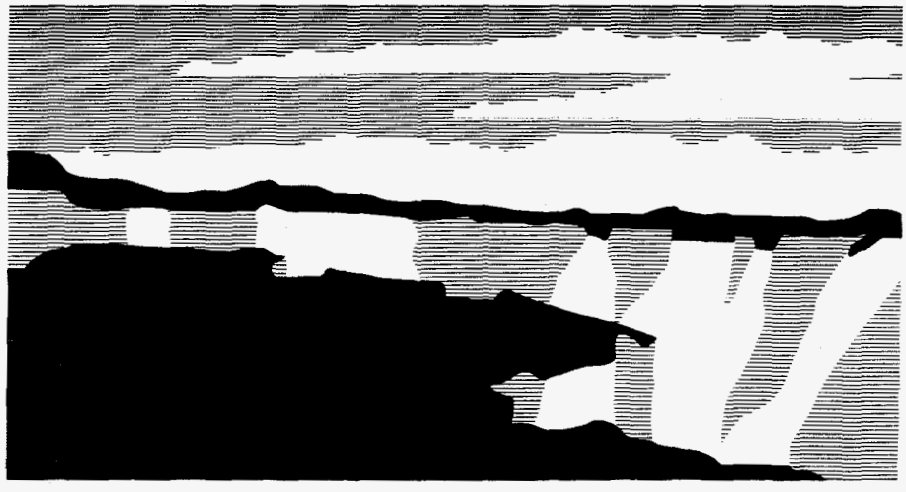
LA-UR- 96-3576

CONF-9611108--1

Title: NEW VLSI COMPLEXITY RESULTS FOR THRESHOLD GATE COMPARISON

Author(s): Valeriu Beiu

Submitted to: 3rd Brazilian Symposium on Neural Networks Recife
November 12-14, 1996
Recife, Brazil



DISCLAIMER

New VLSI Complexity Results for Threshold Gate COMPARISON

Valeriu Beiu *

Los Alamos National Laboratory, Division NIS–1
Los Alamos, New Mexico 87545, USA

Abstract— **The paper overviews recent developments concerning optimal (from the point of view of *size* and *depth*) implementations of COMPARISON using threshold gates. We detail a class of solutions which also covers another particular solution, and spans from constant to logarithmic *depths*. These circuit complexity results are supplemented by fresh VLSI complexity results having applications to hardware implementations of neural networks and to VLSI-friendly learning algorithms. In order to estimate the *area* (*A*) and the *delay* (*T*), as well as the classical AT^2, we shall use the following 'cost functions': (*i*) the connectivity (*i.e.*, sum of *fan–ins*) and the *number–of–bits* for representing the *weights* and *thresholds* are used as closer approximations of the *area*; while (*ii*) the *fan–ins* and the *length* of the wires are used for closer estimates of the *delay*. Such approximations allow us to compare the different solutions —which present very interesting *fan–in* dependent *depth–size* and *area–delay* tradeoffs — with respect to AT^2.**

Keywords— **Threshold gates, COMPARISON, VLSI complexity, circuit complexity.**

I. INTRODUCTION

IN this paper we shall consider feedforward neural networks (NNs) made of linear threshold gates (TGs). A neuron (*i.e.*, linear TG) will compute a Boolean function (BF) $f: \{0, 1\}^n \rightarrow \{0, 1\}$, where the input vector is

$$Z = (z_0, \ldots, z_{n-1}) \in \{0, 1\}^n$$

and

$$f(\mathbf{Z}_k) = sgn\left(\sum_{i=0}^{n-1} w_i z_i + \theta\right),$$

with the synaptic *weights* $w_i \in \mathbb{R}$, $\theta \in \mathbb{R}$ known as the *threshold*, and *sgn* the sign function. The cost functions commonly associated to a NN are *depth* (*i.e.*, number of edges on the longest input to output path, or number of layers) and *size* (*i.e.*, number of neurons).

This research work has been done while the author has been with the Centre for Neural Networks, Department of Mathematics, King's College London, Strand, London WC2R 2LS, UK. It has been entirely supported under the Human Capital and Mobility programme of the European Community by an Individual Research Training Fellowship for a project entitled: "Programmable Neural Arrays" (contract no. **ERBCHBICT941741**).

* Dr. Beiu is on leave of absence from the "Politehnica" University of Bucharest, Computer Science Department, Spl. Independentei 313, RO-77206 Bucharest, România.

The focus of this paper will be on NNs for computing COMPARISON. There are at least two reasons why COMPARISON is an interesting BF. The first one is that the results obtained for COMPARISON can be immediately extended to ADDITION (as the same BFs are used to compute the carries), and thus could lead to faster and/or smaller ADDERS (see [8, 9, 12] and the references therein). The second one is that COMPARISON is a particular $F_{n,m}$ function (functions of n inputs, having m groups of ones in their truth table [26]), namely it is a subclass of $F_{n,1}$; thus, potentially improving on the implementation of a larger class of functions — which has links with decision trees [24] and some constructive learning algorithms [6, 10, 15].

II. PREVIOUS RESULTS

Suppose that $X = x_{n-1}x_{n-2}\ldots x_1x_0$ and $Y = y_{n-1}y_{n-2}\ldots y_1y_0$ are two binary numbers (integers) of n bits each. The COMPARISON of the two numbers is defined as:

$$C_n^{>(\geq)} = C_n^{>(\geq)}(X, Y) = \begin{cases} 1 & \text{if } X > Y \ \ (X \geq Y) \\ 0 & \text{if } X \leq Y \ \ (X < Y) \end{cases},$$

the two BFs being *isobaric* (*i.e.*, BFs which can be implemented by TGs having identical *weights*, but different *thresholds* as $C_n^{\geq}(X, Y) = C_n^{>}(X+1, Y) = C_n^{>}(X, Y-1)$. It is known from previous work that COMPARISON cannot be computed by a single TG with polynomially bounded integer *weights*, but it can be computed by a single TG with exponentially large *weights* [28, 30]. In [2] a *depth*-2 NN with $O(n^4)$ TGs and polynomially bounded *weights* has been detailed. This result has been lately improved in several successive steps. The most important ones are shortly presented in the following four propositions.

Proposition 1 (Theorem 6 from [29]) *The COMPARISON function can be computed in a depth-3 neural network of size 3n with polynomially bounded integer weights.*

This constructive solution (we shall call it **SRK**) has a first layer of n AND gates computing $x_i \wedge \overline{y}_i$, and n OR gates computing $x_i \vee \overline{y}_i$, followed by a layer of $n-1$ AND gates:

$$B_k = (x_k \wedge \overline{y}_k) \wedge \left\{ \bigwedge_{j=k+1}^{n-1} (x_j \vee \overline{y}_j) \right\}$$

and a third layer having one OR gate:

$$C_n^{\geq} = (x_{n-1} \wedge \bar{y}_{n-1}) \vee \left(\bigvee_{k=0}^{n-2} B_k \right).$$

The *depth*-3 NN has:

$$size_{\mathbf{SRK}} = 3n - 1,$$

with *fan–in* $\leq n$, *thresholds* $\leq n$, and all the *weights* ± 1.

