HAL
open science

Area-time optimal VLSI networks based on the cube-connected-cycles

Franco P. Preparata, J. Vuillemin

To cite this version:

Franco P. Preparata, J. Vuillemin. Area-time optimal VLSI networks based on the cube-connected-cycles. [Research Report] RR-0013, INRIA. 1980. inria-00076548

HAL Id: inria-00076548

https://inria.hal.science/inria-00076548v1

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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° 13

AREA-TIME OPTIMAL VLSI NETWORKS BASED ON THE CUBE-CONNECTED-CYCLES

Franco P. PREPARATA
Jean E. VUILLEMIN

Mars 1980

AREA-TIME OPTIMAL VLSI NETWORKS BASED ON THE CUBE-CONNECTED-CYCLES

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
Bât. 490
91405 Orsay
FRANCE

Abstract

We present designs for VLSI circuits computing Cyclic Shifts, Discretre Fourier Transforms, and Integer Multiplication, all based on a machine architecture, the Cube Connected Cycles CCC, introduced by the authors in [10]. All of our designs matchn to within a constant factor, the known theoretical lower bounds [3], [4], [8] for area $\times$(time)^2 products.

Résumé

Nous présentons des schémas de circuits intégrés calculant des décalages cycliques, transformées de Fourier et multiplications entières fondés sur une architecture d'interconnection, les Cubes Connectés en Cycles CCC introduite par les auteurs [10]. Tous ces circuits réalisent, à une constante prêt, le compromis optimal [3],[4], [8] pour le produit (surface)$\times$(temps)^2.

Ce rapport a été produit dans le cadre d'une collaboration entre l'INRIA et l'Université d'Illinois, et paraît simultanément dans les deux organismes.

+ 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 I.R.I.A., Institut 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 methodologies 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. Thus, a meaningful figure-of-merit is represented by the area occupied by the total system, since area captures the complexity of both computation and data communication. As a result, the solution to a given computational problem involves the conception of an interconnection architecture, its layout, and the design of an algorithm for that architecture. It is clear that the traditional hardware-software antinomy disappears in VLSI technology.

Pioneering and fundamental work in the area has been done by Mead-Conway [1] and by Thompson [2], both as regards the development of a VLSI model of computation and in the design of computations (architecture+algorithms) for specific problems. As is typical of the methodology of concrete computational complexity, for a given problem and selected complexity measures one seeks both lower-bounds to these measures which hold for any realization in the computation model and upper-bounds by exhibiting explicit realizations which comply with the model. In spite of the relative novelty, the great interest of the topic is attested to by the additional contributions of Thompson [3], Abelson and Andreae [4], King and Leiserson [5], Guibas et al. [6], Brent and Kung [7,8], Savage [9], and Preparata-Vuillemin [10].

The VLSI computation models of Mead-Conway [1] and Thompson [2] are not significantly different. We briefly recall the latter one for the benefit of the reader. A VLSI computing system (or network) is viewed as a communication graph, whose vertices and edges are called nodes and wires, respectively. Nodes store and process local information ; wires transmit information between nodes. Nodes and wires are laid out on a grid of unit squares, where "unit" relates to the so-called "feature width", a basic parameter characterizing the resolution of current fabrication techniques. Wires have unit width and must be partitionable into no more than ν sets of non intersecting segments, where ν is the number of conducting layers. In this work, we assume that $\nu=2$, the almost universal two layers standard. It is assumed that a bit of information takes unit time to propagate from node to node, independently of the wire length (this implies that longer wires have more powerful drivers, of area proportional to the wire length) ; node processing time is absorbed into wire propagation time, and the total time for a given computation is the number of time units to execute it.

The usual metric selected for complexity is an area-time product $AT^{2\alpha}$, where A is the chip area, T is the computation time, and α is a real parameter satisfying $0\leq\alpha\leq 1$. This metric allows a flexible trade-off (based on α) between the production cost (area) and the incremental cost (time) of computation.

