

The Word-Level Models for Efficient Computation of Multiple-Valued Functions. PART 2: LWL Based Model

Anna M. Tomaszewska*

*Faculty of Computer Science and Information Systems,
Technical University of Szczecin, POLAND, atomaszewska@wi.ps.pl*

Svetlana N. Yanushkevich, Vlad P. Shmerko

*Department of Electrical and Computer Engineering,
University of Calgary, CANADA, <yanush>shmerko@enel.ucalgary.ca*

Abstract

This paper is a continuation of the study of Neural-Like Networks (NLNs) for computation of Multiple-Valued Logic (MVL) functions. NLN is defined as a feedforward network with no learning. In contrast to classical neural network with Threshold Gates (TGs), the proposed NLN is built of so-called Neuron-Like Gates (NLGs). It was shown in our previous study that NLG is modelled by a Linear Arithmetical expression (LAR). In this paper we show even more simple NLG model. We have developed two word-level models, Linear Weighted Logic expressions (LWLs) and a corresponding set of Linear Decision Diagrams (LDDs). We compare the LWL- and LAR-based NLNs. The experimental study on large MVL circuits shows that the number of nodes in the LDDs derived from LWLs is four times less in average compared to those derived from LARs. They are also 2-7 times more compact (require less memory to store the terminal values).

1 Introduction

We consider a **feedforward calculation of MVL functions on an NLN with no learning**. In classic approach, a circuit is modelled as a network of idealized computing elements, known as TGs, connected by the links. Our model, NLG, is a multiple-valued, multi-input, multi-output element. The relationship between the inputs and outputs is expressed by the LWL.

We refer to the initial idea of *McCulloch and Pitts* [9], the fundamental results by *Kautz and Muroga*,

*Supported by PhD grants from State Committee for Scientific Research (KBN), Poland

some recent results, in particular [1, 2], and generalization towards MVL functions [4, 5, 6, 11]. The overview [10] is a useful reference to that. However, all these techniques fail to produce even middle-size functions. This is a motivation for investigation of new methods and aggressive techniques to compute logic functions based on the neural-like paradigm. In paper [12] we introduced an NLN based on a word-level (LAR) model of the NLG. This model and the one proposed in this paper are of linear structure and planar by nature. They both are mapped into LDDs. While related, the proposed models differ in a number of features. We use only logical operations in LWL-based NLG, while the arithmetic operations are used in LAR-based NLG. As a result, we particularly benefit in memory requirements. In general, the contribution of our study is a new understanding of linearization of an MVL circuit models.

It is important to notice that in complex interconnect structures the gate delays are less significant than the routing delays in interconnects. Thus, the linearization and planarization become the important factors of circuit models. The planarization of the circuit models stipulates the advanced conditions to explore the source of the interconnect space; the goal of the planarization is a layout without crossings [3]. As a continuation of our previous research, the proposed algorithm is oriented to large size MVL circuits, and guarantees that an arbitrary MVL circuit is implemented by the LDD-based model planar by its nature.

In [11], a formal model of TG for MVL have been introduced. An NLN proposed in [12], is built of two-input ternary NLGs specified by

$$f = \Xi^\mu\{D\}, \quad D = d_1x_1 + d_2x_2 + d_0. \quad (1)$$

where d_1, d_2 are integer numbers (weights), d_0 is the bias, and Ξ^μ is a masking operator that extract the digits accordingly parameter μ . Expression (1) is an LAR-based **word-level model of NLG**.

The TG model [11] is characterized by the drastic loss of information, due to threshold operation. The word-level model (1) is very effective to represent large MVL circuits by NLN but we need special technique to encode variables and manipulate the huge values of weights [12]. Our goal is to improve the model (1).

2 LWLs of Boolean functions

A technique of Boolean functions manipulation based on so-called *weighted logical form* has been introduced in [8]. We explain this technique by the below example.

Example 1. Let us consider two-output Boolean function f with outputs $f_1 = x_1 \vee x_2$, $f_2 = \bar{x}_1 \vee x_2$. In contrast to traditional word-level form LAR [12] $f = 2^1 f_2 + 2^0 f_1$, the weighted logical sum of a given function is defined via bit-wise operation $\widehat{\vee}$ as follows

$$\begin{aligned} f &= 2^1 f_2 \widehat{\vee} 2^0 f_1 = 2^1(\bar{x}_1 \vee x_2) \widehat{\vee} 2^0(x_1 \vee x_2) \\ &= 2\bar{x}_1 \widehat{\vee} 2x_2 \widehat{\vee} x_1 \widehat{\vee} x_2 = x_1 \widehat{\vee} 2\bar{x}_1 \widehat{\vee} 3x_2. \end{aligned}$$

The bit-wise disjunction of the weighted variable values produces the integer truth table $f = [2 \ 3 \ 1 \ 3]$. For $x_1 x_2 = 01$, $f(01) = 0 \widehat{\vee} 2 \cdot 1 \widehat{\vee} 3 \cdot 1 = 3$ that corresponds $f_1(01) = 1$ and $f_2(01) = 1$. Since f_1 and f_2 does not contain any products of variables, f does not as well.

