# A System for PLA Minimization based on Slight Behavior Modifications

Reinhard Rauscher
University of Hamburg
Dept. of Computer Science
22527 Hamburg, Germany
e-mail: rauscher@informatik.uni-hamburg.de

**Abstract**

*A PLA minimization system is presented, which is dedicated for PLA-realizations of small tables containing integer values. The initial specification of these tables can be modified sligthly. Starting with a concise initial specification a widened specification (e.g. alternative rounding of real values to the next higher or lower integer) can be derived. From this widened specification a modified specification with a minimal power consumption is determined.*

*By using this system, a set of examples has been analyzed. In average, the resulting PLAs have 30 % less power consumption than those obtained from the original specification. The application area of the system could be signal processing applications involving analog functions.*

## 1 Introduction

Since the complexity and the clock frequencies of VLSI chips increase permanently, the power dissipation becomes critical with respect to heat and current dissipation (size of supply nets) [12]. Although a lot of work is done in the field of power dissipation estimation and also in its minimization using CMOS technology (e.g. [12], [9]), there are some problems, especially with PLAs. On one hand PLAs are well suited for automatic design of any combinatoric circuits and they need far less chip area than a functionally equivalent set of standard cells. In particular, Programmable Logic Arrays (PLAs) proved to be a very effective tool for implementing combinational functions [15]. On the other hand their power dissipation is so high, that often additional power pads have to be included in a chip design.

Furthermore, in some cases the PLA generators can not be used because of the high power consumption of the automatically generated PLA realizations. These arguments have been the reason for avoiding the use of PLAs for several practical designs (e.g. [16]).

The objectives of a PLA-optimization are:

- Decrease of power consumption of the PLA
- Decrease of chip area used for the PLA
- Decrease of the PLAs access time

The need for an optimizing tool has been seen from several practical designs as mentioned above. The ASIC for delta modulation [16] will be used as an example to demonstrate the necessity of such a tool:

1. The specification contains two tables of integer values. Both tables do not have $2^{|address|}$ lines and the tables contain the value "0" several times. A realization using two PLAs seems to be more economical than a ROM.

2. The application area is within digital transmission of analog signals. Especially the application area of wireless telephones should be mentioned.

   A minimal power consumption is mandatory, demanding an implementation using CMOS technology *and* a low clock frequency. These constraints contradict with the enormous static power consumption of NMOS-PLAs. So, in the first version of the ASIC-designs no PLA has been used.

3. The modulation and the demodulation unit are of identical structure. So, minimal deviations of the tables in the initial specification can be tolerated. For some of these deviations no differences in the total behavior of both units have been observed.

Especially the last item has been one of the motivations for the work described in this paper. Most often the initial specification has to be completed during a

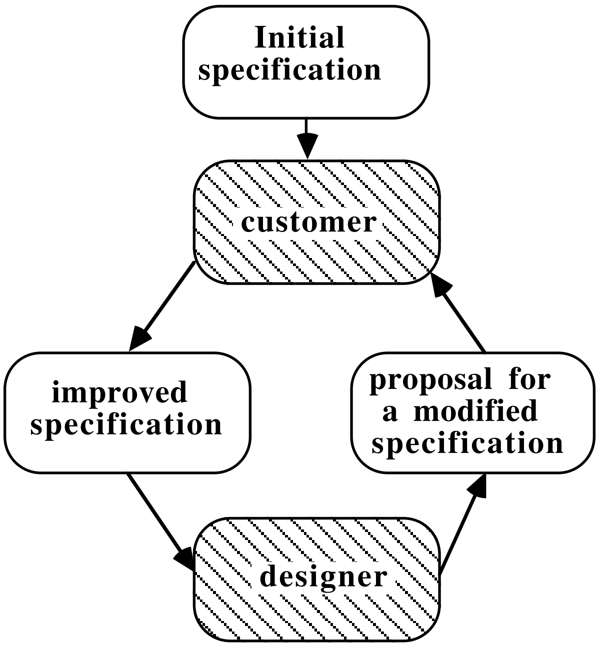


Figure 1: Definition procedure of the specification

top-down design and a lot of iteration steps take place until the final specification is derived.