For several interesting problems, lower-bounds to the area-time product have been obtained. A crucial notion in obtaining such lower-bounds S is the minimal bisection width ω of a given communication graph $G=(V,E)$, which is defined as the smallest integer such that $\omega=|(u,v)\in E \ ; \ u\in V_1, \ v\in V_2\}|$, where $\{V_1,V_2\}$ is a partition of V with $|V_1|\leq|V_2|\leq|V_1|+1$. Thompson has shown [2] that for any n-node communication graph with minimal bisection width ω, $A\geq\omega^2/4$ (in unit squares). Therefore, lower-bounds to $AT^{2\alpha}$ are obtained by bounding the computation time T in terms of ω. In this paper we restrict ourselves to the following problems : cyclic shifts, integer multiplication, and radix-2 Discrete-Fourier-Transform. As regards cyclic shifts, it has been shown in [4] and [10] that $T\geq n/2\omega$ for any VLSI design which performs any cyclic shift of an array of n one-bit terms ; using a technique due also to Thompson [2,p. 72], this leads to the lower-bound $AT^{2\alpha}=\Omega(n^{1+\alpha})$. Since the ability to perform an arbitrary cyclic shift of an n-bit string is reducible to the multiplication of two (n/2)-bit integers, the lower-bound to $AT^{2\alpha}$ for cyclic shift becomes a lower-bound to integer multiplication [4] ; however, an independant proof of the latter -in a slightly more general model- has been supplied by Brent and Kung [8]. Finally, as regards radix-2 DFT, Thompson [2] has shown that

$AT^{2\alpha}=\Omega(n^{1+\alpha}\log^{2\alpha}n)$ for any communication graph which computes the DFT of n numbers each represented with O(logn) bits.

The purpose of this paper is to provide upper-bounds to the chosen metric of complexity for the problems mentioned above. The paper is organized as follows. Section 2 reviews the structure and the layout of a general computation network -called the cube connected-cycles [10]- which is remarkably suited to VLSI design. Section 3 and 4 discuss optimal designs based on the cube-connected-cycles ; specifically Section 3 considers a network for cyclic shifts, while Section 4 considers networks for integer multiplication and radix-2 Fast-Fourier-Transform.

2. THE CUBE-CONNECTED-CYCLES

The cube-connected-cycles (CCC) interconnection has been proposed in [10] as a general-purpose network of processing modules, suited for the implementation of various combinatorial algorithms. The specifications, the operating modes, and the performance of the CCC are now briefly reviewed.

An $h\times 2^s$ CCC-interconnection consists of 2^s cycles, indexed from 0 to 2^s-1 ; each cycle is the circular interconnection of h modules ($h\geq s$), indexed from 0 to h-1. Thus each module is addressed by a pair (ℓ,p), where ℓ and p are respectively the cycle and the module indices, and is denoted $M[\ell,p]$. Each module has three ports : F, B, and L,[1] and the connection is completely specified by

$$F(\ell,p) \leftrightarrow B(\ell,(p+1)\bmod h)$$
$$B(\ell,p) \leftrightarrow F(\ell,(p-1)\bmod h)$$
$$L(\ell,p) \leftrightarrow L(\ell+\varepsilon 2^p,p) \qquad \text{if } p\leq s-1 \text{ (unconnected if } p\geq s)$$

where $\varepsilon=1-2BIT_p(\ell)$ (here $BIT_p(\ell)$ is the coefficient of 2^p in the binary expansion of ℓ). In the hypothesis that modules reduce to nodes (i.e., they can be placed at vertices of a uniform grid of squares) and that wires are laid out on the grid, a layout of a 6×2^4 CCC is shown in figure 1. Notice that if all nodes of every cycle are ideally collapsed into a single node, the resulting set of nodes are connected as a binary s-dimensional cube (s-cube). This justifies the CCC denotation. (In the layout of figure 1, vertical and horizontal wires realise the cycle and cube connections, respectively).

---

(1) F, B, and L are respectively mnemonic for "forward", "backward", and "lateral".