Definition 1. A weighted logical expression that does not include any products of variables be an LWL expression of a multiple-output logic function.

Example 2. $x_1 \widehat{\vee} 2\bar{x}_1 \widehat{\vee} 3x_2$ is an LWL, while $5x_1 \widehat{\vee} 13x_2x_3$ is not.

The procedure of restoration of a separate function from a LWL is formally described by a **masking operator**.

Definition 2. The *masking operator* is an operator to extract the ξ -th digit, $\xi \in \{0, \dots, k\}$, of an m -valued representation of the number $A = m^k a_k + m^{k-1} a_{k-1} + \dots + m^\xi a_\xi + \dots + m^0 a_0$,

$$\Xi^\xi \{A\} = \left\lfloor \frac{A}{m^\xi} \right\rfloor \bmod m = a_\xi. \quad (2)$$

Example 3. Given $A = 10$, $\xi = 2$ and $m = 3$, we obtain $a_2 = \left\lfloor \frac{10}{3^2} \right\rfloor \bmod 3 = 1$.

Given the LWL of an r -output MVL function

$$LWL = a_1 x_1 \widehat{\vee} b_1 \bar{x}_1 \widehat{\vee} a_2 x_2 \widehat{\vee} b_2 \bar{x}_2 \widehat{\vee} \dots \widehat{\vee} a_n x_n \widehat{\vee} b_n \bar{x}_n,$$

$a_i, b_i \in \{0, 1, \dots, m^{r-1}\}$, we can restore the initial single-output functions

$$\begin{aligned} & a_1^{(1)} a_1^{(2)} \dots a_1^{(r)} x_1 \widehat{\vee} b_1^{(1)} b_1^{(2)} \dots b_1^{(r)} \bar{x}_1 \\ & \widehat{\vee} a_2^{(1)} a_2^{(2)} \dots a_2^{(r)} x_2 \widehat{\vee} b_2^{(1)} b_2^{(2)} \dots b_2^{(r)} \bar{x}_2 \widehat{\vee} \dots \\ & \widehat{\vee} a_n^{(1)} a_n^{(2)} \dots a_n^{(r)} x_n \widehat{\vee} b_n^{(1)} b_n^{(2)} \dots b_n^{(r)} \bar{x}_n \\ & = \underbrace{\left[\bigvee_{i=1}^n (a_i^{(1)} x_i \vee b_i^{(1)} \bar{x}_i); \bigvee_{i=1}^n (a_i^{(2)} x_i \vee b_i^{(2)} \bar{x}_i); \dots; \right.}_{\text{Output } f_1} \underbrace{\left. \bigvee_{i=1}^n (a_i^{(r)} x_i \vee b_i^{(r)} \bar{x}_i) \right]}_{\text{Output } f_r} \end{aligned}$$

where upper index of a_i (b_i) denotes the number $(1, 2, \dots, r)$ of a digit in a_i (b_i).

An arbitrary multi-output Boolean function given in a Sum-Of-Product (SOP) form can be represented in a unique way by a weighted logic expression. Moreover, the above mentioned properties are also true for ESOP expressions [8].

In our study, we use logic functions that correspond to elementary gates from a standard library of cells.

We explain motivation of our interest in the LWL description of logic functions by the below example.

Example 4. Let us compare the LWL and Arithmetic expressions (ARs) for the two-output Boolean function $f_1 = x_1 \vee x_2$, $f_2 = \bar{x}_1 \vee x_2$:

$$\begin{aligned} LWL &= x_1 \widehat{\vee} 2\bar{x}_1 \widehat{\vee} 3x_2, \\ AR &= 2f_2 + f_1 = 2(\bar{x}_1 + x_2 - \bar{x}_1 x_2) \\ &+ (x_1 + x_2 - x_1 x_2) \\ &= x_1 + 2\bar{x}_1 + 3x_2 - x_1 x_2 - 2\bar{x}_1 x_2 \end{aligned}$$

We observe, that the obtained AR is not linear (it requires a special technique for linearization). In contrast to LARs [12], an LWL always exists for an elementary Boolean function and is not sensitive to the order of inputs. The problem of linearization of ARs is even more complex for MVL functions, and the LWL model is an alternative solution.

