Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

P.M.B. Vitanyi

Circuit topology, signal propagation delay and the efficiency of VLSI circuits

Department of Computer Science Report CS-R8414 August

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

CIRCUIT TOPOLOGY, SIGNAL PROPAGATION DELAY AND THE EFFICIENCY OF VLSI CIRCUITS

PAUL M.B. VITANYI

Centre for Mathematics and Computer Science, Amsterdam

Using sound electronic principles, a signal propagation delay logarithmic in the length of a wire can be attained in VLSI circuits only if all wires in the layout have the same aspect ratio. This results in a penalty in wire surface area of the order of the square of the length of the wire. Thus, the global complexity of a VLSI circuit is affected. In particular, the complexity becomes very layout dependent. This effect will be truly pronounced in the emerging wafer scale integration technology. There are circuit topologies with the same function (Dictionary machine, Fast Fourier Transform) such that a circuit topology which is optimal under one delay assumption is suboptimal under another. Under constant signal propagation delay, or logarithmic delay, systolic search trees or fast permutation networks can be laid out to perform superior if we assume constant width wires. However, under logarithmic signal propagation delay with the required constant aspect ratio for wires the naive Mesh layout is superior over every tree layout with respect to both Area and Period. Similarly, for thus implementing the Fast Fourier Transform, the naive Mesh layout is superior over every layout for a fast permutation network like the Cube-Connected Cycles in Area, Area$\times$Period and the Area$\times$Execution Time. As a matter of independent interest, with a constant aspect ratio a for wires and using at most c layers, every layout for a complete N-node binary tree, so also the H-tree layout, takes Area $\Omega(N \log^{a/6c} N)$. With the wire aspect ratio and number of layers constants independent of N, it is impossible to layout a complete N-node binary tree using equal length wires.

1980 MATHEMATICS SUBJECT CLASSIFICATION: 68C25, 94C99

1982 CR CATEGORIES: B.7.0, F.2.3

KEYWORDS & PHRASES: very large scale integrated circuits (VLSI), wafer scale integration, logarithmic signal propagation delay, electronic principles, driving long wires, wire aspect ratio, wire length distribution, circuit topology, layout area, computational complexity, efficiency, H-tree layout, Dictionary machine, Cube-Connected Cycles

NOTE: This report will be submitted for publication elsewhere

Report CS-R8414
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009AB, Amsterdam, The Netherlands

1. Introduction

In a realistic complexity theory for the *very large scale integrated (VLSI)* electronic switching circuits, which now become possible, the treatment of signal propagation delay in long wires plays a crucial part. This seems truly the case on the level of the emerging *wafer scale integration.* In this technology in effect chips with 4 inch diameter are manufactured. On a scale like that, the signal propagation delay in long wires becomes a major factor. Synchronization requirements slow down the computation to a clocked switching time in the order of the delay in the longest wire. Such delays can be reduced by change of feature sizes or choice of material. The latter course is taken by the military sensitive investigation of very high speed integrated circuits (VHSIC). For instance, the use of semi-insulating Gallium-Arsenide and Silicon-on-Sapphire insulating substrate technologies allows switching times in the subnanosecond range [EWZL, YYC, TR]. New devices are afoot like the up-and-coming high-electron-mobility transistor (HEMT) [MS]. In MOS (metal oxide semiconductor) technologies metals like tungsten, molybdenum, titanium, tantalum and their silicides are considered to replace and complement polysilicon [SM]. The quest for speed on chips is no luxury; high speed is imperative in real-time applications.

In the computational models rampant in the literature, the eloquently justified assumptions concerning signal propagation delay in long wires range from constant delay (irrespective of the wire length) [Th1, BK1, BK2, Vu, Sa], via logarithmic [PRS, Th2, TR] and linear delay [BPP, CM] to signal propagation delay that is square in the length of wires [BPP, CM, Se]. In all of these models the width (or thickness) of wires is assumed to be a unit, depending on the minimal feature width of the underlying technology. Some of these models are more reasonable than others, while all of them may cover selected real life situations. Most papers cite [MR1, MC] as justification. However, recently [MR2] emphasized once again, that the constant delay assumption cannot be held in general at all, while the logarithmic delay assumption is incompatible with the constant wire width assumption. To achieve a propagation delay logarithmic in the length of the wire, electronic considerations show that all wires need have the same ratio between width and length, that is, the same *aspect ratio.* Thus, in multilayered chips now being manufactured, the communication wires are grouped in metal layers according to length. In a layer with a group of longer wires those wires are proportionally wider and thicker. The surface area occupied by long wires has to be proportional to the squares of their lengths in order to obtain logarithmic propagation delay. This fact heavily influences the resulting complexity theory for VLSI computation models that assume logarithmic propagation delay; in particular it makes the complexity more layout sensitive than previous assumptions. These considerations form the subject matter of the present investigation.

Logarithmic propagation delay in long wires is attractive in VLSI since it is well suited to hierarchically designed circuits like trees [MR1, MC, MR2], and to wire length distributions on actual chips [Do, Vi]. The discussion in [MR1, MC, Section 8.5] apparently has created confusion, in particular concerning the fact that logarithmic

propagation delay is reconcilable with approximately unit width wires. To dispel that misunderstanding we first recapitulate the pertinent electronic facts in the spirit of [MR2], but more precisely. See also [SM]. In Section 3 we interprete the basic electronics of Section 2 to imply that logarithmic signal propagation delay actually requires a constant aspect ratio for the wires. All in all, we can therefore distinguish assumptions of constant, logarithmic, linear or square propagation delay with constant wire width, and logarithmic propagation delay with a constant aspect ratio for the wires. In Section 4 we compare a Mesh topology and a complete binary tree topology for a specific function: the implementation of a systolic search tree under the five different assumptions. It turns out that, for logarithmic signal propagation delay with attending constant aspect ratio for wires, the naive Mesh layout is superior over every tree layout with respect to both Area and Period. In particular, we show that, with a constant aspect ratio a for the wires and using c layers, *every* layout of a complete binary tree with N nodes takes $\Omega(N \log^{a/6c} N)$ area,* so *also* the H-tree layout of [MR1, MC]. With the additional requirement that all wires have the same length, while the aspect ratio and number of layers are constants independent of N, such a layout is *impossible* for large enough N. Assuming constant width wires, only for constant delay can the systolic tree layout match the O(1) Period of the naive Mesh layout [SS]; for linear and quadratic delay even the Execution Time of every systolic tree layout matches or exceeds that of the naive Mesh layout. An application of this exercise is the implementation of a *Dictionary Machine* [ORS, SS] as explained briefly in the Appendix. In Section 5 we do a similar comparison between a Mesh layout and a fast permutation network layout for the Fast Fourier Transform. It appears that the naive Mesh layout performs better than every layout for a fast permutation network like the Cube-Connected Cycles [PV] in Area, Area×Period and Area×Execution Time under logarithmic propagation delay with constant aspect ratio wires. For an extensive discussion on how the propagation delay varies as a function of the wire length and how this influences overall circuit performance, in particular with respect to pipelined computations, see [PRS].

