THE VLSI COMPLEXITY OF BOOLEAN FUNCTIONS

M.R. Kramer and J. van Leeuwen

Department of Computer Science, University of Utrecht
P.O. Box 80.012, 3508 TA Utrecht, the Netherlands

Abstract. It is well-known that all Boolean functions of n variables can be computed by a logic circuit with $O(2^n/n)$ gates (Lupanov's theorem) and that there exist Boolean functions of n variables which require logic circuits of this size (Shannon's theorem). We present corresponding results for Boolean functions computed by VLSI circuits, using Thompson's model of a VLSI chip. We prove that all Boolean functions of n variables can be computed by a VLSI circuit of $O(2^n)$ area and period 1, and we prove that there exist Boolean functions of n variables for which every (convex) VLSI chip must have $\Omega(2^n)$ area.

Keywords and phrases: logic circuit, Boolean function, Lupanov's theorem, Shannon's theorem, VLSI, chip, area, period.

1. Introduction.

The requirements for VLSI circuits are not as relaxed as they are for logic circuits. For example, a VLSI circuit must be embedded in the plane and occupy only a small amount of area, counting what it takes both for gates and wires. Also there can be no unbounded fan-in or fan-out, as basic switching elements can handle only a small number of wires. In this paper we study the quantitative effect of these differences on the "circuit complexity" of arbitrary Boolean functions.

It is well-known that all Boolean functions of n variables can be computed by a logic circuit of $O(2^n/n)$ gates (Lupanov [2]) and that there exist Boolean functions of n variables for which any logic circuit actually requires $\Omega(2^n/n)$ gates (Shannon [6]). We shall prove that suitable analogs of these results hold when VLSI circuits are used, but with the $2^n/n$ replaced by 2^n in the bounding expressions. This suggests that there is a fundamental difference in complexity between traditional logic circuits and their counterparts in VLSI.

To prove upper- and lowerbounds on the VLSI complexity of Boolean functions we adopt Thompson's simplified model of a VLSI chip ([7]). Essentially a chip is a bounded (convex) region in the plane, with the plane divided into unit size cells as in the regular grid (see figure 1). Each cell may contain one basic switching element or (at most) two crossing wires. Wires connect processing elements, and are constraint to run in

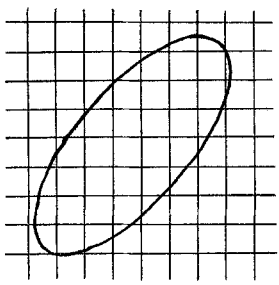

figure 1.

strictly horizontal or vertical directions. (Wires can bend from one direction into another in any cell, but can only cross and never run on top of one another.) The area of a chip is defined as the area of the grid that it occupies. We assume, like Thompson, that all switching elements act synchronously and that signals take unit time to propagate over a wire regardless its length. The period of a circuit is the minimum interval of time that must pass between consecutive sets of input data when the circuit is pipelined. We use the standard notations: A for area, T for time, and P for period.

The paper is organised as follows. In section 2 we recall some concepts from the theory of Boolean functions and provide a simple proof of Lupanov's theorem. It is included for having the opportunity to point out why the resulting construction of a logic circuit is not suited for VLSI. In section 3 we prove that all Boolean functions of n variables can be computed by a VLSI-circuit of $O(2^n)$ area and period 1. In section 4 we prove that there exist Boolean functions of n variables for which any (convex) VLSI circuit requires $\Omega(2^n)$ area. All proofs are elementary.

2. Preliminaries, and a classroom proof of Lupanov's theorem.

A minterm in $x_1,\ldots,x_n$ is any expression of the form $x_1^{\sigma_1}\ldots x_n^{\sigma_n}$ with $\sigma_i \in \{0,1\}$ $(1\leq i\leq n)$, where $x_i^0 \equiv \bar{x}_i$ and $x_i^1 \equiv x_i$. Observe that $x_1^{\sigma_1}\ldots x_n^{\sigma_n} \equiv 1$ if and only if $x_i \equiv \sigma_i$ for all i. If $f(\sigma_1,\ldots,\sigma_n) = 1$, then $x_1^{\sigma_1}\ldots x_n^{\sigma_n}$ is called a minterm for f. It is well-known that every Boolean function f is the sum of its minterms. As there are only 2^n distinct minterm expressions in all, the resulting expression for f is finite. A Boolean function f is uniquely determined by the choice of its minterms. As there are 2^{2^n} distinct sets of minterms, there are 2^{2^n} distinct Boolean functions of n variables.