3 Problem formulation

We consider a feedforward network, i.e. a network of interconnected nodes with no feedback. Traditionally, prototyping of a combinational circuit over library of MVL cells is a mapping of *each cell* of this library to a neuron, or several neurons. In contrast, we represent *every level* of the circuit by an NLG. Keeping in mind LAR-based model [12], we are finding an equivalent mapping via LWLs. So, we formulate the problem as follows: **given** a multiple-valued, multi-level combinational network over the library of gates, **find** an LDD-based model of the NLN to implement the circuit.

The strategy: the task is divided into two sub-problems, namely, (i) describing each level of a given network by an LWL, and (ii) generating an LDD for the corresponding linear expression. The derived set of l LDDs is a model of an l -node network.

4 NLG based on LWL representation

In [12] a word-level neuron model based on LAR expression has been introduced. We continue our study and report another model, that is based on LWL.

Let \odot denotes any of logic operations *MAX*, *MODSUM* or *TSUM*. For instance, $x_1 \odot x_2$ means $x_1 \vee x_2$ if $\odot = \max(x_1, x_2)$. We denote the corresponding digit-wise operation by $\widehat{\odot}$, with respect to \odot function. In case of MIN and TPROD gates we apply DeMorgan's theorem to obtain MAX and TSUM instead.

Example 5. Let us build an LWL model of a two-output function with outputs $f_1 = x_1x_2$ and $f_2 = x_2x_3$. Applying DeMorgan's theorem we get

$$\begin{aligned}\bar{f}_1 &= \bar{x}_1 \vee \bar{x}_2 = \Xi^1\{\overline{LWL}_{MIN}\} \\ \bar{f}_2 &= \bar{x}_2 \vee \bar{x}_3 = \Xi^2\{\overline{LWL}_{MIN}\}\end{aligned}$$

where

$$\begin{aligned}\overline{LWL}_{MIN} &= 3^1\bar{f}_2 + 3^0\bar{f}_1 = 3^0(\bar{x}_1 \vee \bar{x}_2) + 3^1(\bar{x}_2 \vee \bar{x}_3) \\ &= 3^0\bar{x}_1 \widehat{\vee} (3^0 + 3^1)\bar{x}_2 \widehat{\vee} 3^1\bar{x}_3.\end{aligned}$$

Note, the operation *TPROD* can be expressed in the same way: for instance if $f_1 = x_1 \sqcap x_2$ then $\bar{f}_1 = \bar{x}_1 \sqcup \bar{x}_2$.

In general, an LWL expression of an MVL is specified in terms of the digit-wise logical operations.

Definition 3. An LWL of an n -variable MVL function of over *MAX*, *MODSUM* or *TSUM* operation, and also *MIN*, *TPROD* operation (applied DeMorgan's theorem) is expressed by

$$LWL = \bigodot_{i=1}^n w_{i,q} x_{i,q}, \quad (3)$$

where $x_{i,q}$ denotes a variable x_i with inversion (\bar{x}_i), cyclic inversion (\widehat{x}_i), or without inversion, and $w_{i,q} \geq 0$ is an integer number (weight).

Lemma 1. A value f_j of a j -th MVL function, $j \in \{1, \dots, r\}$, can be extracted from an LWL (3) by a masking operator (2) taken $\xi = j - 1$

$$f_j = \Xi^{j-1}\{LWL\} = \left\lfloor \frac{LWL}{m^{j-1}} \right\rfloor \bmod m. \quad (4)$$

Proof-sketch. The above statement follows from the fact that the digit-wise operation $\widehat{\odot}$ of the weights m^{j-1} , $j = 1, 2, \dots, r$, produces the same result as the arithmetical sum, i.e. $A = m^{r-1}a_{r-1} + m^{r-2}a_{r-2} + \dots + m^\xi a_\xi + \dots + m^0 a_0 = m^{r-1}a_{r-1}m^{r-2}a_{r-2} \widehat{\odot} \dots \widehat{\odot} m^\xi a_\xi \widehat{\odot} \dots \widehat{\odot} m^0 a_0$.

Example 6. Given the 3-valued $LWL = 10x_1 \widehat{\odot} 4x_2$, the values of the following functions can be extracted.

Operator \odot	f_3	f_2	f_1
<i>MAX</i>	$x_1 \vee x_2$	x_2	x_1
<i>MODSUM</i>	$x_1 \oplus x_2$	x_2	x_1
<i>TSUM</i>	$x_1 \sqcup x_2$	x_2	x_1
<i>MIN</i>	$\bar{x}_1 \vee \bar{x}_2$	\bar{x}_2	\bar{x}_1
<i>TPROD</i>	$\bar{x}_1 \sqcup \bar{x}_2$	\bar{x}_2	\bar{x}_1

5 Technique of NLN model design

The proposed technique is a three-step strategy: preprocessing, forming LWL-models of NLGs and deriving NLN model (a set of LDDs).