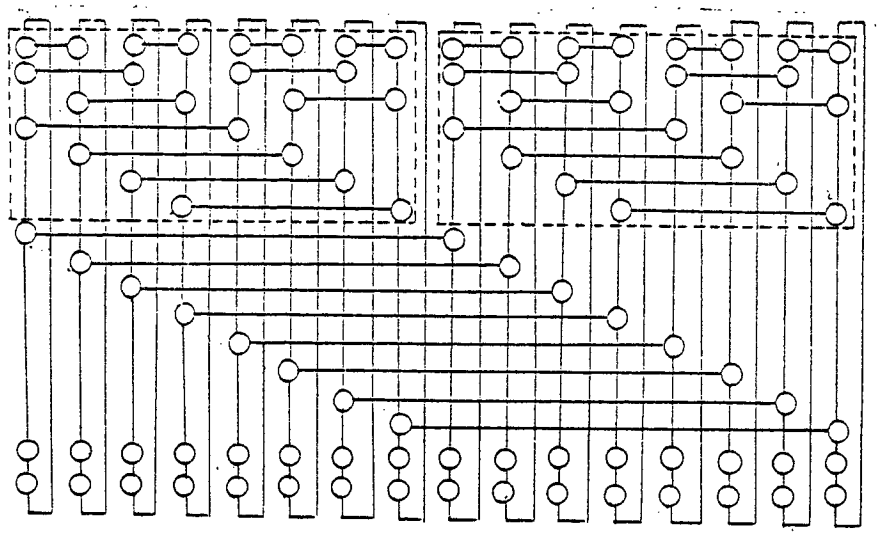

Figure 1. A standard layout for an $h \times 2^s$ CCC (h=6, s=4).

The dimensions of this s-cube are numbered 1,2,...,s, and the set of horizontal wires realizing dimension i are collectively denoted as sheaf i.

As a paradigm of computation, we consider the following type of algorithms. Abstractly, there are $n=2^r$ data items (operands), assigned addresses from 0 to 2^r-1 (or equivalently, each operand is addressed by an r-dimensional binary vector and assumed to be placed at the corresponding vertex of the r-cube). The algorithm is a sequence of $r=\log_2 n$ steps -each executable in parallel- with the property that at each of these steps each operand interacts with another operand, which is adjacent to it one a specified r-cube dimension ; specifically, either the i-th (ASCEND type algorithms) or the (r-i)-th dimension (DESCEND type algorithms) pertains to step i. (Typical instances of such algorithms are the Radix-2 Fast-Fourier-Transform and Bitonic merging of sorted sequences.) We see that these algorithms are supported by an r-cube interconnection.

We now show how algorithms of the type just described can be implemented on a $2^p \times 2^{r-p}$ CCC. Processing occurs in two consecutive phases. Making reference for concreteness to the ASCEND mode of operation, the first phase (referred to conventionally as LOWSHEAVES) pertains to r-cube dimensions 1,2,...,p (which are subsumed by the CCC cycle connection), while the second phase (denoted HIGHSHEAVES) pertains to r-cube dimension p+1, p+2,...,r (which ordely correspond to CCC sheaves 1,2,...,p).

The LOWSHEAVES phase emulates in general the r-cube behavior as follows. Since operand-interaction can occur in the cycles only between adjacent modules, it is necessary to successively realize the adjacencies corresponding to p-cube dimensions 1,2,...,p. The key permutation for this task is the perfect unshuffle [11], and it is shown in [10] that the required adjacencies are globally realizable in time proportional to 2^p, thereby showing that the first phase runs in time $O(2^p)$.

In the second phase the r-cube behavior is emulated as follows. The parallel step pertaining to r-cube dimension p+j can no longer be executed in one time unit ; however, using repeated circular shift within the cycles, each operand can be successively brought to reside for one time unit in that module in its cycle which is connected in sheaf j. Although this processing of all operands in a cycle on sheaf j now requires $O(2^p)$ time units, this computation can be pipelined (overlapped) with the analogous computations corresponding to all other sheaves, according to the scheme illustrated in figure 2. The sequence of steps during which a given sheaf is active is called the active phase of that sheaf (for example, steps 3-6 for sheaf 3 in figure 2).

Thus, the second phase also runs in time $O(2^p)$, and, when p is chosen equal to $\lceil \log_2(r-p) \rceil$, processing time on the CCC is O(logn). We see therefore that, by combining the principles of pipelining and parallelism, the CCC can emulate the cube with no significant loss of performance. In the sequel, we shall assume that

Sheaf \ Step	1	2	3	4	5	6	7
1	X	X	X	X			
2		X	X	X	X		
3			X	X	X	X	
4				X	X	X	X

Figure 2. Illustration of the pipelining of parallel computations. A "X" denotes a step at which a given sheaf is active.

the cycle length 2^p satisfies the limitations corresponding to

$$\lceil \log_2(r-p) \rceil \leq p \leq \lfloor r/2 \rfloor \qquad (1)$$

Finally, as concern the area of the layout, by referring to figure 1 we readily see that a $2^p \times 2^{r-p}$ CCC can be laid out on a chip of height $(2^{r-p}+2^p-r)$ and width $(2^{r-p+1}-1)$ (in the chosen units).