In Figure 1 the usual procedure is illustrated. Starting from an initial specification, a set of alternatives for the completion of the specification will be considered. From the view of the designer, completions or modifications, which reduce the design task and/or the chip area, will be highly desirable. The designer may suggest modifications of the specification to reduce the complexity of the design task and/or the realization cost of the design itself.

The customer will accept only such modifications, which retain the intended behavior and reduce the estimated costs.

Our own experiences show that usually the initial specification and the final specification differ.

Summarized, there is the need for tools which give an estimation of cost reductions/advantages, if a slight modification of the actual specification could be accepted. The output of the tool should be a statement as: "If the specification is modified in the suggested manner, the costs will decrease by a factor of ... ".

Especially within the application area of micro systems, most often the designer has complete control and knowledge of the whole system. So, he can decide, whether 'small' changes of the initally desired behavior could be accepted or not.

Additionally, for VLSI-systems a modification of the initial specification must be done as shown in the following. If the need for the initial modification is accepted, a modification different to the usual one would be accepted, if advantages can be determined.

Furthermore, it should be remarked, that the following approach is only practicable, if both conditions hold:

- The ASIC itself contains information data stored in one or more small tables.
- This information is based on arithmetic functions.

## 2 State of the art

There is a lot of activities in logic synthesis ([10, 3, 4, 5, 8, 11, 6, 17]). Especially, those activities in the field of PLA realizations are restricted to small tables containing only a few terms (maximal amount is 46 in [10]) in the implementation. All of these activities are aimed to determine an optimal or near-by-optimal solution for a given problem specification.

Our approach is another one. Given an initial specification, we determine other possible specifications, and examine improvement in area, power consumption, and access time.

| i | val(i) |
|---|---|
| 0 | 0.00000 |
| 1 | 0.15643 |
| 2 | 0.30900 |
| 3 | 0.45398 |
| 4 | 0.58777 |
| 5 | 0.70709 |
| 6 | 0.80900 |
| 7 | 0.89099 |
| 8 | 0.95104 |
| 9 | 0.98768 |
| 10 | 1.00000 |

Figure 2: Initial part of a specification given as a table

## 3 Basic Approach

A typical example for a PLA design implements the conversion of real (analog) values of the SINE function into bit strings.

This example and modifications of its specification will be discussed in detail. In Figure 2 the initial specification of a table realizing the function $sin(\frac{\pi}{2} \times \frac{i}{10})$, $i \in \{0, ..., 10\}$ is illustrated.

Of course, there must be a compromise between accuracy and implementation costs and therefore an agreement for realization by bit strings of a fixed

$\boxed{\text{Initial specification}}$
$\Downarrow$
$\boxed{\text{Determination of the number of bits}}$
$\Downarrow$
$\boxed{\text{Rescaling}}$
$\Downarrow$
$\boxed{\text{Rounding/ Truncation}}$
$\Downarrow$
$\boxed{\text{Binary representation}}$

Figure 3: Illustration of the procedure to detail the specification

length will be made. Any advantages/disadvantages of decreasing the length of the bit strings can be estimated quite easy.

In Figure 3 the different tasks are sketched. Starting with a discussion of the initial specification (see e.g. Figures 1, 2) the number of bits used for representation is determined.

The next step is named "rescaling". In this procedure a new scaling can be used, so that even so the initial specification did start with integer values, now also real values can be assumed.

In the step "rounding/truncation" the decision whether to choose one of the functions *round* or *truncate* for mapping from REAL values into INTEGER values has to be done. Additionally questions concerning accuracy *should* be discussed (with the customer).

