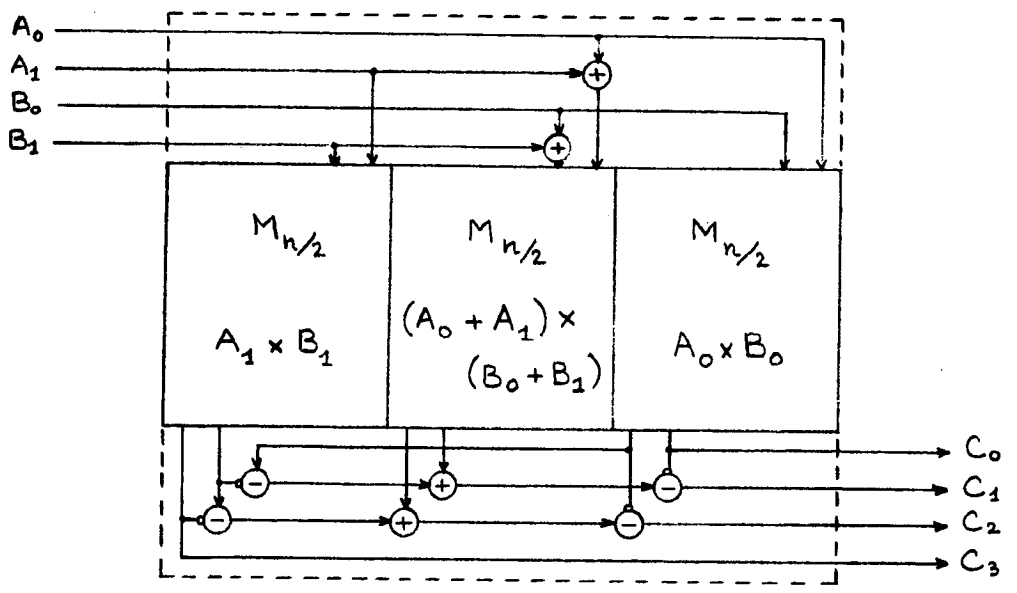


Figure 4

As with the scheme of section 2.1, the basic serial multiplier M_1 can be used at the bottom of the recursion. By inspection of figure 4 we obtain the following recurrence (with the usual convention about width and height) :

$$H_n = W_{n/2} + 8.\frac{n}{2}\lambda \ , \quad W_n = 3H_{n/2}.$$

The period P_n of the convolver M_n is the largest of the time for the serial additions of the coefficients (i.e. kt) and of the period $P_{n/2}$ of the convolver $M_{n/2}$; thus

$$P_n = \max \ \{kt, P_{n/2}\}.$$

the latter recurrence immediately gives $P_n = k \max(t,t_1)$. If we now solve the above recurrences for H_n and W_n , letting n be a power of 2, we find that for p even :

$$H_{2^p} = 16(2^p - 3^{p/2}).\lambda + 3^{p/2}.H_1$$

$$W_{2^p} = 24(2^p - 3^{p/2}).\lambda + 3^{p/2}.W_1,$$

while, for p odd, we have $H_{2^p} = \frac{1}{3} W_{2^{p+1}}$ and $W_{2^p} = 3W_{2^{p-1}}$. We see therefore that both H_n and W_n have length O(n) as long as both H_1 and W_1 are bounded above by $O(n^{1-\alpha/2})$. Since we know $H_1 W_1 = O(k)$, this implies that as long as k is bounded above by $O(n^{2-\alpha})$, the measure AP^2 is bounded by $O(n^2k^2) = O(N^2)$, where again N=nk denotes the problem size. (Note that $k=n^{2-\alpha}$ corresponds to $k=N^{(2-\alpha)/(3-\alpha)} \simeq N^{0.3}$). This shows that $AP^2 = O(n^2k^2) = O(N^2)$, i.e., an AP^2-optimal convolver of the proposed type exists for all values of k comprised in the range $[\log N, N^{0.3}]$.

REMARK : The scheme just described, which allows us to obtain AP^2-optimal convolvers, is merely an instance of the general class of methods called "evaluation-interpolation". Indeed, given $A(x)=a_o+a_1x$, we realize that a_o, a_o+a_1, and a_1 are respectively equal to A(0), A(1), and $\lim_{x\to\infty} A(x)/x$, i.e. the evaluations of A(x) at the distinct values 0,1, and ∞. Once A(0), A(1) and $A(\infty)$ have been obtained, the products $A(0).B(0)=C(0)$, $A(1).B(1)=C(1)$ and

and $A(\infty).B(\infty)=C(\infty)$ are computed and the second degree polynomial C(x) is obtained by interpolation from the values C(o), C(1), and $C(\infty)$.

The fullest exploitation of this approach employs the Discrete Fourier Transform or any other convolution-supporting transformation. Other instances are also possible based on integer evaluation values (rather than primitive roots of unity in a suitable field). However there are considerable difficulties in analyzing such an approach, since both evaluation and interpolation are matrix-vector multiplications, and, besides additions and substractions, also multiplications and division must be executed. It is an interesting question whether there exist appropriate choices of the evaluation values corresponding to simple circuit implementations.