3. CYCLIC SHIFT

Let $T[0:2^r-1]$ be an array of $n=2^r$ one-bit <u>operands</u> (for any other operand length, both the area and the time will be multiplied by a constant).

We describe cyclic shifts to the left by $t<n$ positions of the operands of this array ; although dual implementations are possible, for concreteness, we describe a cyclic shift scheme which corresponds to an algorithm in the ASCEND class.

We now note the following property : Assuming that $T[0:2^{r-1}-1]$ and $T[2^r:2^r-1]$ have both been subjected to a left-cyclic-shift by $t \bmod 2^{r-1}$ positions, the desired final configuration is obtained by the following alternative exchanges :

<u>if</u> $(t>2^{r-1})$ <u>then</u> <u>foreach</u> $k:0\le k\le 2^{r-1}-t\bmod 2^{r-1}-1$ <u>pardo</u> $T[k]\leftrightarrow T[2^{r-1}+k]$ <u>odpar</u>
<u>else</u> <u>foreach</u> $k:2^{r-1}-t\bmod 2^{r-1}\le k\le 2^{r-1}$ <u>pardo</u> $T[k]\leftrightarrow T[2^{r-1}+k]$ <u>odpar</u> (2)

The proof of this property is straightforward (and it is basically supplied by the illustrations in figure 3). Property (2) can be reformulated as follows :

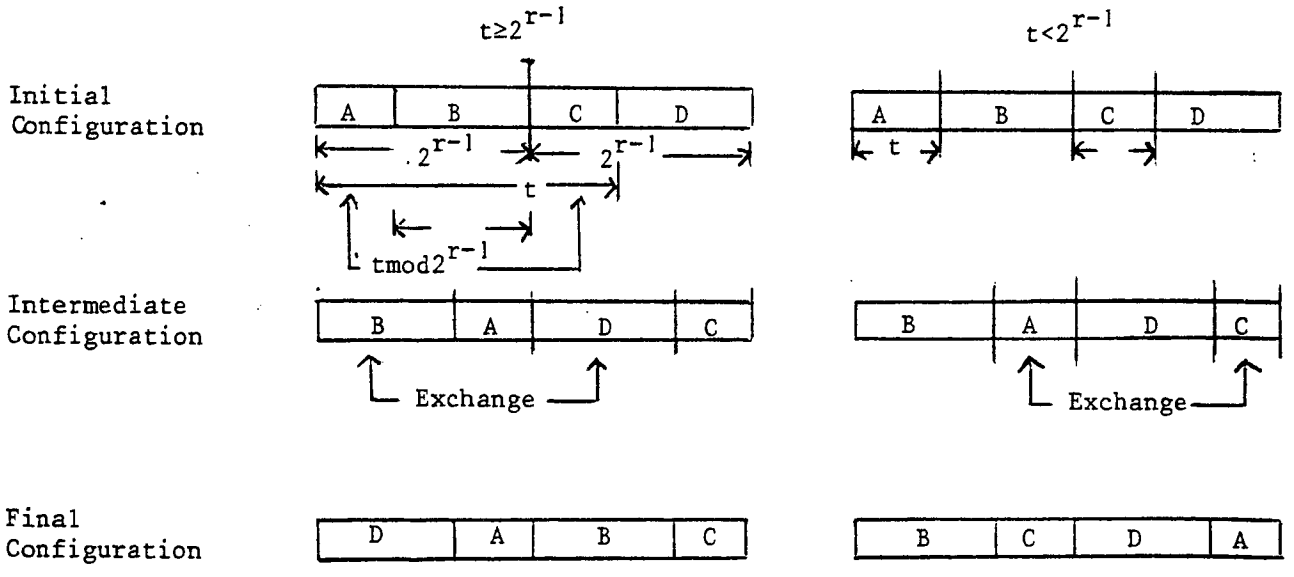

Figure 3. Illustration for the proof of Rule (2)

adjacent pairs on dimension r (i.e., pairs of the form $(j,j+2^{r-1})$) are partitioned into to sets, of respective sizes $t \bmod 2^{r-1}$ and $(2^{r-1}-t \bmod 2^{r-1})$ and the numbers of the former or of the latter set are exchanged depending upon whether $t<2^{r-1}$ of $t \geq 2^{r-1}$. Furthermore, since the two halves of the array $T[0:2^r-1]$ are treated in exactly the same way as regards dimensions $1,2,\ldots,r-1$, it follows by induction on decreasing j that the exchanges pertaining to dimension $1 \leq j \leq r$ are completely described by :