Proposition 2 (Lemma 1 from [7]) *The computation of COMPARISON of two n-bit numbers can be realised by a* Δ*-ary tree of size* $\boldsymbol{O}(n/\Delta)$ *and depth* $\boldsymbol{O}(\log n / \log\Delta)$ *for any integer fan-in* $2 \leq \Delta \leq n$.

Proposition 3 (Theorem 1 from [7]) *The COMPARISON of two n-bit numbers can be computed by a* Δ*-ary tree neural network with polynomially bounded* ($\leq n^k$) *integer weights and thresholds of size* $\boldsymbol{O}(n/\Delta)$ *and depth* $\boldsymbol{O}(\log n / \log\Delta)$ *for any integer fan-in* $3 \leq \Delta \leq c\log n$.

This constructive class of solutions (we shall call them **B_Δ**), firstly proposed in [4, 5], is based on decomposing COMPARISON in a tree like structure. The network has a first layer of 'partial' COMPARISONs $C_\Delta^{>}$ and $C_\Delta^{\geq}$ ($\lfloor \Delta/2 \rfloor$ bits[1] from X and $\lfloor \Delta/2 \rfloor$ bits from Y) followed by a Δ-ary tree of TGs combining these partial results. The fact that the BFs implemented by the nodes are linear separable functions was firstly proven in [4, 5] and will appear also in [12]. The network has:

$$depth_{\mathbf{B\_\Delta}} = \left\lceil \frac{\log n}{\log\Delta - 1} \right\rceil \qquad (1)$$

and

$$size_{\mathbf{B\_\Delta}} = \left\lceil \frac{4(n-1)}{\Delta - 2} \right\rceil - depth \qquad (2)$$

with *fan–in* $\leq \Delta$ (the TGs from the first layer have *fan–in* $= \Delta$, while all the other TGs have *fan–in* $= \Delta - 1$), and *weights* and *thresholds* $\leq 2^{\Delta/2}$ for any integer value of the *fan–in* in the range $3 \leq \Delta \leq n$.

If the *fan–in* is logarithmically bounded, the *weights* and the *thresholds* are polynomially bounded — this being the most interesting case. For *fan–ins* larger than the logarithm of the number of inputs, the *weights* and the *thresholds* are super-polynomial, while for *fan–ins* which are (almost) linear, the *weights* and the *thresholds* are exponential in the number of inputs.

Proposition 4 (Theorem 3 from [27]) *The size complexity of COMPARISON implemented by generalized symmetric functions is* $\Theta(n/\log n)$.

[1] In this paper $\lfloor x \rfloor$ is the floor of x, *i.e.* the largest integer less than or equal to x, and $\lceil x \rceil$ is the ceiling of x, *i.e.* the smallest integer greater or equal to x.
In this paper all the logarithms are to the base 2.

This constructive solution (we shall call it **ROS**) has a first layer of 'partial' COMPARISONs C_i (equivalent to $C_{2m}^{>}$) and $\tilde{C}_i$ (equivalent to $C_{2m}^{\geq}$) having $m = \lceil \log n \rceil + 1$ input bits from X and m input bits from Y. The first layer has $2\lceil n/m \rceil - 1$ TGs of *fan–in* $= 2m$. The second layer has $\lceil n/m \rceil - 1$ AND gates with *fan–in* $= 2, 3, \ldots, \lceil n/m \rceil$:

$$B_k = \tilde{C}_k \wedge \left(\bigwedge_{j=k+1}^{\lceil n/m \rceil} C_j \right).$$

The third layer has just one OR gate:

$$C_n^{\geq}(X, Y) = \bigvee_{k=1}^{\lceil n/m \rceil} B_k$$

with *fan–in* $= \lceil n/m \rceil$. This *depth*-3 NN has:

$$size_{\mathbf{ROS}} = 3 \left\lceil \frac{n}{\lceil \log n \rceil + 1} \right\rceil - 1,$$

with

$$fan\text{–}in_{\mathbf{ROS}} \leq \left\lceil \frac{n}{\lceil \log n \rceil + 1} \right\rceil$$

and *weights* and *thresholds* lower than $2^{\lceil \log n \rceil}$.

Proposition 5 (Corollary 2 from [32]) *The COMPARISON can be computed by a depth-2 linear threshold network of size* $2\lceil n/\lceil \sqrt{n} \rceil \rceil$, *with weight values at most* $2^{\lceil \sqrt{n} \rceil}$ *and with an upper bound of* $2\lceil \sqrt{n} \rceil + 1$ *for the maximum fan-in.*

This constructive solution (we shall call it **VCB**) has a first layer of 'partial' COMPARISONs ($\lceil \sqrt{n} \rceil$ bits from X and $\lceil \sqrt{n} \rceil$ bits from Y), and the second layer computes the carry-out of the 2–1 binary ADDITION with carry. The solution does not seem very attractive as it has exponentially growing *weights*. The complexity results of **VCB** are practically identical to those of a particular **B_Δ** solution: taking $\Delta = 2\lceil \sqrt{n} \rceil$, which leads to **B_2**$\lceil \sqrt{n} \rceil$, the resulting NN has *depth* $= 2$, *size* $= \lceil 2\sqrt{n} \rceil$, with *weights* and *thresholds* of at most $2^{\lceil \sqrt{n} \rceil}$.

For normal length COMPARISONs Vassiliadis *et al.* [32] claim improvements over **ROS** [27], but they do not mention [4, 5]. That is why we present in Table I the same results reported in [32], and the results of **B_Δ** for the particular cases considered there. It can be seen that both **VCB** and **B_Δ** achieve better performances than **SRK** and **ROS**. For *depth* $= 2$, **B_2**$\lceil \sqrt{n} \rceil$ outperforms **VCB** both for 32-bit and for 64-bit operand lengths (the best results are bolded). For *depth* $= 3$, **B_Δ**, which has lower *weights* and *fan–ins* than **VCB**, has (slightly) more gates. Still, **B_Δ** has two main advantages: (*i*) being a class of solutions one can also use it for other *depths* (an example for the case when *depth* $= 4$ can be seen in the last column of Table I); (*ii*) as the *weights* and the *fan–ins* are lower than those of **VCB**, the *area* of the final implementation should also be lower (see the discussions from Section III).

The exact *size* and *depth* of these solutions have been computed and are plotted in Fig. 1.