Related work. In [Do] a theoretical derivation and an experimental study of actual chip layouts shows that the number $f(i)$ of wires of length i in layouts tends to satisfy $f(i) = \lfloor ci^{-\lambda} \rfloor$ $(1 \leq i \leq L)$ and $f(i) \approx 0$ $(L < i)$. In [Vi] we investigate the effects of logarithmic propagation delay with a constant aspect ratio for the wires in layouts with such wire length distributions. While leaving free the topology, placement and routing in the layouts, considerations of wire length distribution alone shows that for some distributions the logarithmic signal propagation delay necessitates an exponential increase in layout area, which implies exponentially long interconnect wires. Thus the gain in greater relative propagation speed is completely lost by the greater distances the

---

* We use the Order-of-Magnitude symbols as follows:
$g(n) \in O(f(n))$ if there exists a positive constant c and $g(n) \leq c\,|f(n)|$ for all but finitely many positive integers n.
$g(n) \in \Omega(f(n))$ if there is a positive constant c and $g(n) \geq cf(n)$ for infinitely many positive integers n.
$\Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$.

signals have to travel. Apart from the fact that we require a constant aspect ratio for the wires to obtain a logarithmic propagation delay, we also need to attach *drivers* to drive the long wires. In [MR1, MC, MR2] and Section 2 it is shown that such drivers need an area proportional to the length of the wire. In [Ra] the worst-case effect of inserting these drivers in a basically fixed layout is investigated while assuming that all wires have the same unit wire width before and after insertion.

2. Electronic basics

The time it takes a minimum transistor to drive a wire of length L, width W and thickness H can be estimated as follows. The wire is assumed to have distance D_l to neighbouring layers and D_w to other wires in the same layer. If W_0 is the minimal width of a wire in the current technology, then the minimal transistor, consisting of a wire crossing, occupies area W_0^2. The total time T to drive a wire is approximated by:

$$T \approx (R_t + R_w)\, C_w \quad , \tag{1}$$

where R_t is the resistance of the minimum transistor, R_w the resistance of the wire and C_w its capacitance. The total time T can be thought of as the sum of the time T_d needed to drive a zero resistance wire of capacitance C_w, and the time $R_w\, C_w$ needed to transport the oppropriate charge from a zero resistance source. Since the resistance of a wire is proportional to its length and inversely proportional to its cross section we have:

$$R_w = \rho_w \frac{L}{WH} \quad , \tag{2}$$

where ρ_w is the resistivity of the considered wire material. The capacitance of a wire is inversely proportional to the distance of its neighbouring wires and layers, and proportional to the area of the side facing that neighbouring layer or wire:

$$C_w = \epsilon_w\, L\, (\frac{H}{D_w} + \frac{W}{D_l}) \tag{3}$$

where ϵ_w is a proportional constant consisting of the product of the permittivity of free space and the dielectric constant of the insulating material (usually SiO_2). Thus,

$$R_w\, C_w = \rho_w\, \epsilon_w \frac{L^2}{WH} (\frac{H}{D_w} + \frac{W}{D_l}) \quad . \tag{4}$$

This suggests a signal propagation time quadratic in L. However, the resistance R_t, of the minimum transistor, dominates in (1) for the magnitudes of L under consideration (smaller than, say, 1 meter). We can decrease that term by fitting a larger driver transistor to the wire. This transistor, in its turn, must be driven by the minimal transistor. Iterating this scheme, cf. [MC], we obtain a sequence of transistors, of which each next one is a factor α larger than the preceding one. The final transistor in the sequence should be large enough to drive the wire in a sufficiently short time. The time to drive a driver with capacitance C_2 by a driver with smaller capacitance C_1 is given by [MC]:

$$\tau \frac{C_2}{C_1} \ , \tag{5}$$

where τ is the time it takes a minimal transistor to charge the gate of another minimal transistor. If C_t is the capacitance of the minimal transistor then we require:

$$\#\,\text{drivers} = \log_\alpha \frac{C_w}{C_t} \ , \tag{6}$$

taking $T_d = \#\,\text{drivers}\,\tau\,\alpha$ time to charge the wire if it had no resistance. The capacitance of the minimum transistor is given by

$$C_t = \epsilon_t \frac{W_0^2}{D_0} \ , \tag{7}$$

where D_0 is the thickness of the gate insulator and ϵ_t is the product of the permittivity of free space and the dielectric constant of the gate insulator. Thus we can drive a zero resistance wire of capacitance C_w through a sequence of # drivers in time:

$$T_d = \alpha\,\tau \log_\alpha \frac{\epsilon_w D_0 L}{\epsilon_t W_0^2} \left(\frac{H}{D_w} + \frac{W}{D_l} \right) \tag{8}$$

From (1), (4) and (8) we obtain an expression for T.

$$T \approx \alpha\,\tau \log_\alpha \frac{\epsilon_w D_0 L}{\epsilon_t W_0^2} \left(\frac{H}{D_w} + \frac{W}{D_l} \right) + \rho_w \epsilon_w \frac{L^2}{WH} \left(\frac{H}{D_w} + \frac{W}{D_l} \right) \tag{9}$$