2.4 - An AP^2-optimal convolver for bounded parallelism

The convolver presented in section 2.3, while optimal for the measure AP^2, requires an area which grows quadratically with the length of the sequences to be convolved. Up to this point we assumed unbounded parallelism, i.e. the convolver can be tailored to the problem size. The quadratic growth of the area, however, is prohibitive for long sequences of coefficients (i.e., high degree polynomials), so it becomes necessary to examine the case of bounded parallelism (i.e., fixed convolver and arbitrary problem size). In the remainder of this section we show that, even in the case of bounded parallelism, an AP^2-optimal convolver can still be designed.

Reduction of the area of the circuit presented in the preceding section is achieved by using it in a pipeline fashion.

We merely observe that the convolver designed for multiplying two degree n-1 polynomials can be used, with few changes (i.e., changes which will not influence the overall area), to multiply two degree nq-1 polynomials.

Indeed, the nq coefficients of a degree nq-1 polynomial can be split into n strings, each with p consecutive coefficients. In order to perform the product C=A.B of two degree nq-1 polynomials, the 2n strings corresponding to A and B are fed in parallel, while, within each string, the q coefficients are fed serially. The k bits of each coefficient are still fed serially, resulting

in qk bits fed serially on each input line. The 2nq-1 coefficients of the product C are delivered to the output according to a similar format.

Using the same scheme as in section 2.3, a circuit M_{nq} can, again, be defined recursively to multiply two degree nq-1 polynomials. The recursion leaves us with degree q-1 polynomials, and the basis of the recursion is now a circuit M_q^* which performs the multiplication of degree q-1 polynomials, where the 2kq input bits are read serially. One such circuit has been proposed by Kung and Leiserson [4] and consists of 2q integer multipliers and accumulators which are functionnaly connected as a linear array but may physically be laid out -in a "snake-like" fashion- to obtain a nearly square module M_q^*. For given technology-dependent constants a" and t", the circuit requires an area $A_q=W_q.H_q=2q\ a"$, while its period is given by $P_q=2qkt"$. From this, we deduce the characteristics of circuit M_{nq} through recurrence relations identical to those of the circuit of section 2.3 . We obtain (for n a power of 4, and with $a=\log_2 3$) :

$$W_{nq} = n^{\alpha/2} W_q + w" n,$$
$$H_{nq} = n^{\alpha/2} H_q + n" n,$$
$$P_{nq} = P_q.$$

Thus (provided $W_q=H_q$), we have

$$A_{nq} = W_{nq}.H_{nq} = n^{\alpha}A_q + O(n^2).$$

Again, letting N=nqk denote the size of the problem, this yields :

$$AP^2 = O[N^2 + n^{\alpha}(qk)^3].$$

Since P=O(qk), we conclude that circuit M_{nq} is optimal with respect to the measure AP^2 for any walue of the period in the range :

$$k \leq P \leq N^{\frac{2-\alpha}{3-\alpha}} \simeq N^{0.3}.$$

3 - CONCLUSION

In this work, we show how a theoretical result -$AP^2=\Omega(N^2)$- provides a powerful guide in the design of asymptotically optimal convolution circuits. The techniques used are very simple : a careful wiring of a well-known polynomial product algorithm provides a large, yet AP^2-optimal convolver ; grafting the general technique of pipelining upon this basic design then yields a family of AP^2-optimal convolver, for all choices of P in a wide interval. The practical potential of such designs remains to be proved. However, there are many favorable signs in that respect :

- the basic recursive construction technique meets all of the basic constraints of current VLSI practice : regular, systematic, repetitive and dense layout ; modularity and generality of a design which can potentially be fully automated.

- if one is willing to use a specific convolver as part of a more general signal-processing chip, the area-time compromises demonstrated in section 2.4 should be helpful in finding the proper balance between control and computing power within the chip.

REFERENCES

1 Brent,R.P., and Kung,H.T., The chip complexity of binary arithmetic. Proceedings of the 12-th Annual ACM Symposium on Theory of Computing, Los Angeles, California, 1980, pp. 190-200.

2 Foster,M.J., and Kung,H.T., Design of special-purpose VLSI chips : examples and opinions, Carnegie-Mellon University, Computer Science Technical Report, September 1979.

3 Karatsuba,A., and Ofman,Y., Multiplication of multidigit numbers on automata, Doklady Akademiia Nauk SSSR, Vol. 145, 1962, pp. 293-294 (in Russian).

4 Kung,H.T., and Leiserson,C.E., Algorithms for VLSI processor arrays, in [5], 1980, pp. 271-292.

5 Mead,C.A., and Conway,L.A., Introduction to VLSI Systems, Addison-Wesley, Reading, Mass., 1980.

6 Preparata,F.P., and Vuillemin, J.E., Area-time optimal VLSI networks based on the Cube-Connected-Cycles, Rapport de Recherche INRIA, No. 13, March 1980.

7 Preparata,F.P., and Vuillemin,J.E., Area-time optimal VLSI networks for multiplying matrices, 14-th Princeton Conference on Information Sciences and Systems, March 1980.

8 Thompson,C.D., Area-time complexity for VLSI, Proceedings of the 11-th Annual ACM Conference on Theory of Computing, Atlanta, Georgia, May 1979, pp. 81-88.

9 Vuillemin,J.E., A combinatorial limit to the computing power of VLSI circuits, to appear.