# FOUR RESULTS ON THE COMPLEXITY OF VLSI COMPUTATIONS

Thomas Lengauer and Kurt Mehlhorn

A 83/01

Fachbereich 10
Universität des Saarlandes
D-6600 Saarbrücken
West Germany

# FOUR RESULTS ON THE COMPLEXITY OF VLSI COMPUTATIONS

Thomas Lengauer and Kurt Mehlhorn*

Abstract: We present four results on the complexity of VLSI computations:

a) We further justify the Boolean circuit model [Vu, Sa, LS] by showing that it is able to model multi-directional VLSI devices (e.g. pass transistors, pre-charged bus drivers).

b) We prove a general cutting theorem for compact regions in $R^d$ ($d \geq 2$) that allows us to drop the convexity assumption in lower bound proofs based on the crossing sequence argument.

c) We exhibit an $\Omega(n^{1/3})$ asymptotically tight lower bound on the area of strongly where-oblivious chips for transitive functions.

d) We prove a lower bound on the switching energy needed for computing transitive functions.

Keywords: Complexity Theory, Lower Bounds, VLSI Models, Switching Energy.

---

*A preliminary version appeared in the Proceedings of the CMU-Conference on VLSI Systems and Computations, Pittsburgh PA, Oct. 19-21, 1981.

## 1. THE BOOLEAN CIRCUIT MODEL AND MULTI-DIRECTIONALITY OF VLSI CIRCUITS

In a number of recent papers on the complexity of VLSI computations [Vu, Sa, LS] VLSI chips are modelled as synchronous Boolean circuits that are embedded into the plane. More specifically, the following model is proposed in [LS].

A chip consists of a synchronous boolean machine and its embedding into the plane (the layout). The boolean machine is built using two-input gates and and- and or-gates of arbitrary fan-in. A layout assigns a connected, compact region of the plane to every component (wire, gate, port) of the circuit such that each point in the region lies inside a square of size $\lambda^2$ that is completely contained in the region, such that every point of the plane belongs to at most $\nu$ regions and such that regions in the layout intersect whenever the corresponding circuit components are connected in the boolean machine. The area A of a chip is the area of the union of all regions assigned to circuit components, the computation time T of the chip is the time (in cycles) required by the underlying boolean machine. The period P is the minimum distance in time between two problem instances which are to be fed into the machine. We refer to the model described above as the LS-model. The LS-model encompasses all Boolean circuit models used in the literature. Despite its generality (arbitrary fan-in) all known lower bounds for the familiar $AT^2$-measure hold true for the LS-model.

In Boolean circuits wires have a specific direction (from the output of one gate to inputs of one or more other gates). Moreover, this fact is explicitly used in numerous lower bound arguments (see [LS]). However, current MOS technology supplies us with a number of truly multi-directional devices, such as pass

transistors and precharged bus drivers, and these devices are extensively used in circuit design (cf. [MC]). (Note that Thompson [Th] relaxes the uni-directionality of wires somewhat. But he is not able either to model, e.g., a bus driver with many ports and constant delay).

In this section we introduce a class of models for VLSI circuits that are also able to model multi-directional components. We show that the circuits in these models can be simulated by circuits in the LS-model (and vice versa) with area and time increased only by a constant factor. Thus we give evidence for the fact that the uni-directionality of Boolean circuits is indeed no serious restriction if we are only concerned with asymptotic analysis.
These results contrast experiences made in the area of switch-level simulators for MOS circuits, where due to the greater level of detail in which the circuits have to be modelled - especially because one has to take special care in modelling faulty circuits - multi-directional circuit models have great advantages over the Boolean circuit model (cf. [Br]).

The multi-directional circuit models (MD-models) are closely related to models used in switch level simulators for MOS circuits. Their main characteristic is a different concept of a wire. A wire in an MD-model is not a passive but an active device. It determines its value as a symmetric function of the values on its terminals. The ingredients of an MD-model are an MD-circuit and a layout fulfilling the following requirements:

1. The MD-circuit consists of components which are either wires or gates. It operates on Boolean values (0 and 1) at any of a finite number of strengths, increasing from 1 to a constant r. Thus a value is a pair $v = (v_b, v_s) \in B := \{0,1\} \times \{1,\ldots,r\}$.

2. Each gate and wire of the circuit have a finite number of terminals. Gates may have at most k terminals, where k is some specified constant. (We do not distinguish between input

and output terminals.) Each terminal of a wire (resp. gate) is either an input or an output to the MD-circuit, or is coincident with exactly one terminal of a gate (resp. wire).

3. Each gate g has an associated transition function $\delta_g$ that computes the values on the terminals of the gate. Here $\delta_g$: $B^m \to B^m$ where $m \leq k$ is the number of terminals of the gate, maps values on the terminals before switching to values on the terminals after switching of the gate.

4. The circuit operates fully synchronously. Each step consists of four phases:

   a) Each gate reads the values on its terminals.
   b) Each gate uses its transition function to determine the new values on each of its terminals.
   c) Each wire reads the values on all of its terminals.
   d) Each wire determines a value $w \in B$ to be put on all of its terminals. This value is the strongest Boolean value put on any of its terminals. In a case of a tie between 0 and 1 the value w is undefined. (Essentially a wire is a special sort of gate that can have arbitrarily many inputs and that computes some kind of threshold function. Gates and wires switch alternately.)

5. The layout is defined analogously to the LS-model.

Different MD-models can vary in the number of strength levels and the kinds of gates they allow. As an example we give the following MD-model: There are three levels of strength: 1 (isolated charge), 2 (connection to GND resp. VDD through a pull-up resistor), and 3 (direct connection to GND resp. VDD). There is only one kind of gate, namely the MOS transistor T. T has three terminals s (source), d (drain), and g (gate). The transition function $\delta_T$ is defined as follows: $\delta_T(s,d,g) = (s',d',g')$, where $g' = (g_b,1)$, and $s' = (s_b,1)$ if $g_b = 0$, otherwise $s' = d'$ is the strongest of the values s and d. In case of tie resolve arbitrari-

ly, or set the value to be undefined. Thus a transistor with a 1 on the gate behaves exactly like a wire with two terminals. It is straightforward to show the following

Theorem 1: Each circuit in the LS-model can be simulated by a circuit in the above MD-model with area, time, and period only increased by a constant factor.

Sketch of Proof: We can simulate inverters as well as AND- and OR-gates with arbitrary fan-in in the MD-model by using the NOR-gate implementations common in NMOS. These implementations translate easily into the MD-model. Since wires in the MD-model can have arbitrary shape it is possible to simulate each gate in the LS-model "in place". Any undefined values occuring in the MD-circuit have to have been introduced through incorrect operation of the simulated LS-circuit. □

The following theorem shows that the reverse of Theorem 1 holds for all MD-models. This means that circuits in the LS-model are powerful enough to model also multi-directional VLSI circuits. In the proof of the theorem the properties of the gates in the MD-model chosen do not play a significant role. They only influence the constant factor.

Theorem 2: Choose any MD-model. Any circuit in the model can be simulated by a circuit in the LS-model with area, time, and period only increased by a constant factor. The factor depends on the MD-model chosen.

