ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 45-50

# Efficient Decomposition of Comparison and Its Applications[①]

Valeriu Beiu[†,‡], Jan Peperstraete[†], Joos Vandewalle[†], and Rudy Lauwereins[†,②]

[†] Katholieke Universiteit Leuven, Department of Electrical Engineering
Division ESAT, Kardinaal Mercierlaan 94, B-3001 Heverlee, Belgium

[‡] on leave of absence from Bucharest Polytechnic Institute, Department of Computer Science, Spl. Independentei 313, 77206 Bucharest, România

**Abstract.** The paper deals with transformation algorithms for decomposing Boolean neural networks (NNs); this being done with a view to possible VLSI implementation of NNs using threshold gates (TGs). We detail a possible tree decomposition for *COMPARISON*, and show how this can be used for the decomposition of Boolean functions (BFs) belonging to $\mathcal{F}_{N,m}$ – the class of BFs of $N$ variables that have exactly $m$ groups of ones. Complexity estimates are given, the results being: (i) linear size; (ii) logarithmic depth; (iii) constant weights; and (iv) constant threshold.

## 1. Introduction

In the last several years we have witnessed a growing number of TG theoretical investigations, beside the marked revival of interest in neural networks. TGs have received more attention as microelectronic technology has evolved up to the mature point where it is possible to realize small NNs in silicon, and to foresee the VLSI implementation of large NNs [11, 12]. TGs are a challenging alternative to classic Boolean solutions due to their intrinsic nonlinearity, and the solid theoretical background [13, 16, 17]. The new interest in TGs is proven by many articles from the late 80s and 90s [1, 2, 5–10, 15, 18–22], as well as proposals of implementations [2, 5, 18, 21, 22], or even designs [14]. This paper falling in this trend, is the result of an ongoing work at KULeuven focused towards reducing the complexity of Boolean NNs, with a view to their efficient VLSI implementation using TGs. The meaning of "reducing a NN" is that after applying such an algorithm to an "input NN", the "output NN" will be "simpler" with respect to: (i) fan-in of neurons [6, 7], (ii) precision of weights [3, 4], and (iii) approximation of the sigmoidal output function [5].

As decomposition is an immediate solution for fan-in reduction, the paper will examine how to decompose TG circuits. We will focus on *COMPARISON* as one of the most interesting $\mathcal{F}_{N,1}$ function. By using the solution we propose in this article, together with the one detailed by the authors in [6, 7], we develop a systematic algorithm for the class $\mathcal{F}_{N,m}$. Complexity bounds are proven. Conclusions and further directions of research end the paper.

---

① This research work was partly carried out in the framework of a Concerted Action Project of the Flemish Community, entitled: *"Applicable Neural Networks"*. The scientific responsibility is assumed by the authors.

② Senior Research Assistant of the Belgian National Fund for Scientific Research.

ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 45-50

## 2. Comparison

The following notations will be used in this section: $\tilde{B}$ and $\tilde{C}$ (the "constant") are two binary numbers of $N$ bits each $B = b_{N-1}b_{N-2}\ldots b_1b_0$, $C = c_{N-1}c_{N-2}\ldots c_1c_0$. *COMPARISON*, which is a $\mathcal{F}_{N,1}$ function, has received a lot of attention from the "TG community" [2, 18, 19, 22]. It can be defined as:

$$\Omega(B,C) = \begin{cases} 1 & \text{if } \tilde{B} > \tilde{C} \\ 0 & \text{if } \tilde{B} \le \tilde{C} \end{cases} \quad (1)$$

The classical way to look at *COMPARISON* is the well known serial algorithm: "Compare bit by bit the two numbers starting from $i = N-1$ (the MSB); if for the current bits $b_i > c_i$ then $\Omega(B,C) = 1$; if $b_i = = c_i$ then go to the next bits (decrement i, and if $i < 0$ then $\Omega(B,C) = 0$); if $b_i < c_i$ then $\Omega(B,C) = 0$". Another way is to try to compute *COMPARISON* in parallel, and that is what we shall do.