It is therefore clear that the signal propagation time heavily depends on the various dimensions and materials involved in the chip. In [MR2] it was observed that by keeping the derivatives, with respect to L, of the two terms in the righthand side of (9) balanced:

$$\frac{\alpha\,\tau}{L \ln \alpha} \approx 2 \rho_w \epsilon_w \frac{L}{WH} \left(\frac{H}{D_w} + \frac{W}{D_l} \right) \ , \tag{10}$$

T grows logarithmic in L. Viz.,from (9) we obtain by assumption of equality (10):

$$T \approx \frac{\alpha\tau}{\ln \alpha} \left\{ \ln \left[\frac{\epsilon_w D_0 L}{\epsilon_t W_0^2} \left(\frac{H}{D_w} + \frac{W}{D_l} \right) \right] + \tfrac{1}{2} \right\} \ . \tag{11}$$

Having the minimum transistor drive the total wire outright, we obtain from (1), (4), (5) and (7):

$$\begin{aligned} T &\approx \tau \frac{C_w}{C_t} + C_w R_w \\ &= \left(\frac{\tau D_0}{\epsilon_t W_0^2} + \rho_w \frac{L}{WH} \right) \epsilon_w L \left(\frac{H}{D_w} + \frac{W}{D_l} \right) \ . \end{aligned} \tag{12}$$

The above shows that we can reduce the signal propagation time by employing new materials with more favorable characteristics, like Gallium-Arsenide and Silicon-on-Sapphire technologies [YYC]. We can also change the size of the wires and the interwire- and interlevel separation. Thus, for long interconnect wires extra layers with wider and thicker wires are used.

3. Logarithmic Delay Assumption, Constant Aspect Ratio for Wires, Area, Time, and Implementation Details

Logarithmic Delay and Constant Aspect Ratio. Under assumption (10) we can obtain a logarithmic signal propagation delay by, all other things being equal, maintaining:

$$L^2\left(\frac{1}{W\,D_w}+\frac{1}{H\,D_l}\right)=\text{constant}\ , \tag{13}$$

rather than by just keeping L^2 proportional to WH as in [MR2]. Keeping the interwire distance proportional to the wire width, and the interlayer distance proportional to the wire height, we observe that if W, H and L are kept in proportion the desired logarithmic propagation delay is attained. (Note that we cannot reach this effect by keeping the wire width the same but using very 'tall' wires or vice versa.) The *aspect ratio* of a wire is the quotient of its length and width. To obtain a logarithmic signal propagation delay we thus need the fixed constant aspect ratio following from (10) and (13) for all wires in the layout. In designing a high speed layout we therefore need to install drivers to drive the long wires and to design all wires with a constant aspect ratio. The area taken by such a driver is linear in the length of the wire [MR2]: the minimal transistor occupies area W_0^2, the next driver area αW_0^2, and so on for $\log_\alpha l$ terms for an l-length wire. The total driver area for an l-length wire becomes $W_0^2(l-1)/(\alpha-1)$. This area is required at the lowest silicon layer of the chip; the long interconnect wires are executed in the upper metal layers.

Area and Length. The area for a VLSI layout is expressed in A area units. The area unit is the square of the basic length unit which is the feature width of the underlying technology. This is currently $4{\cdot}10^{-6}$ - $10{\cdot}10^{-6}$ meter and is expected to decrease to $1.5{\cdot}10^{-6}$ - $4{\cdot}10^{-6}$ meter in the near future [SM, TR, YYC]. Below we make the following assumptions about the Area, for a VLSI layout [PRS]. Such assumptions need to be made in particular with respect to lower bound arguments.

1. The *Area* is taken to be the area of the smallest convex region enclosing the layout.
2. There is a *cross-over* constant $c>0$ such that no unit circle encloses points of more than c different edges (wires) or nodes (components or transistors).

(In case we allow an unlimited amount of cross-over, we should consider the worst-case 'area $\times$ cross-over' product instead of the area. Effectively, we then consider 3-dimensional 'layered' chips which is outside the scope of this paper but is used in [Vi].)

Time. The *Execution Time* of a problem instance is the time elapsed between the entering of the first bit of the problem instance in the circuit and the leaving of the last

bit of the answer from the circuit. In *pipelined* and especially in *systolic* computations [MC] the *Period* is important. The Period is the time elapsed between entering the first bit of a problem instance and the first bit of a next problem instance. In 'moving belt' type computation the Period can be substantially less than the Execution Time. Below the Period of a (systolic) computation appears to be more sensitive for the propagation delay assumption than the overall Execution Time. If the signal propagation delay depends on the length of the wire the signal has to traverse, then the *minimax edgelength* in the layout will determine the Period in a systolic network. The minimax edgelength (or wirelength) $e(.)$ of a layout for a given circuit is the minimum over all layouts, implementing the circuit, of the length of the longest wire in such a layout. See also [PRS].

Implementation Details of Area and Time. We may assume that the circuits are laid out on a Manhattan grid. In [Le1] algorithms are presented to embed easily separated graphs efficiently in grids. The considerations below assume that the processing elements have unit area and the links between them have unit bandwidth. This view captures the underlying communication structure. This leaves free the precise implementation. For example, to communicate a word of k bits between two processing elements, one can either use a link of bandwidth k and one cycle or use a link of bandwidth 1 and k cycles. Let each of the P processing elements actually fit in area U and let the total area used by the bandwidth 1 links be L. Let W be the bandwidth of the precise implementation, e.g., word-parallel or word-serial communication. Using methods of [Le1], a rough upper bound on the total actual chip area is given by $A_p \in O(PU)$ plus $A_w \in O(LW^2)$, where A_p is the area taken by the processing elements and A_w is the area for the wires, for both the search tree and the Mesh network. We follow the normalization custom generally adhered to, so by a layout *Area* A we mean $P+L$, that is, we normalize the processing elements to unit area and the wires to unit bandwidth. "Once the issues such as bandwidth of links are resolved this [estimate] gives the basis for a bound on the actual chip area," [ORS]. Concomitant with the assumption of unit bandwidth, we also assume that each communication between processors concerns a unit (bit). Once the bandwidth and sizes of messages are resolved, the estimates give a basis for the actual chip times.