Sketch of Proof: Let C be the circuit in the MD-model that is to be simulated. Let C run in area A, time T, and period P. We will simulate C "in place" by a Boolean circuit C'. It will turn out that we can fit a layout of C' inside a blowup by a constant factor of the layout of C. This yields the bound on the area. The bounds on time and period follow directly from the definition of C'.

For the purpose of the simulation, values $v \in B$ will be encoded in a unary fashion, i.e., by unit Boolean vectors of length 2r. The

vector $(v_1,\ldots,v_{2r})$ with the unique 1 being element $v_i$ encodes the value $v = ((i+1) \bmod 2, (i+1) \operatorname{div} 2)$.

Since each gate has at most $k = O(1)$ terminals its function can be simulated by a Boolean circuit with $A,T,P = O(1)$ that fits into a blowup by a constant factor of the layout of the gate in C. (Note that we can, in addition, assure the proper location of the terminals in the circuit C'. However, this may require different layouts of the circuit simulating gate g, for different copies of the same gate g in C.)

It remains to show how to simulate the wires. Let $(v_1,\ldots,v_n)$ with $v_i = (v_{i,1},\ldots,v_{i,2r})$ be the unit vectors encoding the values on the terminals of the wire before switching. The "output" value w of the wire is encoded by a vector $(w_1,\ldots,w_{2r})$, where for $1 \le i \le 2r$

$$w_i = \bigwedge_{k=2\lceil \frac{i}{2} \rceil + 1}^{2r} \left( \bigwedge_{j=1}^{n} \overline{v_{j,k}} \right) \wedge \bigvee_{j=1}^{n} v_{j,i}$$

(An undefined value is here encoded by a vector which is not a unit vector). If we allow AND- and OR-gates with arbitrary fan-in this formula has depth 2. For the simulation of one step on the wire 2r such functions have to be computed in parallel. The layout of the circuitry necessary for this can fit inside a blowup by a constant factor (depending on r) of the layout of the wire in C. □

Note that in the proof of Theorem 2 we make extensive use of the fact that in the LS-model AND- and OR-gates with arbitrary fan-in are allowed, as well as of the liberty we can take with respect to the shape of the gates. At first sight this may seem an unrealistic assumption. Notice, however, that in all our models T assumes unit delay on wires, independent of wire length. Thus, in effect, T measures the number of clock cycles taken by a synchronous machine, rather than the physical time delay of the computation. From this view point unbounded fan-in becomes a reasonable assumption since

it can be implemented just using wire delay (which is not considered in our models) and not computational delay.

The unit delay assumption is the major drawback of our models. If T is supposed to measure the actual physical delay, this assumption is only realistic in technologies where gate delay dominates wire delay. However, as far as lower bounds are concerned models incorporating wire delay can only yield stronger results, thus our lower bounds are also valid in such models (see [CM], [MC] for alternative assumptions on delay). The synchronicity assumption is not a severe restriction. Call a network asynchronous if its gates do not switch at every cycle but only upon arrival of new data. By an easy transformation blowing up time and area only by a constant factor each such network can be made synchronous. We only have to add some gate enable circuitry.

## 2. THE CONVEXITY ASSUMPTION

Most papers on the complexity of VLSI circuits assume the convexity of their layout. (The notable exceptions are Thompson's original paper [Th] and a paper by J.E. Savage [Sa]. However, both authors need to make restrictions on the geometry of the layout, namely, Thompson assumes that the layout is embedded into the planar grid and Savage assumes that all wires are composed of straight line segments). We believe that this assumption is unsatisfying, because:

a) Convexity is not an inherent property of VLSI circuits.

b) Dropping the convexity assumption and modelling a chip as a compact region of the plane (holes allowed!) will considerably strengthen our lower bounds. It will allow us to measure the area occupied by only those circuit components that actually process information, and affect the yield during chip production.

c) Taking power and ground into account non-convexity may be called for (cf. the discussion in [CM] on how convexity plus power consumption imply large area).

In the following we will prove a cut theorem for compact regions in the plane that will allow us to drop the convexity assumption. Note that the lower bound argument used in [Th, Vu, Sa, LS] calls for the circuit layout to be cut into two halves such that about the same number of bits out of a certain pre-specified set of I/O-bits are communicated between the chip and its environment through each half of the layout. Here the length of the cut has to be short, i.e., $O(\sqrt{A})$. We will associate with each I/O bit a point in the plane through which the bit is communicated. We call this point the "port". We can assume different bits to be communicated through different ports because we make no assumption about the minimum separation of ports.

Theorem 3: Let M be a compact plane region (the layout). Let $p_1,\ldots,p_r$ be r different points inside the interior of M (the ports) of which n are specially marked. (The marked ports correspond to the bits with respect to which to balance the halves of the chip.) Then M can be cut by a set C of three straight cuts into two compact sets $M_\ell$ and $M_r$ (the left and right half) such that:

1. $M_\ell \cup M_r = M$, $M_\ell \cap M_r = C$
2. $\lg(C) \leq 2\sqrt{A}$ where A is the area of M and $\lg(C)$ is the length of C.
3. $M_\ell$, $M_r$ both have at most $2n/3$ marked points of $p_1,\ldots,p_r$ in their interior and none of the points $p_1,\ldots p_r$ on their border.

Proof: First we note that because the set $P = p_1,\ldots,p_r$ is finite M can be put into a Cartesian coordinate system such that no line parallel to the x- or y-axis contains more than one point in P. We devise the following procedure for cutting M.

W.l.o.g. assume that $\lceil n/2 \rceil$ of the marked points lie above the x-axis, $\lfloor n/2 \rfloor$ of the marked points lie below the x-axis, and no point in P lies on the x-axis. Let $M_+ = \{(x,y) \in M;\ y \geq 0\}$ and $M_- = \{(x,y) \in M;\ y \leq 0\}$. Let $A_+$ be the area of $M_+$ and $A_-$ be the area of $M_-$ ($A_+ + A_- = A$). The method for cutting M has two steps:

Step 1: For any real $\ell$, let $C_\ell = \{(x,y) \in M;\ y = \ell\}$. We choose two cuts of M parallel to the x-axis at distances $y = \ell_t > 0$ and $y = \ell_b < 0$. $C_{\ell_t}$ and $C_{\ell_b}$ cut M into three parts $M_t, M_m$, and $M_b$ as suggested by Figure 1.