Divide $B$ in $\pi$ groups (not necessarily equal, but for the ease of notations we will consider them equal) $\beta_{\pi-1},\beta_{\pi-2},\ldots,\beta_2,\beta_0$, where $\beta_i = b_{(i+1)N/\pi-1}\ldots b_{iN/\pi}$ (we have also considered $n$ divisible by $\pi$, which is also not really necessary but simplifies notations); similarly divide $C$ in $\pi$ groups $\chi_{\pi-1},\chi_{\pi-2},\ldots,\chi_2,\chi_0$, where $\chi_i = c_{(i+1)N/\pi-1}\ldots c_{iN/\pi}$. If we use + and × instead of the logical operators OR and AND, we can write:

$$\Omega_\pi(B,C) = \sum_{i=0}^{\pi-1}\left\{\Omega_1\left(\beta_{\pi-1-i},\chi_{\pi-1-i}\right)\times\left[\prod_{j=0}^{i-1}\Omega_1\left(\beta_{\pi-i+j},\chi_{\pi-i+j}-1\right)\right]\right\}, \quad (2)$$

where $\prod_0^{-1} = 1$, and the subscripts indicate how many groups are compared.

**Proposition 1.** ***COMPARISON* can be decomposed in $\pi$ sub-*COMPARISON*s using only linearly separable functions in a tree-like fashion.**

**Proof.** The proof of this statement is by induction. For $\pi$=2 equation (2) becomes:

$$\Omega_2(B,C) = \Omega_1\left(\beta_1,\chi_1\right) + \Omega_1\left(\beta_1,\chi_1-1\right)\times\Omega_1\left(\beta_0,\chi_0\right) \quad (3)$$

which is of the form $x + y \times z$, and can be implemented by a TG: $\langle 2x + y + z\rangle_{1.5}$ (the notations are from [13, 17]).

Now suppose it is true for $\pi$; we want to prove it for $\pi$+1. Refining equation (2) we can determine a recursive version:

$$\Omega_{\pi+1}(B,C) = \Omega_1\left(\beta_\pi,\chi_\pi\right) + \Omega_1\left(\beta_\pi,\chi_\pi-1\right)\times \times\sum_{i=0}^{\pi-1}\left\{\Omega_1\left(\beta_{\pi-1-i},\chi_{\pi-1-i}\right)\times\left[\prod_{j=0}^{i-1}\Omega_1\left(\beta_{\pi-i+j},\chi_{\pi-i+j}-1\right)\right]\right\} =$$

$$= \Omega_1\left(\beta_\pi,\chi_\pi\right) + \Omega_1\left(\beta_\pi,\chi_\pi-1\right)\times\Omega_\pi(B,C) \quad (4)$$

From equation (4) one way to decompose *COMPARISON* is shown: $\langle 2x + y + \langle 2x' + y' + \ldots\rangle_{1.5}\rangle_{1.5}$; it is not interesting as it leads to an unbalanced $\pi$ leveled tree, but we should mention the fact that the weights are either 1 or 2 (i.e. $2^0$ and $2^1$). If one tries to determine

the weights of the compound function starting from the inequalities arising from the truth table of the $x + y \times z$ function, might be quite amazed to find out that no weights satisfy the system of inequalities (i.e. the function "seems" to be not linear separable). The authors when taking this approach have also been puzzled; the clue being that the input variables are not "independent input variables", and some combinations cannot occur (i.e. the function is incompletely specified). Assigning the unspecified (don't-cares) values to 0 and 1 in a proper (smart) way, leads always to a determined system of inequalities, thus justifying that the compound function is linear separable. Although correct, this line of proving is manageable, but quite difficult.

A better idea is to built the $\Omega_{\pi+1}$ as a $2\pi + 1$ inputs TG: $2\pi - 1$ inputs being copies of the previous $\Omega_\pi$ TG, and the two more inputs coming from $\Omega_1(\beta_\pi, \chi_\pi)$ and $\Omega_1(\beta_\pi, \chi_\pi - 1)$, with weights $2^\pi$ and, respectively $2^{\pi-1}$; assign $2^\pi - 1/2$ as the threshold of this $\Omega_{\pi+1}$ TG. This TG satisfies equation (4):