The representation by minterms can be rephrased in computational terms. Consider the lexicographic tree of minterms, modified such that the nodes "compute" the proper x_i or $\bar{x}_i$ and "and" it to the partial term values in x_1 to x_{i-1} $(1\leq i\leq n)$. The resulting tree will be called the (general) mintree for n variables and is shown in figure 2(a). It is a circuit with $2^i + 2^{i-1}$ logic gates in the i^{th} level and (hence) $\sum_{i}^{n}(2^i+2^{i-1}) =$ $= 3.(2^n-1)$ gates total. Every leaf of the mintree corresponds to a unique minterm in x_1 to x_n. Every Boolean function of n variables corresponds to a selection of leaves

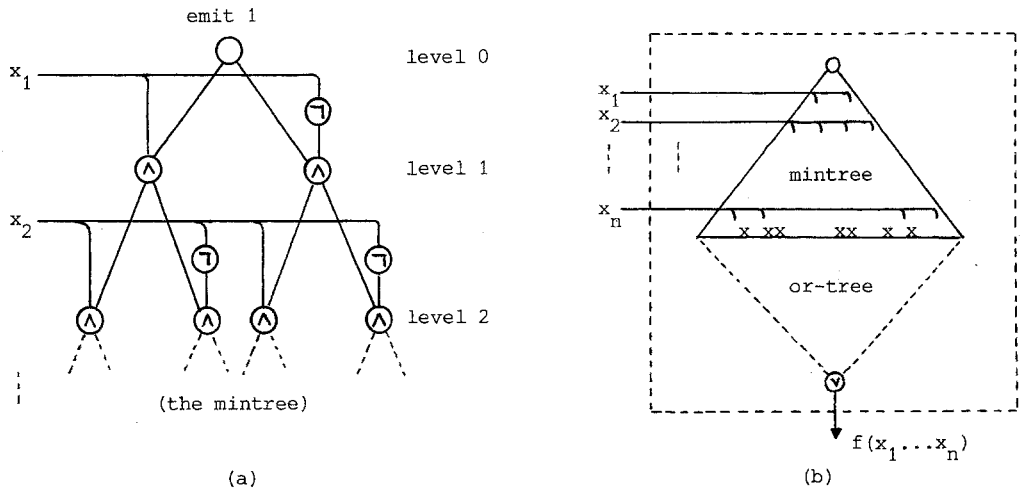

figure 2.

of the mintree. Let the selected leaves be marked (by x, see figure 2.b) and suppose another tree circuit is added that sums ("or-s") the computed values at the marked leaves. The resulting circuit is shown in figure 2(b) and clearly is a logic circuit of $O(2^n)$ gates for computing any Boolean function by the proper masking of components. We shall ignore the "or-" tree and concentrate on the mintree part. By optimizing it we shall be able to derive a weak form of Lupanov's theorem ([2]) in a simple way.

Let f be an arbitrary Boolean function in x_1 to x_n and let T be the mintree with marked leaves representing f. Cut T at level s, and consider the 2^s subcircuits (subtrees) attached to the nodes in this level. (The value of s will be fixed later, but