The result of the procedure in Figure 3 is a (partially defined) binary representation, which could be used as an input for a PLA generation procedure.

| i | bit string | value |
|---|---|---|
| 0 | 0000000 | 0 |
| 1 | 0010100 | 20 |
| 2 | 0100111 | 39 |
| 3 | 0111010 | 58 |
| 4 | 1001011 | 75 |
| 5 | 1011010 | 90 |
| 6 | 1100111 | 103 |
| 7 | 1110001 | 113 |
| 8 | 1111001 | 121 |
| 9 | 1111101 | 125 |
| 10 | 1111111 | 127 |

Figure 4: Specification after first discussion concerning the number of bits

In Figure 4 a result of such a compromise between accuracy and ease of implementation is illustrated. It can be seen, that a realization by bit strings of 7 bits has been choosen. Additionally, in Figure 4 the third column describes the integer value of the bit string. This values can be generated using the following formula:

$tab[i] = round(127 * sin(\frac{\pi}{2} \times \frac{i}{10})),\ i \in \{0, ..., 10\}$

Another approach would be to ask whether the rounding as described above would be necessary. Alternatively, there could be realized one of both integer values described by the interval $[trunc(x), min\{maxvalue, trunc(x)+1\}]$. Most often, this results only in a slight —may be neglectable— modification of the specification, but could result in a considerable reduction of realization costs. For the example of the SINE function the resulting specification is illustrated in Figure 5.

| i | min.value | max.value |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 19 | 20 |
| 2 | 39 | 40 |
| 3 | 57 | 58 |
| 4 | 74 | 75 |
| 5 | 89 | 90 |
| 6 | 102 | 103 |
| 7 | 113 | 114 |
| 8 | 120 | 121 |
| 9 | 125 | 126 |
| 10 | 127 | 127 |

Figure 5: Specification after rescaling and introduction of intervals

Obviously, for at least some lines of the initial table these intervals can be widened.

The basic approach can be summarized as follows:

1. Discussion with the customer, which accuracies may be mandatory. For some items of the table widened intervals can be used or the values may in general differ from the initial by up to e.g. 5 %.

2. Determine the best suited modification of the initial specification with regard to the restrictions coming from item 1.

3. Change the specification.

Thus, there is a crucial need for a automatic estimation of implementation advantages for slight modifications of the specification. If a tool like this is available, it would also be possible to suggest a specification not included in the demands coming from item 1 and to discuss the advantages and disadvantages of such a modification with the customer.

### 3.1 Basic definitions

The table (PLA) can be considered as a $n \times m$-Matrix $M$ with components $a_{ij} \in \{0,1\}$. Usually, also don't care components valued "*" can occur, but in this work it is assumed that $M$ is totally specified.

The *first specification (fs)* of a PLA can be considered as a table of integer values with $n$ items denoted by *fs[i]*, $i \in \{1,...,n\}$. The *resulting final specification (rs)* can be defined analogous.

The *widened specification (ws)* is a table of $n$ pairs $(t_{min}[i], t_{max}[i])$. Each pair denotes the minimum and the maximum value of the item numbered by $i$.

Of course, the following condition holds:

$$t_{min}[i] \leq fs[i], rs[i] \leq t_{max}[i] \quad \forall i \in \{1,..,n\}.$$

Obviously, there exists a correspondence between $M$ and the different specifications based on integer values. This can be seen using the usual interpretation of bit strings (without don't cares) as unsigned integers. Of course, each bit string can be transformed into an interval as defined above.

Furthermore, there is a one-to-one-mapping between single integer values and bit strings. In the following the procedures *bit2int* and *int2bit* are used for this conversion.

This is not valid for the mapping from an interval to bit string: Starting from representation as an interval of integer values, there need not exist a bit string (may be, including the value "*") covering all the values within the interval. An example for this is the interval [3,4]. The binary representation would be either "...00011" or "...00100".

Therefore, both representations are used within the procedures.

This is shown in Figure 6.

As it can be seen from Figure 6, four different values have to be distinguished within the bit string representation. The difference between "*" and "-" is: A bit valued by "*" is a *don't care* as usual. A set of bits valued by "-" (named in the following *could_be_one*) denotes, that at least one of these bits has to be set to 1 for meeting the specification. Setting one of these bits would affect the remaining choices for the other ones. In other words, there are values in $[t_{min}[i], t_{max}[i]]$, for which the corresponding value (*int2bit*) would be '1' and there are other values within the same interval, for which a value '0' must be used. The task is to determine values from {0,1}, for all *could_be_one* values, so that the solution is feasible.