- if $\Omega_1(\beta_\pi, \chi_\pi) = 1$, then $\Omega_{\pi+1}(B,C) = 1$ as $2^\pi > 2^\pi - 1/2$;
- if $\Omega_1(\beta_\pi, \chi_\pi) = 0$, but $\Omega_1(\beta_\pi, \chi_\pi - 1) = 1$, the decision can be taken only by considering $\Omega_\pi(B,C)$:
  - if $\Omega_\pi(B,C) = 1$ then $\Omega_{\pi+1}(B,C) = 1$ as $2^{\pi-1} + 2^{\pi-1} > 2^\pi - 1/2$;
  - if $\Omega_\pi(B,C) = 0$ then $\Omega_{\pi+1}(B,C) = 0$ as $2^{\pi-1} < 2^\pi - 1/2$;
- if $\Omega_1(\beta_\pi, \chi_\pi) = 0$ and $\Omega_1(\beta_\pi, \chi_\pi - 1) = 0$, then $\Omega_{\pi+1}(B,C) = 0$ for any value of $\Omega_\pi(B,C)$ as $2^{\pi-1} < 2^\pi - 1/2$.

This concludes the proof. □

## 3. Application to $\mathcal{F}_{N,m}$ decomposition

Beside the useful class of symmetric functions [6, 7, 22], another known class is $\mathcal{F}_{N,m}$: the class of BFs with a given number of groups of ones ($\mathcal{F}_{n,m}$ in the original article [16], but has been changed to be consistent with the notations we have used in [6, 7]). If we allow $m$ to grow exponentially (not very interesting) with respect to $N$, $\mathcal{F}_{N,m} = \mathcal{B}_N$, the set of all $n$-ary BFs. Here $N$ represents the number of input variables, and $m$ the number of groups of ones in the truth table of the function to be implemented $f \in \mathcal{F}_{N,m}$. As basic subfunctions for constructing the TG circuit for $f \in \mathcal{F}_{N,m}$ Red'kin [16] uses $\psi_i^{\ddagger}(\tilde{x})$ and $\varphi_i(\tilde{\sigma})$ which are *COMPARISON*s.

### 3.1 Size and depth

Because there are only $m$ groups of ones we can use $2m$ TGs to compare the limits for these groups of ones (in the worst case, as it might be possible to need only $2m$-1 or even $2m$-2 TGs – the exceptions being for the first and the last of the $\pi$ groups); one more TG is needed to "sum" the outputs from this first layer in an alternate way: +, −, +, ... (see fig.1). The $2m$ TGs forming the first layer have weights $\leq 2^{N-1}$ (exponential); threshold $\leq 2^{N-1}$ (exponential); and fan-in $N$ (linear). The TG from the second layer has weights ± 1 (i.e. it is a MAJ gate); threshold 0.5; and fan-in $2m$. Now let us impose the same fan-in limitation as in [6, 7]: "allow

ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 45-50

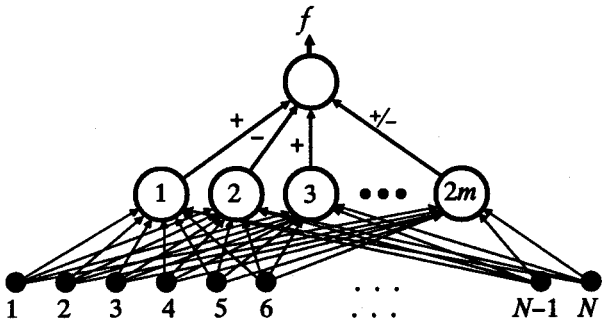


**Fig. 1.** The two layers TG circuit for $f \in \mathcal{F}_{N,m}$.

only TGs with fan-in $n \leq 2^{l_n}$.

The TG from the second layer can be decomposed as has been shown in [6, 7]. The tree has depth $\log(2m/n)$, and size $2m - n + 1$. As the other $2m$ TGs from the first layer are *COMPARISONs*, we can decompose each one as shown in section 2. For each of these $2m$ TGs we can build a tree (see fig.2) of size $2(N - n/2 + 1)$, and depth $\log(2N/n)$.