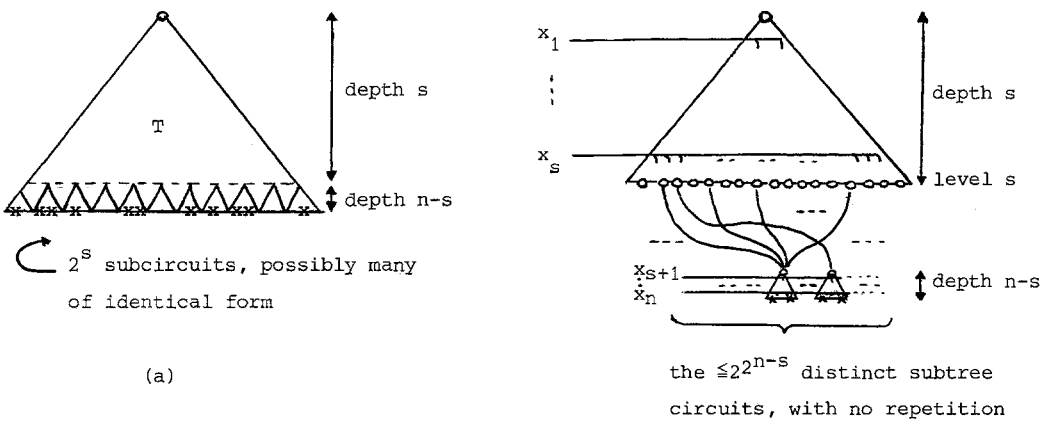

figure 3.

will be "almost equal" to n.) As circuits with input x_{s+1} to x_n the subtrees are identical, but they may differ through their marking. Because the subtrees are small but large in number, many must have the same marking and (hence) perform essentially the same computation. Thus, to save gates, we might as well split the subcircuits off from T and weed out all duplicates, and connect every intermediate node of level s to the proper and unique copy of the subcircuit that was originally attached to it. See figure 3(a)(b) for a pictorial illustration of this optimization of T. When phrased in terms of Boolean functions the construction is known as the "Lupanov decomposition" of f. Observe that every subcircuit of depth n-s has 2^{n-s} leaves, and thus there are $2^{2^{n-s}}$ distinct subcircuits if we take all possible markings into account.

Theorem 2.1(Lupanov, 1958). Every Boolean function of n variables can be computed by a logic circuit of $O(2^n/n)$ gates, with no unlimited fan-in's.

Proof

Count the number of gates in the Lupanov decomposition of f's circuit. There are $3.(2^s-1)$ gates in the top part. Next there are $\leq 2^{2^{n-s}}$ small tree circuits with $3.(2^{n-s}-1)$ gates each. Suppose i_j nodes from the original level s are connected to the j^{th} subtree circuit $(1 \leq j \leq 2^{2^{n-s}})$. To avoid the unlimited fan-in, we can combine them through an or-tree with i_j-1 gates. This accounts for an extra of $\sum_j (i_j-1) < \sum_j i_j = 2^s$ gates. Finally the results at the marked leaves must be summed in an or-tree to obtain $f(x_1 \ldots x_n)$ as output. Because we optimized the lower part of the mintree, there may actually be fewer marked nodes in the subcircuits altogether that still (correctly) represent f. Note that each of the $2^{2^{n-s}}$ subtrees can be paired to one with a complementary marking, to contribute a total of 2^{n-s} marked leaves. Thus there are at most $\frac{1}{2}.2^{n-s}.2^{2^{n-s}}$ marked leaves, and the or-tree summing for f needs to have no more gates than this. The total number of gates in the circuit is bounded by:

$$\begin{aligned} & 3.(2^s-1) + 3.(2^{n-s}-1).2^{2^{n-s}} + 2^s + \tfrac{1}{2}.2^{n-s}.2^{2^{n-s}} \leq \\ & \leq 4.2^s + 3\tfrac{1}{2}.2^{n-s}.2^{2^{n-s}} \leq \\ & \leq 4.(2^s + 2^{n-s}.2^{2^{n-s}}). \end{aligned}$$

Now choose s (integer) such that $2^{n-s} \leq n - \alpha \log n \leq 2^{n-s+1}$, for some α to be fixed later. It follows that $2^s \leq 2.2^n/n - \alpha \log n$ and $2^{n-s}.2^{2^{n-s}} \leq (n-\alpha \log n).2^n/n^\alpha \leq 2^n/n^{\alpha-1}$. Thus the number of gates in the circuit is bounded by

$$\begin{aligned} & 4.(2.\frac{2^n}{n-\alpha \log n} + 2^n/n^{\alpha-1}) = \\ & = 2^n/n.(8.\frac{n}{n-\alpha \log n} + 4/n^{\alpha-2}) = \\ & = O(2^n/n). \end{aligned}$$

, for any $\alpha \geq 2$. □

Note that we proved only a simplified version of Lupanov's theorem. The classical result states that by additional techniques one can bring the constant factor in the $O(2^n/n)$ term down to 1.