<u>if</u> $(t \bmod 2^j > 2^{j-1})$ <u>then</u> <u>for each</u> k: $k=2s.2^{j-1}+v$, $0 \leq v \leq 2^{j-1}-t \bmod 2^{j-1}-1$,
$0 \leq s \leq 2^{r-j}-1$ <u>pardo</u> $T[k] \leftrightarrow T[k+2^{j-1}]$ <u>odpar</u>

<u>else</u> <u>for each</u> k: $k=2s.2^{j-1}+v$, $2^{j-1}-t \bmod 2^{j-1} \leq v \leq 2^{j-1}-1$, $0 \leq s \leq 2^{r-j}-1$
<u>pardo</u> $T[k] \leftrightarrow T[k+2^{j-1}]$ <u>odpar</u> (3)

Notice that in the above rule (3) a crucial role is played by the parameter v, which defines the range of the pairs to be exchanged.

We now propose to implement the described cyclic shift operation on a CCC-like network. We select a $2^p \times 2^{r-p}$ CCC-interconnection so that dimension j of the previous abstract description (for $j>p$) corresponds to CCC sheaf $j-\phi$. We now observe the following facts :

(i) Refering to the standard layout of figure 1, in sheaf ℓ, for ℓ in the range $(1,r-p)$, adjacent pairs on the same orizontal line are characterized by the same value of the parameter v, as defined above. This means that all pairs on a given horizontal line of the layout will behave identically during the execution of the shift algorithm, whence the behavior of sheaf ℓ is completely specified by the behavior of modules $M[i,\ell-1]$ in cycles $i=0,1,\ldots,2^{\ell-1}-1$. Therefore, as regards sheaf ℓ, we may restrict ourselves to the subarray $T[0:2^{p+\ell-1}]$; in particular according to rule (3), this array is partitioned into $T[0:2^{p+\ell-1}-t \bmod 2^{p+\ell-1}-1]$ and $T[2^{p+\ell-1}-t \bmod 2^{p+\ell-1}:2^{p+\ell-1}-1]$, and either the first or the second of them exchanged on sheaf ℓ.

(ii) $t \bmod 2^{p+\ell-1} = q_\ell 2^p + t \bmod 2^p$, where $q_\ell = (t \bmod 2^{p+\ell-1})/2^p$. Thus modules $M[i,\ell-1]$ $(0 \leq i \leq 2^{\ell-1}-1)$ are divided into three sets, $\{M[0,\ell-1],\ldots,M[2^{\ell-1}-q_\ell-2,\ell-1]\}$, $\{M[2^{\ell-1}-q_\ell-1,\ell-1]\}$, and $\{M[2^{\ell-1}-q_\ell,\ell-1],\ldots,M[2^{\ell-1}-1,\ell-1]\}$, such that modules in the first and third set have fixed behavior during their active phases (either exchange or no-exchange), while $M[2^{\ell-1}-q_\ell-1,\ell-1]$ changes its behavior after the first $2^p-t \bmod 2^p$ steps of its active phase.

(iii) For any value of ℓ (i.e., for all sheaves) the quantity $(2^p-t \bmod 2^p)$ is independent of ℓ. This means that all modules with mixed behavior, the change of behavior (from exchange to no-exchange, or vice versa) occurs after the same number of steps during their active phases.

(iv) The LOWSHEAVES phase is void. Indeed the effect of a left cyclic shift by $t \bmod 2^p$ positions within each cycle is implicitly achieved by the timing of the exchanges in the HIGHSHEAVES phase. All that is needed initially for each cycle is a toward cyclic shift by one position so that $T[i2^p-1]$ resides in $M[i,0]$, for $i=0,1,\ldots,2^{r-p}-1$.