1. **Preprocessing.** We apply a regular procedure that sorts the gates of a circuit with respect to their type by the rule: every level of a circuit has to include the gates of the same type, for example, MIN or MAX gates only. Note, that after this procedure the number of levels in a circuit can increase.

2. **Designing the NLN network.** We represent every level of the obtained circuit by a corresponding LWL.

So, after this step the circuit is modelled by a set of LWLs. In contrast with [12], the LWL expressions utilize bit-wise logical operations.

3. Forming LDD model. We construct an LDD for each LWL. Then, we connect LDDs with a procedure of data transmission between the LDDs.

Example 7. Let us build an LWL-based model of an NLG to implement a 2-input, 3-output level of an MVL circuit consisting of the gates $f_1 = x_1 \odot x_2$, $f_2 = \bar{x}_1 \odot x_2$, $f_3 = x_1 \odot \bar{x}_2$. Recall, that \odot operation denotes MAX, MODSUM, TSUM (in case of MIN and TPROD DeMorgan's theorem should be applied). The LWL model of the level is

$$\begin{aligned} & 3^2 f_3 \hat{\odot} 3^1 f_2 \hat{\odot} 3^0 f_1 \\ &= 3^2(x_1 \odot \bar{x}_2) \hat{\odot} 3^1(\bar{x}_1 \odot x_2) \hat{\odot} 3^0(x_1 \odot x_2) \\ &= (3^2 + 3^0)x_1 \hat{\odot} (3^1 + 3^0)x_2 \hat{\odot} 3^1 \bar{x}_1 \hat{\odot} 3^2 \bar{x}_2 \\ &= 10x_1 \hat{\odot} 4x_2 \hat{\odot} 3\bar{x}_1 \hat{\odot} 9\bar{x}_2. \end{aligned}$$

So, we can calculate the outputs f_1, f_2 and f_3 as follows

$$[f_3 \ f_2 \ f_1] = \begin{bmatrix} 10 \cdot 0 & \hat{\odot} & 4 \cdot 0 & \hat{\odot} & 3 \cdot 2 & \hat{\odot} & 9 \cdot 2 \\ 10 \cdot 0 & \hat{\odot} & 4 \cdot 1 & \hat{\odot} & 3 \cdot 2 & \hat{\odot} & 9 \cdot 1 \\ 10 \cdot 0 & \hat{\odot} & 4 \cdot 2 & \hat{\odot} & 3 \cdot 2 & \hat{\odot} & 9 \cdot 0 \\ 10 \cdot 1 & \hat{\odot} & 4 \cdot 0 & \hat{\odot} & 3 \cdot 1 & \hat{\odot} & 9 \cdot 2 \\ 10 \cdot 1 & \hat{\odot} & 4 \cdot 1 & \hat{\odot} & 3 \cdot 1 & \hat{\odot} & 9 \cdot 1 \\ 10 \cdot 1 & \hat{\odot} & 4 \cdot 2 & \hat{\odot} & 3 \cdot 1 & \hat{\odot} & 9 \cdot 0 \\ 10 \cdot 2 & \hat{\odot} & 4 \cdot 0 & \hat{\odot} & 3 \cdot 0 & \hat{\odot} & 9 \cdot 2 \\ 10 \cdot 2 & \hat{\odot} & 4 \cdot 1 & \hat{\odot} & 3 \cdot 0 & \hat{\odot} & 9 \cdot 1 \\ 10 \cdot 2 & \hat{\odot} & 4 \cdot 2 & \hat{\odot} & 3 \cdot 0 & \hat{\odot} & 9 \cdot 0 \end{bmatrix}$$

$$= \begin{bmatrix} 000 & \hat{\odot} & 000 & \hat{\odot} & 020 & \hat{\odot} & 200 \\ 000 & \hat{\odot} & 011 & \hat{\odot} & 020 & \hat{\odot} & 100 \\ 000 & \hat{\odot} & 022 & \hat{\odot} & 020 & \hat{\odot} & 000 \\ 101 & \hat{\odot} & 000 & \hat{\odot} & 010 & \hat{\odot} & 200 \\ 101 & \hat{\odot} & 011 & \hat{\odot} & 010 & \hat{\odot} & 100 \\ 101 & \hat{\odot} & 022 & \hat{\odot} & 010 & \hat{\odot} & 000 \\ 202 & \hat{\odot} & 000 & \hat{\odot} & 000 & \hat{\odot} & 200 \\ 202 & \hat{\odot} & 011 & \hat{\odot} & 000 & \hat{\odot} & 100 \\ 202 & \hat{\odot} & 022 & \hat{\odot} & 000 & \hat{\odot} & 000 \end{bmatrix} = \begin{bmatrix} c_0 & b_0 & a_0 \\ c_1 & b_1 & a_1 \\ c_2 & b_2 & a_2 \\ c_3 & b_3 & a_3 \\ c_4 & b_4 & a_4 \\ c_5 & b_5 & a_5 \\ c_6 & b_6 & a_6 \\ c_7 & b_7 & a_7 \\ c_8 & b_8 & a_8 \end{bmatrix}$$