The optimization problem can be defined as follows:

Given an extended specification as a list of intervals. For each input value of the PLA a minimum ($t_{min}[i]$) and a maximum value ($t_{max}[i]$) is given. The task is to determine at least one solution $rs$ as a table of integer values, for which the following conditions hold:

- $t_{min}[i] \leq rs[i] \leq t_{max}[i]$ and
- the power consumption of the PLA-realization of $rs$ is minimal.

### 3.2 Strategies for searching the optimal PLA-specification

The power consumption of a NMOS-PLA depends linearly on the product of the number of terms and the number of input/output lines. The number of input/output lines is determined uniquely in the specification, the optimization procedure can be restricted in determining a solution with the minimal number of terms.

From [14] it is known, that the problem of determining a minimal PLA (for a specification as *fs*) belongs to the NP-complete problems. So, only heuristics can be used to determine the optimal solution *rs* starting from an initial solution *fs*.

Altogether five strategies to determine the best suited PLA-specification have been realized so far. It should be mentioned, that these strategies are considered as alternatives. For each example the best suited is choosen for the implementation. First, the strategies themselves are sketched.

**Strategy 1:**
In a simple approach, we first determine the power consumption of the specifications given by analyzing only minimum or maximum values of *ws*. The first solution for the case of considering minimum values is determined as follows: $rs[i] = ws_{min}[i]$. The second solution based on the maximum values of $ws$ is analogous. Additionally, the results are used for determining further decisions.

As a third approach it is determined, how much of the least significant bits in $M$ can be set to the value 0. This approach would be similiar to the trivial idea to decrease the number of columns within $M$. The results of this trivial approach are compared to the results of the following more sophisticated strategies.

**Strategy 2:**
The second strategy is based on the idea to minimize

the number of '1'-values in $rs$. For each row $i$ of the table a value $x$ is determined, for which the following conditions hold:

1. $x \in [t_{min}[i], t_{max}[i]]$
2. $\forall y \in [t_{min}[i], t_{max}[i]]$ : the number of '1's in $int2bit(x)$ is less or equal to the number of '1's in $int2bit(y)$.

It should be mentioned, that there is a difference to the third approach of strategy 1. That difference can be seen considering the case, that more than one alternative for determination of the value of $x$ exists. In this case a value is choosen due to strategy 3.

**Strategy 3:**
In this approach the determination of smaller intervals is done by analyzing the binary representation of the adresses.

Let be given $i, j \in \{1, ..., n\}$. If the Hamming distance between $int2bit(i)$ and $int2bit(j)$ is equal to 1, and the intervals $[t_{min}[i], t_{max}[i]]$ and $[t_{min}[j], t_{max}[j]]$ are not disjoint, then the new interval for $i$ and $j$ is determined as: $[t_{min}[i], t_{max}[i]] \cap [t_{min}[j], t_{max}[j]]$.

**Strategy 4:**
This strategy can be considered to be an extension of strategy 2. Successively for each column the number of the distinct values is counted. As mentioned above, we distinguish between the four following values: '0', '1', 'dont care', and 'could_be_one'.

Now, for each column the number of all these values is determined. In the second step a dominant value is extracted from these values. This is the value '1', if the number of '1' is larger than the number of '0'. In the third step, for each of the rows having the value 'could_be_one' the value of the bit string is set to this dominant value. In the last step the resulting interval is determined.

**Strategy 5:**
In Strategy 5 a variant of the well known method of Quine and McCluskey is realized. The idea in this approach is that the *could_be_one* values of different rows are independent from each other. The values *could_be_one* are set to '1' successively, if more terms can be merged than in the case that the respective values are set to '0'.