4. The Case of the Systolic Search Tree

In [ORS] a realization of a *Dictionary machine* on the basis of a *systolic search tree* [Le1] is described. This requires the implementation of two complete binary trees, or of X-trees [Le1, ORS], back-to-back. If we allow a constant amount of layers for the VLSI layout it suffices to consider the implementation of a complete binary tree [Le1]. For a dictionary capacity N there exists a systolic search tree of Area $O(N)$. The dictionary operations *SEARCH*, *INSERT* and *DELETE* are processed pipeline fashion. With the assumption of a constant signal propagation time, as in [Th1], the Period of an operation is $O(1)$ and the Execution Time $O(\log N)$. Under the linear delay time assumption of [CM], while retaining the Area $O(N)$, the Period is $O(\sqrt{N}) \cap \Omega(\sqrt{N}/\log N)$, cf. below,

and the Execution Time $O(\sqrt{N})$. In [SS], however, the authors show that the necessary systolic search tree can be implemented in a simple systolic Mesh architecture, of $O(N)$ Area, O(1) Period and $O(\sqrt{N})$ Execution Time, under the linear delay assumption. The exhibited architecture has also a simpler layout and uses smaller absolute area. In this section we want to investigate how the two topologies for implementing the systolic search tree compare under five different signal propagation delay assumptions. In an Appendix we briefly explain the Dictionary machine as proposed by [ORS] and later [SS].

The five propagation delay assumptions identified in Sections 1, 2 and 3 are:

(A) *Constant delay:* the wire has unit width and the propagation time is independent of its length.

(B) *Logarithmic delay:* the wire has unit width and the propagation delay is logarithmic in the length of the wire.

(C) *Linear delay:* the wire has unit width and the propagation delay is linear in the length of the wire.

(D) *Quadratic delay:* the wire has unit width and the propagation delay is quadratic in the length of the wire.

(E) *Logarithmic delay & constant aspect ratio.* the propagation delay is logarithmic in the length of the wire and all wires in the layout have a constant aspect ratio. Thus, wires occupy an area square in their lengths.

Since in the naive layout of the *Mesh* topology the wires do not need to exceed a constant fixed length, its performance is invariant under a change of the propagation delay assumption; it *always* has, for a Dictionary of N items:

- Area $O(N)$;
- Period O(1);
- Execution time $O(\sqrt{N})$ for a command.

For the complete binary tree topology matters are different. Each layout of a complete binary tree with N leafs contains wires of length about $\sqrt{N} / \log N$. This is easily seen as follows. If nodes cannot lie on top of each other, then the total Area A covered by the layout is at least $2N$ times the unit area covered by a node. So we can find two points p and q in the layout which are at least $\sqrt{2N}$ units apart. p and q can be nodes or locations on a wire. To go from p to q along the edges of the tree cannot cause us to traverse more than $2 \log N$ edges. Hence, there is an wire in the layout of length $\sqrt{2N} / 2 \log N$ units. Note that this holds for *every* layout which satisfies the restriction of not having more than one (or a constant number of) nodes on top of each other. Similarly, if we have a layout covering A Area, then there must be a wire in the layout of length $\sqrt{A} / 2 \log N$; for instance, if $A \in \Omega(N \log N)$ then the minimax edgelength for a layout is $e(N) \in \Omega(\sqrt{N / \log N})$. Moreover, there is a path from the root to a leaf of total length of at least $\Omega(\sqrt{A})$. When the leafs are constrained to be on the convex circumference of the layout these lower bounds can be considerably increased

[PRS].

The area occupied by the wires in the systolic tree stays the same under assumptions (A) through (D), since the wire width is one unit under all of them. Under assumption (E) the situation is different. Recall that the H-tree layout for a complete binary rooted tree [MR1] achieves Area less than $4N$ for an N-node complete binary tree, under the unit wire width assumption. Now assume (E), so all wires in the layout have a constant aspect ratio.

Upper bound. Analyse the area occupied by an H-tree layout with N leaves and no overlap. Let the ratio between the length of the wires at two consecutive levels be α. That is, the ratio between the length of a lower level wire and a wire on the next level above is $0<\alpha<1$. Let the *aspect ratio* of the wires be a, that is, the quotient of width and length is a. Let $N = 2^m$. Recall the familiar H-tree layout with constant width wires for complete binary trees as depicted in, for instance, [MC, Le1]. Considering that layout, it is not to difficult to see that, with constant aspect ratio a $(0<a<1)$,

$$\alpha^{-k} \geqslant a\alpha^{-k-1} + 2\alpha^{-k+2} \ ,$$

for each level k between 1 and m, suffices to layout the H-tree compactly with no overlap of wires and nodes. Consequently we obtain

$$0 < a \leqslant \alpha(1-2\alpha^2) \ ,$$

and, for $a, \alpha > 0$,

$$0<\alpha<\frac{\sqrt{2}}{2} \ \ \& \ \ 0<a\leqslant\frac{2}{3}\sqrt{\frac{1}{6}} \ .$$

Note that in the limit for $\alpha \to \sqrt{2}/2$ we have that $a \to 0$. For given α and a in the appropriate ranges, an *upper bound* $A(2^m)$ on the total wire area plus node area for the H-tree layout is computed as follows:

$$\begin{aligned} A(2^m) &\approx a\sum_{k=0}^{\infty} 2^k \alpha^{2k-2m} + 2^m \\ &= \frac{a\alpha^{-2m}}{1-2\alpha^2} + 2^m \ . \end{aligned}$$

Therefore,

$$A(N) \approx CN^{-2\log_2\alpha} + N \ ,$$

with $C = a/(1-2\alpha^2)$. From the relation between a and α it follows that $C \leqslant \alpha$ for $0<\alpha<\sqrt{2}/2$. Setting $a = (1-2\alpha^2)/2$, so $1/2 \leqslant \alpha < \sqrt{2}/2$ and therefore $0 < a \leqslant 1/4$ and $C = 1/2$, yields

$$A(N) \approx \tfrac{1}{2}N^{1-\log_2(1-2a)} + N \ .$$

Therefore, for each $\epsilon>0$ there is an $\alpha < \sqrt{2}/2$ such that $A(N) \in O(N^{1+\epsilon})$. Since a must be greater than 0 this ϵ remains greater than 0 as well.