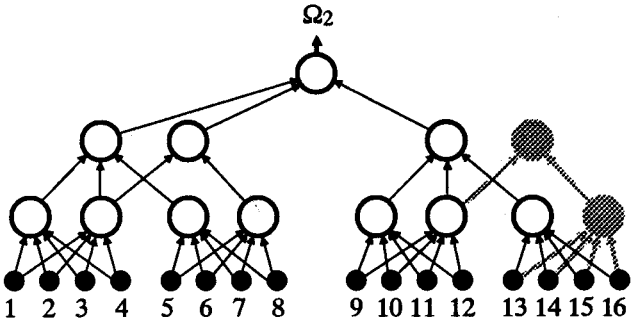


**Fig. 2.** Decompodition of *COMPARISON* for $N$=16 and $n$=4 (TGs not used are tinted).

Linking these results, we have for $\forall f \in \mathcal{F}_{N,m}$:

$$N_G(N,m,n) = \left(2m - n + 1\right) + 4m \cdot \left(N - n/2 + 1\right) = \mathbf{O}\left(mnN\right), \text{ and} \quad (5)$$

$$N_L(N,m,n) = \log\left(2m/n\right) + \log\left(2N/n\right) = \mathbf{O}\left[\log\left(mN/n^2\right)\right]. \quad (6)$$

As $n$ can be considered a constant for a given technology, the size and the depth of the network are $\mathbf{O}(mN)$, and $\mathbf{O}(\log mN)$ respectively.

### 3.2 Weights and thresholds

An interesting thing to analyze is what has happened to the weights and the thresholds.

**Proposition 2. Any of the $2m$ TGs from the first (input) layer has weights and thresholds reduced to less than $2^{n/2}$.**

**Proof.** The value 2 comes from the assignment of the weights suggested in proposition 1. The value $n/2$ of the exponent comes from the fact that each TG can have at most $n$ inputs (due to the imposed fan-in

ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 45-50

limitation), and the weights are: 1, 1, 2, 2, 4, 4, ... (in fact we will always use only $n-1$ out of the $n$ allowed inputs, see fig.3).

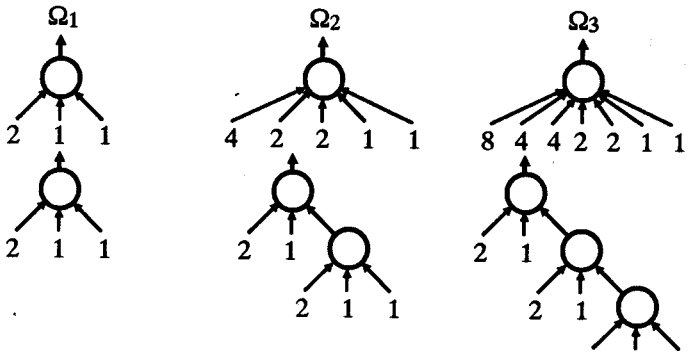


**Fig. 3.** TG for $\Omega_1$, $\Omega_2$, $\Omega_3$ showing the series of the weights: 1, 1, 2, 2, 4, 4, ... .

Because $n$ has been a considered constant, these values are also constants. □

## 4. Conclusions

The paper presents a new decomposition algorithm for $\mathcal{F}_{N,m}$ functions. It improves on the known algorithms with respect to the well known cost function "number of gates" (size) and "number of layers" (depth) [see also 6, 7].

While $N_G$ is related to the area of the circuit ($A \approx$ size), $N_L$ relates to delay ($T \approx$ depth). The VLSI complexity measures $AT$ and $AT^2$ can be estimated by $N_G \cdot N_L$ and $N_G \cdot N_L^2$ leading to $AT \approx mN \cdot \log(mN)$, and $AT^2 \approx mN \cdot \log^2(mN)$.

A significant problem to reveal is that of the error made by assuming $A \approx N_G$ might be quite large. Due to the fact that weights have to be stored in some way in the case of TGs as opposed to classical Boolean gates (area of a capacitor, size of a resistor, ...), the size of a TG depends on its weights' values. That is why a more correct estimate would be the cost function we have introduced in [5]: *the number obtained by summing all the weights (in modulus).*

The claim made in [6, 7] that "while decomposing, we also reduce the range of the weights" is supported by the direct relation between the limited fan-in value and the limited value of the weights.

As further direction of research we are interested in bridging the gap between the two classes of BFs for which algorithms have been defined [6, 7]. Work is under progress, and it seems that, at least for simple logical combinations of two functions, one belonging to one class and the other belonging to the other class, a similar efficient decomposition should be possible.

## References

1. A. Albrecht: On Bounded-Depth Threshold Circuits for Pattern Functions. In I. Aleksander and J. Taylor (eds.): Artificial Neural Networks 2 (Proc. ICANN'92, Brighton, Sept. 4-7), Elsevier Sci. Publ., Vol. 1, 135-138 (1992).