3. Area/time-efficient VLSI circuits for Boolean functions.

We now change our point of view to VLSI. We adopt all conventions as laid down in Thompson's model except that, for the moment, we will allow unlimited fan-outs by free "tapping" of a wire. The main issues are not gate counts, but area and time.

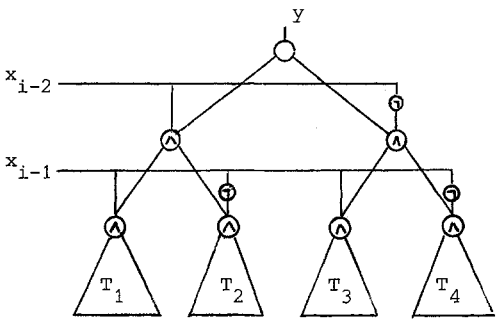

figure 4.

Trees are among the graphs that are easiest to lay out (see e.g. Thompson [7]) and only require "linear" area. The circuit lay-outs presented in this section derive from the common H-pattern design for (perfect) binary trees originally proposed by Mead and Rem [4].

Consider the representation of an arbitrary Boolean function f as suggested in fig. 2(b). For convenience we assume n even. The circuit of fig. 2(b) has a recursive structure that can be exploited to obtain an area-efficient lay-out. The essential idea is shown in figure 4. Build up the tree from the lowest level by adding two variables at the time, with the necessary logic to gate them into the four identical (but perhaps differently marked) subtrees that are needed on the variables x_i to x_n.

The construction suggests that a Lupanov-type optimization (see theorem 2.1) does not immediately lead to desirable effects for VLSI: (i) the many subtree circuits that Lupanov optimizes are all in the lowest part of the mintree and (hence) uniformly spread over the chip, and it would require much extra area and long wires to set the circuits apart for general access, (ii) by the same token it would lead to many more wire-crossings, which does not enhance the practicality for VLSI production and (iii) the unlimited fan-in (or: the or-trees that simulate it) is not equal at all places and (thus) creates severe problems of synchronization if the circuit is to be pipelined unless we "pretend" the or-trees to be all equal and complete. Redundancy has never been an issue in logic circuits, but it may be unavoidable in VLSI circuits!

We shall now rid ourselves of the assumption of unlimited fan-out. Rather than input the x_1 to x_n one after the other at separate levels of the mintree, we input them simultaneously in parallel at the top and distribute the signals down the tree while the computation of partial minterms in x_1 to x_i $(i \to n)$ is taking place. The idea is given in figure 5, where □-cells are used to represent the subcircuit that appends a next x_i to the computed terms at level i. Recursively continuing it leads

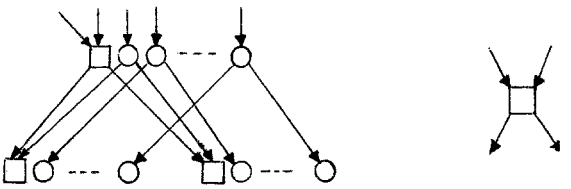

figure 5.

to another tree that computes all minterms at its leaves, the parallel mintree. Every Boolean function f of n variables can be represented by a marking of the leaves of the parallel mintree.

The parallel mintree has an area-efficient lay-out (see figure 6). The technique derives from the H-pattern construction and consists of adding two variables x_{i-1} and x_{i-2} at a time (for $i \to 1$) while combining the four recursively obtained sub-

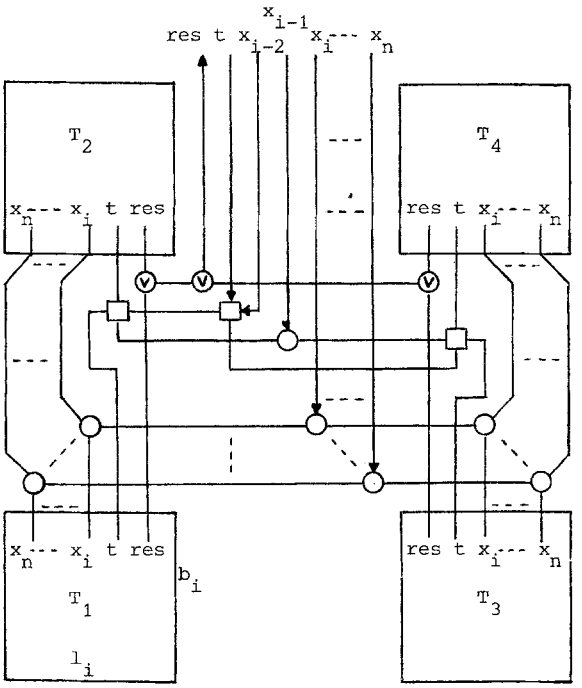

figure 6.