In particular, given the assignments $x_1 x_2 \bar{x}_1 \bar{x}_2 = \{0022\}$ and $\hat{\odot} = \hat{\vee}$, the values of functions are equal to $a_0 = 0 \odot 0 \odot 0 \odot 0 = 0 \vee 0 \vee 0 \vee 0 = 0$, $b_0 = 0 \odot 0 \odot 2 \odot 0 = 0 \vee 0 \vee 2 \vee 0 = 2$, $c_0 = 0 \odot 0 \odot 0 \odot 2 = 0 \vee 0 \vee 0 \vee 2 = 2$.

Then, since $\hat{\odot} = \hat{\vee}$, the $LWL = 10x_1 \hat{\odot} 4x_2 \hat{\odot} 3\bar{x}_1 \hat{\odot} 9\bar{x}_2$, a level of a circuit consisting, for instance, of three MAX gates can be modelled in a way proposed in [11],

i.e. using a triple of neurons to implement one MVL gate (Figure 1b). In our approach, the NLN consists of one node, NLG (Figure 1c), and the word-level model is a four-node LDD (Figure 1d). Hence, an LWL-based model can be created for any level of an MVL circuit.

Example 8. Let us compare the models of a level of an MVL circuit based on (i) TG (ii) LWL and (iii) Arithmetic expressions (ARs) for the level of a ternary circuit with three MAX gates given in Figure 1a.

The corresponding TG model that maps every logic gate to three neurons is given in Figure 1b.

By (3), we obtain

$$LWL = 3^0 x_1 \vee 3^0 x_2 \vee 3^1 x_3 \vee 3^1 x_4 \vee 3^2 x_5 \vee 3^2 x_6,$$

and using the approach described in [12] we get

$$\begin{aligned} LAR &= 15897 + 5x_1^\circ + x_2^\circ + 135x_3^\circ + 27x_4^\circ \\ &+ 3645x_5^\circ + 729x_6^\circ. \end{aligned}$$

We observe that two types of word-level expressions are different in the values of their coefficients, and in the variables (pseudo-variables in case of LAR). This will cause the different terminal values in the corresponding decision diagrams.

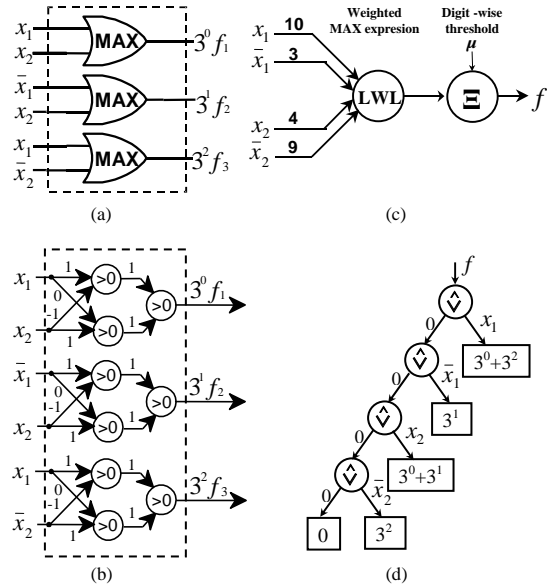


Figure 1. Models of the MVL circuit level

An LWL can be described by an LDD. To explain the decomposition type used in nodes of LDDs, let us

analyze the $LWL = x_1 \widehat{\vee} 2\bar{x}_1 \widehat{\vee} 3x_2$. Due to the associative law that holds for each operator $\widehat{\vee}$, the LWL can be rewritten as $LWL = x_1 \widehat{\vee} (2\bar{x}_1 \widehat{\vee} 3x_2)$, where the integer coefficient of x_1 is equal to 1 and $(2\bar{x}_1 \widehat{\vee} 3x_2)$ is a value of the LWL if x_1 is substituted with 0. Hence, the node of the LDD corresponds to the expression: $LWL = d_0x \widehat{\vee} LWL|_{x=0}$.

Example 9. The LDD for the LWL $(3^0 + 3^2)x_1 \widehat{\vee} 3^1\bar{x}_1 \widehat{\vee} (3^0 + 3^1)x_2 \widehat{\vee} 3^2\bar{x}_2$, that describes a ternary circuit level with three MAX gates, derives the single-node network given in Figure 1c. The LDD corresponding to the above LWL is shown in Figure 1d. So, our model requires a single-node structure while the TG model [11] involves 9 neurons.