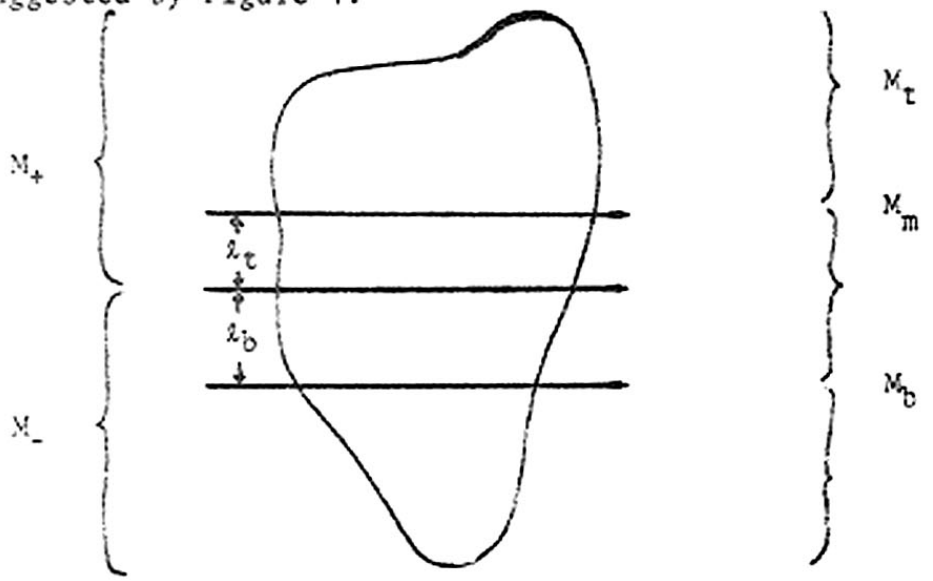



Figure 1

We choose $\ell_t$ such that $\lg(C_{\ell_t}) + \ell_t \leq \sqrt{2A_+}$ and $\ell_t \leq \sqrt{2A_+}$. Analogously we choose $\ell_b$ such that $\lg(C_{\ell_b}) - \ell_b \leq \sqrt{2A_-}$ and $\ell_b \geq -\sqrt{2A_-}$. Such $\ell_t$ and $\ell_b$ can be found. For assume that such an $\ell_t$ does not exist. Then

$$A_+ = \int_0^\infty \lg(C_\ell)d\ell \geq \int_0^{\sqrt{2A_+}} \lg(C_\ell)d\ell > \int_0^{\sqrt{2A_+}} (\sqrt{2A_+} - \ell)d\ell = A_+$$

which is a contradiction. In fact we can assume that no point of P lies on $C_{\ell_t}$ since the set $\{\ell;\ 0 < \ell \le \sqrt{2A_+}$ and $\lg(C_\ell) + \ell \le \sqrt{2A_+}\}$ must have a positive measure. An analogous argument applies to $\ell_b$.

Now, if $M_m$ contains no more than 2n/3 marked points of P then we are done. Because $M_t$ and $M_b$ both have at most n/2 marked points of P we only have to choose $M_\ell$ to be the one of the three sets with the most marked points of P in it. If, however, $M_m$ has more than 2n/3 marked points in it then we have to continue cutting.

Step 2: We bisect $M_m$ with a line $x = \ell_v$ such that half the marked points in $M_m$ are either side on the line. More precisely, the cut consists of the segment $\{(x,y);\ x = \ell_v$ and $\ell_t \ge y \ge \ell_b\}$. Now clearly the half of $M_m$ containing the most marked points of P will serve as $M_\ell$.

The total length of the cut C is bounded by

$$\lg(C) \le \lg(C_{\ell_t}) + \lg(C_{\ell_b}) + (\ell_t - \ell_b) \le \sqrt{2A_+} + \sqrt{2A_-}$$

$$\le \sqrt{2A_+} + \sqrt{2(A-A_+)} \le 2\sqrt{A}$$

since the function $z(x) = \sqrt{x} + \sqrt{1-x}$ has a maximum of $\sqrt{2}$ at $x = 1/2$. □

The above proof is patterned after a similar proof of a separator theorem for planar graphs given in [LT].

Theorem 3 can be generalized to sets of $d \ge 2$ dimensions. Thus the convexity assumption can also be dropped in three-dimensional models for VLSI that have been mentioned in the literature ([Ro]).

Theorem 4: Let M be a compact region $\mathbb{R}^d$ ($d \geq 2$). Let $\{p_1,\ldots,p_r\}$ be a set of points inside the interior of M of which n are specially marked. Then M can be cut by a set C of at most 2d-1 straight cuts (i.e. subsets of hyperplanes) into two sets $M_\ell$ and $M_r$ such that:

1. $M_\ell \cup M_r = M \qquad M_\ell \cap M_r = C$

2. $\mathrm{vol}_{d-1}(C) \leq (2d-3/2)\cdot 2^{1/d}\cdot \mathrm{vol}_d(M)^{(d-1)/d}$

3. $M_\ell$, $M_r$ both have at most 2n/3 marked points of $p_1,\ldots,p_r$ in their interior and none of the points $p_1,\ldots,p_r$ on their border.

(Here $\mathrm{vol}_d(M)$ denotes the d-dimensional volume of the set $M \subset \mathbb{R}^d$).

Proof: We iterate step 1 of the cutting procedure given in the proof of Theorem 3 through the dimensions 1,...,d-1 of M. After step i we either stop (if $M_i$ contains at most 2n/3 marked points) or continue cutting some subset $M_i$ of M. We start out with $M_1 = M$. In general for $1 \leq i \leq d-1$, we assume without loss of generality that $\lceil n/2 \rceil$ of the marked points inside $M_i$ lie in the positive half-space with respect to dimension i. (If this is not the case we first translate $M_i$ by an appropriate amount). Then we cut $M_i$ normal to the i-th dimension at distances $\ell_{t,i} > 0$ and $\ell_{b,i} < 0$ from the origin as in step 1 of the proof of Theorem 3, except that we choose $\ell_{t,i}$ such that $\mathrm{vol}_{d-1}(C_{\ell_{t,i}}) \leq 2\ \mathrm{vol}_d(M_{i,+})^{(d-1)/d}$ and $\ell_{t,i} \leq 1/2\ \mathrm{vol}_d(M_{i,+})^{1/d}$, and $\ell_{b,i}$ such that $\mathrm{vol}_{d-1}(C_{\ell_{b,i}}) \leq 2\ \mathrm{vol}_d(M_{i,-})^{(d-1)/d}$ and $\ell_{b,i} \geq -1/2\ \mathrm{vol}_d(M_{i,-})^{1/d}$. ($C_{\ell_{t,i}}$ and $C_{\ell_{b,i}}$ are here subsets of hyperplanes normal to the i-th dimension.) If after the (d-1)-st step we end up with a set $M_d$ containing more than 2/3n marked points we cut $M_d$ exactly in half by a cut normal to dimension d. The cut surface can this time be limited by all the 2(d-1) cuts done before and will have a (d-1)-dimensional volume of at most

$$\prod_{i=1}^{d-1} (\ell_{t,i} - \ell_{b,i}) \leq \prod_{i=1}^{d-1} [1/2\ \mathrm{vol}_d(M_{i,+})^{1/d} + 1/2\ \mathrm{vol}_d(M_{i,-})^{1/d}]$$

Thus the total volume of C is at most

$$\mathrm{vol}_{d-1}(C) \quad \sum_{i=1}^{d-1} [2\ \mathrm{vol}_d(M_{i,+})^{(d-1)/d} + 2\ \mathrm{vol}_d(M_{i,-})^{(d-1)/d}]$$

$$+ \sum_{i=1}^{d-1} [1/2\ \mathrm{vol}_d(M_{i,+})^{1/d} + 1/2\ \mathrm{vol}_d(M_{i,-})^{1/d}]$$

We know that for all $1 \leq i \leq d$ $\mathrm{vol}_d(M_{i,+}) + \mathrm{vol}_d(M_{i,-}) = \mathrm{vol}_d(M_i) \leq V := \mathrm{vol}_d(M)$. Also, the function $x^\epsilon + (a-x)^\epsilon$ is maximized at $x = a/2$ for all $\epsilon > 0$. Thus

$$\mathrm{vol}_{d-1}(C) \leq (d-1) \cdot 4 (V/2)^{(d-1)/d} + (V/2)^{(d-1)/d}$$

$$= (2d-3/2) \cdot 2^{1/d} \cdot V^{(d-1)/d}. \qquad \square$$

Note that for d = 2 the constant factor in the upper bound on the volume of the cut is greater in Theorem 4 than in Theorem 3.