circuits in x_i to x_n. The □-cells in figure 6 actually are small circuits of some bounded size. The design uses a "t-" wire for accumulating partial minterms on the way down the tree, and a "res-" wire for summing the results from the marked leaves.

Theorem 3.1. Every Boolean function of n variables can be computed by a VLSI-circuit of $O(2^n)$ area, with a compute time of $O(n)$ and a period of $O(1)$.

Proof

Let $C=C_1$ be the resulting VLSI-circuit for f obtained by laying out the parallel mintree as suggested. Let the i^{th} intermediate circuit C_i of the recursive construction (starting with i equal to n-1) fit in a rectangle of size $b_i \times l_i$. From figure 6 we conclude that $b_{i-2} = 2b_i+(n-1) + 9$ and $l_{i-2} = 2l_i+(n-i) + 8$. (The additive constants are really larger, because the detailed circuits for the □-cells require more than unit area.) It follows that $b_1 = O(2^{n/2}-n)$ and $l_1 = O(2^{n/2}-n)$ and (hence) that C occupies no more than $O(2^n)$ area.

The time bound of $O(n)$ for computations with C is obvious. The period of $O(1)$ follows because C can process an entire input tuple in parallel as a single wavefront going down, with the possibility of sequencing consecutive wavefronts at unit time distance. □

4. Optimal area/time bounds for Boolean VLSI-circuits.

For particular Boolean functions f of n variables the bounds of theorem 3.1. may not be best possible. (See e.g. Thompson [7] for many examples of essentially Boolean functions that require only $O(n)$ or $O(n^2)$ area, or $O(\log n)$ time.) The question is rather how tight the bounds are if the theorem is taken as a uniform statement for the entire class of Boolean functions of n variables. It is clear that the $O(1)$ bound on the period of circuits can not be improved. We show that none of the other bounds in theorem 3.1. can be improved either.

Theorem 4.1. There are Boolean functions of n variables such that any circuit to compute them requires $\Omega(n)$ time on some inputs.

(The result is a simple variant of the fact that there are Boolean functions that require circuit depth n, see Savage [5]. The proof is omitted.)

A natural question is whether the $O(2^n)$ bound on the area requirement for general Boolean functions can be improved.

Proposition 4.2. Suppose all Boolean functions of n variables can be computed by circuits that fit on a fixed chip of area A. Then necessarily $A = \Omega(2^n)$.

Proof

Recall that a chip was defined to be a connected region of the unit grid. Assume there is a fixed chip R of area A such that for all Boolean functions f of n variables there is a circuit to compute it that can be embedded in R. According to Thompson's model each cell of R can be occupied by at most one of the finitely many different constructs, say c altogether. Thus at most c^A different circuits can be embedded in R. On the other hand, the number of distinct Boolean functions of n variables is 2^{2^n}. It follows that $c^A \geq 2^{2^n}$, or $A = \Omega(2^n)$. □

The assumption that there be a "general chip" is too strong to measure the area requirement of Boolean functions. We shall now consider circuits fitted on their own, optimal area chips. To be realistic we require that the chips are convex. (This is a very common constraint, see e.g. Brent and Kung [1]). By the following two lemmas we shall be able to "mold" every circuit that fits on a convex chip of area A into a form that fits in a standard (iso-oriented) rectangle on the grid of area O(A).

Lemma 4.3. Every convex chip of area A can be enclosed by a (tilted) rectangular box of area 2A.