## 4 Complexity observations

The number of possible PLA-specifications which can be derived from the representation by minimum and maximum integer values is given by:

| i | min.value | max.value | binary represent. |
|---|---|---|---|
| 0 | 0 | 0 | 0000000 |
| 1 | 19 | 20 | 0010--- |
| 2 | 39 | 40 | 010---- |
| 3 | 57 | 58 | 01110-- |
| 4 | 74 | 75 | 100101* |
| 5 | 89 | 90 | 10110-- |
| 6 | 102 | 103 | 110011* |
| 7 | 113 | 114 | 11100-- |
| 8 | 120 | 121 | 111100* |
| 9 | 125 | 126 | 11111-- |
| 10 | 127 | 127 | 1111111 |

Figure 6: Example used for explanation of the strategies

$$\prod_{i=0}^{n-1}(t_{max}[i] - t_{min}[i] + 1).$$

Keeping in mind, that PLA-generators are able to handle don't cares, a more sophisticated estimation is:

$$\prod_{i=0}^{n-1} mp_i,$$

where $mp_i$ denotes the minimal number of prime implicants needed for covering all values between $t_{max}[i]$ and $t_{min}[i]$.

## 5 System Overview

In Figure 7 the system environment is illustrated. Additionally, it shows the usual way for PLA generation (using the thick lines).

Figure 7 also shows the most important components of the PLA optimization system (in shaded rectangles).

The first component (*generation of input table*) is used to generate academic examples and/or to generate $ws$ from $fs$ and user inputs.

The second component (*optimizer*) determines a set of specifications according to the strategies described above.

The third component named *evaluate differences* determines some statistical information concerning the difference between the original specification and the modified one. For example, it gives the maximum relative and absolute deviation from the original values. Furthermore, it verifies, that the proposed solution meets the initial interval constraints.

The component (*optimizer*) is used for the estimation of the advantages in area and power consumption. It contains two different programs, first an optimizer, second the generator itself. The optimizer would not be essential for our approach, but demon-

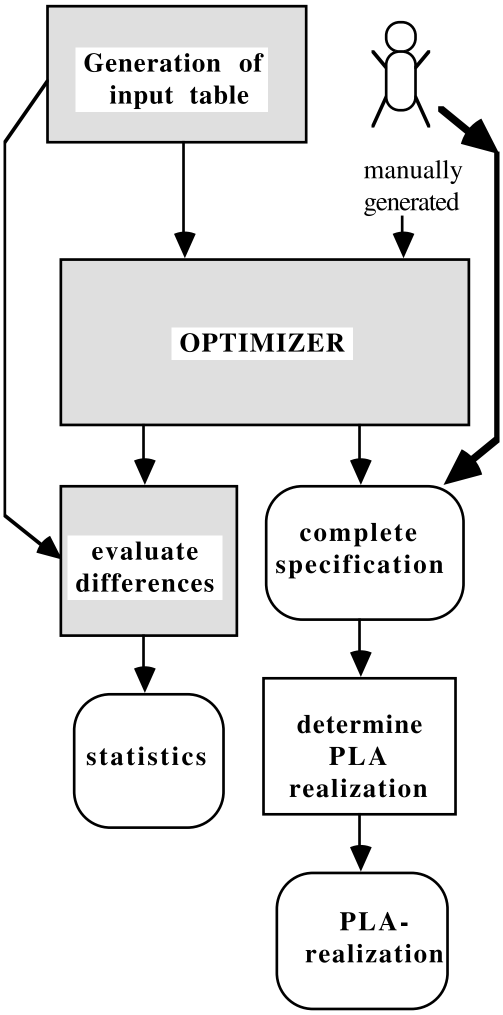


Figure 7: System overview

strates that any other optimizer could be included. Both programs are part of the SOLO 1400 system [13]. The first one, the program SYNTHESIZE (containing ESPRESSO [7]), performs an optimization of the modified PLA. This component can be substituted by any other optimizing component. The second component is the program GENERATE, which constructs a PLA-description to be used within a SOLO 1400 layout.

## 6 Examples

In order to evaluate the efficiency of our approach we analyzed a set of examples. Here we had to distinguish between handcrafted examples and automatically generated examples. For all examples we allowed a deviation of at most 5 % from the original value. Alternative rounding to the next higher and lower integer value has also been allowed.

The procedure to generate the example values is the following:

$$ws_{min}[i] := \lfloor (fs[i] \times 0.95) \rfloor$$
$$ws_{max}[i] := \lceil (fs[i] \times 1.05) \rceil$$

## 7 Implementation / Results

The results for several examples are shown in Figures 8, 9, and 10.