The LWLs of a circuit level that consists of $n/2$ (n is an even integer) 2-input ternary gates are presented in Table 1. Compared to [12], the LWL models utilize variables, not pseudo-variables. A drawback of the LWL model are that the level has to consist of the same type of gates. The other drawback is that it is necessary to apply DeMorgan's theorem in case of MIN and TPROD functions.

Table 1. The LWLs of a circuit level that includes $n/2$ 2-input ternary gates

MAX	$\widehat{\bigvee}_{j=1}^{n/2} 3^{j-1}(x_{2j-1} \vee x_{2j})$
$MODSUM$	$\widehat{\bigoplus}_{j=1}^{n/2} 3^{j-1}(x_{2j-1} \oplus x_{2j})$
$TSUM$	$\widehat{\prod}_{j=1}^{n/2} 3^{j-1}(x_{2j-1} \sqcup x_{2j})$
\overline{MIN}	$\widehat{\bigvee}_{j=1}^{n/2} 3^{j-1}(\bar{x}_{2j-1} \vee \bar{x}_{2j})$
\overline{TPROD}	$\widehat{\prod}_{i=1}^{n/2} 3^{i-1}(\bar{x}_{2j-1} \sqcup \bar{x}_{2j})$

The above explanation allow us to introduce our main result formulated as a theorem.

Theorem 2. An arbitrary l level MVL circuit over a standard library of cells can be represented by an NLN with l NLGs; every NLG is formally described by an LWL that is mapped into an LDD, so that the resulting NLN is mapped into l LDDs.

Proof. Since a level of a circuit is represented by an NLG, the NLN description required l NLGs. Also, the

formal model of an NLG, LWL, is mapped to an LDD; consequently l NLG will correspond to l LDDs. \square

6 Algorithm

The algorithm to generate an LDD-based NLN model from an MVL network is given in Figure 2a. An important part of the algorithm is a data transmission between LDDs (procedure *Mail*). We utilize (with some modification) the concept of data exchange between two cells in linear systolic array [7]. Two arbitrary LDDs are connected through a mailbox (Figure 3). The procedures of transferring the data between the two LDDs and the algorithm to generate an NLN is given in Figure 2b.

```

procedure Build LDD(circuit)
{
  for( $\forall$  level  $l$  in a network)
  {
    Create LDD for the level  $l$ ;
    for( $\forall$  gate $_t$  in the level  $l$ )
    {
      calculate a weight $_t$  for gate $_t$ ;
      for( $\forall$  gate $_t$  in the level  $l$ )
      {
        for( $\forall$  input  $in$  in the gate $_t$  equal to the inputs in the gate $_t$ )
        {
          if( $in$  is not presented in the LDD for the level  $l$ )
          {
            then add a new node with the variable  $t_{in}$ 
            corresponding to the given input  $in$  of
            the gate $_t$ ;
          }
          add a value weight $_t$  to the existing terminal node
          with the variable  $t_{in}$  ;
        }
      }
    }
  }
}

```

(a)

```

procedure Mail(no_of_LDD $_1$ , no_of_LDD $_2$ , ...,
no_of_node $_1$ , no_of_node $_2$ , ..., value,  $\mu$ )
{
  transmit value and  $\mu$  to no_of_node $_1$ -th node
  in no_of_LDD $_1$ -th, and next diagrams
}

```

(b)

Figure 2. Algorithm to generate an NLN (a) and data transmission procedure (b)

7 Experiments

We modelled the benchmark circuits (with about 5000 gates) from LGSynth'93¹ base on PC Pentium III 450 MHz processor with 128 MB RAM. The binary gates of the benchmarks were replaced with ternary gate as follows: AND gate with MIN gate, and OR gate

¹http://zodiac.cbl.ncsu.edu/CBL_Docs/lgs93.html

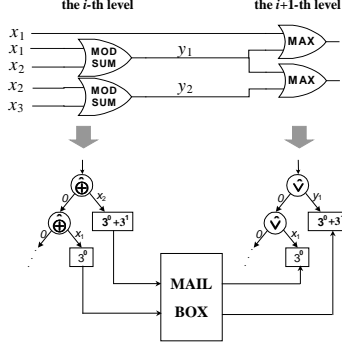


Figure 3. A mailbox procedure

with MAX gate. Our program *NetDesign* was written in C++.² We compared LWL and LAR based NLNs on the ISCAS benchmarks (Table 2) where **In** and **Out** is a number of inputs and outputs, **#G** is a number of gates, **#L** is a number of levels, and **MEM** is memory in bytes required to represent circuit in ISCAS format.