Proof

(The easy argument can be derived from e.g. Yaglom and Yaglom [8], solution to problem 120a.) Let R be a bounded convex region of area A (see figure 7). Let AB be the longest chord. The tangents to R at A and B are perpendicular to AB. Now "enclose" R by drawing the (two) lines of support that are parallel to AB. Let the

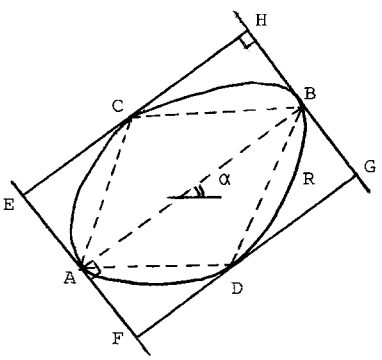

figure 7.

lines touch on R at C and D (see figure 7). In figure 7 is indicated that R is now enclosed by $\square$EFGH, formed by the four intersecting tangents. By convexity $\square$ADBC $\subseteq$ R and (hence) $\square$ADBC has area $\leq A$. On the other hand, by geometric reasoning it follows immediately that $\square$ABHE = 2.ΔABC and that $\square$ABGF = 2.ΔABD and (hence) that $\square$EFGH = = 2.$\square$ADBC $\leq$ 2A. $\square$

Lemma 4.4. Let circuit C be embedded on the unit grid within a rectangle of area A that is tilted by angle $\alpha (0 \leq \alpha \leq \pi/4)$. Then C can be embedded on the unit grid within an iso-oriented rectangle of area bounded by $(1 + 2\tan\alpha - \tan^2\alpha)A \leq 2A$.

Proof

Let C be embedded on the grid such that it is enclosed by a rectangle of size lxw ($l \geq w$ and $l.w = A$) whose longest side makes an angle α with the positive x-axis of the grid (see e.g. figure 7). It is no restriction to assume that $0 \leq \alpha \leq \pi/4$, for otherwise the argument holds for a properly reflected or rotated version of the problem. We also assume that no cell cut by the boundary of the rectangle is used by C.

Divide the rectangle (and correspondingly, the circuit) into parts A_1, A_2 and B as shown in figure 8. Parts A_1 and A_2 are "padded" to iso-oriented rectangular boxes

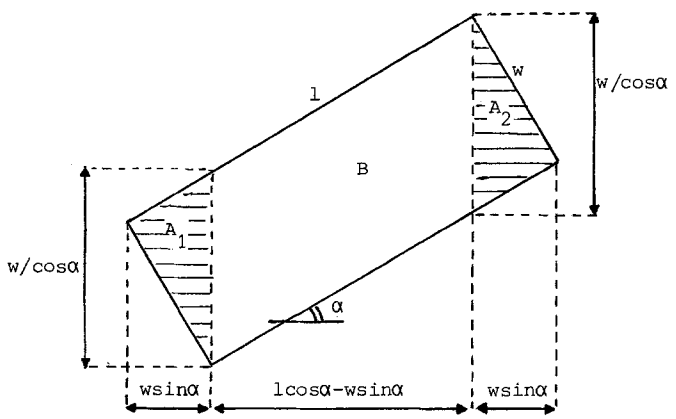

figure 8.

of size $w/\cos\alpha \times w\sin\alpha$ each. Part B consists of $l.\cos\alpha - w.\sin\alpha$ columns of $w/\cos\alpha$ cells each. Because $\alpha \leq \pi/4$, each column can differ by at most one cell at either end from the immediately preceding column (see figure 9). It means that we can bring B down to an iso-oriented form by shifting each of its columns downwards by at most one cell with respect to the immediately preceding column, resulting in a circuit as shown in figure 10. Figure 9 shows that this can be done within the constraints of Thompson's model, provided that we include a spacing of one extra column in between every pair

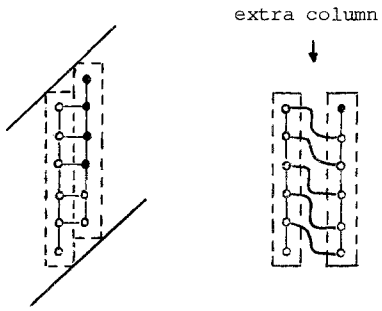

figure 9.

of columns. This explains that in figure 10 the length of the B-part is at most doubled. Thus circuit C appears embedded in an iso-oriented rectangle of width w/cosα and length about 2.l.cosα, and hence area 2.l.w = 2A. By being a little more accurate, one can