## 3. I/O CONVENTIONS AND BOUNDS ON AREA

Two I/O conventions prevail in the literature:

a) Times and locations at which the input and output bits are available at I/O ports are independent of the input [BK, Vu]. Such chips are called when- and where-oblivious in [LS]. Several linear lower bounds on the area for when- and where-oblivious chips are known ([BK] for integer multiplication, [Vu] for transitive functions, [B] for surjective functions which depend on all arguments).

b) Only the locations are fixed. In this case the chip actively requests the inputs at its input ports. We distinguish two cases here: Either the chip requests inputs by name, i.e., it may request $x_{17}$, and hence the chip environment has to adapt to input dependent requests, e.g., by means of a random access memory for each port. Output bits are also produced at input-dependent times, and the chip identifies each output value when it produdes it. [LS] call such chips where-oblivious.

We add the following other possibility:

c) The chip may only request the next input bit at each port, i.e., a queue is associated with each port and the ordering in these queues is independent of the input. Similarly the order in which output bits are produced at each output port is fixed. We call such chips strongly where-oblivious.

Definition [Vu]: A function $f(x_1,\ldots,x_n,s_1,\ldots,s_p):\{0,1\}^{u+p} \to \{0,1\}^n$ is transitive of degree n if there is a transitive permutation group G operating on $\{1,\ldots,n\}$ such that for every $g \in G$ there is an assignment $\alpha$ to the control inputs $s_1,\ldots,s_p$ such that $y_{g(i)} = x_i$ for all i. Here $(y_1,\ldots,y_n) = f(x_1,\ldots,x_n,s_1,\ldots,s_p)$. We call $x_1,\ldots,x_n$ the permutation inputs, and $y_1,\ldots,y_n$ the (permutation) outputs in the sequel.

Theorem 5: Let f be a transitive function of degree n. Consider any strongly where-oblivious chip computing f with storage area $A_S$, input area $A_I$, and output area $A_O$. Then $A_I A_O (A_S + \lambda^2/v) \geq (\lambda^6/v^3)\cdot n$. In particular

$$A \geq (\lambda^2/v)\cdot(n^{1/3}-1).$$

Proof: Let $f(x_1,\ldots,x_n,s_1,\ldots,s_p) = (y_1,\ldots,y_n)$ be a transitive function of degree n (cf. [Vu]) with p control inputs. Assume that the

chip under consideration has $k_I$ input and $k_O$ output ports. Assume further that through input port $i$ the bits $x_{in(i,1)},\ldots,x_{in(i,p_i)}$ are read in this order. Here $p_1+\ldots+p_{k_I} = n$. We do not consider the times at which the $s_i$ are read in. Analogously define the outputs $y_{out(j,1)},\ldots,y_{out(j,q_j)}$, i.e. bits $y_{out(j,1)},\ldots,y_{out(j,1)},\ldots,y_{out(j,q_j)}$ are produced at output port $j$ in that order. Again $q_1 + \ldots + q_{k_o} = n$. Let $Out_1 = \{y_{out(j,1)};\ 1 \le j \le k_O\}$ be the set of output bits that have to be output first.

Let $G$ be the group computed by $f$. Consider any fixed $y \in Out_1$. Then $\{g^{-1}(y); g \in G\}$ is a multiset with exactly $|G|$ elements; moreover, each $x_i, 1 \le i \le n$, appears exactly $|G|/n$ times in that multiset. Hence each $x_i$ appears exactly $k_O|G|/n$ times in the multiset $G^{-1}(Out_1) = \{x_j; g(x_j) \in Out_1,\ g \in G\}$.

Now let for $b \in N$ be $In_b = \{x_{in(i,\ell)}; 1 \le i \le k_I,\ 1 \le \ell \le \min(b,p_i)\}$, i.e., $In_b$ is the set of all $x_i$ that are in the first $b$ positions of the queues associated with all input ports. Certainly $|In_b| \le k_I b$.

We define $witness_b(g)$ for all $g \in G$ to be an arbitrary element of $In_b \cap g^{-1}(Out_1)$, if one exists. Let $b_{all} = \min\{b;\ witness_b(g)$ exists for all $g \in G\}$. We prove the following claim.

Claim: At least $b_{all}-1$ bits have to be stored by the chip.

Proof of Claim: Assume the chip is only able to store $b < b_{all}-1$ bits. Then there is some $g \in G$ such that $witness_{b+1}(g)$ does not exist. Set $s_1,\ldots,s_p$ such that the chip computes $g$. Since $witness_{b+1}(g)$ does not exist, we have $In_{b+1} \cap g^{-1}(Out_1) = \emptyset$, i.e., at least $b+2$ inputs have to be read before the first output is produced. Thus the chip must be able to store at least $b+1$ bits. This is a contradiction. □

Note that by our counting argument above, we have $|\{g;\ x_i \in g^{-1}(Out_j)\}| = k_0|G|/n$ for any $x_i$ with $1 \le i \le n$. Thus $|\{g;\text{witness}_b(g) \text{ exists}\}| \le k_I b \cdot k_0|G|/n$, by our upper bound on the size of $In_b$. For $b = b_{all}$ the left side of this inequality becomes $|G|$ and we get

$$|G| \le k_I b_{all} \cdot k_0 |G|/n.$$

Since $A_I \ge \lambda^2 k_I/\nu$, $A_0 \ge \lambda^2 k_0/\nu$ and $A_S \ge \lambda^2 (b_{all}-1)/\nu$ the theorem follows. The lower bound on A is a consequence of the formula $A \ge \max(A_I, A_0, A_S)$. □

For certain transitive functions the lower bound given in Theorem 5 can be matched up to a constant factor with an upper bound. We consider here the function $f_{CS}$ computing cyclic shifts, i.e. the function $f_{CS}(x_0,\ldots,x_{n-1},k) = (y_0,\ldots,y_{n-1})$ where $0 \le k \le n-1$ and $y_i = x_{(i-k) \bmod n}$.

Theorem 6: There is a strongly where-oblivious chip computing $f_{CS}$ with area $O(n^{1/3})$.

Sketch of Proof: Since we are only concerned with an upper bound on the area we will not take special care to make the chip fast.