2. N. Alon and J. Bruck: Explicit Construction of Depth-2 Majority Circuits for Comparison and Addition. Res. Rep. RJ 8300 (75661), IBM Almaden, San Jose, CA, 1-13 (1991).
3. C. Alippi: Weight Representation and Network Complexity Reductions in the Digital VLSI Implementation of Neural Nets. Res. Note RN/91/22, Dept. CS, Univ. College London, 1-20 (1991).
4. T. Baker and D. Hammerstrom: Modifications to Artificial Neural Network Models for Digital Hardware Implementation. Tech. Rep. CS/E 88-035, Dept. CS&E, Oregon Graduate Center, 1-10 (1988).
5. V. Beiu, J.A. Peperstraete and R. Lauwereins: Using Threshold Gates to Implement Sigmoid Nonlinearity. In I. Aleksander and J. Taylor (eds.): Artificial Neural Networks 2 (Proc. ICANN'92, Brighton, Sept. 4-7), Elsevier Sci. Publ., Vol. 2, (1992).
6. V. Beiu, J.A. Peperstraete and R. Lauwereins: Algorithms for Fan-In Reduction. In IJCNN'92 (Beijing, 3-6 Nov.), PHEI Press, Vol. 3, 204-209 (1992).
7. V. Beiu, J.A. Peperstraete and R. Lauwereins: Simpler Neural Networks by Fan-In Reduction. In Proc. NeuroNimes'92 (Nimes, 2-6 Nov.), EC2, Nanterre, 589-600 (1992).
8. N.N. Biswas and R. Kumar: A New Algorithm for Learning Representations in Boolean Neural Networks. Current Science 59(12), 595-600 (1990).
9. J. Bruck: Harmonic Analysis of Polynomial Threshold Functions. SIAM J. on Disc. Math. 3(2), 168-177 (1990).
10. J. Bruck and R. Smolensky: Polynomial Threshold Functions, $AC^0$ Functions and Spectral Norms. SIAM J. on Comput. 21(1), 33-42 (1992).
11. H.P. Graf, E. Sackinger, B. Boser and L.D. Jackel: Recent Developments of Electronic Neural Nets in USA and Canada. In Proc. Conf. Microelectronics for Neural Networks (Münich, 16-18 Oct.), Kyrill&Method Verlag, 471-488 (1991).
12. Y. Hirai: Hardware Implementation of Neural Networks in Japan. In Proc. Conf. Microelectronics for Neural Networks (Münich, 16-18 Oct.), Kyrill& Method Verlag, 435-446 (1991).
13. S.T. Hu: Threshold Logic. University of California Press, Berkeley&Los Angeles, 1965.
14. R. Lauwereins and J. Bruck: Efficient Implementation of a Neural Multiplier. In Proc. Microelectronics for Neural Networks (Münich, 16-18 Oct.), Kyrill &Method Verlag, 217-230 (1990).
15. E. Mayoraz: On the Power of Networks of Majority Functions. In A. Prieto (ed.), Lecture Notes in Computer Science 540, Proc. IWANN'91 (Grenade), Springer-Verlag, 78-85 (1991).
16. N.P. Red'kin: Synthesis of Threshold Circuits for Certain Classes of Boolean Functions. Cybernetics 6(5), 540-544 (1973).
17. C.L. Sheng. Threshold Logic. Academic Press, New York, 1969.
18. K.-Y. Siu and J. Bruck: Neural Computation of Arithmetic Functions. Proc. IEEE 78(10), 1669-1675 (1990).
19. K.-Y. Siu and J. Bruck: On the Power of Threshold Circuits with Small Weights. SIAM J. on Disc. Math. 4(3), 423-435 (1991).
20. K.-Y. Siu and J. Bruck: On the Dynamic Range of Linear Threshold Elements. SIAM J. on Disc. Math., to appear.
21. K.-Y. Siu, J. Bruck and T. Kailath: Depth Efficient Neural Networks for Division and Related Problems. Res. Rep. RJ 7946 (72929), IBM Almaden, San Jose, CA, 1-20 (1991).
22. K.-Y. Siu, V. Roychowdhury and T. Kailath: Computing with Almost Optimal Size Threshold Circuits. Tech. Rep., Information System Lab., Stanford Univ., 1-24 (1990).