For $\alpha \to \sqrt{2}/2$ (so the aspect ratio a of the wire becomes very small):

$$A(2^m) \to a \sum_{k=0}^{m} 2^k \alpha^{2k-2m} + 2^m$$

$$= am\alpha^{-2m} + 2^m$$

$$= aN \log N + N \in \Theta(N \log N)$$

Lower bound. Let $N = 2^m$. Let a be the constant aspect ratio of the wires. Let c be the maximal amount of cross-over. For each i, $1 \leqslant i \leqslant m$, let A_i be the *minimum layout Area* for a complete binary tree T_i with 2^i nodes. Imagine, for the sake of the argument, a virtual layout for T_m, such that each maximal subtree determined by a node of T_m takes minimal area. Selecting wires from the maximal lengths paths in these subtrees, we sum their areas while taking care that each such wire is counted only once.

Claim 1. Let T_i be a complete binary tree of $i+1$ levels with the root at level i and the leaves at level 0. If the minimal layout Area of T_i is A_i then there is a path in T_i of at most $2i$ edges and of length at least $\sqrt{A_i/c}$ in the layout. The area taken by this path must therefore exceed $aA_i/2ic$, where a is the aspect ratio of the wires and c the maximal amount of cross-over.

Proof. By arguments concerning the diameter of the smallest convex area containing T_i it is easy to see that there is a path of length $\sqrt{A_i/c}$ in the layout with $1 \leqslant j_i \leqslant 2i$ wires. The sum of the area of the wires in such a path is therefore $\Omega(aA_i/cj_i)$. □

Claim 2. Let T_i be a complete binary tree of $i+1$ levels with the root at level i and the leaves at level 0. A path through the tree is disjoint from at least 2 maximal complete binary subtrees of $j+1$ levels with roots at level j of T_i for all j, $i-2 \geqslant j \geqslant 0$, such that all $2(i-1)$ complete binary subtrees concerned are pairwise disjoint. (This is easy to verify from a simple picture.)

So by Claims 1 and 2, we can give a lower bound on the Area A_i of T_i by adding the minimal possible area of a $\sqrt{A_i/c}$-length path in T_i to twice the sum for j ranges from 0 through $i-2$ of the areas of subtrees T_j:

$$A_i > \frac{aA_i}{2ci} + 2\sum_{j=0}^{i-2} A_j.$$

Unfolding this inequality we obtain

$$A_i > \sum_{j=0}^{i} \frac{aF_{i-j}A_j}{2cj} \quad ,$$

with F_{i-j} the $(i-j)$-th element of the Fibonacci-like sequence generated by the recurrence relation $F_i = F_{i-1} + 2F_{i-2}$ with $F_0 = 1$ and $F_1 = 0$. Therefore,

$$F_i = \frac{2^i + 2(-1)^i}{3} \approx \frac{2^i}{3} \quad .$$

Substituting $A_i = g(i)2^i$ in the inequality above yields

$$g(i) > \frac{a}{6c} \sum_{j=0}^{i} \frac{g(j)}{j}$$

which is satisfied for $g(i) \in \Omega(i^\epsilon)$ with $\epsilon > a / 6c$. Hence,

$$A_m \in \Omega(N \log^{a/6c} N) \ .$$

Crudely derived, considering only volume without considerations of placement and routing, this lower bound is yet nonlinear and reflects both the necessary influence of the aspect ratio a and the cross-over coefficient c.

Under assumptions (A) through (D) the *Area*, *Period* and total command *Execution Time* are easy to compute. They are summarized in Table 1 below. Let again A_m be the minimal Area needed for the layout of a complete binary tree T with $N = 2^m$ nodes. The Period is computed from the minimax wire length $e(N) \in \Omega(\sqrt{A_m} / \log N)$. The Execution Time follows by summing the delays along an $\Omega(\sqrt{A_m})$-length path from the root to a leaf. Before we have argued that there is such a path with at most about $\log N$ nodes to determine the minimax edge length which determines the Period under the different signal propagation delay assumptions. To determine the Execution Time one also needs to establish that there are $\sqrt{A_m}$-length paths with at least about $\log N$ nodes. Consider a complete binary tree T of depth $\log N$ and the $1 + \sqrt{N}$ subtrees resulting from slicing the edges from the $\lfloor (\log N) / 2 \rfloor$-th level to the next level in T. Note that in T there can be only $\sqrt{N}$ nodes within a $\lfloor (\log N) / 2 \rfloor$ edge-length path from each other. Now divide A_m into approximately $\sqrt{N}$ equal size nonoverlapping squares. Then two such squares, which are in the order of the diameter of the layout Area A_m apart, must harbor nodes between which there is a node disjoint path with $\Omega(\log N)$ wires with a total length of $\Omega(\sqrt{A_m})$.

Delay	Area	Period	Execution time
Constant	$\Theta(N)$	$\Theta(1)$	$\Theta(\log N)$
Logarithmic	$\Theta(N)$	$\Theta(\log N)$	$\Theta(\log^2 N)$
Linear	$\Theta(N)$	$\Omega(\sqrt{N}\ /\log N)$	$\Omega(\sqrt{N})$
Square	$\Theta(N)$	$\Omega(N\ /\log^2 N)$	$\Omega(N\ /\log N)$
Log.&Asp.Ratio	$\Omega(N\log^{a\ /6c} N)\cap O(N^{1-\log_2(1-2a)})$	$\Theta(\log N)$	$\Theta(\log^2 N)$

Table 1. Minimal area with unequal length wires. Here a is the wire aspect ratio and c is the cross-over coefficient or number of layers.

If we want to synchronize then it may be preferable to have layouts with only *equal length* wires. Under the constant wire width assumption, the least such wire length for a layout of a complete binary tree is $N\ /\log^2 N$ with simultaneous least Area of $\Theta(N^2\ /\log^2 N)$ [PRS]. Table 2 summarizes the effect of the requirement of equal length wires on layouts of a complete binary N-node tree.