In summary the shift operation can be controlled as follows. Each module of the CCC is assigned two bits, b_1 and b_2, which respectively control the module operation during the first $(2^p-t \bmod 2^p)$ and the last $t \bmod 2^p$ steps of the module active phase. Bit b_j is set to 1 denote "exchange" and to 0 otherwise. The timing of the possible change of behavior (between step $(2^p-t \bmod 2^p)$ and step $(2^p-t \bmod 2^p+1)$ is controlled by the bit sequence $(0)^{2^p-t \bmod 2^p}\,1\,(0)^{t \bmod 2^p}$, which circulates in each cycle along with the operands. Thus we conclude that three control bits for module are sufficient, i.e., the cyclic shift operation has a finite-state module control.

Since the layout of figure 1 is used without modification, (and the nodes have constant area) we reach the following conclusions. For $p=\lfloor r/2 \rfloor$ we obtain a CCC whose computation time is $O(2^p)=O(\sqrt{n})$, i.e. a "slow" realization, which, however is optimal for the $AT^{2\alpha}$ metric $(0\le\alpha\le1)$: in fact, referring to the expression for the CCC height and width obtained at the end of Section 2, we have : $A=O(2^r)$ and $T=O(2^{r/2})$, whence $AT^{2\alpha}=O(n^{1+\alpha})$. If, on the other hand, we seek minimum computation time $O(\log n)$, the corresponding "fast" CCC is obtained by choosing $p=\lceil \log_2(r-p) \rceil \simeq \mathrm{llog}_2 n$.[1] In this case we obtain $T=O(2^p)=O(\log n)$ and $A=O((n/\log n)^2)$, whence setting $\alpha=1$ in $AT^{2\alpha}$, we obtain $AT^2=O(n^2)$, i.e. the network is optimal (notice that this occurs only for $\alpha=1$).

4. INTEGER MULTIPLICATION AND DISCRETE FOURIER TRANSFORM

The design of hardware multipliers is not a new problem in computer science. The classical shift and add method multiplies two n bits integers in time $T=O(n)$, within a circuit area $A=O(n)$. Furthermore, such a circuit is laid out in a

[1] The notation "llog" is used in this paper instead of the more common "loglog".

rectangle of constant width O(1), corresponding to a few wires, and height O(n) proportional to the number of bits. Although this multiplier does not meet the $AT=O(n^{3/2})$ and $AT^2=O(n^2)$ bounds of Brent-Kung [8], it proves to be useful in designing optimal VLSI for the DFT and binary multiplication.

4.1. CIRCUITS FOR DFT

In [10], we indicate that the radix-2 FFT algorithm can be implemented on the CCC, in time T proportional to the cycle size h. Each module of the machine performs one of seven tasks at a given time : it may transmit operands, in either direction, on one of its three communication lines, or it may be performing an internal operation. Internal operations, in this context, are linear combinations of the form $(U,V)\leftarrow(U+\alpha V,\ U-\alpha V)$ where U et V are two operands present in the module, and α is an appropriate power of $\omega=e^{\frac{2i\pi}{n}}$, a primitive n-th root of unity. In fact, the successive values of α vary with time, taking the form $\omega_0.\omega_1^T$, where ω_0 and ω_1 are appropriate powers of ω ; keeping the value of ω_1, in a special register allows to update α with a single multiplication. Internal operations can thus be computed in each module by a multiplication $\alpha\leftarrow\alpha.\omega_1$, another multiplication $V\leftarrow\alpha.V$, and a final add-substract step $(U,V)\leftarrow(U+V,\ U-V)$. Using a shift and add multiplier, and a few registers, such a <u>butterfly</u> module can be implemented on a chip of area O(logn) proportional to the number of bits used for representing each of the n inputs. As for the multiplier, this butterfly can be laid out in a rectangle of width O(1) and height O(logn). The n butterflies are then placed on the CCC of figure 1 as indicated in figure 4. It should be pointed out that in figure 4 the horizontal wires realizing the sheaf connection can be interleaved with the horizontal wires belonging to the butterfly modules; this interleaving at most doubles the height of the latter modules. Thus, the width of this new layout for a $2^p \times 2^{r-p}$ CCC (with $n = 2^p$) is $O(2^{r-p}) = O(\frac{n}{2^p})$; we shall now evaluate the height of the CCC.