We give the chip $n^{1/3}$ input ports. The i-th input port receives the inputs $x_{i\cdot n^{2/3}},\ldots,x_{(i+1)\cdot n^{2/3}-1}$ in this order ($0 \le i \le n^{1/3}-1$). These are $n^{2/3}$ inputs per port. We will give the chip slightly fewer output ports, namely $n/(n^{2/3}+n^{1/3})$ output ports. Each output port produces $n^{2/3}+n^{1/3}$ output bits. Output port j produces the bits $y_{j\cdot(n^{2/3}+n^{1/3})},\ldots, y_{(j+1)\cdot(n^{2/3}+n^{1/3})-1}$ for $0 \le j \le n/(n^{2/3}+n^{1/3})-1$.

The idea of this arrangement is, informally, that for each amount of shift there will be one output port j and one input port i, such that the first input bit to be read by input port i has an index that it at most $n^{1/3}$ smaller than the first output bit to be produced by output port j. Thus, if we start reading inputs from input port i

we have to store at most $n^{1/3}$ input bits before we can directly output the input bits read starting at output port j. We continue reading inputs in a clockwise fashion and directly produce the corresponding output. At the end we produce the input bits read and stored at the beginning.

Formally, we proceed as follows. Suppose that we use k to denote the amount of shift. Define $j = j(k)$ such that

$$(j(k)-1)\cdot n^{1/3} < k \bmod n^{2/3} \leq j(k)\cdot n^{1/3}$$

and let $i(k) = j(k) - (k \operatorname{div} n^{2/3})$. We start reading at input port $i(k)$ and producing output at output port $j(k)$. Then the first bit to be output is $y_{j(k)\cdot(n^{2/3}+n^{1/3})} = x_{[j(k)\cdot(n^{2/3}+n^{1/3})-k] \bmod n} = x_{[i(k)\cdot n^{2/3}+m] \bmod n}$ where $m = j(k)\cdot(n^{2/3}+n^{1/3})-k-i(k)n^{2/3} = j(k)n^{1/3} - k \bmod n^{2/3} \in [0, n^{1/3}-1]$.

Thus we have shown that for the above algorithm $A_I, A_O, A_S = O(n^{1/3})$. We still have to argue that the computation necessary for selecting the correct routing in dependence of k can be done in small area.

We propose the following layout (see Figure 2). Input ports are located at the leaves of an H-tree with $n^{1/3}$ leaves. Similarly, the output ports are located at the leaves on an H-tree with $n/(n^{2/3}+n^{1/3})$ leaves. In addition, there is a memory unit (which may be realized as a shift register) which can store up to $n^{1/3}$ bits and there is a control logic part. The control logic part receives the amount of shift, say k, as input and computes i(k) and j(k) as defined above. It then generates a number of control signals for the input and output trees and for the memory. All control signals for the input and output trees are sent into the trees at the roots and then propagate down the tree to the leaves. Clearly we only need a constant amount of control circuitry for each node in the tree and thus the area of

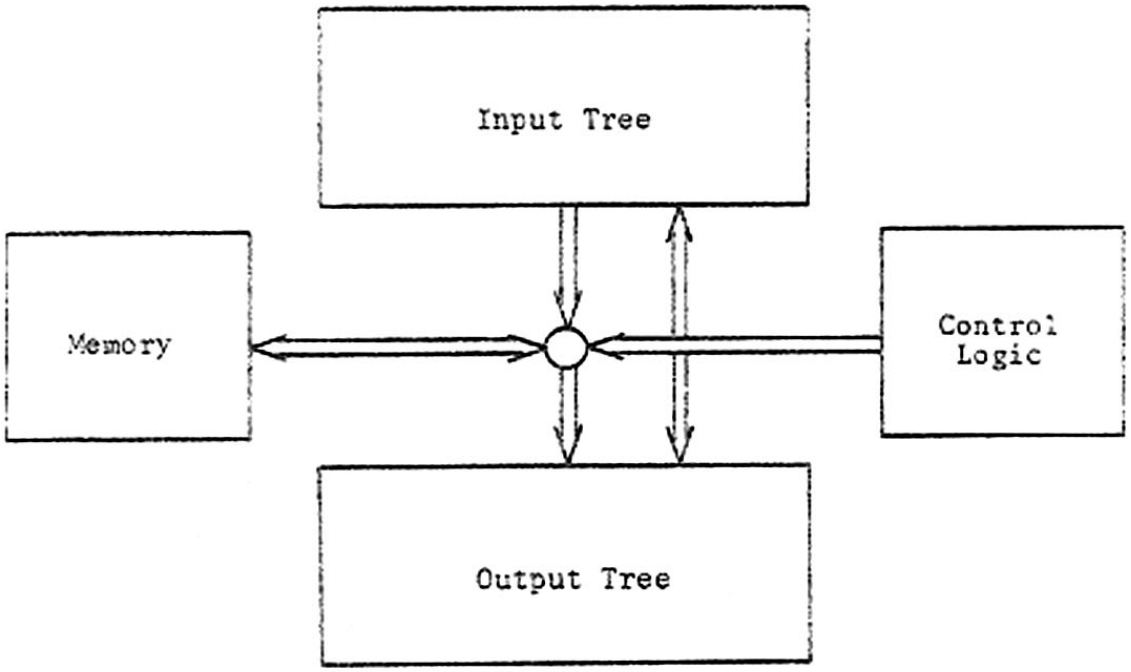



Figure 2: Layout for cyclic shift chip of small area

the input and output trees stays linear in the number of leaves and hence is $O(n^{1/3})$. The control signals for the trees are used to open one path from a leaf to the root for communication and to close all other paths. The chip now operates as follows. After computing i(k) and j(k) the chip starts reading inputs at port i(k). The first n-1 (m as defined above) input bits are sent to the memory and stay there almost to the and of the computation. The m-th input bit and all following bits are sent to output port j(k). Whenever an input queue is exhausted reading continues at the next input port and whenever an output port queue is full output is continued at the next output port. Finally, the computation is completed by outputting the bits stored in the memory at the appropriate port.

The bound on area is now easily established. Memory, input and output trees have area $O(n^{1/3})$ and the control logic has area $O((\log n)^a)$ for some small constant a (a = 2 suffices). Thus total area is $O(n^{1/3})$. The chip operates in time T = O(n). □

Theorem 6 shows that there are strongly where-oblivious chips for $f_{CS}$ that are strictly smaller than any where- and when-oblivious chip for $f_{CS}$. If we consider all where-oblivious chips, we can achieve a further reduction in area.

Lemma 7: There is a where-oblivious chip computing $f_{CS}$ with $A = O(\log n)$.

Proof: The chip has one input port and one output port. Outputs are always produced in the order $y_0,\ldots,y_{n-1}$. Inputs are requested in the right order to be produced directly at the output port. The necessary computation for requesting the input bits can be done in area $A = O(\log n)$. □

Note that it is not known yet whether there is a strongly where-oblivious VLSI circuit computing cyclic shifts that fulfills $AT^2 = O(n^2)$ and $A = O(n^{1/3})$ simultanously. The chip described in the proof of Theorem 6 is not optimal with respect to the $AT^2$ complexity measure.