Delay	Area	Period	Execution time
Constant	$\Theta(N^2\ /\log^2 N)$	$\Theta(1)$	$\Theta(\log N)$
Logarithmic	$\Theta(N^2\ /\log^2 N)$	$\Theta(\log N)$	$\Theta(\log^2 N)$
Linear	$\Theta(N^2\ /\log^2 N)$	$\Theta(N\ /\log^2 N)$	$\Theta(N\ /\log N)$
Square	$\Theta(N^2\ /\log^2 N)$	$\Theta(N^2\ /\log^4 N)$	$\Theta(N^2\ /\log^3 N)$
Log.&Asp.Ratio	$\Omega(aN^3\ /\ c\log^4 N)$	$\Omega(\log N)$	$\Omega(\log^2 N)$

Table 2. Minimal area with equal length wires, with a the wire aspect ratio and c the cross-over coefficient or number of layers.

With respect to wires with a constant aspect ratio, viz. the row row of the table, notice that under the requirement of unique wire length $e(N)$ for all wires, cross-over number c

(number of layers) and an aspect ratio a, the following relations have to hold for any complete binary N-node tree layout.

$$\frac{ae(N)^2N}{c} \leqslant A(N) \leqslant 4e(N)^2\log^2 N \quad .$$

Viz., on the one hand the Area must accommodate all wires, on the other hand the diameter of the layout cannot exceed the length of the longest path ($2\log N$ edges). Therefore,

$$\frac{a}{c} \leqslant \frac{4\log^2 N}{N} \quad ,$$

and, for fixed constant a and c independent of N, the desired layout is impossible for large enough N.

In any case, the Mesh is always superior in both Area and Period (under constant delay ex equo) over the tree, especially so in the case of equal length wire layouts.

5. The Case of the Cube-Connected Cycles

In [Th2] Thompson gives 9 designs for VLSI layouts that compute the N-element Fourier Transform. Below we compare the complexity of two of these, the Mesh network and the Cube-Connected Cycles network. For more detail see the reference. The Mesh network consists of a mesh of N processors formed by $\sqrt{N}$ rows and $\sqrt{N}$ columns fitted with word-parallel interconnections. This is essentially the ILLIAC IV architecture, with the difference that each processor in the Mesh is capable of running its own program. Under the constant wire width assumption, the total Area of the Mesh is $O(N\log^2 N)$, since there are N processors of $O(\log^2 N)$ area each. The processors are connected by constant length wires. The processors are laid out with a square aspect ratio so that the internal processor wires have length $O(\log N)$. The Mesh can perform an N-element FFT in $O(\log N)$ steps of computation, the routing accounts for $O(\sqrt{N})$ additional operations [Th2]. Under *all* delay assumptions (A) -(D) we therefore obtain:

- Area $O(N\log^2 N)$;
- Period $O(\sqrt{N})$;
- Execution Time $O(\sqrt{N})$.

Under delay assumption (E) only the internal wires of the processors are widened to match the length; the interprocessor wires all are of constant length. Thus the processor area increases up to $\log^3 N$, yielding a total Area of $O(N\log^3 N)$. The Period and Execution Time do not change.

An important class of networks in VLSI are fast permutation networks like the Fast Fourier Transform network, the Shuffle-Exchange network and the Cube-Connected Cycles (CCC) network. We shall consider the CCC network [PV] in more detail. For each positive integer k consider the k-dimensional hypercube C_k. The set of k-length binary strings can be used to name the 2^k corners of H_k. Each corner is connected by an

edge with the k corners which are named by strings one bit different. Thus, the nodes have degree k. By replacing each corner node by a k-length cycle, and attaching each of the k edges connected to the original node to a particular node of the replacing cycle, the degree of no node in the new network exceeds 3. The resulting network C_k has $N = k2^k$ nodes and $3k2^{k-1}$ edges. A CCC interconnection for N cells is capable of performing an N-element FFT in $O(\log N)$ multiply-add steps. Under the constant wire width assumption each cell takes $O(\log N)$ area [Th2]. So all cells together take $O(N \log N)$ area. However, the area taken up by wires dominates, as is seen as follows. The *minimum bisection width* of a graph is the number of vertices which must be cut to divide the graph in two subgraphs with an equal (give or take one) number of vertices. Since CCC's can realise an arbitrary permutation in $O(\log N)$ communication steps, and between two halves of a CCC $\Omega(N)$ data items may need to be swapped, the minimum bisection width of a CCC network is $\Omega(N / \log N)$. Since the area of a layout (with wires of constant width) is at least the square of the minimum bisection width, the Area A_N of an N-node CCC network is $A_N \in \Omega(N^2 / \log^2 N)$ with constant width wires. Moreover, this lower bound can be matched by an upper bound of the same order of magnitude [Th2]. Under the logarithmic delay assumption (E) we can now reason as follows.

Lower Bound. Since there are $3N / 2$ wires and N nodes of size $\log N$, and under the constant wire width assumption the total densely packed Area of the layout is $\Omega(N^2 / \log^2 N)$, the *average* length of a wire is $\Omega(N / \log^2 N)$. Consequently, under the logarithmic delay assumption with a constant aspect ratio for all wires in the layout, the Area A_N of the N-node CCC layout is bounded below by the product of the number of wires and the area occupied by the average wire. Taking the wire aspect ratio a and the cross-over number c to be fixed constants below, we obtain

$$A_N \in \Omega(N^3 / \log^4 N) \ .$$

According to [Th2], the time used by the CCC consists for both Period and Execution Time of two components: $\log N$ multiply-add steps in the processing nodes and $\log N$ routing steps which move O(1) words over each interprocessor connection. It is assumed that each multiply-add calculation of a processing node takes $O(\log N)$ time under (A), (B) and (E). The Period for the CCC is also its Execution Time. This time is for assumptions (A), (B), (E) dominated by the multiply-add steps, and for assumptions (C), (D) by the routing steps. The time for a routing step is related to the minimax edgelength of the CCC topology. Since we can get from each node to each other node in $2\log N$ edges, and there are parts of the layout which are $\sqrt{A_N}$ apart, the minimax edgelength $e(N)$ is at least $\sqrt{A_N} / 2\log N$, so with constant width wires

$$e(N) \in \Omega(N / \log^2 N) \ ,$$

and with constant aspect ratio wires

$$e(N) \in \Omega(N^{1.5} / \log^3 N) \ .$$

Since the degree of each node in the CCC network is 3, we can argue similarly to the case of tree layouts, that there is an $\Omega(\log N)$ edge-lengths path in each layout for C_k of length $\Omega(\sqrt{A_N})$, with A_N the minimum layout Area. The minimax edgelength under constant aspect ratio wires will not significantly influence the Period or Execution Time under (E), since the same order of magnitude would result under any polynomial in N. We summarize the results on the CCC network in Table 3 below.