TABLE I
SIZE, WEIGHTS AND FAN-INS FOR SOME OPERAND LENGTHS.

Length	Solution	Vassiliadis [32]	**Beiu** [4, 5, 7]	Siu [29]	Roychowdhury [27]	**Vassiliadis** [32]	**Beiu** [4, 5, 7]	**Beiu** [4, 5, 7]
		depth = 2		*depth* = 3				*depth*=4
32–bit operands	*Size*	13	**12**	95	19	**18**	19	45 ✓
	Max. *weight*	64	**32**	32	64	16	**8**	3
	Max. *fan–in*	13	**12**	32	12	9	**8**	5
64–bit operands	*Size*	17	**16**	191	31	**26**	39	63
	Max. *weight*	256	**128**	64	128	64	**8**	5
	Max. *fan–in*	17	**16**	64	14	13	**8**	7

III. LINKS TO VLSI COMPLEXITY

The classical *depth* and *size* measures used in Section II, can be linked to the *delay* ($T \approx depth$) and the *area* ($A \approx size$) of a VLSI chip. It is known that a VLSI design is considered optimum when judged on a combined measure [31]: AT^2, thus leading to $size \times depth^2$. By substituting the previous results we get:

$$AT^2_{SRK} = (3n-1)\times 3^2 = \boldsymbol{O}(n)$$

$$AT^2_{B\_\Delta} < \left\lceil \frac{4(n-1)}{\Delta-2} \right\rceil \times \left\lceil \frac{\log n}{\log\Delta - 1} \right\rceil^2 = \boldsymbol{O}\left(\frac{n \log^2 n}{\Delta \log\Delta} \right)$$

$$AT^2_{ROS} = 3\left\lceil \frac{n}{\lceil \log n \rceil + 1} \right\rceil \times 3^2 = \boldsymbol{O}\left(\frac{n}{\log n} \right).$$

Exact AT^2 values have been computed and are plotted in Fig. 2.

A. Area

The results presented in Fig.2 are quite far from any practical implementation. This can easily be explained because: *"comparing the number of nodes is inadequate for comparing the complexity of neural networks as the nodes themselves could implement quite complex functions"* [34].

For VLSI this is due to the following facts:

- the *area* of one neuron can be related to its associated *weights*; and also
- the *area* of the connections is completely neglected.

That is why the *size* complexity measure is not the best criteria for ranking different solutions when going for silicon [8]. Other different measures (or 'cost functions') have already been used:

- the total *number–of–connections* or $\sum$*fan–ins* has been used by several authors [1, 19, 22, 25];
- the total *number–of–bits* needed to represent the *weights* and the *thresholds* $\sum(\sum\lceil \log|w_i| \rceil + \lceil \log|\theta| \rceil)$ has been used by others [17, 18, 34];
- the sum of all the absolute values of the *weights* and *thresholds* $\sum(\sum|w_i| + |\theta|)$ has also been advocated [7, 8, 16].

The sum of all the absolute values of the *weights* and *thresholds* has been used as an optimum criterion for: (*i*) linear programming synthesis [23]; (*ii*) defining the minimum-integer TG realisation of a function [20]. Very recently [3], the same measure (under the name of *"total weight magnitude"*) has been used in the context of computational learning theory applied to neural learning for pattern classification problems. By using it, several bounds which improve the standard VC-theory bounds have been proved !

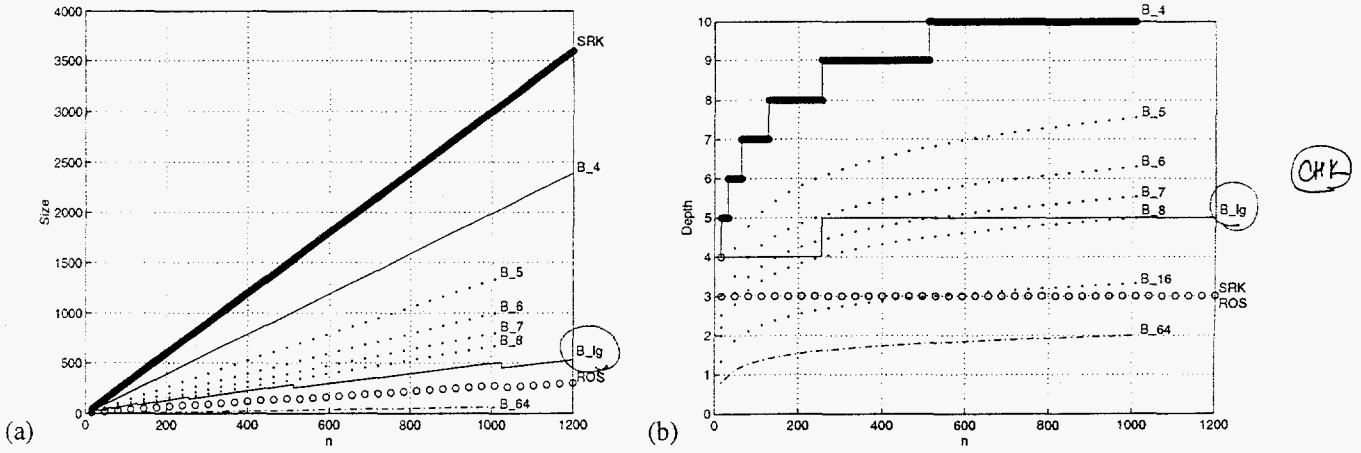


Fig. 1. Several known solutions for COMPARISON: (a) *size* and (b) *depth*. Because $n = 1024$ **B_2**$\lceil\sqrt{n}\rceil$ is in fact **B_64** (equivalent to **VCB**).