## 4. BOUNDS ON ENERGY CONSUMPTION

Thompson [Th] derives a lower bound on energy consumption based on the assumption that one unit of energy is consumed by one unit of chip area every time that it is involved in the transmission of a signal. However, there is a definite difference between transmitting a 0 followed by a 0, i.e., maintaining a state, and transmitting a 0 followed by a 1, i.e., switching a state. In the first case only "static" energy is expended for maintaining the value. In the second case "switching" energy is expended in addition, for changing the value. Whereas static energy consumption is dominant in the NMOS process, switching energy consumption is dominant in processes like CMOS. Moreover, switching energy is the energy concept that is more closely related to computational complexity, and it is the central energy concept introduced in [MC]. Thompson bounds static energy from below.

We derive a lower bound on switching energy consumption based on the following assumption:

Every unit of chip area on every layer of the chip consumes one unit of energy every time it changes its state (from 0 to 1 or vice versa).

Our argument is based on the following ideas:

1) Consider any cut through the chip. If considerable amount of information has to be transported across the cut in small amount of time then the circuit components intersecting the cut must change state very frequently. This is made precise in Lemmas 10 - 13.

2) In any chip for a transitive function we can identify a large set of cuts such that considerable amout ($\Omega(n)$) of information has to be transported across most (namely $\Omega(n/T)$) cuts in the set. This is made precise in Lemma 8.

   Summing the state changes associated with each cut over the set of cuts yields the lower bound.

For the proof of Theorem 9 we make the following assumption (even with these assumptions the proof is quite involved):

a) chips are when- and where-oblivious

b) no two inputs or outputs share a port

c) the chip is laid out on the rectangular grid of mesh size $\lambda$

, i.e. gates and ports have degree at most 4 and are located in the centers of cells of the grid and wires run either vertically or horizontally and connect adjacent cells of the grid. The area of the chip is then the number of cells which are occupied by gates, ports or wires. This model was used in [Th].

Consider now a computation of the chip and concentrate on any grid cell. At any point of time the circuit component occupying that grid cell carries either a logic value 0 or a logic value 1. The switching energy consumed by the cell is the number of times it switches its logic state. The switching energy consumption of the computation is the sum of the consumptions of all the cells of the chip. Finally, the worst case switching energy, which we denote by E, is the maximal energy consumption on any input.

Theorem 9: Let f be a transitive function of degree n. Let E be the worst case switching energy consumed by any when- and where-oblivious chip computing f. Let A be the (active) chip area and let T be the computing time. Then

$$c_1AT^2 \geq ET \geq \frac{c_2n^2}{\log\frac{c_3AT^2}{n^2}} \geq 0$$

for appropriate constants $c_1, c_2, c_3 > 0$.

Proof: Consider any when- and where-oblivious chip computing f. The transitive function f has n + p inputs and n outputs $f(x_1,\ldots,x_n, s_1,\ldots,s_p) = (y_1,\ldots,y_n)$. As in Section 3 we call $s_1,\ldots,s_p$ the control inputs, $x_1,\ldots,x_n$ the permutation input bits and $y_1,\ldots,y_n$ the permutation output bits. The upper bound on ET is trivial. The lower bound argument has three steps. In step 1 we identify a large set of cuts with large information transfer. In the second step we count state changes and in the last step we relate state changes and energy consumption.

Step 1: Identifying a large set of cuts with large information transfer.

In the first step we cut the chip according to a technique devised by [Th]. We define cuts $C_1,\ldots,C_Q$ bisecting the chip as shown in Figure 3. (Q to be determined later).

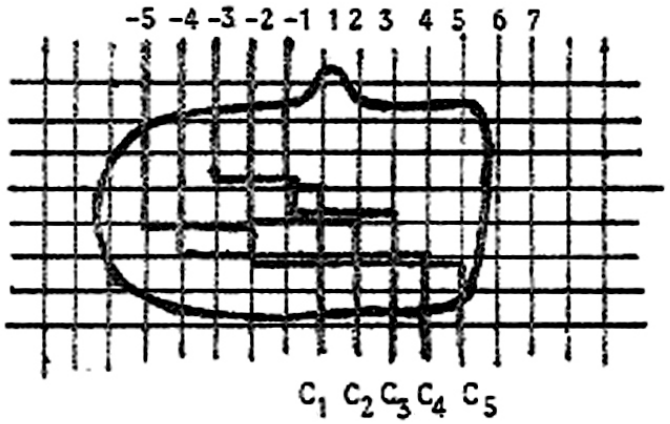
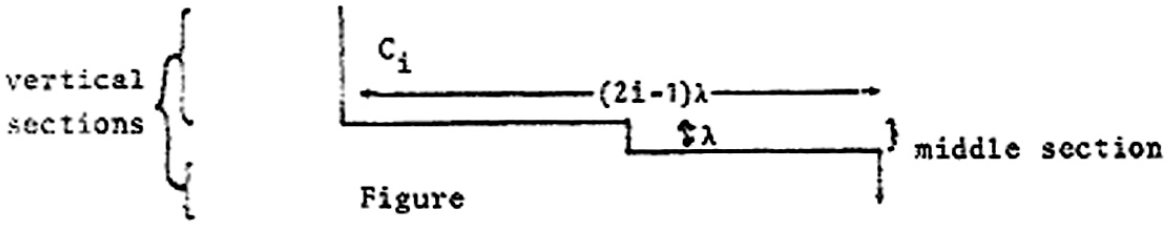



Figure 3

Hereby each cut consists of several sections as shown in Figure 4.




Figure

The middle section of cut $C_i$ has a length of at most $2i\lambda$. The vertical sections of all cuts $C_i$ are disjoint. Each cut induces a partition of the permutation input bits A into sets X,Y and of the output bits B into sets I,J such that $|X| + |I| = |Y| + |J| = n$. The input bits in X and the output bits in I have their ports to the left of the cut and the input bits in Y and the output bits in I have their ports to the right of the cut. The existence of cuts $C_1,C_2,\ldots$ can be seen as follows. Move a vertical line from left to right across the chip until $|X| + |I| \geq n$. Label this grid line by +1 and label all other grid lines as indicated in Figure 3, i.e. grid lines are labelled ..., -2, -1, +1, +2, ... from left to right. By the definition of the grid line labelled +1 we have: If $i \in \mathbb{N}$ then the

number of input and output ports to the left of grid line - i is less than n and the number of ports to the left of line + i is at least n. The cut $C_i$ uses grid line - i for its upper vertical section and + i for its lower vertical section. The horizontal part of cut $C_i$ is found as follows. Conceptually move a horizontal line connecting the two vertical parts from bottom to top and always count the number of ports to the left of the cut. This number is $< n$ when the horizontal line is in its down-most position and it is $\geq n$ when it is in its up-most position. Thus there has to be an intermediate position which does the job.