In certain experiments we compared the memory requirements (Table 3), the number of nodes in LDDs, the maximal weights of terminal node (Table 4), and run time of calculation based on the LDD models (Table 5).

Table 2. Circuits in ISCAS format

Name	In	Out	#G	#L	Memory
C17	5	2	6	3	1303
C432	36	7	160	18	18991
C499	41	32	202	12	21989
C880	60	26	383	50	37844
C1355	41	32	546	25	59573
C1908	33	25	880	71	78574
C2670	233	64	1193	68	110025
C3540	50	22	1669	119	145359
C5315	178	123	2307	86	220596
C6288	32	31	2416	126	255406
C7552	207	107	3512	75	311939
Total					1.261.599

The results on comparison of memory needs for the approach given in [12] (denoted **MEM** [12]) and method proposed in this paper (denoted **MEM**) are given in Table 3. We observe that the proposed approach requires more than two times less memory compared to [12]. Also, the traditional ISCAS circuit format **MEM** (Table 2) consumes in average about 5 times more memory than ours.

In the next experiment, summarized in Table 4, we compared the LDDs derived from LAR [12] and from LWL with respect to (i) the total number of nodes **#N**, (ii) the number of nodes in the LDD with the

²The package is available from the authors

Table 3. Comparison of the size of memory

Test	I/O	MEM [12]	MEM (LWL)
C17	5/2	201	48
C432	36/7	4601	2688
C499	41/32	5427	3672
C880	60/26	8306	5832
C1355	41/32	16850	9576
C1908	33/25	22491	14980
C2670	233/64	36305	20760
C3540	50/22	54757	32329
C5315	178/123	107392	48246
C6288	32/32	68572	52778
C7552	207/107	271970	73632
Total		596872	264541

biggest number of nodes N_{max} , (iii) the total number of digits to represent LDD terminal weights **#T**, and (iv) the number of digits in the LDD with the biggest number of nodes TW_{max} . Notice, the number of digits to represent the LDD terminal weights is one of the main characteristics of the word-level expressions and diagrams. They are huge exponential numbers, and we use a special technique to encode them. We applied our technique to both types of LDDs and observed that, for example, in a case of circuit C6288, the number of nodes equals $\#N=3616$, whereas in LDDs constructed from LARs the number of nodes is $\#N=6917$, i.e. about two times more. In [12], the largest LDD includes $N_{max}=256$ nodes, four times more in comparison with the proposed LWL-based approach ($N_{max}=60$). The terminal weights are 6.48 times less for LWL-based models ($\#T=4798$ digits) than for LAR-based models ($\#T=31126$). Considering the biggest LDDs, LWL-based LDDs also required less digits. For example, in a case of C6288 circuit, the biggest LDD requires $T_{max}=510$ digits, about 35 percent less than $T_{max}=768$ needed in [12]).

Table 4. Comparison of the number of nodes

Test	LAR-based [12]		LWL-based	
	$\#N(N_{max})$	$\#T(T_{max})$	$\#N(N_{max})$	$\#T(T_{max})$
C17	23(5)	103(6)	9(3)	12(4)
C432	503(46)	2263(63)	263(45)	336(54)
C499	880(105)	3960(200)	267(64)	408(64)
C880	860(59)	3870(148)	587(32)	729(62)
C1355	1354(129)	6093(192)	779(96)	1064(128)
C1908	1140(71)	5130(101)	1206(79)	1498(122)
C2670	1715(228)	7717(372)	1699(102)	2076(220)
C3540	2397(202)	10786(311)	2257(179)	2939(283)
C5315	3977(356)	17896(646)	3206(202)	4386(448)
C6288	6917(256)	31126(768)	3616(60)	4798(510)
C7552	5935(407)	26707(561)	4847(624)	6136(814)
Total	25701(1864)	115651(3368)	18727(1486)	24371(2709)

In Table 5 we give two characteristics, namely CPU time to design LDD based NLN and runtime to calculate MVL functions on the NLN models. The columns

LDD [12] and **LDD** contain CPU time to transform a circuit given in ISCAS format to the LDD format, for LDDs derived from LAR and LWL, respectively. In the column **Mailbox** we place CPU time to design links between LDDs. In the next three columns we present a results on a simulation of functions calculation for 10,000 randomly generated input assignments. The columns **Circuit**, **LDD**[12] and **LDD** contain the average CPU time for ISCAS format, LDD format derived from LAR [12] and the proposed LDD format derived from LWL, accordingly. We observe that CPU time of all model is almost the same.