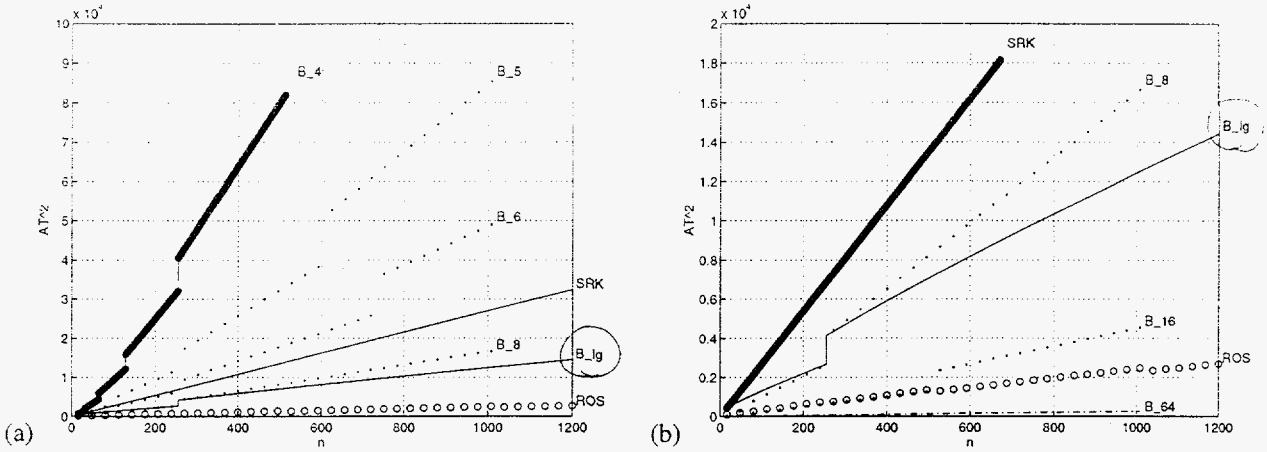


Fig. 2. AT^2 complexity of COMPARISON as *size* × *depth* ^{2}: (a) for several solutions; and (b) detail (here **B_64** is equivalent to **VCB**).

Finally, another quite similar 'definition of complexity' is $\sum(\sum w_i^2 + \theta^2)$ which has been used in the context of genetic programming for reaching 'minimal' neural networks [36].

All the above mentioned 'cost functions' have links to VLSI by the assumptions one makes on how the *area* of a chip scales with the *weights* and the *thresholds* [8, 16]. Here are several of the most common alternatives:

- for purely digital implementation the *area* scales at least with the cumulative size of the *weights* and *thresholds*, as the bits for representing the *weights* and the *thresholds* have to be stored;
- for certain types of analog implementations (*e.g.*, using resistors or capacitors) the same type of scaling is valid (although it is possible, in some particular cases, to come up with analog implementations which have binary encoding of the parameters — for which the *area* would scale at least with the cumulative *log*–scale size of the parameters);
- there are some types of implementations (*e.g.*, transconductance ones) which offer a constant size per element, thus in principle scaling only with the number of parameters (*i.e.*, with $\sum$*fan–ins* as the total *number–of–connections*).

It is worth emphasising that it is anyhow desirable to limit the range of parameter values [35] for VLSI implementations —be they digital or analog— because:

- the maximum value of the *fan–in* [33] and
- the maximal ratio between the largest and the smallest *weight* cannot grow over a certain (technological) limit.

The first normal extension of the circuit complexity results (presented in Section II) towards VLSI complexity ones is by closer estimates of the *area* (instead of the obvious *area* ≈ *size*). Based on the arguments given above (for the different estimations of the *area*), the following two propositions can be stated.

Proposition 6 (this paper) *If the area of a neural network is estimated as* $\sum$*fan–ins, the computation of* COMPARISON *of two n-bit numbers occupies between* $\boldsymbol{O}(n)$ *and* $\boldsymbol{O}(n^2)$ *area:*

$$A_{\mathbf{SRK}} = \frac{n^2 + 11n - 6}{2} = \boldsymbol{O}(n^2)$$

$$A_{\mathbf{B\_\Delta}} < \Delta \times \left\lceil \frac{4(n-1)}{\Delta - 2} \right\rceil + \left\lceil \frac{\log n}{\log\Delta - 1} \right\rceil + \left\lceil \frac{4n}{\Delta} \right\rceil = \boldsymbol{O}(n)$$

$$A_{\mathbf{ROS}} < 4n + \frac{1}{2} \cdot \left\lceil \frac{n}{\lceil \log n \rceil + 1} \right\rceil^2 + \frac{3}{2} \cdot \left\lceil \frac{n}{\lceil \log n \rceil + 1} \right\rceil = \boldsymbol{O}\left(\frac{n^2}{\log^2 n} \right).$$

Proof The **SRK** solution has a first layer of n AND gates and n OR gates (in fact only $n-1$ as the OR gate computing $x_0 \vee y_0$ is not used) of *fan–in* = 2. The second layer has $n-1$ AND gates with *fan–in* = 2, 3, … n. The output layer has one OR gate of *fan–in* = n. These lead to:

$$A_{\mathbf{SRK}} = 2(2n-1) + (2 + 3 + \ldots n) + n = \frac{n^2 + 11n - 6}{2}.$$

For the class **B_Δ**, the first layer has $2\left\lceil \frac{n}{\Delta/2} \right\rceil$ TGs of *fan–in* = Δ (here again, one TG is not used). All the other TGs have *fan–in* = Δ − 1. Using eq. (1) and (2) we have:

$$A_{\mathbf{B\_\Delta}} = \left(\left\lceil \frac{2n}{\Delta} \right\rceil - 1 \right) + \left(\left\lceil \frac{4(n-1)}{\Delta - 2} \right\rceil - \left\lceil \frac{\log n}{\log\Delta - 1} \right\rceil \right)(\Delta - 1),$$

and by keeping only the additive terms the result follows.

Finally, for **ROS** we use a similar counting argument. We remember that $m = \lceil \log n \rceil + 1$ (Proposition 4). The first layer has $2\lceil n/m \rceil - 1$ TGs of *fan–in* = 2m. The last two of the TGs might also have a lower *fan–in* due to the fact that n might not be a multiple of m. Their *fan–in* can be computed by subtracting the *fan–ins* of all the other

TGs from the first layer from the number of inputs $2n$: *fan–in* $= 2\,(n - m\lfloor n/m \rfloor)$. The second layer has $\lceil n/m \rceil - 1$ AND gates with *fan–in* $= 2, 3, \ldots, \lceil n/m \rceil$. The third layer has just one OR gate with *fan–in* $= \lceil n/m \rceil$. It follows that:

$$A_{\mathrm{ROS}} = 2m\left(2\left\lceil\frac{n}{m}\right\rceil - 3\right) + 4\left(n - m\left\lfloor\frac{n}{m}\right\rfloor\right) + \\ + \left(2 + 3 + \ldots + \left\lceil\frac{n}{m}\right\rceil\right) + \left\lceil\frac{n}{m}\right\rceil.$$

After substituting m and neglecting the subtractive terms the proof is concluded. ❑

Proposition 7 (this paper) *If the area of a neural network is estimated as* $\sum(\sum|w_i|+|\theta|)$, *the computation of COMPARISON of two n-bit numbers occupies between* $O(n)$ *and* $O(n^2)$ *area:*

$$A_{\mathrm{SRK}} = \frac{2n^2 + 15n - 9}{2} = O(n^2)$$

$$A_{\mathrm{B\_\Delta}} < \frac{2^{\Delta/2}}{\Delta} \cdot \frac{8n\Delta - 6n - 5\Delta}{\Delta - 2} = O\left(\frac{n \cdot 2^{\Delta/2}}{\Delta}\right)$$

$$A_{\mathrm{ROS}} < \left(4n - \frac{5}{2}\right) \times \left\lceil\frac{n}{\lceil\log n\rceil + 1}\right\rceil + \left\lceil\frac{n}{\lceil\log n\rceil + 1}\right\rceil^2 \\ = O\left(\frac{n^2}{\log n}\right).$$