First of all the abbreviations in the columns of Figures 8, 9, and 10 have to be explained in detail:

- I/O: number of input lines, number of output lines
- org.a.: chip area of the solution generated from the original specification (in $mm^2$)
- org.t.: the number of product terms for the original specification (after optimization using ESPRESSO)
- org.p.: power consumption of the solution generated from the original specification (in mA)
- b.s: index of the best strategy
- mod.t.: the number of product terms for the modified specification (after optimization using ESPRESSO)
- mod.p.: power consumption of the solution generated from the modified specification (in mA)
- abs.d.: Maximum value of the differences between original and modified specification. $abs.d. := max\{abs(fs[i] - rs[i])\}$
- rel.d.: Maximum value of the relative differences between original and modified specification. $rel.d. := max\{abs(fs[i] - rs[i])/fs[i]\}$
- sqr.d: Sum of squares of the differences. $sqr.d := \frac{1}{n+1} \times \sqrt{\sum_{i=0}^{n} (fs[i] - rs[i])^2}$
- red.p: Reduction of power consumption in %. $red.p := 100 \times (org.p. - mod.p)/org.p$

Now, the examples themselves will be discussed.

- The example named "increasing" is a small table consisting of 64 items between the integer values 3 and 40.
- The example named "square root" determines the square root of six bit integers.
- "100/x" is a PLA-specification with 6 bit input and 7 bit output lines.
- $[(2^n - 1) \times \frac{log_2(x+1)}{n}]$ has been taken from [18]. It has 6 input lines and 7 output lines in the first example, in the second one the number of input and

| | | *original* | | | | *modified* | | *statistics* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| name | I/O | org.a. | org.t | org.p. | b.s. | mod.t | mod.p. | abs.d. | rel.d. | sqr.d | red.p. |
| increasing | $6 \times 6$ | 0.149 | 51 | 10.53 | 3 | 18 | 4.32 | 2 | 0.33 | 0.11 | 59 |
| square root | $6 \times 3$ | 0.072 | 20 | 4.32 | 3 | 7 | 1.88 | 1 | 0.50 | 0.08 | 56 |
| 100/x | $6 \times 7$ | 0.090 | 23 | 5.45 | 3,4 | 13 | 3.57 | 5 | 1.00 | 0.16 | 34 |
| $[(2^n - 1) \times \frac{log_2(x+1)}{n}]$ | $6 \times 7$ | 0.132 | 43 | 9.02 | 5 | 20 | 4.70 | 3 | 0.23 | 0.08 | 48 |
| $[(2^n - 1) \times \frac{log_2(x+1)}{n}]$ | $8 \times 9$ | too large | >147 | >22.94 | 2 | 41 | 9.21 | 13 | 0.06 | 0.37 | >60 |
| SINE | $7 \times 10$ | 0.295 | 94 | 19.36 | 2 | 36 | 8.65 | 48 | 0.06 | 1.96 | 55 |
| delta_mod1. | $6 \times 12$ | 0.187 | 56 | 12.60 | 2,4 | 45 | 10.53 | 93 | 0.50 | 2.35 | 16 |
| delta_mod2. | $4 \times 8$ | 0.061 | 13 | 3.76 | 2,3 | 12 | 3.57 | 7 | 0.06 | 0.78 | 05 |
| SINE.small | $4 \times 7$ | 0.042 | 10 | 3.01 | 4 | 5 | 2.07 | 7 | 0.05 | 1.20 | 31 |

Figure 8: Results concerning a deviation up to 5%.