As we see from figure 1, in each cycle there are two sets of modules: the set of the sheaf-modules, whose lateral port is used, and the possibly empty set of non-sheaf modules, which we shall consider first. Let H be the module height.

Each row of 2^{r-p} non-sheaf modules (there are $2^p-(r-p)$ such rows) can be laid out in an obvious way, using height H; the chip height used to

accommodate these rows is thus $(2^p-r+p)H$. As regards sheaf modules — although more compact placements are possible — we just assume the standard placement shown in figure 4, where sheaf i uses height $2(H+2^i)$ (2). Thus the (r-p) sheaves contribute height $2^{r-p+2}+2H(r-p)$, and the total chip height is $(2^{r-p+2}+H(2^p+r-p)) = O(\frac{n}{2^p} + 2^p\log n)$, since $r = \log n$ and $H = O(\log n)$. It follows that (provided 2^p, the cycle length, is upper-bounded by $\sqrt{n/\log n}$; we already know that $2^p \geq \log n$) the total CCC area is $A = O(n^2/2^{2p})$.

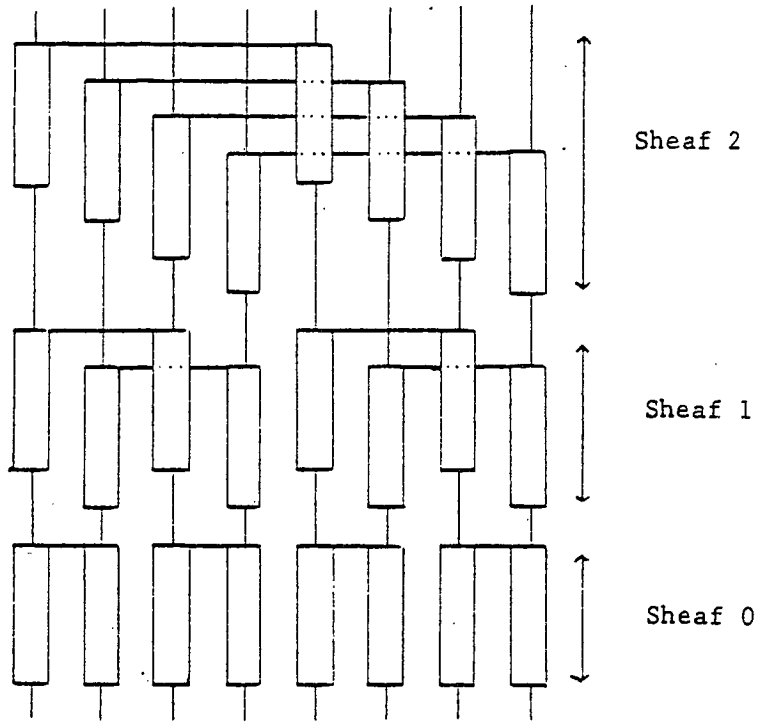

Figure 4. Placement of the butterflies (vertical rectangles) on the CCC.

Processing time is devoted to butterfly operations and operand transmission, each of which requires time $O(\log n)$. There are $2^p+\log(n/2^p)$ steps in the computation, thus total time is $T = O(2^p\log n)$. Since we have just observed that 2^p is bounded as $\log n \leq 2^p \leq \sqrt{n/\log n}$, for any choice of T within the bounds $\log^2 n \leq T \leq \sqrt{n\log n}$ we have just designed networks of area $A = O(n^2\log^2 n/T^2)$, thus achieving the lower-bound of Thompson [2].

Of independent interest is the product AT which, as observed by Thompson [2], is proportional to the <u>amount of energy spent in the computation</u>. In this regard, a lower bound $AT = \Omega(n^{3/2}\log n)$ is obtained in [2], for the computation of the DFT; from the lower bound $AT = \Omega(n^{3/2})$ obtained by Brent-Kung [8]

(2) The factor 2 is due to the interleaving of module wires and sheaf connections.

for binary multiplication, it is straightforward however to conclude that any DFT circuit satisfies the bound $AT = \Omega((n\log n)^{3/2})$. This last bound is met by a "slow" CCC design, with the choice of $2^P = O((n/\log n)^{1/2})$ for cycle size, yielding a circuit for the DFT with values $A = O(n\log n)$ and $T = O(\sqrt{n\log n})$.