Proof The **SRK** solution has all the *weights* ± 1 which makes:

$$\sum_{NN}\sum_{i}|w_i| = \sum fan\text{–}ins = \frac{n^2 + 11n - 6}{2}.$$

As there is an infinity of solutions, we shall take average values for the *thresholds*: $-k + 0.5$ for a k–input AND, and -0.5 for a k–input OR. We do remember that: in the first layer there are n AND gates and $n-1$ OR gates of *fan–in* $= 2$; the second layer has n–1 AND gates with *fan–in* $= 2, 3, \ldots n$; the output layer has one OR gate of *fan–in* $= n$. The sum of the absolute *thresholds* can now be easily computed as:

$$\sum_{NN}|\theta| = \frac{3n}{2} + \frac{n-1}{2} + \frac{1}{2}\cdot\sum_{k=3}^{2n-1} k + \frac{1}{2} = \frac{n^2 + 4n - 1}{2}$$

and hence

$$A_{\mathrm{SRK}} = \sum_{NN}\sum_{i}(|w_i| + |\theta|) = \frac{2n^2 - 15n - 9}{2}.$$

For computing the *area* of the **B_Δ** class we rely on the proof that the BFs implemented by the nodes of the Δ-ary tree are linear separable functions. The proof is based on a recursive version of the BFs implemented by the nodes and determines the *weights* and the *thresholds* constructively by induction on the value of the *fan–in*. The proof is not technically involved but is quite long; a first sketch has appeared in [4, 5], while the full proof can be found as *Lemma 2* (together with *Corollary 1*) in [7, 12]. We shall only use the fact that the *weights* of a TG-node have already been determined to be: $1, -1, 2, -3, 2, -5, 5, \ldots, -5\cdot 2^i, 5\cdot 2^i, \ldots, -5\cdot 2^{\Delta/2-4}, 5\cdot 2^{\Delta/2-4}$. It is now straightforward to compute: (*i*) first, the *area* of one TG; (*ii*) second, the *area* of the first layer of the Δ-ary tree; and (*iii*) third, the *area* of the subsequent layers. By adding the *area* of the first layer to that of the subsequent layers, we shall obtain the total *area* of the NN (Δ-ary tree).

The first layer has $[2n/(\Delta/2)] - 1 \cong 4n/\Delta$ TGs of:

$$a_{\mathrm{TG\text{–}I}} = 2\sum_{i=0}^{\Delta/2-1} 2^i = 2\cdot(2^{\Delta/2} - 1) \cong 2^{\Delta/2+1}$$

area each, such that the first layer will occupy:

$$A_{\mathrm{I}} \cong \frac{4n}{\Delta}\cdot 2^{\Delta/2+1} = \frac{8n\cdot 2^{\Delta/2}}{\Delta}.$$

All the other layers have $size_{\mathrm{B\_\Delta}} - (4n/\Delta)$ TGs, each of:

$$a_{\mathrm{TG}} = (1+1+2+3+2) + 10\sum_{j=0}^{\Delta/2-4} 2^j = 9 + \frac{5\,(2^{\Delta/2} - 1)}{4}$$

area (see the *weights* of a TG-node mentioned above). The *area* of all the subsequent layers will be:

$$A_{\mathrm{TREE}} = a_{TG}\,[size_{\mathrm{B\_\Delta}} - (4n/\Delta)] \cong \frac{5\,(2n-\Delta)\cdot 2^{\Delta/2}}{\Delta\,(\Delta-2)},$$

which leads to:

$$A_{\mathrm{B\_\Delta}} = A_{\mathrm{I}} + A_{\mathrm{TREE}} < \frac{2^{\Delta/2}}{\Delta}\cdot\frac{8n\Delta - 6n - 5\Delta}{\Delta - 2}.$$

Turning now to **ROS**, we start from Proposition 6 and replace the *fan–in* of each TG from the first layer by:

$$a_{\mathrm{TG}}(i) = \sum|w_i| = 2\sum_{j=0}^{i-1} 2^j \qquad (3)$$

as the *threshold* of these TGs can be taken 0. We remember that $m = \lceil\log n\rceil + 1$, and that the first layer has $2\lceil n/m\rceil - 1$ TGs (see Proposition 6 for the two TGs having a lower *fan–in*); the second layer has $\lceil n/m\rceil - 1$ AND gates with *fan–in* $= 2, 3, \ldots, \lceil n/m\rceil$; and the third layer has one OR gate with *fan–in* $= \lceil n/m\rceil$. Using the corresponding *thresholds* for the AND and OR gates, it follows that:

$$A_{\mathrm{ROS}} = a_{\mathrm{TG}}(m)\left(2\left\lceil\frac{n}{m}\right\rceil - 3\right) + 2a_{\mathrm{TG}}\left(n - m\left\lfloor\frac{n}{m}\right\rfloor\right) + \\ + 2\left(2 + 3 + \ldots + \left\lceil\frac{n}{m}\right\rceil\right) - \frac{1}{2}\left(\left\lceil\frac{n}{m}\right\rceil - 1\right) + \frac{1}{2} + \left\lceil\frac{n}{m}\right\rceil.$$

Now, substitute m and $a_{\mathrm{TG}}(i)$ as given by eq. (3), and working on this expression the result follows. ❑

B. Delay

With respect to *delay*, two VLSI models have been commonly in use [31]:

- the capacitive one assumes that the *delay* is proportional to the total capacitance, hence a TG introduces a *delay* proportional to its *fan–in*;
- the diffusion one assumes a distributed resistance and capacitance along any wire, hence the *delay* for propagating a signal from one TG to another is proportional to the *length* of the connecting wire.

The second extension of the circuit complexity results (presented in Section II) towards VLSI complexity ones is by closer estimates of the *delay* (instead of the obvious *delay* ≈ *depth*). For the two different approximations of the *delay* suggested, the following two propositions can be stated.