| | | *original* | | | | *modified* | | *statistics* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| name | I/O | org.a. | org.t | org.p. | b.s. | mod.t | mod.p. | abs.d. | rel.d. | sqr.d | red.p. |
| increasing | $6 \times 6$ | 0.149 | 51 | 10.53 | 3 | 18 | 4.32 | 2 | 0.33 | 0.11 | 59 |
| square root | $6 \times 3$ | 0.072 | 20 | 4.32 | 3 | 7 | 1.88 | 1 | 0.50 | 0.08 | 55 |
| 100/x | $6 \times 7$ | 0.090 | 23 | 5.45 | 3 | 15 | 3.95 | 1 | 0.50 | 0.07 | 27 |
| $[(2^n - 1) \times \frac{log_2(x+1)}{n}]$ | $6 \times 7$ | 0.132 | 43 | 9.02 | 3 | 18 | 4.51 | 2 | 0.10 | 0.14 | 40 |
| $[(2^n - 1) \times \frac{log_2(x+1)}{n}]$ | $8 \times 9$ | too large | >147 | >22.94 | 3,4 | 30 | 7.14 | 6 | 0.04 | 0.16 | >60 |
| SINE | $7 \times 10$ | 0.295 | 94 | 19.36 | 3 | 38 | 8.84 | 18 | 0.03 | 0.75 | 54 |
| delta_mod1. | $6 \times 12$ | 0.187 | 56 | 12.60 | 2 | 48 | 11.09 | 35 | 0.50 | 0.84 | 12 |
| delta_mod2. | $4 \times 8$ | 0.061 | 13 | 3.76 | 3 | 10 | 3.20 | 3 | 0.02 | 0.45 | 15 |
| SINE.small | $4 \times 7$ | 0.042 | 10 | 3.01 | 4 | 7 | 2.44 | 3 | 0.06 | 0.56 | 19 |

Figure 9: Results concerning a deviation up to 2%

output lines is increased to 8 and 9, respectively. Note, that the PLA-generator failed on generating a PLA directly from the initial specification.

- The example named "SINE" is a table of 100 items valued by $1000 * sin(\frac{\pi}{2} * \frac{i}{n})$.
- The example named "delta.mod1" is a table which has been used in the design of a delta modulation chip [16]. It countains 64 items valued between 0 and $2^{12} - 1$.
- The example named "delta.mod2" is the second table used in the design of a delta modulation chip. It countains 16 items between '0' and '255'.
- The last example called "SINE.small" refers to the example shown in Figures 2, 4, and 5.

The only difference between the Figures 8, 9, and 10 is, that for the first one a deviation of at most 5 % has been allowed, while for the second one we generated examples with a maximum deviation from the original value of 2%. In the last one (Figure 10) our system has been allowed to choose only truncation or rounding.

The results in Figures 8, 9, and 10 show that an impressive reduction in power consumption of PLAs can be achieved using our tool. Considering the average decrease of power consumption there has been a reduction of at least 30 %.

It should be remarked that the improvements are only due to widening the initial specification and choosing a well-suited modified specification. Other available tools for PLA-optimization (instead of ESPRESSO) can still be used to further reduce chip area and power consumption.

Considering Figures 8 and 9 there are some astonishing results. Partially the better results are obtained using a maximum deviation of 2 % in comparison to 5 %. Partially, the heuristics make better results if smaller deviations are choosen.

## 8 Further activities

Other heuristics for determination of the best suited final specification will be developed and evaluated using additional examples (especially those taken from [1]).

| | | *original* | | | | *modified* | | *statistics* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| name | I/O | org.a. | org.t | org.p. | b.s. | mod.t | mod.p. | abs.d. | rel.d. | sqr.d | red.p. |
| increasing | $6 \times 6$ | 0.149 | 51 | 10.53 | 5 | 31 | 6.77 | 1 | 0.33 | 0.07 | 35 |
| square root | $6 \times 3$ | 0.072 | 20 | 4.32 | 4 | 11 | 2.44 | 1 | 0.50 | 0.09 | 43 |
| 100/x | $6 \times 7$ | 0.090 | 23 | 5.45 | 3 | 17 | 4.32 | 1 | 0.50 | 0.05 | 20 |
| $[(2^n-1) \times \frac{log_2(x+1)}{n}]$ | $6 \times 7$ | 0.132 | 43 | 9.02 | 3,5 | 29 | 6.39 | 1 | 0.10 | 0.09 | 29 |
| $[(2^n-1) \times \frac{log_2(x+1)}{n}]$ | $8 \times 9$ | too large | >147 | >22.94 | 3 | 81 | 16.54 | 1 | 0.02 | 0.04 | >27 |
| SINE | $7 \times 10$ | 0.295 | 94 | 19.36 | 3 | 86 | 17.86 | 1 | 0.01 | 0.07 | 07 |
| delta_mod1. | $6 \times 12$ | 0.187 | 56 | 12.60 | 2,3 | 49 | 11.09 | 1 | 0.50 | 0.15 | 12 |
| delta_mod2. | $4 \times 8$ | 0.061 | 13 | 3.76 | 3 | 10 | 3.20 | 1 | 0.02 | 0.45 | 15 |
| SINE.small | $4 \times 7$ | 0.042 | 10 | 3.01 | 3 | 9 | 2.82 | 1 | 0.05 | 0.20 | 05 |