Let G be the transitive permutation group computed by f. For $g \in G$ and cut $C_i$ let $M(g,i)$ be the set of inputs which have to be output on the other side of the cut, i.e. $M(g,i) = \{x;\ x \in X,\ g(x) \in J$ or $x \in Y$ and $g(x) \in I\}$ where $X,Y,I,J$ are defined with respect to cut $C_i$.

Lemma 8: There is $g_o \in G$ such that $|M(g_o,i)| \geq n/7$ for at least $Q/8$ cuts $C_i$

Proof: Assume w.l.o.g. that $|X| \geq n/2$. Note first that $|X| + |Y| = n = |I| + |J|$, $|X| + |I| = n = |Y| + |J|$ implies $|X| = |J|$. Note next that for $x \in X$ there are exactly $|G||J|/n$ group elements g with $g(x) \in J$. Hence

$$\sum_{g \in G} M(g,i) \geq n/4\ |G|$$

and therefore

$$\sum_{i=1}^{Q} \sum_{g \in G} M(g,i) \geq n|G| \cdot Q/4$$

Thus there is $g_o \in G$ such that $\sum_{i=1}^{Q} |M(g_o,i)| \geq n\ Q/4$. This implies our claim. Assume otherwise. Then

$$\sum_{i=1}^{Q} |M(g_o,i)| < \frac{Q}{8} \cdot n + (Q-\frac{Q}{8}) \frac{n}{7} = \frac{n\,Q}{4},$$

a contradiction. □

In the sequel we denote the set of cuts identified by Lemma 8 by $\Phi$.

Step 2: Counting state changes.

Before we start the actual argument we discuss an encoding that expresses bit strings in terms of the state changes happening in them.

Definition: Let w be an arbitrary bit string. Assume that w starts with a 0; the definition being symmetric in the other case. Then $w = 0^{\ell_1}\,1^{\ell_2}\,0^{\ell_3} \ldots {}^{\ell_t}$ for some integers $\ell_1,\ell_2,\ldots,\ell_t$. Let

a) $s(w) := t$ be the number of state changes in bit strings w plus one.

b) bin($\ell$) be the bit string obtained from the binary representation of integer $\ell$ by substituting 00 for 0 and 11 for 1.

c) compress(w) = 0 bin($\ell_1$) 01 bin($\ell_2$) 01 bin($\ell_3$) 01 ... 01 bin($\ell_t$),

i.e. compress(w) is the concatenation of a 0 (indicating that w starts with a 0) followed by the "binary" representations of integers $\ell_1,\ldots,\ell_t$ separated by delimiter 01. □

Example: compress(00011) = 0 11 11 01 11 00

Lemma 9: $|\text{compress}(w)| \le 4s(w) + 2s(w) \log \frac{|w|}{s(w)}$

Proof: $|\text{compress}(w)| \le 2s(w) + 2\sum_{j=1}^{s(w)} (1 + \log \ell_j)$

$$\le 4s(w) + 2\sum_{j=1}^{s(w)} \log \ell_j$$

$$\le 4s(w) + 2\log((\sum_{j=1}^{s(w)} \ell_j)/s(w))^{s(w)}$$

$$\le 4s(w) + 2s(w)\log \sum_{j=1}^{s(w)} \ell_j/s(w)$$

$$\le 4s(w) + 2s(w)\log \frac{|w|}{s(w)}$$

Here the next to last inequality is derived using the well known fact that the geometric mean is no greater than the arithmetic mean. □

We will count state changes happening on the vertical section of the cuts. Let $L_i$ be the number of circuit components crossing $C_i$ and let $\ell_i$ be the number of circuit components crossing the vertical sections of $C_i$. Then $\ell_i \ge L_i - c_o i$ for some appropriate constant $c_o > 0$.

We now consider an arbitrary but fixed cut $C_i$ in $\Phi$. $|M(g_o,i)| \ge n/7$ by Lemma 8. Select any $n/7$ permutation inputs which have to be transported across cut $C_i$, choose an arbitrary but fixed assignment $\alpha$ for the remaining $6n/7$ permutation inputs and let the $n/7$ chosen permutation inputs vary in all $2^{n/7}$ possible ways. Thus we generate $2^{n/7}$ different bit sequences $(w_{ij}(\alpha), h_{ij}(\alpha))$ across $C_i$. Here $w_{ij}(\alpha)$ summarizes all bits crossing $C_i$ in the vertical sections and $h_{ij}(\alpha)$ summarizes all bits crossing $C_i$ in the middle section ($1 \le j \le 2^{n/7}$). To simplify notation we will write $w_{ij}$, $h_{ij}$ instead of $w_{ij}(\alpha)$, $h_{ij}(\alpha)$ in Lemma 10 and 11.

Lemma 10: Let $\{w_1,\ldots,w_r\}$ be a set of r different bit strings. Then

$$\sum_{1\le j\le r} |\text{compress}(w_j)| \ge (\lfloor \log r \rfloor - 2)\, r.$$

Proof: The mapping $w \to \text{compress}(w)$ is injective. Thus

$$\sum_{1\le j\le r} |\text{compress}(w)| \ge \sum_{1\le j<\lfloor \log r\rfloor} j2^j + (r-2^{\lfloor \log r\rfloor})\lfloor \log r\rfloor \ge (\lfloor \log r\rfloor - 2)r.$$

□

Lemma 11: $\sum_{1\le j\le 2^{n/7}} |\text{compress}(w_{ij})| \ge (\frac{n}{7} - c_o iT - 3)2^{n/7}$.

Proof: Since the length of the middle section of cut $C_i$ is at most $2iA$ there are at most $c_o i$ circuit components that intersect the middle section of cut $C_i$. Thus there are at most t, $1 \le t \le 2^{c_o iT}$, different sequences $h_{ij}$. For each such sequence there is a number $u_z$ of different sequences $w_{ij}$ such that $(w_{ij},h_{ij})$ is generated by some input as described above. We have $U := \sum_{1\le k\le t} u_k \ge 2^{n/7}$ and by Lemma 10

$$\sum_{1\le j\le 2^{n/7}} |\text{compress}(w_{ij})| \ge \sum_{1\le k\le t} (\log u_k - 3)u_k$$

$$\ge U \log(U/t) - 3U$$

$$\ge 2^{n/7}(n/7 - c_o iT - 3).$$

Here we used the fact that $\sum_{1\le k\le t} u_k \log u_k$ is minimum if for all $k, u_k = U/t$. □

Using the upper bound on $|\text{compress}(w)|$ derived in Lemma 9 we can now give a lower bound on the average number of state changes in all sequences $w_{ij}(\alpha)$. Let

$S_i = \sum_{\alpha=1}^{2^{6n/7}} \sum_{j=1}^{2^{n/7}} s(w_{ij}(\alpha))/2^n$ be the average number of state changes occuring at cut $C_i$.