Proposition 8 (this paper) *If the delay of one neuron is proportional to its fan–in, the neural network computing the COMPARISON of two n-bit numbers requires between* $\boldsymbol{O}(\log n)$ *and* $\boldsymbol{O}(n)$ *time:*

$$T_{\mathbf{SRK}} = 2n+2 = \boldsymbol{O}(n)$$

$$T_{\mathbf{B\_\Delta}} < (\Delta-1)\times\left\lceil\frac{\log n}{\log\Delta-1}\right\rceil+1 = \boldsymbol{O}\left(\frac{\Delta\log n}{\log\Delta}\right)$$

$$T_{\mathbf{ROS}} < 2\times\left(\left\lceil\frac{n}{\lceil\log n\rceil+1}\right\rceil+\lceil\log n\rceil+1\right) = \boldsymbol{O}\left(\frac{n}{\log n}\right).$$

Proof The **SRK** solution has *fan–in* = 2 TGs in the first layer; the second layer has *fan–in* = 2, 3, … *n* TGs, thus the *delay* of this layer will be *n* (as it is determined by the largest *fan–in*); the third layer has one *fan–in* = *n* TG. The overall *delay* is:

$$T_{\mathbf{SRK}} = 2+n+n = 2n+2\,.$$

For the **B_Δ** class of solutions, the TGs from the first layer have *fan–in* = Δ, while all the subsequent layers have TGs with *fan–in* = Δ − 1. As there are *depth* layers, we have:

$$T_{\mathbf{B\_\Delta}} = \Delta + (depth-1)(\Delta-1)$$

and substituting *depth* as given by eq. (1), the result follows.

For **ROS** the first layer has *fan–in* = 2*m* TGs (remember that $m=\lceil\log n\rceil+1$); the second layer adds $\lceil n/m\rceil$ to the *delay*, as having gates with *fan–in* = 2, 3, … , $\lceil n/m\rceil$; the third layer has just one gate with *fan–in* = $\lceil n/m\rceil$. It follows that:

$$T_{\mathbf{ROS}} = 2m+\lceil n/m\rceil+\lceil n/m\rceil = 2m+2\lceil n/m\rceil$$

and by substituting *m* the proof is concluded. ❑

Proposition 9 (this paper) *If the delay in a neural network is proportional to the length of the wires, the neural network computing the COMPARISON of two n-bit numbers requires* $\boldsymbol{O}(n)$ *time:*

$$T_{\mathbf{SRK}} = (3n-1)/2,\quad T_{\mathbf{B\_\Delta}} < n,\quad T_{\mathbf{ROS}} < n\,.$$

Proof For computing the *length* of the wires one has to know the position of the TG on a 2-dimensional grid. We make here the simplest assumption: namely, that the implementation keeps the 3-layer structure of the feed-forward NNs. The inputs are all on only one side of width $2n-1$. Clearly, it is possible to reduce the *delay* by placing the inputs on all four sides of a chip, but this involves only constant factors (which are very important in practice !). We shall also assume that the layers are 'centered' on top of each others.

For **SRK**, the lowest value is obtained by centering the second layer on B_0. The delay will be: 1 from inputs to the first layer; $(2n-3)/2$ from the first layer to the second (due to the width of $2n-1$); $n/2$ from the second layer to the output. The result follows.

Both for **B_Δ** and for **ROS**, if the layers are properly centered, only the signal from the most (or least) signifi-

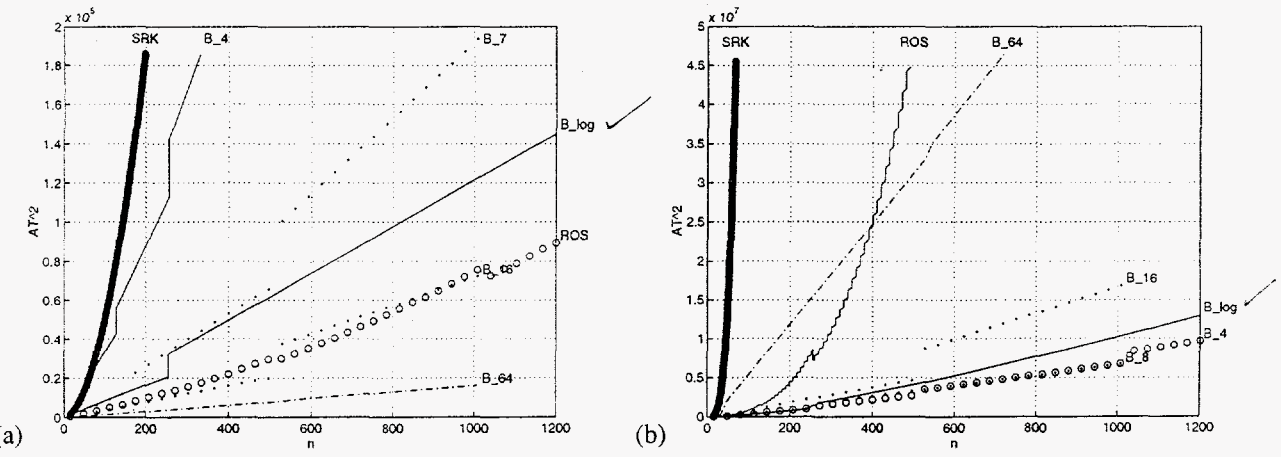


Fig. 3. AT^2 complexity of COMPARISON if the *area* is $A \propto \sum fan\text{–}ins$, and the *delay* is estimated as: (a) $T \propto depth$; or (b) $T \propto fan\text{–}in$ (for $n = 1024$, **B_64** is equivalent to **VCB**).

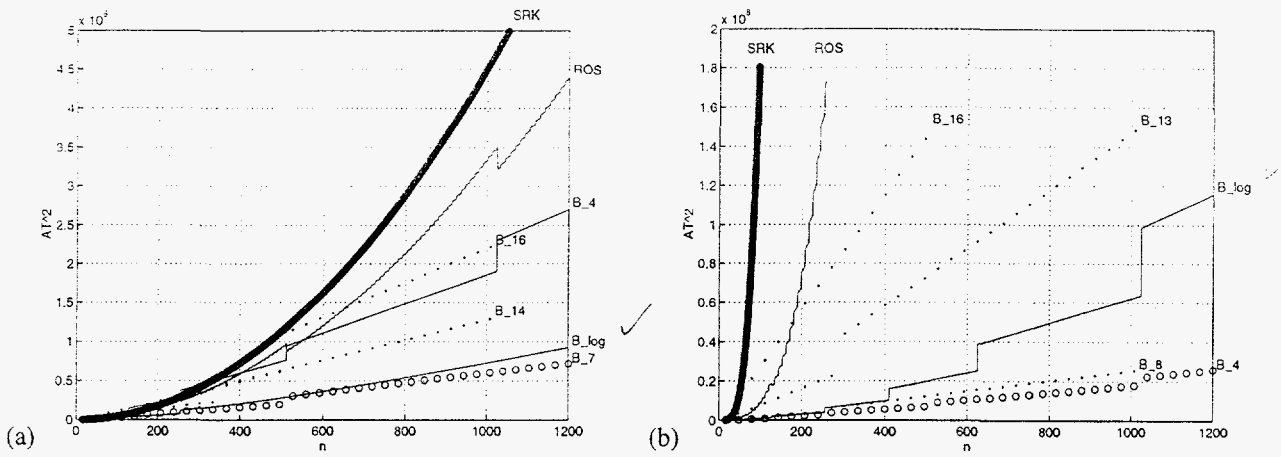


Fig. 4. AT^2 complexity of COMPARISON if the *area* is $A \propto \sum (\sum |w_{ji}| + |\theta|)$, and the *delay* is estimated as: (a) $T \propto$ *depth*; or (b) $T \propto$ *fan–in*.