The fastest circuit, among those described, is obtained for a cycle size $2^P = O(\log n)$; it uses an area $A = O(n^2/(\log n)^2)$ for computing DFT in time $O((\log n)^2)$.

4.2. CIRCUITS FOR INTEGER MULTIPLICATION

It is well-known [13] that the Discrete Fourier Transform allows to compute convolution products. From the preceding section, we know that we can construct circuits computing the convolution of two sequences of q integers, each integer being represented on $\log_2 q$ bits, with the following characteristics :

$$A.T^2=O(q^2\log q^2) \text{ for any } T \text{ such that } (\log q)^2 \le T \le \sqrt{q\log q}.$$

In [13], Schönage and Strassen show that, if $\omega=e^{2i\pi/q}$ and its powers are represented with $5\log_2 q$ bits, and the arithmetic is carried out with this precision, the approximate error in computing convolutions via FFT remains confined to the fractional parts of the terms involved.

In order to compute the product of two n bits integers, we divide each operand into $q = \frac{n}{\log n}$ blocks of length $\log n$ bits each, and we compute the convolution of the two sequences of q integers. The exact product is then found by "releasing the carries", in a straightforward manner.

Setting $q= \frac{n}{\log n}$ in the expression above for convolution shows that we have designed circuits for binary integer multiplication having the characteristics :

$$A.T^2=O(n^2) \text{ for any } T \text{ such that } (\log n)^2 \le T \le \sqrt{n}.$$

These circuits meet the $AT^2=\Omega(n^2)$ bound of Brent and Kung [8] ; in this class, the slow circuit corresponding to $T=O(\sqrt{n})$ also meets the $AT=\Omega(n^{3/2})$ lower bound [8].

Although circuits in section 4 appear to be too complex for being feasible on one chip in the present state of the technology, the sheer existence of, say, an $A=O(n)$, $T=O(\sqrt{n})$ multiplier raises interesting very practical prospects.

5. REFERENCES

[1] C. Mead and L. Conway, Introduction to VLSI Systems, Addison-Wesley, Reading, Mass. 1979.

[2] C.D. Thompson, "A complexity theory for VLSI", Ph.D. Thesis, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Penn., September 1979.

[3] C.D. Thompson, "Area-time complexity for VLSI," Proc. of the 11th Annual ACM Symposium on the Theory of Computing (SIGACT), pp. 81-88, May 1979.

[4] H. Abelson and P. Andreae, "Information transfer and area-time trade-offs for VLSI multiplication", to appear in the Communications of the ACM (1980).

[5] H.T. Kung and C.E. Leiserson, "Algorithms for VLSI processor arrays," Symposium on Sparse Matrix Computations, Knoxville, Tenn., Nov. 1978.

[6] L.J. Guibas, H.T. Kung and C.D. Thompson, "Direct VLSI implementation of combinatorial algorithms," Proc. Conference on VLSI Architecture, Design, Fabrication, Calif. Inst. of Techn., January 1979.

[7] R.P. Brent and H.T. Kung, "A regular layout for parallel adders," Research Report, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Penn., June 1979.

[8] R.P. Brent and H.T. Kung, "The area-time complexity of binary multiplication," Research Report, Department ofComputer Science, Carnegie-Mellon University, Pittsburgh, Penn., July 1979.

[9] J.E. Savage, "Area-time trade-offs for matrix multiplication and transitive closure in the VLSI model," Proc. of the 17th Annual Allerton Conference on Communications, Control, and Computing, October 1979.

[10] F.P. Preparata and J. Vuillemin, "The cube-connected-cycles : a versatile network for parallel computation," Proceedings of 20-th Annual IEEE Syposium on Foundations of Computer Science, Puerto Rico, October 1979.

[11] H.S. Stone, "Parallel processing with the perfect shuffle", IEEE Transactions on Computers, Vol. C-20, pp. 153-161 ; 1971.

[12] C.S. Wallace, "A Suggestion for a Fast Multiplier", IEEE Transactions on on Computers, Vol. 12, pp. 14-17, February 1965.

[13] A. Schönage and V. Strassen, "Schnelle Multiplikation grosser Zahlen," Computing, Vol. 7, pp. 281-292, 1971.