Figure 10: Results concerning rounding/truncation

## References


[1] S. Yang: Logic Synthesis and Optimization Benchmark User Guide, Version 3.0, MCNC, Jan. 1991

[2] S. Yang, M. Ciesielski: PLA decomposition with generalized Decoders, IEEE Int. Conf. on Computer-aided Design, Santa Clara, CA, 1989, P 312-315

[3] S. Yang, M. Ciesielski: PLA decomposition with generaized Decoders, IEEE Int. Conf. on Computer-aided Design, Santa Clara, CA, 1989, P 312-315

[4] T. Sasao: Mulitiple-valued Logic and Optimization of Programmable Logic Arrays, IEEE Computer, Vol. 21, No. 4, pp 71-80, 1988

[5] R. Brayton, F. Somenzi: Boolean Relations and the Incomplete Specification of Logic Networks, VLSI 89, Elsevier, North Holland, pp231-240, 1989

[6] R. Brayton, G. Hachtel, A. Sangiovanni-Vincentelli: A Survey of Optimization Techniques for Integrated Circuit Design, IBM Watson Research Center, Yorktown Heights, Report RC 8882, 1981

[7] R. Brayton, G. Hachtel, L. Hemachandra, A. Newton, A. Sangiovanni-Vincentelli: A comparison of logic minimization strategies using ESPRESSO. An APL program package for partitioned logic minimization; Proc. Int. Symp. on Circuits and Systems, pp. 43-49, Rome, May 1982

[8] R. Brayton, G. Hachtel, C. McMullen, A. Sangiovanni-Vincentelli: Logic Minimization Algorithms for VLSI, Kluwer Academic Press, 1986

[9] S. Chakroun, E. Melcher, W. Röthig, M. Dana: Power Consumption Estimation Using Statistical Signal Properties. Proc. EUROMICRO 92, Paris, Sept. 14.-17., pp. 691-696

[10] D. Dietmeyer, M. Doshi: Automated PLA Synthesis of the Combinational Logic of a DDL Description, Design Automation and Fault-tolerant Computing, Vol. 3, Issue 3/4, pp 241-257, 1980

[11] G. DeMicheli, M. Hofmann, R. Newton, A. Sangiovanni-Vincentelli: A System for the Automatic Synthesis of Programmable Logic Arrays, in Advances in Computer-Aided Engineering, A. Sangiovanni-Vincentelli editor, Jay Press, 1984

[12] F.Dresig, P. Lanches, O. Rettig, U. Baitinger: Simulation and Reduction of CMOS Power Dissipation at Logic Level, Proc. EUROASIC 93, Paris, Feber, 22-25, 1993, pp. 341-346

[13] European Silicon Structures Ltd., Solo 1400 Release 3.1 Reference Manual, Berkshire, 1991

[14] E.McCluskey: Introduction to the theory of Switching Circuits, McGraw-Hill, 1965

[15] C. Mead, L. Conway, Introduction to VLSI Systems, Addison-Wesley, 1980

[16] R. Rauscher: Entwurf und Implementierung eines ICs zur krümmungsadaptiven Deltamodulation, University of Hamburg, internal Report Nr. 210, Sept. 1992

[17] T. Sasao: Logic Synthesis and Optimization, Kluwer Academic Press, 1993

[18] T. Sasao: Input Variable Assignment and Output Phase Optimization of PLA's, IBM Yorktown Heights Res. Center, RC 10003, 1983