cant bit has to travel to the 'centre', thus the *delay* being $(2n - 1)/2$ and the proof is concluded. □

For the different estimations of A and T given by Propositions 6–9, exact simulation have been performed. The results —for the different solutions presented— of these simulations are shown in Figs. 3–5. These closer approximations of AT^2 (than *size* × *depth*^{2}) support the claim that *"small constant fan–in digital NNs are VLSI–optimal"* [11]: see the circle plots in Figs. 3(b), 4 and 5 which show that the AT^2 are minimized by **B_4**, **B_6** or **B_7**.

IV. Conclusions and Open Problems

The paper has focused on TG implementations of COMPARISON. Four recent constructive solutions have been presented and their *size* and *depth* complexity discussed. Using different cost functions —which are closer estimates for the *area* and the *delay* of a VLSI chip— the same solutions have been analysed and compared with respect to their VLSI complexity: *area*, *delay* and the combined AT^2 measure. The main conclusions of relevance to VLSI designers are that:

- the VLSI–optimal solutions are not the *size*–optimal ones;
- there exist quite interesting *fan–in* dependent tradeoffs for *depth–size* and *area–delay*;
- the AT^2–optimal solutions are obtained for small constant *fan–in* values.

Future work should concentrate on:

- extending these results to $\boldsymbol{F}_{n,m}$ and to direct synthesis (*e.g.*, using some constructive algorithms);
- linking these results with the entropy of the data–set (for classification problems) and with principles like the 'Occam's razor' and/or the 'minimum description length' (see also [3]);
- finding closer estimates (*i.e.*, cost functions) for optimal mixed analogue/digital implementations.

Preliminary results on these lines can be found in [6, 13, 14].

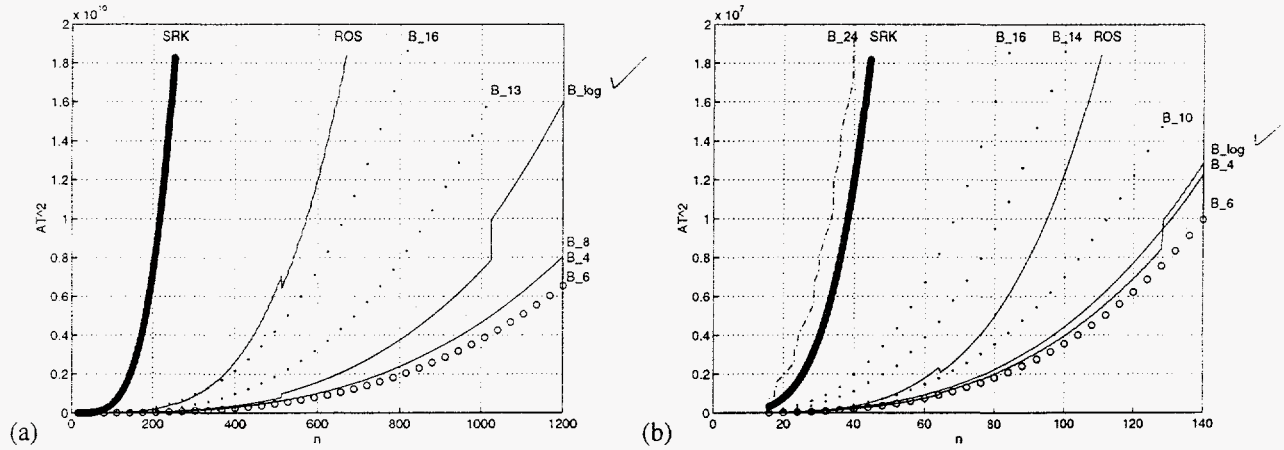


Fig. 5. AT^2 complexity of COMPARISON as $\{\sum(\sum|w_{ji}| + |\theta|)\} \times length^2$: (a) complexity trend; and (b) normal lengths operands (for $n = 128$, **B_24** is equivalent to **VCB**).

REFERENCES

[1] Abu–Mostafa, Y.S., "Connectivity Versus Entropy," in D.Z. Anderson (ed.), *NIPS*, New York, NY: American Inst. of Physics, pp. 1-8, 1988.

[2] Alon, N. & Bruck, J., "Explicit Construction of Depth–2 Majority Circuits for Comparison and Addition," *Tech. Rep. RJ 8300* (75661), IBM Almaden, San Jose, CA, 1991.

[3] Bartlett, P.L., "The Sample Complexity of Pattern Classification with Neural Networks: The Size of the Weights Is More Important than the Size of the Network," *Tech. Rep.*, Dept. of Systems Eng., Res. Sch. of Information Sci. and Eng., Australian Nat. Univ., Canberra 0200 Australia, 7 May 1996 (**ftp:** syseng.anu.edu.au/pub/peter/TR96d.ps.Z).

[4] Beiu, V., Peperstraete, J.A., Vandewalle, J. & Lauwereins, R., "Efficient Decomposition of COMPARISONand Its Applications," in M. Verleysen (ed.), *ESANN'93*, Brussels, Belgium: Dfacto, pp. 45-50, 1993.

[5] Beiu, V., Peperstraete, J.A., Vandewalle, J. & Lauwereins, R., "COMPARISONand Threshold Gate Decomposition," in D.J. Myers and A.F. Murray (eds.), *MicroNeuro'93*, Edinburgh, UK: UnivEd Tech. Ltd., pp. 83-90, 1993.

[6] Beiu, V., Peperstraete, J.A., Vandewalle, J. & Lauwereins, R., "Learning from Examples and VLSI Implementation of Neural Networks," in R. Trappl (ed.), *Cybernetics and System Research '94*, Singapore: World Scientific Publishing, pp. 1767-1774, 1994.

[7] Beiu, V., Peperstraete, J.A., Vandewalle, J. & Lauwereins, R., "Area-Time Performances of Some Neural Computations," in P. Borne, T. Fukuda and S.G. Tzafestas (eds.), *SPRANN'94*, Lille, France: GERF EC, pp. 664-668, 1994.