Delay	Area	Period	Execution time
Constant	$\Theta(N^2/\log^2 N)$	$\Theta(\log^2 N)$	$\Theta(\log^2 N)$
Logarithmic	$\Theta(N^2/\log^2 N)$	$\Theta(\log^2 N)$	$\Theta(\log^2 N)$
Linear	$\Theta(N^2/\log^2 N)$	$\Omega(N/\log N)$	$\Omega(N/\log N)$
Square	$\Theta(N^2/\log^2 N)$	$\Omega(N^2/\log^3 N)$	$\Omega(N^2/\log^3 N)$
Log.& Asp.Ratio	$\Omega(N^3/\log^4 N)$	$\Theta(\log^2 N)$	$\Theta(\log^2 N)$

Table 3. The CCC performance with minimum Area.

Consequently, the Mesh is superior for the Area, the Area×Period and the Area×Execution Time measures.

The inspiration for the present investigation stems from a lecture by Martin Rem in the 1980/1981 Computer Science Colloquium "Complexity and Algorithms" at the (former) *Mathematisch Centrum*. Talks with Andries Brouwer, Aart Blokhuis, Lambert Meertens and Sape Mullender have improved this work.

REFERENCES

BPP Bilardi, G., M. Pracchi & F.P. Preparata, A critique and an appraisal of VLSI models of computation. In: Kung, H.T., B. Sproull & G. Steele (Eds.), *VLSI Systems and Computations*. Springer verlag, Berlin, 1981, 81-88.

BK1 Brent, R.P. & H.T. Kung, The chip complexity of binary arithmetic, Proc. 12th ACM Symposium on Theory of Computing, 1980, 190-200.

BK2 Brent, R.P. & H.T. Kung, The area-time complexity of binary multiplications, *J. ACM* **28** , 521-534.

CM Chazelle, B. & L. Monier, A model of computation for VLSI with related complexity results, Proc. 13th ACM Symposium on Theory of Computing, 1981, 318-325.

Do Donat, W.E., Wire length distribution for placement of computer logic, *IBM J. Res. Develop.* **25** (1981) 152 - 155.

EWZLEden, R., B.M. Welch, R. Zucca & S.I. Long, The prospects for ultra-high speed VLSI GaAs digital logic, *IEEE Journal of Solid-State Circuits,* SC-14 (1982) 221 - 239.

Le1 Leiserson, C.E., *Area Efficient VLSI Computation.* The MIT Press, 1982.

Le2 Leighton, F.T., *Complexity Issues in VLSI.* The MIT Press, 1983.

MC Mead, C. & L. Conway, *Introduction to VLSI Systems.* Addisson-Wesley, 1980.

MR1 Mead, C. & M. Rem, Cost and Performance of VLSI Computing structures, *IEEE J. on Solid State Circuits,* Vol. SC-14, 1979, 455 - 462.

MR2 Mead, C. & M. Rem, Minimum propagation delays in VLSI, *IEEE J. on Solid State Circuits,* vol. SC-17 (1982) 773 - 775. Correction: *Ibid,* (1984) 162.

MS Morkoc, H., and P.M. Solomon, The HEMT: a superfast transistor, *IEEE Spectrum,* February, 1984, 28 - 35.

ORS Ottmann, T., A.L. Rosenberg & L.J. Stockmeyer, A dictionary machine (for VLSI). Res. Rept. RC 9060, IBM Th.J. Watson Res. Center, Yorktown Heights, NY, 1981.

PRS Paterson, M.S., W.L. Ruzzo & L. Snyder, Bounds on the minimax edge length for complete binary trees. 11th ACM Symposium on Theory of Computing, 1981, 293 - 299.

PV Preparata, F., and J. Vuillemin, The Cube-Connected Cycles: a versatile network for parallel computation, Proceedings 20th IEEE Annual Symposium on Foundations of Computation, 1979, 140 - 147.

Ra Ramachandran, V., On driving many long lines in a VLSI layout, Proceedings 23rd IEEE Symposium on Foundations of Computation, 1982, 369 - 378.

SM Saraswat, K.C. & F. Mohammadi, Effect of scaling of interconnections on the time delay of VLSI circuits, *IEEE J. of Solid State Circuits,* vol. SC-17, 275-280, 1982.

Sa Savage, J., Planar circuit complexity and performance of VLSI algorithms, Proc. CMU Conference on VLSI Systems and Computations (H.T. Kung et. al. Eds.), Computer Science Press, 1981, 61-68.

SS Schmeck, H. & H. Schröder, Dictionary machines for different models of VLSI, Manuscript, Inst. Inf. & Pract. Math., Christian-Albrechts Univ. Kiel, 1983.

Se Seitz, C.L., Ensemble architectures for VLSI - A survey and taxonomy, Proc. of MIT Conference on Advanced Research in VLSI (P. Penfield, Jr., Ed.), Artech House, 1982, 130-132.

Th1 Thompson, C.D., *A Complexity Theory for VLSI.* Ph.D. Thesis, Dept. of Computer Science, Carnegie-Mellon University, 1980.

Th2 Thompson, C.D., Fourier transforms in VLSI, *IEEE Transactions on Computing,* (1983)

TR Thompson, C.D., & P. Raghavan, On estimating the performance of VLSI circuits, Tech. Rep. UCB/CSD 84/138, Computer Science Division (EECS), University of California, Berkeley, September 1983.

Vi Vitányi, P.M.B., Signal propagation delay, wire length distribution and the efficiency of VLSI circuits. Tech. Rept. CS-R8412, Centre for Mathematics and Computer Science (C.W.I.), Amsterdam, July 1984.

Vu Vuillemin, J., A combinatorial limit to the computing power of VLSI circuits, Proc. 21th IEEE Symposium on the Foundations of Computer Science, 1980, 294-300.