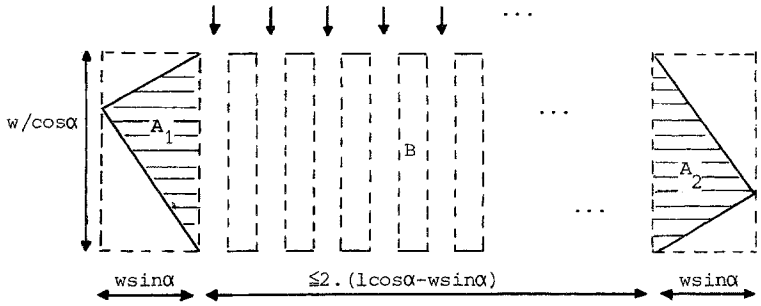

figure 10.

see that a down-shift over one cell (hence, the need for an extra column) is required only once in every 1/tanα columns. It follows that the length of α can remain at (1+tanα) times the original. Hence C fits in an iso-oriented rectangle of size (1+tanα) - l.cosα + (1-tanα).w.sinα by w/cosα, which has an area bounded by (1+2tanα--$\tan^2\alpha$).A. □

We conclude that the convexity assumption in VLSI-theory degenerates and does not go much beyond assuming that chips are rectangular! (In all fairness we should add that this only applies if one adopts the grid-model for the chip surface.)

Corollary 4.5. Every circuit that fits on a convex chip of area A can be embedded in an ordinary iso-oriented rectangle of area ≦4A.

Theorem 4.6. There are Boolean functions of n variables such that every convex chip for them requires $\Omega(2^n)$ area.

Proof

Suppose that all Boolean functions of n variables can be computed by convex chips of area $\leq A$. By corollary 4.5. we may as well assume that all circuits fit in an iso-oriented rectangle of area $\leq 4A$. Now note that there are at most $\sqrt{4A} = 2.\sqrt{A}$ different (i.e., non-isomorphic) iso-oriented rectangles of area $\leq 4A$, and that each of these rectangles can have at most c^{4A} different circuits. (The constant c is as in the proof of proposition 4.2.) It follows that necessarily $2\sqrt{A}.c^{4A} \geq 2^{2^n}$, and (hence) that $A = \Omega(2^n)$. □

By an only slightly more sophisticated argument one can see that theorem 4.6. also holds without the convexity assumption. Note that every chip of area $\leq A$ is fully described by its contour which, when "projected" onto the grid-lines, can be encoded into a string of length $\leq 4A$ over the alphabet {L,R,U,D} (L for "one square left", etc.). It follows that there are at most $\sum_{1}^{4A} 4^l \leq \frac{4}{3}.4^{4A} \leq d^A$ (some d) chips of area $\leq A$. On each chip at most c^A different circuits can be fitted. To accomodate all Boolean functions of n variables one must necessarily have $c^A.d^A = (cd)^A \geq 2^{2^n}$, and (hence) that $A = \Omega(2^n)$. By a more detailed counting argument it follows that almost all Boolean functions of n variables require $\Omega(2^n)$ area.

5. References.

[1] Brent, R.P. and H.T. Kung, The area-time complexity of binary multiplication, J.ACM 28 (1981) 521-534.

[2] Lupanov, O.B., A method of circuit synthesis, Izv. V.U.Z. Radiofiz. 1 (1958) 120-140.

[3] Mead, C.A. and L.A. Conway, Introduction to VLSI systems, Addison-Wesley, Reading, Mass., 1980.

[4] Mead, C.A. and M. Rem, Cost and performance of VLSI computing structures, IEEE J. Solid State Circuits SC-14 (1979) 455-462.

[5] Savage, J.E., The complexity of computing, John Wiley & Sons, New York, N.Y., 1976.

[6] Shannon, C.E., The synthesis of two-terminal switching circuits, Bell Syst. Techn. J. 28 (1949) 59-98.

[7] Thompson, C.D., A complexity theory for VLSI, (Ph.D.Thesis), Techn. Rep. CMU-CS-80-140, Dept. of Computer Science, Carnegie-Mellon University, Pittsburgh, P.A., 1980.

[8] Yaglom, A.M. and I.M. Yaglom, Challenging mathematical problems with elementary solutions, Holden-Day, San Francisco, CA., 1967.