[8] Beiu, V., Peperstraete, J.A., Vandewalle, J. & Lauwereins, R., "On the Circuit Complexity of Feedforward Neural Networks," in M. Mariano and P. Marassao (eds.), *Artificial Neural Networks IV*, London, UK: Springer-Verlag, pp. 521-524, 1994.

[9] Beiu, V., Peperstraete, J.A., Vandewalle, J. & Lauwereins, R., "Placing Feedforward Neural Networks Among Several Circuit Complexity Classes," *Proc. WCNN'94*, Hillsdale: Lawrence Erlbaum and INNS Press, pp. 584-589, 1994.

[10] Beiu, V. & Taylor, J.G., "VLSI Optimal Neural Network Learning Algorithm," in D.W. Pearson, N.C. Steele and R.F. Albrecht (eds.), *Artificial Neural Nets and Genetic Algorithms*, Wien, Austria: Springer-Verlag, pp. 61-64, 1995.

[11] Beiu, V., "Constant Fan-In Neural Networks Are VLSI-Optimal," accepted for publication in *Annals of Mathematics and Artificial Intelligence*, 1995 (**http:**//www.mth.kcl.ac.uk/~beiu/postscripts/math_NN_apl95.ps.Z).

[12] Beiu, V. & Taylor, J.G., "On the Circuit Complexity of Sigmoid Feedforward Neural Networks," accepted for publication in *Neural Networks*, 1995 (**http:**//www.mth.kcl.ac.uk/~beiu/postscripts/neural_nets96.ps.Z).

[13] Beiu, V., "Direct Synthesis of Neural Networks," *Proc. MicroNeuro'96*, Los Alamitos, CA: IEEE CS Press, pp. 257-264, 1996.

[14] Beiu, V., "Entropy Bounds for Classification Algorithms," *Neural Network World*, **6**(4), pp. 497-505, 1996.

[15] Beiu, V., "Optimal VLSI Implementation of Neural Networks," ***Chapter 18*** in J.G. Taylor (ed.), *Neural Networks and Their Applications*, Chichester, UK: John Wiley, pp. 255-276, 1996.

[16] Beiu, V., *VLSI Complexity of Discrete Neural Networks*, accepted for publication, Newark, NJ: Gordon and Breach, 1997.

[17] Bruck, J. & Goodmann, J.W., "On the Power of Neural Networks for Solving Hard Problems," in D.Z. Anderson (ed.), *NIPS*, New York, NY: American Inst. of Physics, pp. 137-143, 1988. Also in *J. of Complexity*, **6**, pp. 129-135, 1990.

[18] Denker, J.S. & Wittner, B.S., "Network Generality, Training Required and Precision Required," in D.Z. Anderson (ed.), *NIPS*, New York, NY: American Inst. of Physics, pp. 219-222, 1988.

[19] Hammerstrom, D., "The Connectivity Analysis of Simple Association –or– How Many Connections Do You Need," in D.Z. Anderson (ed.), *NIPS*, New York, NY: American Inst. of Physics, pp. 338-347, 1988.

[20] Hu, S., *Threshold Logic*. Berkeley, CA: Univ. of California Press, 1965.

[21] Krishnamoorthy, A.V., Paturi, R., Blume, M., Linden, G.D., Linden, L.H. & Esener S.C., "Hardware Tradeoffs for Boolean Concept Learning," *Proc. WCNN'94*, Hillsdale: Lawrence Erlbaum and INNS Press, pp. 551-559, 1994.

[22] Mason, R.D. & Robertson, W., "Mapping Hierarchical Neural Networks to VLSI Hardware," *Neural Networks*, **8**(6), pp. 905-913, 1995.

[23] Minnik, R.C., "Linear–Input Logic," *IRE Trans. on Electr. Comp.*, **10**, pp. 6-16, 1961.

[24] Murthy, K.V.S., "On Growing Better Decision Trees from Data," *PhD Thesis*, Johns Hopkins Univ., Baltimore, 1995 (**http:**//www.cs.jhu.edu/~murthy/).

[25] Phatak, D.S. & Koren, I., "Connectivity and Performances Tradeoffs in the Cascade Correlation Learning Architecture," *IEEE Trans. Neural Networks*, **5**(6), pp. 930-935, 1994.

[26] Red'kin, N.P., "Synthesis of Threshold Circuits for Certain Classes of Boolean Functions," *Kibernetika* **5**, pp. 6-9, 1970. English translation in *Cybernetics*, **6**(5), pp. 540-544, 1973.

[27] Roychowdhury, V.P., Orlitsky, A. & Siu, K.-S., "Lower Bounds on Threshold and Related Circuits Via Communication Complexity," *IEEE Trans. on Information Theory*, **40**(2), pp. 467-474, 1994.

[28] Siu, K.-Y. & Bruck, J., "On the Power of Threshold Circuits with Small Weights," *SIAM J. Discrete Mathematics*, **4**(3), pp. 423-435, 1991.

[29] Siu, K.-Y., Roychowdhury, V.P. & Kailath, T., "Depth-Size Tradeoffs for Neural Computations," *IEEE Trans. on Computer*, **40**(12), pp. 1402-1412, 1991.

[30] Siu, K.-Y. & Bruck, J., "Neural Computing with Small Weights," in J.E. Moody, S.J. Hanson and R.P. Lippmann (eds.), *NIPS 4*, San Mateo, CA: Morgan Kaufmann, pp. 944-949, 1992.

[31] Ullman, J.D., *Computational Aspects of VLSI*, Rockville, MA: Computer Science Press, 1984.

[32] Vassiliadis, S., Cotofana, S. & Berteles, K., "2-1 Addition and Related Arithmetic Operations with Threshold Logic," accepted for publication in *IEEE Trans. on Computer*, 1996 (**http:**//einstein.et.tudelft.nl:80/~sorin/papers.html).

[33] Walker, M.R., Haghighi, S., Afghan, A. & Akers, L.A., "Training a Limited-Interconnect, Synthetic Neural IC," in D.S. Touretzky (ed.), *NIPS 1*, San Mateo, CA: Morgan Kaufmann, pp. 777-784, 1989.

[34] Williamson, R.C., "ε-Entropy and the Complexity of Feedforward Neural Networks," in R.P. Lippmann, J.E. Moody & D.S. Touretzky (eds.), *NIPS 3*, San Mateo, CA: Morgan Kaufmann, pp. 946-952, 1990.

[35] Wray, J. & Green, G.G.R., "Neural Networks, Approximation Theory, and Finite Precision Computation" *Neural Networks*, **8**(1), pp. 31-37, 1995.

[36] Zhang, B.-T. & Mühlenbein, H., "Genetic Programming of Minimal Neural Networks Using Occam's Razor.," Tech. Rep. GMD 734, Schloß Birlinghoven, Sankt Augustin, Germany, 1993.