YYC Yuan, H.-T., Y.-T. Lin & S.-Y. Chiang, Properties of silicon, sapphire, and semi-insulating gallium arsenide substrates, *IEEE J. of Solid State Circuits,* Vol. SC-17, 1982, 269-274.

if $i=1$ **and** $r_1 \neq \varnothing$ **then** RETURN(k_1, r_1) **else** RETURN("Dictionary empty");
$(k_i, r_i) \leftarrow (k_{i+1}, r_{i+1})$

Appendix

We briefly explain the Dictionary machine of [ORS]. Let K be an ordered set of *keys*, and let R be a set of *Items*. A *dictionary* is a finite subset F of $K \times R$, such that for each key k at most one item r exists with $(k,r) \in F$. A key k is *in* F when there is a r with $(k,r) \in F$. Every $(k,r) \in F$ is called an *entry* in F. For all $k \in K$ we have that $F(k)$ is defined as $\{ (k,r) \mid (k,r) \in F \}$. Therefore, $F(k)$ is a singleton set if k is in F and empty otherwise. The dictionary machine must support:

INSERT(k,r): $F \leftarrow (F - F(k)) \bigcup \{(k,r)\}$;

DELETE(k): $F \leftarrow F - F(k)$;

SEARCH(k): RETURN $F(k)$ or "not there" ;

XMIN: $F \leftarrow F - F(k_{\min})$; RETURN $F(k_{\min})$ or "dictionary empty".

If, before an INSERT(k,r), the key k is already in F, or before a DELETE(k) not in F then that operation is called *redundant*. We can avoid such operations if we execute a SEARCH before every INSERT and DELETE. However, this would prevent *pipelining* of the operations. Therefore, we want the dictionary machine to be able to execute also redundant operations correctly.

The *Systolic Search Tree* solutions of [ORS], using N memory elements $S_1, S_2, \ldots, S_N$, can store at most N items. Every S_i can directly communicate with S_{i-1} and S_{i+1} ($1 \leqslant i \leqslant N$ with S_0 and S_{N+1} nonexistent). Each S_i has a *contents* (k_i, r_i) as well as (k_{i-1}, r_{i-1}) and (k_{i+1}, r_{i+1}), with the *least* key $k_0 = -\infty$, the *greatest* key $k_{N+1} = \infty$ and the *empty* item $r_0 = r_{N+1} = \varnothing$. Every memory element can determine the relative order of keys and whether an item is the empty one. Initially, all memory elements contain $(\infty, \varnothing)$. The memory elements constitute the *leafs* of a complete binary tree (so we assume $N = 2^k$ for some natural number k), the *root* of which accepts the input commands and delivers the required output. The keys are stored in ascending order in $S_1, S_2, \ldots, S_N$. For each memory element S_i, $1 < i \leqslant N$, the operations are defined as follows.

SEARCH(k):

if $k_i = k$ **and** $r_i \neq^*$ **then** RETURN(k_i, r_i) **else** RETURN ("not there")

INSERT(k,r):

if $k_{i-1} < k \leqslant k_i$ **then** $(k_i, r_i) \leftarrow (k,r)$; **if** $k_{i-1} = k$ **then** $(k_i, r_i) \leftarrow (k, ^*)$; **if** $k < k_{i-1}$ **then** $(k_i, r_i) \leftarrow (k_{i-1}, r_{i-1})$

DELETE(k):

if $k_i = k$ **then** $(k_i, r_i) \leftarrow (k, ^*)$

XMIN:

COMPRESS:

if $r_i = *$ **and** $r_{i+1} \neq *$ **then** $(k_i, r_i) \leftarrow (k_{i+1}, r_{i+1})$; **if** $r_i \neq *$ **and** $r_{i-1} = *$ **then** $(k_i, r_i) \leftarrow (k_i, *)$

The commands are propagated from the root of the tree down to the leafs constituting the memory elements, and the answers from the individual memory elements are percolated upwards from the leafs to the root. This latter process happens in such a way that at a node, where two different answers come together, the positive one always prevails and is the one which is transmitted upwards. Since there are in fact two trees, one to broadcast the input commands down to the leafs, and one to transmit the answers from the individual leafs to the root, meanwhile combining them, the commands can be processed *pipeline fashion.* The time in between the processing of two commands is called the *Period*. The Period is at least the time it takes to execute the commands at the leaf level. The *holes* * created by redundant INSERTs or nonredundant DELETEs may necessitate, to be shifted out, at most $N-1$ COMPRESS operations. However:

Lemma [ORS]. *By following each INSERT, XMIN and DELETE by two COMPRESS's, every initial Dictionary segment of length n will contain no more than* $\lfloor n/2 \rfloor$ *holes at any time.*

Thus we need at most N extra memory elements to obtain a Dictionary machine of capacity N which also allows redundant operations. Under the constant signal propagation delay assumption, [ORS] obtain $O(1)$ Period, $O(\log N)$ execution time for the individual commands, and $O(N)$ Area for the VLSI layout of the systolic tree. However, under the linear signal propagation delay assumption, since the *minimax edgelength* $e(N)$ of a wire in the tree is $\Omega(\sqrt{N}/\log N)$, if the layout Area is to be $O(N)$, the Period increases to $\Omega(\sqrt{N}/\log N)$ and the Execution Time to $\Omega(\sqrt{N})$, cf. Section 4. To get rid of synchronization problems due to unequal length wires, we may require all wires to have equal length. In that case the length is at least $\Theta(N/\log^2 N)$ while the Area increases to $A = \Theta(N^2/\log^2 N)$, see [PRS], yielding a Period of $\Theta(N/\log^2 N)$ and a Execution Time of $\Theta(N/\log N)$. The systolic Mesh layout proposed in [SS] yields, under both the constant and linear signal propagation delay assumption, a layout Area $O(N)$, a Period $O(1)$ and Execution Time $O(\sqrt{N})$. There the N memory elements are laid out in a rectangular $\sqrt{N} \times \sqrt{N}$ Mesh, such that the ith row contains, from left to right, the $(i-1)\sqrt{N}+1$ -th through $i\sqrt{N}$ -th memory elements. It is easy to see that, if the commands c.q. answers are transmitted to the first c.q. last element of each column, and the communication amongst the memory elements, to shift items and key pairs or holes, happens along the rows, the claimed performance is reached.