Table 5. Comparison of the RUN time

Test	CPU time to design [s]			CPU time to calculate [ms]		
	LDD[12]	LDD	Mailbox	Circuit	LDD[12]	LDD
C17	0.00	0.00	0.00000	0.00	0.00	0.00
C432	0.04	0.01	0.0014	0.10	0.15	0.12
C499	0.15	0.02	0.0017	0.14	0.16	0.14
C880	0.10	0.05	0.0083	0.18	0.33	0.30
C1908	0.12	0.39	0.0760	0.24	0.25	0.74
C1355	0.25	0.07	0.0200	0.24	0.28	0.45
C2670	0.91	0.24	0.1900	0.37	0.49	1.02
C3540	1.14	0.29	0.2400	0.44	0.59	1.49
C5315	2.06	0.51	0.5150	1.24	1.58	2.29
C6288	1.43	0.42	0.4950	1.10	2.19	2.60
C7552	3.69	0.69	0.9600	2.01	2.34	3.83
Total	9.70	2.66	2.696	6.06	8.36	12.98

Conclusion

We have extended the LAR-based model NLG proposed in [12]. We demonstrated a number of useful properties of the LWL models, in particular, the planarity of their diagrams, LDDs, and simplicity of a corresponding NLN modeled by a set of the LDDs. This is the main contribution of our study. We also showed in a series of experiments, that the NLN, as a data structure, allows to manipulate large MVL circuits.

The expected high "cost" of achieving the linearity and planarity of the proposed circuit models has not been observed. Instead, the LDD format exhibits drastically less memory needs compared with some traditional circuit formats (2-7 times compared to ISCAS format), whereas the run time of calculation by NLN and the initial circuit is comparable. The results of comparison with EDIF format are even more impressive.

We continue our experimental study with complex interconnected networks. If successful for industrial networks, the proposed approach can tackle with the problems of realization of complex interconnected systems like neural networks.

Acknowledgment. The authors thank to *Prof. V. Malyugin* (Russian Academy of Science, Russia) for long-term interest and support in the area of study, and also the anonymous reviewers for their useful comments.

References

- [1] S. T. Chakradhar, V. D. Agrawal, M. L. Bushnell, *Neural models and algorithms for digital testing*, Kluwer Academic Publishers, Boston, USA, 1991
- [2] S. Cotofana, S. Vassiliadis, Periodic Symmetric Functions, Serial Addition, and Multiplication with Neural Networks, *IEEE Trans. on Neural Networks*, 1998, vol. 9, no. 6, pp. 1118 - 1128
- [3] V. Feinberg, A. Levin, E. Rabinovich, *VLSI Planarization Methods, Models, Implementation*, Kluwer Academic Publishers, 1997
- [4] H. Fujiwara, Three Valued Neural Networks for Test Generation, *Proc. 20th IEEE Int. Symp. on Fault Tolerant Computing*, 1990, pp. 64-71
- [5] Y. Hata, T. Hozumi, K. Yamato, Gate Model Networks for Minimization of Multiple-Valued Logic Functions *Proc. 23rd IEEE Int. Symp. on Multiple-Valued Logic*, 1993, pp. 29 - 34
- [6] T.Hozumi, N. Kamiura, Y. Hata, K. Yamato, On Minimization of Multiple-Valued Sum-of-Products Expression with Multiple-Valued TRSUM, *Int. Journal on. Multiple-Valued Logic*, vol. 2, 1997, pp. 141 - 158
- [7] G. Kukharev, V. Shmerko, S. Yanushkevich, *Technique of Binary Data Parallel Processing on VLSI*, Vysheyschaya Shkola Publishers, Minsk, Belarus, 1991 (In Russian)
- [8] V. Malyugin, Parallel Calculation by Means of Ayrthmetical Polynomials, *Physical and Mathematical Publishing Company, Russian Academy of Sciences* Moscow, Russia, 1997 (In Russian)
- [9] W. S. McCulloch, W. Pitts, A Logical Calculus of the Ideas Immanent in Neural Nets, *Bulletin of Mathematical Biophysics*, no. 5, 1943, pp. 115 - 137
- [10] C. Moraga, R. Heider, New Lamps for Old (Generalized Multiple-Valued Neurons), *Proc. 26th IEEE Int. Symp. on Multiple-Valued Logic*, 1999, pp. 36 - 41
- [11] Z. Tang, O. Ishizuka, Q. Cao, H. Matsumoto Algebraic Properties of a Learning Multiple-Valued Logic Network, *Proc. 23rd IEEE Int. Symp. on Multiple-Valued Logic*, 1993, pp. 196 - 201
- [12] S. N. Yanushkevich, V. P. Shmerko, P. Dziurzanski, The Word-Level Models for Efficient Computation of Multiple-Valued Functions, PART 1: LAR Based Model, *Proc. 32nd IEEE Int. Symp. on Multiple-Valued Logic*, 2002