Lemma 12: $(n/7 - c_0 iT - 3) \leq 4S_i + 2S_i \log \frac{T\ell_i}{S_i}$

Proof: We have

$$2^n(n/7 - c_0 iT - 3) \leq \sum_\alpha \sum_j |\text{compress}(w_{ij}(\alpha))|, \text{ by Lemma 11}$$

$$\leq \sum_\alpha \sum_j (4s(w_{ij}(\alpha)) + 2s(w_{ij}(\alpha))\log(|w_{ij}(\alpha)|/s(w_{ij}(\alpha)))) \text{ by Lemma 9}$$

$$\leq 4 \cdot S_i \, 2^n + \sum_\alpha \sum_j 2s(w_{ij}(\alpha))\log(T\ell_i/S_i)$$

since $|w_{ij}(\alpha)| \leq T\ell_i$ for all $j$ and $\alpha$. Finally observe that the sum above is maximum if $s(w_{ij}(\alpha)) = S_i$ for all $j$ and $\alpha$. Thus

$$2^n(n/7 - c_0 iT - 3) \leq 4 S_i \, 2^n + 2 S_i \, 2^n \log(T\ell_i/s(w_{ij}(\alpha))) \qquad \square$$

We now sum over all cuts $C_i$ in $\Phi$. Let $\ell := \sum_{i=1}^{Q/8} \ell_i$ and $S := \sum_{i=1}^{Q/8} S_i$. Thus $S$ is the average number of state changes in the sequences $w_{ij}$ across all cuts in $\Phi$.

Lemma 13: $\sum_{1 \leq i \leq Q/8} (n/7 - c_0 QT - 3) \leq 4S + 2S \log (T\ell/S)$.

Proof: We manipulate the formula in a way similar to the proof of Lemma 12. $\square$

Choosing $Q = (n-21)/14\, c_0 T$ yields

Lemma 14: For suitable constants $c_2, c_3 > 0$ and sufficiently large $n$ we get

$$ST \geq \frac{c_2 n^2}{\log \frac{c_3 T^2 \ell}{n^2}}$$

Proof: Substituting the value for $Q$ into Lemma 13 yields

$$\left(\frac{n-21}{14}\right)^2 \frac{1}{c_0 T} \leq 4S \left(1 + \frac{1}{2} \log \frac{T\ell}{S}\right). \qquad (*)$$

Thus

$$\log S \geq \log \left(\left(\frac{n-21}{14}\right)^2 \frac{1}{4c_o T}\right) - \log\left(1 + \frac{1}{2}\log\frac{T\ell}{S}\right).$$

Applying the inequality $\log(1 + x) \leq x$ we get

$$\log S \geq 2 \log \left(\left(\frac{n-21}{14}\right)^2 \frac{1}{4c_o T}\right) - \log(T\ell).$$

Substituting this into (*) proves the lemma. □

We are left with relating S to the switching energy and bounding $\ell$ from above. Since $\ell_i$ is the number of wires crossing the vertical part of cut $C_i$ we clearly have $\ell = O(A)$. Furthermore $S = O(E + A)$ since we can charge a state change on a wire across cut $C_i$ to the $\Omega(1)$ area within the strip of width $\lambda/2$ on both sides of cut $C_i$. Note that we have to add A here to count the state changes in $w_{ij}$ that can occur when switching from one wire crossing $C_i$ to the next and also to account for the fact that s(w) is defined as 1 + the number of state changes in string w. Thus

$$AT \geq \frac{c_2 n^2}{\log \frac{c_3 AT^2}{n^2}} - c_4 A.$$

If we assume that on each circuit component there will be at least one state change, say, during initialization, then the last term can be omitted proving the theorem. □

Note that if the chip is $AT^2$ optimal, i.e., if $AT^2 = O(n^2)$ then the above lower bound on E is tight up to a constant factor. This shows that $AT^2$ optimal chips use their computing resources to capacity (up to a constant factor), i.e. in any time unit a fixed fraction of the chip has to be active.

In Theorem 9 we put some restriction on the circuit computing a transitive function. First it had to be when- and where-oblivious. It turns out that this restriction is not necessary. The cutting

argument used in Theorem 9 also holds for strongly where-oblivious chips. (See [Ko,Le] for details.) Second we assumed that each input resp. output bit has its own private port. This assumption can be dropped as well, at the expense of decreasing the constant factor in the lower bound. Specifically, if we assume that at most cn I/O bits share a port, where $0 < c < 1$, then we can find a sequence of cuts each cutting the chip such that at most $(c+1)n/2$ bits enter respectively leave the chip on each side of the cut. Clearly, if more than cn bits share a port then $T > cn$. Since the permutation inputs respectively outputs can assume all possible configurations we therefore have $E = \Omega(n)$ and thus $ET \geq \Omega(n^2)$. Finally, we assumed that the chip is laid out on a rectangular grid. This restriction can be overcome by simple superimposing a grid of mesh size $\lambda$ upon the layout. Then $L_i$ has to be defined as the number of circuit components which intersect cut $C_i$ and $\ell_i$ is the number of circuit components which intersect the vertical sections of $C_i$.

## 5. ACKNOWLEDGEMENTS

Bob Tarjan pointed out to us how to generalize the cutting theorem (Theorem 3) to dimensions $d > 2$.

## 6. REFERENCES

[Ba] G.M. Baudet "On the Area Required by VLSI Circuits," CMU Conference on VLSI Systems and Computations (1981), 100-107.

[BK] R.P. Brent, H.T. Kung, "The Area-Time Complexity of Binary Multiplication," JACM 28,3 (July 1981), 521-534.

[Br] R.E. Bryant "A Switch-Level Model of MOS Logic Circuits," in VLSI 81 (ed. John Gray), Academic Press (August 1981), 329-340.

[CM] B. Chazelle, L. Monier, "A Model of Computation for VLSI with Related Complexity Results," 13th Ann. Symp. on Theory of Computing, ACM (May 1981), 318-325.

[Ko] R. Kolla, "Untere Schranken für VLSI", Diplomarbeit, Fachbereich 10, Universität des Saarlandes, Saarbrücken, West-Germany (1982).

[Le] T. Lengauer, "The Communication Complexity of VLSI Circuits", Proceedings of an AMS Short Course on Computer Communication, Denver, CO (1985).

[LS] R.J. Lipton, R.S. Sedgewick, "Lower Bounds for VLSI", 13th Ann. Symp. on Theory of Computing, ACM (May 1981), 300-307.

[LT] R.J. Lipton, R.E. Tarjan, "A Separator Theorem for Planar Graphs," SIAM J. Appl. Math. Vol 36 (1979), 177-189.

[MC] C. Mead, L. Conway, Introduction to VLSI Systems, Addison Wesley (1980).

[Ro] A.L. Rosenberg, "Three Dimensional Integrated Circutry", Proceeding of the CMU-Conference on VLSI-Systems and Computations, Pittsburgh, PA (1981), 69-80.

[Sa] J.E. Savage, "Planar Circuit Complexity and the Performance of VLSI Algorithms," CMU-Conference on VLSI Systems and Computations (1981), 61-68.

[Th] C.D. Thompson, "A Complexity Theory for VLSI," Ph. D. Thesis, Carnegie-Mellon University, Pittsburgh, PA (1980).

[Vu] J. Vuillemin, "A Combinatorial Limit to the Computing Power of VLSI Circuits," 21st Symp. on the Foundations of Computer Science, IEEE (1980).